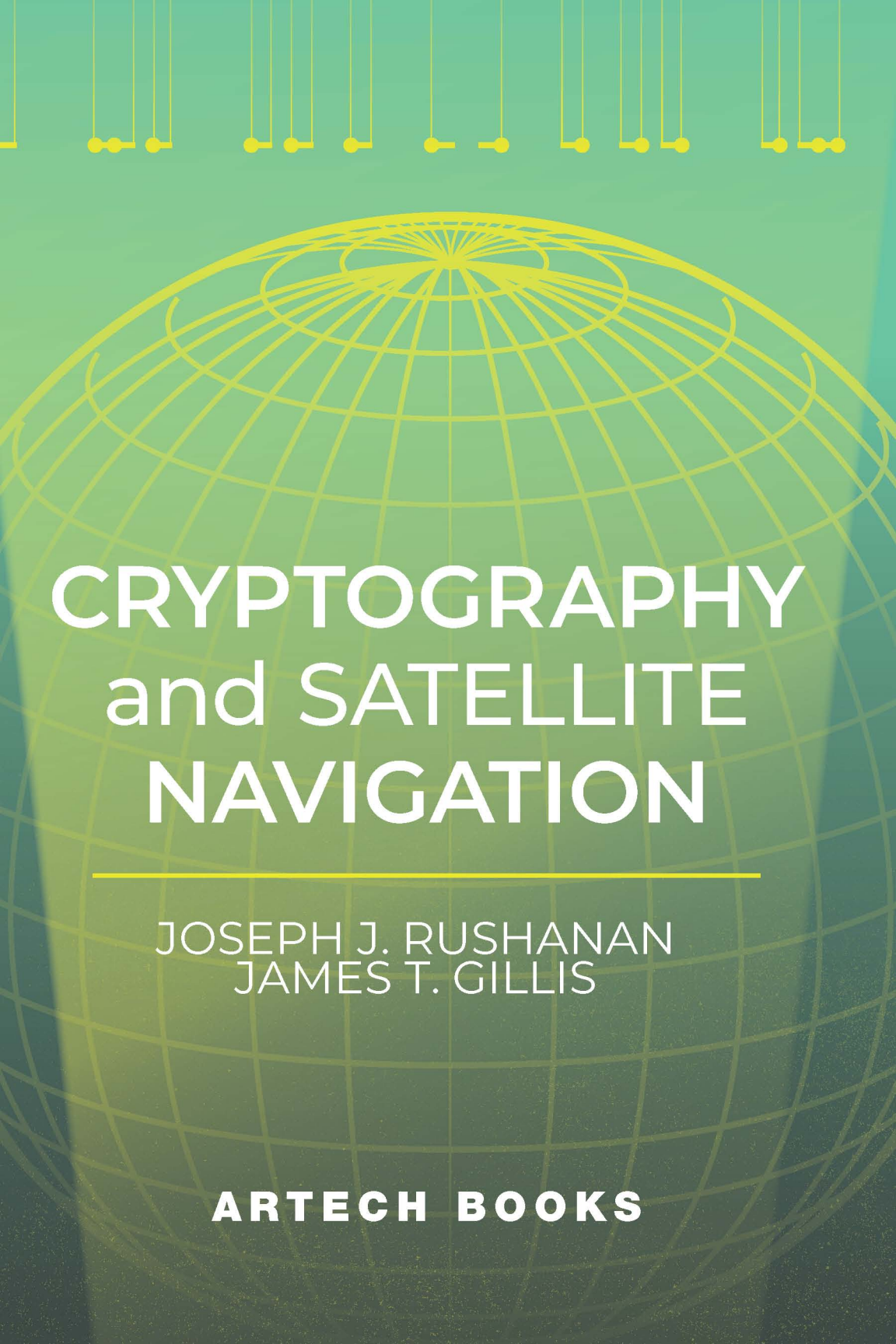

CRYPTOGRAPHY and SATELLITE NAVIGATION

JOSEPH J. RUSHANAN
JAMES T. GILLIS

ARTECH BOOKS

Cryptography and Satellite Navigation

For a complete listing of titles in the
Artech House GNSS Technology and Applications Library,
turn to the back of this book.

Cryptography and Satellite Navigation

Joseph J. Rushanan

James T. Gillis

ARTECH
HOUSE

BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library.

Cover design by Joi Garron

ISBN 13: 978-1-68569-031-1

685 Canton Street
Norwood, MA 02062

Approved by the MITRE Corporation for Public Release; Distribution Unlimited. Public Release Case Number 24-2487.

Joseph J. Rushanan's affiliation with the MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author.

James T. Gillis's affiliation with the Aerospace Corporation is provided for identification purposes only, and is not intended to convey or imply Aerospace's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author.

10 9 8 7 6 5 4 3 2 1

Contents

	Preface	***xvii***
	Acknowledgments	***xix***
	Part I **Satellite Navigation**	**1**
1	**Introduction**	**3**
1.1	What Is This Book About?	3
1.2	Who Is This Book For?	4
1.3	What Do We Mean by Assurance?	5
1.3.1	Confidentiality	6
1.3.2	Integrity	7
1.3.3	Availability	7
1.3.4	Authentication and Identity	7
1.3.5	Nonrepudiation	8
1.4	What Do We Mean by PNT Assurance?	8
1.4.1	Assurance, Integrity, and All That	8
1.4.2	Confidentiality	9
1.4.3	Integrity	9
1.4.4	Availability	9
1.4.5	Continuity	9

1.4.6 Authentication and Identity 9
1.4.7 Nonrepudiation 10

1.5 What Is Cryptography, and How Can It Help? 10

1.6 How Is This Book Organized? 11

End Notes 12

References 13

2 Overview of Satellite Navigation 15

2.1 Navigation Signals from Space 16

2.2 The Segments of Satnav Systems 19
2.2.1 Space Segment 19
2.2.2 User Segment 20
2.2.3 Mission Segment 22
2.2.4 Control Segment 24

2.3 Descriptions of Orbits 25

2.4 Specific Constellations 27

2.5 Satnav Signals 27

2.6 Correlation 31
2.6.1 An Instructive Example 32
2.6.2 Misalignment Compensation 35
2.6.3 The Importance of Correlation 36

2.7 Satnav Signal Data 36

2.8 Takeaways 37

End Notes 37

References 38

Part II
Cryptography 41

3 Symmetric Cryptography 43

3.1 Classic Ciphers 44
3.1.1 The Ancient World and Child's Play 44
3.1.2 To the Digital Age 45

3.2	An Overview of Ciphers	46
3.2.1	Block Ciphers	47
3.2.2	Stream Ciphers	47
3.2.3	Much of the Same	48
3.3	Desired Properties	48
3.3.1	Information Theory, or Shannon, Part 1	49
3.3.2	Confidentiality	50
3.3.3	Pseudorandom Generation	54
3.3.4	Avalanche Property	54
3.4	General Principles for Creating Ciphers	55
3.4.1	Shannon, Part 2	55
3.4.2	Substitution-Permutation Construction	56
3.4.3	Feistel Construction	57
3.4.4	Stream Ciphers	59
3.4.5	Performance	59
3.5	AES	59
3.5.1	Choosing the AES Cipher	60
3.5.2	AES Structure	60
3.5.3	AES Performance	62
3.5.4	AES Takeaway	63
3.6	Lightweight Crypto	63
3.6.1	Add-Rotate-XOR Ciphers	63
3.6.2	ChaCha20	63
3.6.3	Simon and Speck	64
3.7	Cipher Modes	65
3.7.1	Overview	65
3.7.2	Electronic Codebook	66
3.7.3	Counter Mode	67
3.7.4	Cipher Block Chaining	69
3.7.5	Which Mode to Use?	70
3.8	Steganography	70
3.9	Foreshadowing: Navigation Examples	72
3.10	Takeaways	72
	End Notes	72
	References	73

4	**Hashing**	**77**
4.1	Hash Functions and Their Goals	78
4.1.1	Preimage Resistance	79
4.1.2	Collisions	80
4.1.3	A Birthday Surprise	80
4.1.4	Other Properties	82
4.2	Construction	82
4.2.1	SHA-2 Family	84
4.2.2	SHA3	86
4.3	Message Authentication Codes	87
4.3.1	What Doesn't Work	88
4.3.2	Hash-Based Message Authentication Code	89
4.4	Ciphers and Authentication	89
4.4.1	Cipher Modes	90
4.4.2	Counter Mode with CBC-MAC	91
4.4.3	Galois Counter Mode	92
4.4.4	Authenticated Ciphers	93
4.5	Application: Digital Fingerprints	94
4.5.1	Message Digest	94
4.5.2	Secure Verification	94
4.5.3	Watermarking	95
4.6	Application: Chains, Trees, and Blockchain	96
4.6.1	Hash Chains	96
4.6.2	Merkle Trees	97
4.6.3	Blockchain and Cryptocurrency	97
4.7	Application: Bit Commitment	98
4.8	Application: TESLA	99
4.8.1	Main Idea	100
4.8.2	Data Stream	100
4.8.3	Hash Chain of Keys	101
4.8.4	Message Stream and MACs	101
4.8.5	Authentication via Digital Signatures	101
4.8.6	Properties	102
4.9	Foreshadowing: Navigation	103
4.10	Takeaways	103

End Notes 104
References 105

5 Public Key Cryptography 109

5.1 Motivation and History 110
5.1.1 Physical Analogs 110
5.1.2 History 111

5.2 Public Key Cryptography Goals 111
5.2.1 Encryption 111
5.2.2 Signatures 112
5.2.3 Key Exchange 114

5.3 Math Foundations 115

5.3.1 Number Theory 101 115
5.3.2 Hard Number Theory Problems 117

5.4 RSA 118
5.4.1 Definition 118
5.4.2 Justification 119
5.4.3 Implications 120
5.4.4 Signatures 120
5.4.5 Practicalities 120
5.4.6 Attacks 121
5.4.7 Strengths 121
5.4.8 Performance 121

5.5 Digital Signature Algorithm 122
5.5.1 Definition 123
5.5.2 Justification 124
5.5.3 Implications and Attacks 124
5.5.4 Elliptic Curve Methods 125
5.5.5 Strengths 127
5.5.6 Performance 128

5.6 Diffie-Hellman Key Exchange Protocol 128
5.6.1 Protocol 129
5.6.2 Implications and Attacks 130
5.6.3 Elliptic Curve Methods 131
5.6.4 Strengths 131

5.7 Applications: More on Signatures 131

5.7.1	Time Stamps	131
5.7.2	Blinding and Blind Signatures	132
5.7.3	Double Signatures	132
5.8	Applications: PKI	133
5.8.1	PKI Structure	134
5.8.2	PKI Operations	134
5.8.3	Certificates	135
5.8.4	Important Points	136
5.9	Applications: Secure Email	136
5.9.1	Goals	137
5.9.2	General Framework	137
5.9.3	Nuances	139
5.10	Foreshadowing: Navigation	139
5.11	Takeaways	140
	End Notes	140
	References	141
6	**Cryptographic Protocols**	**143**
6.1	Protocol Principles	144
6.1.1	The Need for Trust	145
6.1.2	Ensuring Trust	145
6.1.3	There's Always Risk	146
6.1.4	Trust in the Protocol	147
6.2	Identity Methods	147
6.2.1	General Traits	147
6.2.2	Cryptographic Identification	148
6.2.3	Zero-Knowledge Proofs	150
6.3	Managing Keys	152
6.3.1	Key Management Ingredients	152
6.3.2	Public Key Methods	153
6.3.3	Symmetric Methods	156
6.3.4	Comparisons	158
6.4	The Network Stack	158
6.4.1	The Network Stack	159
6.4.2	Layer 1: The Physical Layer	160
6.4.3	Layer 2: The Link Layer	161

6.4.4 Layer 3: The Network Layer 163
6.4.5 Layer 4: The Transport Layer 164
6.4.6 Upper Layers: Applications 167
6.4.7 Observations 167

6.5 Other Protocols 168
6.5.1 Other Communication Protocols 168
6.5.2 Internet of Things 168
6.5.3 Zero Trust 169

6.6 Takeaways 170

End Notes 170

References 172

Part III
Securing Satellite Navigation 175

7 Cryptography and the Satnav Enterprise 177

7.1 Space and Security 177

7.2 Protocols and Satnav 178
7.2.1 Trust 178
7.2.2 Identity 179
7.2.3 Key Management 180

7.3 Satnav Infrastructure 181
7.3.1 Space Segment 182
7.3.2 Control Segment 183
7.3.3 Mission Segment 183
7.3.4 User Segment 185
7.3.5 Threats 186
7.3.6 What about Satnav Signals? 187

7.4 A Summarizing Sample Enterprise 187
7.4.1 Common Elements 189
7.4.2 Mission Segment 190
7.4.3 Control Segment 190
7.4.4 Ground Antenna 191
7.4.5 Monitoring Stations 191
7.4.6 Space Segment 191
7.4.7 Ephemeris Service 192
7.4.8 Key Management 192

7.4.9	User Equipment	193
7.5	Satnav Signals	194
7.5.1	Satnav Signal Layer and Security Goals	195
7.5.2	Taxonomy of Threats	197
7.6	A Brief Relevant History	200
7.6.1	The Need: Pre-2000	200
7.6.2	2000s	201
7.7	Takeaways	202
	End Notes	202
	References	203
8	**Navigation Message Authentication**	**209**
8.1	Protecting Navigation Data	209
8.1.1	Goals	210
8.1.2	Constraints	210
8.1.3	Measures	211
8.2	History	211
8.2.1	Original Thoughts	212
8.2.2	Recent Trends	212
8.3	General Methods for NMA	212
8.3.1	Using Only Digital Signatures	213
8.3.2	Using TESLA	215
8.3.3	Out-of-Band	219
8.3.4	Summary	220
8.4	Galileo OSNMA	221
8.4.1	Overview	221
8.4.2	Overall Architecture	222
8.4.3	Cryptographic Methods	223
8.4.4	Performance	226
8.4.5	Future Plans	227
8.5	Chimera	227
8.5.1	NMA on Chimera	227
8.5.2	Baseline Chimera	228
8.5.3	TESLA Chimera	229
8.5.4	An Experimental Design	231

8.6 Summary and the Future 231
End Notes 232
References 232

9 Spreading Code Protection 235

9.1 Protecting the Signal Lower Layers 235
9.1.1 Goals 236
9.1.2 Constraints 236
9.1.3 Measures 237

9.2 History 237

9.3 General Cryptographic Methods 238
9.3.1 Cryptography and Spreading Codes 238
9.3.2 Shared Key Methods 239
9.3.3 Bit Commitment Methods 240

9.4 Applying Cryptography to Spreading Codes 241
9.4.1 Ciphertext as Spreading Codes 241
9.4.2 Ciphertext as Markers 242
9.4.3 Ciphertext Modifying Spreading Chips 243

9.5 Chimera Markers 245
9.5.1 L1C Spreading Code Structure 246
9.5.2 Chimera Marker Overview 246
9.5.3 Cryptographic Processing 247
9.5.4 Creating the Markers 249
9.5.5 Performance 249

9.6 Takeaways 250
End Notes 250
References 250

10 Hybrid Protection of a Satnav Signal 253

10.1 The Issue and Options 253
10.1.1 The Problem 253
10.1.2 The (Only?) Choices 254
10.1.3 A Holy Grail 254

10.2 Protecting Both the Signal and Data 255
10.2.1 Goals 255

10.2.2 Constraints 256
10.2.3 Measures 256

10.3 History 256

10.4 General Methods 257
10.4.1 Hybrid Approaches Using Bit Commitment 257

10.5 NTS-3 258
10.5.1 Baseline Chimera Overview 259
10.5.2 TESLA Chimera Overview 263
10.5.3 Processing 265
10.5.4 Performance 265
10.5.5 Multichannel Chimera 266

10.6 Galileo Signal Authentication Service 270

10.7 Takeaways 272
End Notes 272
References 272

11 Other Things and Going Forward 275

11.1 Other Navigation Examples 275
11.1.1 Snapshot Receivers 275
11.1.2 Using Location in Cryptography 277

11.2 But What About ...? 278
11.2.1 Homomorphic Encryption 278
11.2.2 Blockchain 279
11.2.3 Physical Layer Key Exchange 279
11.2.4 Implementation Security 280

11.3 Future Trends 281
11.3.1 Future Satnav 281
11.3.2 Other Navigation Methods 281

11.4 Our Final Thoughts 282
References 283

Appendix
The Influence of Quantum 285

A.1 The Components of Quantum Computing 286

A.1.1	States and Superposition	287
A.1.2	Matrices: Hermitian, Unitary, and Measurement	288
A.1.3	Entanglement	289
A.1.4	No Cloning	293
A.2	Quantum Computing Algorithms	293
A.2.1	Deutsch's Algorithm	294
A.2.2	Grover's Algorithm	296
A.2.3	Shor's Algorithm and Impacts	299
A.3	Post-Quantum Cryptography	303
A.3.1	Symmetric Algorithms	303
A.3.2	Asymmetric Cryptography	304
A.3.3	Complexity Theory	304
A.3.4	Code and Lattice-Based Cryptography	307
A.3.5	Multivariate Cryptography	310
A.4	Takeaways: Quantum and Post-Quantum	312
	End Notes	312
	References	314
	About the Authors	**319**
	Index	**321**

Preface

It is hard to underestimate the profound impact satellite navigation, like Global Positioning System (GPS), has had on the world. With such usefulness and ubiquity comes the emergence of bad actors, using cheap jammers and misleading signals. That in turn requires methods that provide security.

This book has sprung from our efforts in protecting and validating satellite navigation, mainly through the use of cryptography. We have worked together on this problem since the late 1990s, while working more generally on security problems since the 1980s. Our greatest hope is that this book helps the reader understand the difficult and unique problems that come with securing satellite navigation. And perhaps helps inspire the next generation of mathematicians and engineers working to secure these precious systems.

Acknowledgments

Together, we would like to thank several individuals who have influenced, inspired, and, at times, cajoled us in our years working on satellite navigation. Jon Anderson, as a Captain, Colonel, and contractor, has been there from the start of our collaboration, always appearing when things got dull and needed a push to get ideas moving. Then Captain Katie Carroll was the driving force behind our Chimera work, which led indirectly to our tutorials on cryptography and ultimately this book. Katie's initial direction has been wonderfully continued by Drs. Joanna Hinks and Madeleine Naudeau at AFRL, who always demand realizable creativity. Logan Scott has been a constant font of ideas and challenges, always interesting and challenging.

Besides being an important colleague in our work, Renee Yazdi has done an amazing job reviewing this book. The foibles and errors are ours, but are much fewer because of Renee.

We also want to thank our other reviewers: Dr. Alex Cerruti, Adam Woodbury at MITRE; Roger Knobbe, Dr. David Goldstein, Dr. Joe Touch, Bob Lindell, Dr. Phil Dafesh, Dr. Christine Black, Rachel Allen at Aerospace; and Ignacio Fernandez-Hernandez at EC.

Finally, Julia Sayger at Artech House has been great, explaining the process and keeping us almost on schedule.

Dr. John Betz brought me to GPS in 1998, followed soon after with Dr. Chris Hegarty getting me to wonder how one could authenticate GPS messaging. Through the years at MITRE, there have been many, many colleagues who have guided and collaborated on various security and/or navigation endeavors; too many to name, but not too many that I don't appreciate each one. A special shout-out goes to MITRE's embedded security team, who have helped me both

learn and teach novel uses of cryptography. It has been a privilege in the past decade to be a lecturer at Northeastern University's Khoury College of Computer Science for the course in Applied Cryptography. Needless to say, this book owes much to that experience.

My family has always supported me, especially my dear sister Ann Marie, and the memories of my parents, Joe and Vicky, guide me daily. I have been blessed with numerous friends, some known for over 50 years, some much newer. There is nothing like smart, fun, and close friends to keep one driven, entertained, and humble. Finally, I dedicate my part of this book to my children, Rita and Jared: sometimes a cipher, always a joy.

—Joe J. Rushanan

I'd like to acknowledge the late William Feess for getting me involved in GPS integrity in the early 1980s with L-Band monitoring and Rita Lollock and John Clark for providing fantastic growth opportunities while working on the GPS at the GPS Wing. Paul Timmel was an immeasurable part of my learning about cryptography. Rawna Haddad has been a great mentor and partner for the last 20 years in my authentication work, along with Karl Kovach Michael Cole, and, over the previous 15 years, Bob Lindell.

Calvin Miles at the FAA, Dr. Andrew Hanson at the Volpe Transportation Center, and Karen Van Dyke at the Department of Transportation have been great to work with. The team at Stanford's Center for Navigation deserves a shout-out, too. There are many others, too—all wonderful.

No accomplishment happens without the support of one's family, I'm grateful to my wife, Fran, for her forbearance, love, and encouragement. My daughter Emi for her disposition and moxie. And my son Ethan for his perseverance and sense of dedication. As I write, he is with the Marine Corps. Third Reconnaissance Battalion in Okinawa, Japan.

—J. T. Gillis

Part I
Satellite Navigation

1

Introduction

Our goal is to show how cryptography could, is, and should be used to achieve assurance for satellite-based positioning, navigation, and timing (PNT), which we denote as satnav. This introduction gives a foundational overview of the central concepts in assured navigation and their associated history in preparation for subsequent deeper explorations of the supporting topics.

1.1 What Is This Book About?

Most personal electronic devices give position and time automatically, and from that, navigation easily follows. In practice, satnav systems like a Global Navigation Satellite System (GNSS), infrastructure such as the Global Positioning System (GPS), or a Regional Navigation Satellite System (RNSS) set of satellites such as the Quasi-Zenith Satellite System (QZSS) contribute to that position, navigation, and timing data.[1] As with any other kind of information service, the assumption is that this information is true—that one can believe the position and time are what we are given. That belief is our informal definition of PNT assurance: the degree to which a user can trust that PNT is accurate. This notion is formalized later.

Assurance, in general, is achieved by putting protections in place to mitigate anything that would cause the information to deviate from truth. In cybersecurity, the whole subject of system security engineering exists there to develop methods to provide assurance. Adapting that thinking to the PNT world has been considered, as discussed later in this chapter, but delving into those broad areas is beyond our scope.[2]

Instead, we focus on one aspect of security engineering: cryptography, which is a large class of protection methods applied to digital information. Cryptographic methods have been applied to satnav even though much of the information being protected is not traditional digital information. These nontypical uses of cryptography for satnav are often challenging for users, who may not be familiar with cryptography. Our goal is to alleviate that challenge, bringing us to this book's mission, which is:

> Present the how and why of cryptographic methods and show where they are and can be applied to assure satnav.

As a corollary, the authors have spent the last decades working in the intersection of these two fields. Our biggest hope is that our experience will educate and motivate our reader.

1.2 Who Is This Book For?

Any book that tries to blend together two technical and almost nonoverlapping areas has the challenge of establishing its audience. As we mentioned, the use of cryptography in satnav is often obscure to navigation professionals, who may lack in-depth knowledge in cryptography, and thus those individuals are a prime target for this book. Our discussion of cryptography, arguably brief when compared to the numerous, excellent books devoted specifically to that topic, will hopefully suffice as a gentle reference and reminder as new methods are developed.

At the same time, we also have a separate target audience and one perhaps more challenging to please. We have observed that explaining the need for cryptographic methods in navigation to a cryptography professional is challenging, largely because the information that is being protected, namely the navigation signals broadcast from a satellite, are not the traditional focus of cryptography. For example, how would one digitally sign a radio frequency (RF) signal? Thus, this book addresses where assurance methods are needed in navigation, and how cryptography can be applied. Accomplishing that requires that we offer a description of satnav that is enough to show where the cryptographic methods are applied, but not too much to overwhelm those who are not navigation engineers by training.

In summary, our intended reader is interested in the union of these two questions:

- For the navigation engineer, what are cryptographic methods and how are they used in satnav?

• For the cryptographer, what are the needs of satnav for cryptography?

This intent then begs the question: what are the expectations on the reader? We do not assume any in-depth familiarity with the two broad areas. In general, we do assume some level of technical sophistication, say at the undergraduate level. We present equations and algorithms with some details with no more than a reminder of the necessary background. There is some math, but again nothing more than at the undergraduate level (although perhaps not necessarily included in the reader's particular undergraduate math classes). The chapter end notes offer more details not germane to the main topic of the chapters.

More specifically, for the navigation engineer, the discussion of satnav should be a refresher with almost surely the occasional "Why was that emphasized?" or "Why was that omitted?" Even so, some of the navigation topics may be somewhat new, such as the variety of GNSS signals and how navigation messaging is done. For those familiar with cryptography, our cryptographic discussion will also be a recap, although we are sure there will be new topics. In particular, the material in the appendix on quantum technologies may be new to many readers. For the general security engineer or cybersecurity specialist, we expect that the application of common concepts will be novel when considered for satellite navigation.

To summarize, our expectations for the reader is that they are curious about these subjects and willing to see interconnections between them.

1.3 What Do We Mean by Assurance?

There are many ways that assurance can be defined. Consider this definition for assurance from the National Institute of Standards and Technology (NIST) [1]:

> The justified confidence that the system functions as intended and is free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted as part of the system at any time during the life cycle.

In more informal terms, assurance is confidence (belief, trust) that we achieve the behavior that we expect, whether it is in data, communications, networks, or even PNT. We achieve assurance by preparing for possible exploitations and adapting protections against them. Terms that describe assurance are as numerous as ways to define assurance. For example, additional terms include security, robustness, resilience, and integrity. We necessarily establish our own vernacular for the book, much of which is presented in this chapter, and we will translate other concepts/terms as needed. For example, we say something is *secure* or *robust* if it is well-protected from an exploit. That is, it can withstand

an attack. Going further, we say that something is *resilient* if it can bounce back or recover from an attack.

These two concepts usually require different protection methods. For example, personal identification measures can ensure that a credit card is protected from fraudulent use. The ability to quickly cancel a credit card and issue a new one is a way to ensure recovery when those protection methods fail.

A high level of PNT assurance means that a user has high confidence that their PNT solution is accurate, whether in the face of adversaries or in the presence of other faults. It is related to being robust against faults, which we define as maintaining expected behavior in the face of natural phenomena. The two concepts are related, for example, random interference to an RF signal will often be indistinguishable from adversarial produced interference (i.e., jamming). But the two concepts require different solutions, as we elaborate throughout this book. The difference between adversarial concerns and faults also shows how even as simple a word as "protection" can be overworked, (i.e., we say "an umbrella protects me from the rain" even though the rain is not an actual attack).

Defining assurance in terms of confidence in the face of adversaries requires defining specific goals that we want to maintain in their presence. The rest of this section defines those goals. The assurance goals defined for cybersecurity security purposes are as a rule denoted by the triad of *confidentiality*, *integrity*, and *availability* (CIA) [1], although we call out other goals as needed. These concepts are important in that specific cryptographic methods are designed to achieve certain goals. Furthermore, as we define the navigation signals that we want to protect, defining "protection" means defining confidentiality, integrity, and availability.

1.3.1 Confidentiality

Confidentiality means keeping information private and revealing it only to someone who is authorized to have the information. Confidentiality is often nuanced in that perhaps only certain information need be kept private. For example, email messages may have the message body encrypted, but the list of recipients is left open, usually necessarily by the mail protocols. Deciding what data needs to be confidential leads to examining the consequence if the data is exposed. Such considerations suggest various solutions, such as segregating data based on its importance and protection methods.

Related to the notion of confidentiality is that of exclusivity. Here the desire is not so much that the information is private but that it cannot be used except by an authorized user; of course, information that is private is also exclusive. An example of exclusive information is a process that generates random

numbers in an unpredictable manner, such as is found in some authentication devices. The actual numbers are not confidential once they are generated, but the process is such that only the authorized user will have access to these numbers at that time.

1.3.2 Integrity

Information integrity means that the information is not modified. That is, the user can be assured that the information is in the state it should be. Integrity is often achieved by various protections from errors, such as parity checks and error correcting codes. Such methods give assurance that the data remains intact. In fact, the concept of integrity often is used purely to mean that the information can perform its desired function (e.g., that the system is operating error-free). However, as we see later, such methods don't protect against some threats, and thus additional methods are needed. These observations lead to an important point:

> *The Many Meanings of Integrity*
> Integrity as a concept applies both to freedom from faults and resistant to certain threats.

1.3.3 Availability

The third item of the CIA triad is availability, which is the concept that information is usable when desired. While perhaps the most fundamental of security goals—what use is information if it is not available—it is also one of the hardest to achieve. Attacks against availability are often called *denial of service*, and such attacks can take the form of cutting a wire, flooding a network, jamming a signal, timing out an access, and so on.

The concept of continuity is related to availability: it is availability desired for a span of time. Many services, such as networks, communication, and PNT desire not only that the service be available at some instance, but that the service is available at every instance. Denial of service attacks apply here perhaps more so, since intermittent denial of service can defeat the expected continuity goal—reliable service—even if the service is available on occasion.

1.3.4 Authentication and Identity

Information is generated by some source. The goal of authentication and identity are the desire to assure that the source is correct and can be verified. These concepts are related to integrity: if the source was modified/changed/spoofed

then the information has been modified. That said, we feel it is important to highlight authentication/identity separately, since, for example, someone capturing information and resending it has violated authentication even if the information itself has not been modified.

1.3.5 Nonrepudiation

Nonrepudiation is the goal of the source not being able to deny or repudiate that it is the source of the information. A common example in the financial world are checks, and the desire that someone who has signed a check cannot later repudiate that check and claim they did not sign it. Nonrepudiation is related to how much trust there is in the entities. Almost by definition, if we trust the source of the information, then we do not worry about repudiation.

1.4 What Do We Mean by PNT Assurance?

The focus of this book is on cryptographic methods to achieve assurance for satnav. That focus requires first defining assurance for PNT; our focus on satnav is expanded more in Chapter 2. As with the general concept of assurance in the previous section, PNT assurance is all a matter of applying security goals to achieving PNT. Before recapping the various goals and how they manifest for PNT, we first need to discuss terminology, which requires reviewing some history.

1.4.1 Assurance, Integrity, and All That

The story begins with the concept of *PNT integrity*. The idea is that if the satnav system can bound system errors due to faults and other random events, then we can predict PNT performance such as accuracy and availability. This binding is part of the various satnav system requirements. GPS, as an example, specifies specific system bounds [2]. The word integrity here is used in the sense of general system integrity; that is, that the system functions as it should and not strictly as in "information integrity."

However, before long the notion of integrity also began to be synonymous with a more general sense of security or assurance. A thorough discussion can be found in [3]. See also the products by the Department of Homeland Security (DHS) [4]. Our take is that in a book about cryptography, it will get confusing to use the word integrity in a manner separate from that given in the last section. That is, we want to use "integrity" to represent the CIA triad, where integrity is the "I." Thus we stress and repeat:

Avoiding PNT Integrity
While PNT integrity is often used to denote what we call PNT assurance, we restrict use of integrity to the usual cybersecurity meaning.

In other words, since assurance is defined by achieving the various security goals, PNT assurance is defined to be the same, only when the goals are specifically applied to PNT. We offer some specifics here; more details are given throughout the book.

1.4.2 Confidentiality

Confidentiality applies to satnav in a few ways. There may be enterprise information that needs to be kept hidden, such as in the use of private services. Such services often create features in signals that they themselves should be private, at least until the signal is broadcast.

1.4.3 Integrity

Integrity applies both to the enterprise that provides PNT and to the signals produced. For the enterprise, the system in a very broad sense should be resistant to modification; a satellite should be inviolate to being changed in an unauthorized manner. For signals, integrity means the signals have content and perform the way they are supposed to. Indeed, this was the motivation behind PNT integrity; the satellite, and consequently its signals, should behave as they are supposed to.

1.4.4 Availability

Satnav systems are meant to be literally ubiquitous and available at any time. That goal applies here. Indeed, the GPS Program Office used to have as its motto "Any time, any place, right time, right place." Note there are many ways availability can fail: jammed signals, satellite failures, and the like. Another factor that affects availability is the system design itself.

1.4.5 Continuity

Similar to availability, continuity applies to a satnav system in terms of its reliability: How long does the system stay up for usage? Continuity has the same kinds of threats and faults as availability.

1.4.6 Authentication and Identity

Information generated by the satnav system should be authenticated to assure that it came from the system and not someone else. This goal means authen-

ticating the information and establishing the identity of the system. The most common attacks on authentication are false signals.

1.4.7 Nonrepudiation

Nonrepudiation is usually not a common goal for PNT assurance mainly because we often have full trust in the satnav system, and so it would be unlikely that the system would repudiate information that it sent (i.e., attesting that it did not send that signal). That said, as the world moves more and more to using different entities and systems, methods that prevent repudiation will arise, often as a natural consequence of methods used to establish identity.

1.5 What Is Cryptography, and How Can It Help?

There are many ways to achieve the assurance goals for information. For example, confidentiality can be achieved by limiting access to the information to just a few, or limiting exposure, such as locking in a safe or changing the information often. However, those types of solutions end up being specific to the application. It is natural instead to devise methods that can be applied universally to achieve the goals. Developing those methods is the subject of cryptography.

Cryptography means "hidden writing," the science of creating procedures that hide information. That usage dates back millennia. For example, Julius Caesar used a cipher to obscure his dispatches. Creating ciphers for that purpose are the most common examples of cryptography, but cryptography can be used to achieve other goals besides confidentiality. Cryptography can ensure integrity, can establish identity, and can prevent repudiation. Cryptography can even help prevent some denial-of-service attacks.

Part II of this book presents a brief, broad view of cryptography with separate discussions on the three main categories: ciphering, hashing, and public key methods. Figure 1.1 shows this taxonomy along with the various goals that are achieved. Symmetric cryptography concerns ciphering, which the traditional use of cryptography concerned with algorithms used to hide information. Even so, such algorithms can be used to provide other goals, such as integrity via message authentication codes. Hashing concerns algorithms that give secure fingerprints or digests of information. They allow one to fully believe that information has not been modified and can be used with or without shared information like keys. Finally, asymmetric cryptography or public key methods are algorithms that can achieve the goals of the previous two categories, but not requiring that some privately held information be shared ahead of time. As such, they can be used for ciphering or signatures. They can also be used for key exchange, which is an enabler for the first two categories.

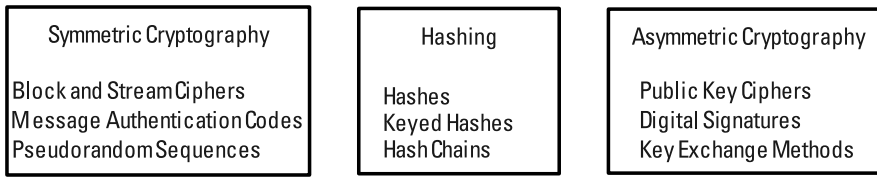

Figure 1.1 A cryptography taxonomy.

There are many good sources for cryptography.[3] Our necessarily broad view wants to give enough background to motivate and describe those techniques for satnav. Unfortunately, such descriptions also need to be abridged with regard to the technical details. (Unfortunate because such an abridgment is painful to us as mathematically trained authors.) Again, the cited sources are useful if the reader desires more technical details.

1.6 How Is This Book Organized?

Applying cryptography to satellite navigation requires covering both topics first in general before the specifics. This book thus is divided into three parts, as shown in Figure 1.2. Part I consists of this introduction and Chapter 2, which describes the main components to satnav with a focus on navigation signals.

Part II comprises the full range of cryptography needed for this book; four chapters that cover cryptographic algorithms and protocols. Chapter 3 describes the goals, algorithms, and associated methods using symmetric or shared-key ciphers. Chapter 4 discusses hashing, which are methods that yield digital fingerprints: a small distillation of data that can ensure integrity. Chapter 5 describes asymmetric cryptography, which differs from symmetric in that

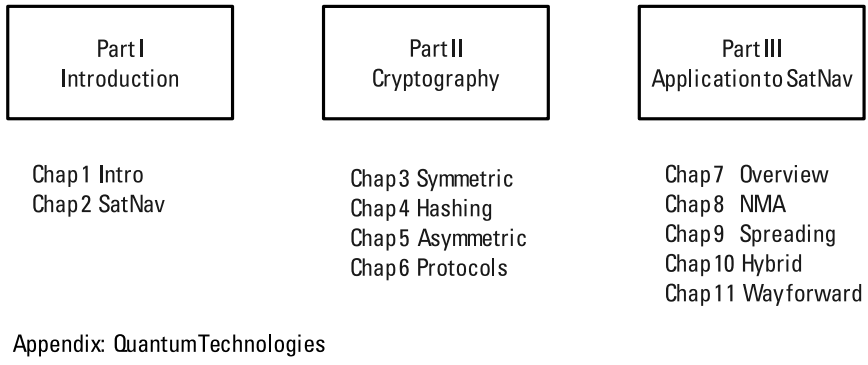

Figure 1.2 A map of this book.

entities do not share the same cryptographic material; asymmetric cryptography enables encryption to just a single party, digital signatures, and methods to create a shared key. Finally, Part II closes with Chapter 6, which defines general protocols that stitch cryptographic algorithms together.

The final Part III covers the applications of cryptography to satnav divided by the focus area of the methods. Chapter 7 extends the discussion of protocols to applications in the satnav system in general before we segue to the focus on satnav signals. Chapter 8 describes Navigation Message Authentication (NMA), which are methods to protect data in a satnav signal. Chapter 9 looks at methods to protect the timing in a satnav signal by considering methods applied to the signal spreading codes. Chapter 10 combines these two approaches to hybrid approaches such as the Chips Message Robust Authentication (Chimera) protocol. We finish the book in Chapter 11 with some miscellaneous topics and a summary of the most important takeaways.

An appendix gives a very brief overview of quantum technologies and their impact on cryptography.

End Notes

1. For example, most people likely get their time information through the cellular phone system, which begs the question: where does that system get its time? GPS provides that time, which is in turn furnished by the U.S. Naval Observatory working with international standards.

2. For example, we do not touch on risk management, such as in the Risk Management Framework [5]. Issues such as policy and legal aspects are also not discussed. Finally, emerging standards (at the time of this writing) such as Institute of Electrical and Electronics Engineers (IEEE) P1952 [6] for a resilient PNT platform are relevant in that the methods in this book fit into creating such a platform, but the standard itself is not discussed.

3. These are cryptography references we find helpful:

   - [7] Gives cryptography from the perspective and fundamental problems being solved;
   - [8] While predating such modern algorithms as AES, this book offers an excellent overview of the underlying mathematics and reasoning behind cryptography;
   - [9] A thorough presentation of all of cryptography;
   - [10] Another good textbook on cryptography;
   - [11] Focuses on the application of the cryptographic algorithms, although predates some modern algorithms;
   - [12] A more popular introduction to the history of cryptography.

References

[1] National Institute of Standards and Technology, *NIST Computer Security Resource Center Glossary,* https://csrc.nist.gov/glossary.

[2] GPS Program Office, *NAVSTAR GPS Space Segment/Navigation User Segment Interfaces IS-GPS-200.*

[3] National Institute of Standards and Technology, *Federal Radionavigation Plan,* Technical Report, Washington, DC: Department of Defense, Department of Transportation, and Department of Homeland Security, 2021, doi DOT-VNTSC-OST-R-15-01, https://rosap.ntl.bts.gov/view/dot/63024/dot_63024_DS1.pdf.

[4] DHS, *Positioning, Navigation, and Timing (PNT) Program,* https://www.dhs.gov/science-and-technology/pnt-program.

[5] National Institute of Standards and Technology, *NIST Risk Management Framework,* https://csrc.nist.gov/projects/risk-management/about-rmf.

[6] IEEE, *P1952 Standard for Resilient Positioning, Navigation and Timing (PNT) User Equipment,* https://sagroups.ieee.org/p1952/.

[7] Rosulek, M., *The Joy of Cryptography,* 2021, https://joyofcryptography.com/pdf/book.pdf%20https://joyofcryptography.com.

[8] Menezes, A. J., P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography,* Boca Raton, FL: CRC Press, 2001., http://www.cacr.math.uwaterloo.ca/hac/.

[9] Oppliger, R., *Cryptography 101: From Theory to Practice,* Norwood, MA: Artech House, 2021.

[10] Paar, C., and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners,* Berlin: Springer, 2014.

[11] Schneier, B., *Applied Cryptography, Second Edition,* Indianapolis, IN: John Wiley & Sons, 1995, pp. 191.

[12] Singh, S., *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography,* in collaboration with Internet Archive, Anchor, 2000, http://archive.org/details/codebook00simo.

2

Overview of Satellite Navigation

This book focuses on using cryptographic methods to secure satellite navigation. This chapter lays a foundation for space-based PNT, which, as noted previously, is also referred to as satnav, as it exists at the time of this writing. We introduce the various aspects of a satnav enterprise, with special emphasis on the features of the navigation signals. In Part III of the book, we will show how the various cryptographic algorithms and protocols defined in the next several chapters can be used to provide protection.

The purpose of the satnav enterprise ultimately is to enable navigation signals for the users. As such, this chapter begins with a brief discussion of how navigation signals enable PNT. A satnav system essentially is a ranging system, analogous to a bistatic radar system, with the added trait that the signals provide data to facilitate the PNT calculation from the range estimates, which makes the signals self-contained.

Maintaining navigation signals of sufficient quality is the goal of the satnav enterprise. This enterprise comprises three main areas. First is the *space segment*, which consists of the transmitting satellites. Second is the ground segment, which we further divide into two separate parts: the *control segment*, which provides control and scheduling of the satellites, and the *mission segment*, which is responsible for tracking the satellites. Finally, the *user segment* represents the PNT receivers. We follow the enterprise discussion with examples of various constellations and some details about orbits.

We then focus on this book's main area of interest: navigation signals. Two main features of the signals are highlighted. The first feature is the range measurements computed from the signal, which involves the carrier and modulation of the radio frequency signal. Processing the signal yields correlation measurements, which depend on the physical layer of the signal. The second feature

is the data messages that reside on the signal, which provides information about the satellite's health and location. We highlight security goals throughout the discussion using terminology from Chapter 1. Discussion on threats to these security goals is found in Chapter 7.

We develop concepts briefly and as needed. There are many more thorough references for this chapter, such as [1–6].

2.1 Navigation Signals from Space

Satnav is all about enabling navigation signals, which are used for ranging. Figure 2.1 shows the satnav enterprise and a very high-level view of signals being broadcast from transmitting satellites to receivers; the enterprise is defined in Section 2.2. A signal consists of two parts. The first part is the aspects of the signal that enable ranging measurements: the carrier, the modulation, and spreading codes. The second part is data modulated onto the signal. The data tells the receiver where the transmitter was at the time the signal was emitted; that is, how to interpret the information in the first part. The receiver will use these two parts to determine their position, time, and perhaps other facts, such as, velocity or ionospheric properties. There are important variations, such as pilot signals that do not have data, in which case they borrow the data from other parts of the signal. Augmentation signals may have data but no ranging information. But in general, we consider a navigation signal having these two parts.

Satnav systems are ranging systems; the satellites are radio beacons that are used to determine the receiver's PNT from range measurements, as seen in Figure 2.1. The architecture of the ranging signal is a primary: the receiver must be able to obtain the range measurements through its processing. These ranging measurements are not useful unless they can be used to derive PNT information. That derivation requires data—specifically, the location and time

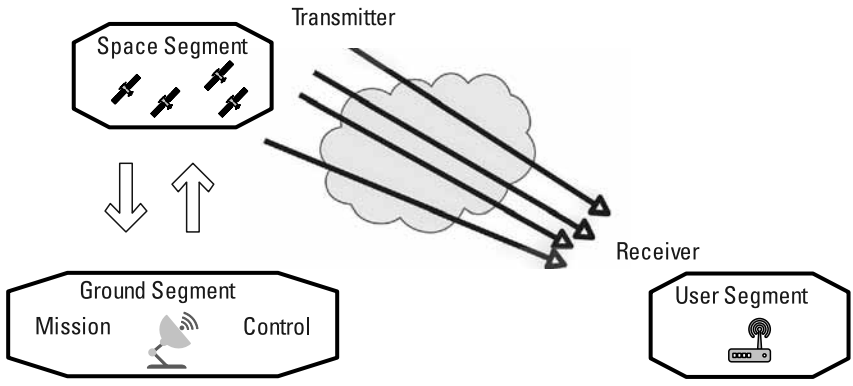

Figure 2.1 A navigation signals and the satnav enterprise.

of transmission. The threats against the two features of the signal are different, as are their protection methods.[1]

Ultimately, it does not matter where the relevant data is collected, only that it is correct. We emphasize the data aspects of the signal because satnav systems try to be self-contained even if there are other avenues for obtaining the data. Contrast that with the ranging measurement, which can only be taken between the transmitter and the receiver. For reasons that will be explained shortly, the ranging measurements will be referred to as *pseudorange* measurements (see Sections 2.1 and 2.2.2).

Satnav covers all space-based systems: global, augmentation, and regional.

- A *GNSS* refers to global coverage systems and solely regional or augmentation systems. Examples include GPS, Galileo, Global Navigation Satellite System (GLONASS), and Beidou.
- A s*pace-based augmentation system (SBAS)* refers to a system that is designed to augment a regional or global PNT system. An example is the *Wide Area Augmentation System (WAAS)* or the *European Geostationary Navigation Overlay Service (EGNOS)*.
- There are satnav systems that are stand-alone regional systems. We call these regional satnav services if we need to distinguish them from GNSS systems. Examples include Japan's QZSS, India's *Navigation with Indian Constellation (NavIC)* and Korea's *Korea Augmentation Satellite System (KASS)*. The Beidou system is a GNSS that also has a regional augmentation.

The fundamental principle, similar to radar, is that the flight time of a radio signal can be used to determine distance. This works because the speed of propagation is known ($c \approx 2.998 \times 10^8$ meters/second). If a signal is emitted from the known satellite position S at the start second time t^s and arrives at unknown position P at final time t^f then the distance is:

$$d(S,P) = c\left(t^f - t^s\right) \tag{2.1}$$

here the distance function is:

$$d(S,P) = \sqrt{\left(X^s - X^p\right)^2 + \left(Y^s - Y^p\right)^2 + \left(Z^s - Z^p\right)^2}$$

where $S = [X^s, Y^s, Z^s]^T$ and likewise $P = [X^p, Y^p, Z^p]^T$.

Since S is known, then (2.1) is an equation in three unknowns, P. An immediate problem is in the right-hand side of (2.1), namely the presumed

time synchronization between the transmitter and the receiver. One part of this problem is easily solved because the nature of the ranging portion of the signal embeds the time as part of the signal.

A second part of this problem that needs to be considered is that the clocks of the transmitters and the receiver are not synchronized. The unknown time difference between the transmitters and receiver is modeled.

We now have an equation in four unknowns—three for the position, P, and one for the time bias, b. For these unknowns to be unambiguously solved, we would need four such measurements, *if* all the transmitter times are synchronized. This synchronization is one of the jobs of the satnav mission segment. It is important to note that since the mission segment coordinates the clocks on the space vehicles, including information in the navigation message, and the user is using the same receiver clock in all measurements, then there is only *one clock bias* for all measurements in each piece of user equipment.[2] This observation yields four range equations:

$$\sqrt{\left(X_i^s - X^p\right)^2 + \left(Y_i^s - Y^p\right)^2 + \left(Z_t^s - i^p\right)^2} = c\left(t_i^f - t_i^s + b\right) \quad i = 1,2,3,4 \qquad (2.2)$$

These equations are nonlinear but well-behaved. If there are more than four measurements, then the solution is overdetermined; however, there are error terms from various sources, so additional measurements increase accuracy. The system of (2.2) can be solved in an iterative fashion, for examples, by the Newton-Raphson algorithm or steepest descent algorithm [1, 2, 6].

One simplification in the above discussion is that the speed of light is constant. Radio waves are refracted in the ionosphere and much less in the troposphere. Fortunately, the degree of refraction is frequency-dependent, and thus, measurements at two radio frequencies will allow the elimination of this error. The governing variable is the total electron content of the ionosphere. In the troposphere, the main thing is moisture content along the path. These types of errors affect the measurements and resulting equations.

It is also worth noting that the positions, S_i, in (2.2) have to be determined very accurately. For satellites, this accuracy means dealing with orbital mechanics, discussed in Section 2.4. Information about the orbits is broadcast as data; the need for the system's data will compete directly with the need to determine the pseudoranges. This is a fundamental tension in the design of satnav systems.

As mentioned, the system data does not have to come from the source *S*; it just needs to be accurate. For example, under normal circumstances, cell phones get data about the sources, S_i, from a cell tower; this method is faster than getting the data from the satnav signal [7]. Such methods of data collection are known as "out-of-band" (OOB). Looking ahead, securing OOB data is

much easier than securing in-band data, (i.e., through the signal S). See Chapter 8, particularly Section 8.3.3, for OOB security approaches.

The seemingly straightforward equations in this section demonstrate the purpose of satnav. This purpose leads to the general principle of security goals for satnav:

> *Fundamental Satnav Security Goal*
> The information in the range measurement has integrity; it comes from the true source at the proper time.

Attaining that purpose requires the interaction of several different segments of the satnav enterprise, which we present next.

2.2 The Segments of Satnav Systems

There are four segments of a satnav system: space, control, mission, and user, as shown in Figure 2.1. As previously mentioned, it is common to group the mission and the control together as the *ground segment*. Figure 2.1 indicates the signal in space, which is the raison d'être for a satnav system. We discuss the PNT signals in more depth in the following sections.

2.2.1 Space Segment

The space segment is the collection of spacecraft responsible for emitting the signals used for pseudoranges by the receiver. The payload portion of the space vehicle is the most pertinent for our purposes. The components of a payload are:

- The antenna system
- The radio frequency subsystem, which comprises
  - Waveform generation, including the carrier signal and signal combining methods, which allow for multiple signals to be broadcast;
  - Upconversion to the carrier frequency, which is usually at L-band;
  - Amplification, which often uses nonlinear amplifiers to improve the efficiency of power amplifiers.
  - Filtering to fix the effects of the nonlinear amplifiers; see the International Telecommunication Union (ITU) agreements [8].
- Timing subsystem, what is usually atomic clocks, including clock steering algorithms.
- Baseband processing, which consists of

- Spreading code generation;
- Data message generation An example payload is shown in Figure 2.2.

Security Goals for Space Segment
As with any system, a satellite should have the requisite cybersecurity to ensure it has integrity and that only authorized parties can make any modifications. Its output transmissions should match what is expected. That is, the codes, data, and timing should be correct.

2.2.2 User Segment

The user segment is the collection of individual radio receivers (i.e., the collective of the *user equipment (UE)*) that calculates the pseudoranges and provide the actual user with the position and timing information. This segment is also responsible for integration with mapping functions to provide the user with navigation information.

The user segment is an interesting contrast to the other segments, mainly because UEs often have severe constraints on size-weight-power and cost. For example, the UE is often a component of consumer items that run on rechargeable batteries. Unit cost in the tens of dollars is considered prohibitive in some applications, and the power consumption of the UE systems is a serious consideration [10]. Comparatively, the space segment and control segment are

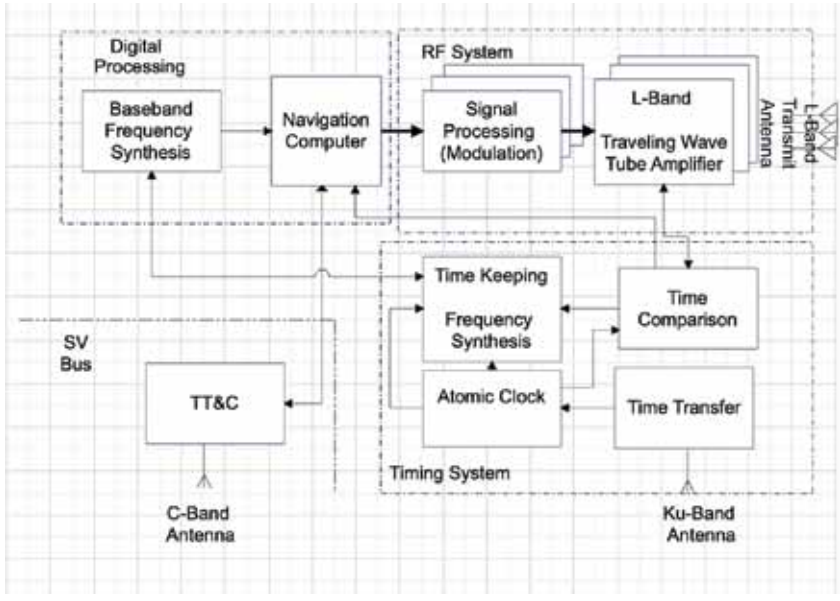

Figure 2.2 A high-level payload diagram modeled after QZSS [9].

resource-rich. However, as we can see, there are still some notable constraints on each segment. Space, after all, is an unforgiving environment. Constraints, both on the user and space vehicle, will affect the possible cryptographic solutions.

The functional layout is given in Figure 2.3. The amount of digitization varies; many current devices have a significant analog component due to cost and energy consumption considerations. Some items from the figure:

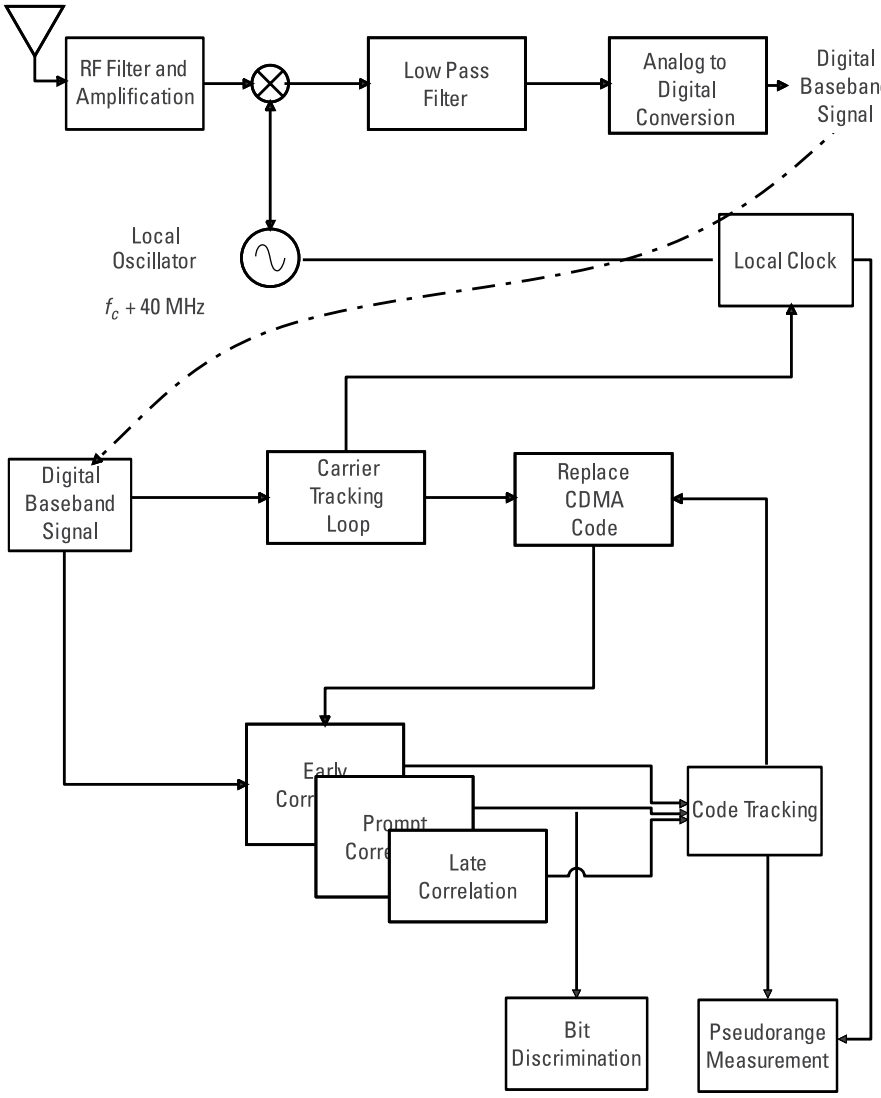

Figure 2.3 Block diagram of a satnav UE.

- The radio frequency signal is converted to either baseband or an intermediate frequency in a process variously called mixing, downconversion, or superheterodyning using a local oscillator near the carrier frequency.
- When downconverting, the difference is retained (via filtering).
- Carrier tracking follows to track the phase of the signal.
- Code tracking uses the replica of the spreading codes to allow for data demodulation.
- The resulting pseudorange measurements are passed on to generate the PNT solution.

Security Goals for User Segment
The UE should have the requisite cybersecurity to protect its functions, specifically the PNT computation and protection of any information stored that should be protected. Any data stored should be known to be correct.

2.2.3 Mission Segment

The mission segment computes the orbital information for use by receivers and disciplines and synchronizes the clocks on the space segment. A list of its functionality includes the following:

Sensor network: This network of sensors allows calibration of the various components of the system, especially the orbital and clock information for each space vehicle. Data from the sensor network is combined with extensive models, such as detailed gravity models, solar radiation pressure models, ephemeris data, and space vehicle health issues such as solar panel degradation.

Filtering: The actual calibration process that computes orbits and orbital predictions, clock corrections, and other important parameters such as time reference (e.g., the relationship to Universal Coordinated Time).

Database: The mission segment populates a database of values needed by receivers to compute the pseudorange accurately. This database includes many parameters that the system operators need to maintain the system's correct operations. Data includes orbital predictions, some of which end up as almanac, which is long-term but less accurate information about the space-segment orbits, and ephemeris, which is shorter-term information, more accurate orbital data. Also, satellite clock behaviors, including information that will be packaged into the navigation message, as well as in-

ternal performance data needed by the operators to help anticipate when they might want to move to a redundant atomic clock on a space vehicle.

The mission segment consists of sensors and processing of the sensed data. The sensor network needs to be geometrically dispersed over the coverage area to allow proper calibration of the system (see Figure 2.4). So for a local augmentation like the QZSS, it will be local to the service area, and for a global system like GPS or Galileo, it will be a global network of sensors.

In general, the mission segment contains a predictive model for the orbits, as the position of each space vehicle, coordinated with the time of transmission of the ranging signals are critical. The heart of this is a Kalman filter, a recursive least-squares estimator applied to the orbits. The Kalman filters contain

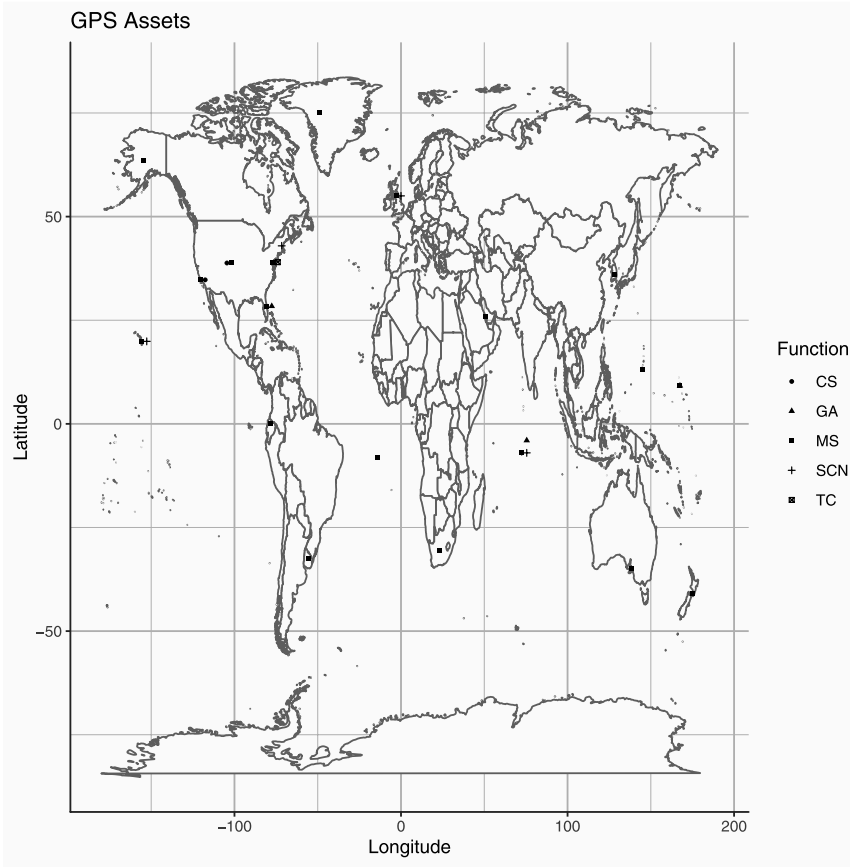

Figure 2.4 The location of various components of the GPS mission and control segments. CS: control segment, GA: ground antenna (dedicated telemetry, tracking, and control), MS: monitoring station, SCN: satellite control network (shared telemetry, tracking, and control), TC: time control.

detailed spacecraft models, including solar pressure models, and extensive gravity models. Kalman filters come up a lot in navigation problems, general references are [11–13], and an interesting paper estimating the GPS system can be found in [14].

Security Goals for Mission Segment
The security goals for the mission segment are similar to the space segment and consist of the cyber goals needed to protect the assets in the segment. In addition, access to information needed to monitor the satellites is required.

2.2.4 Control Segment

The control segment controls space vehicles and is responsible for flying the constellation. It uses the information from the mission control that measures where the vehicles are, but it is also responsible for controlling the day-to-day operations, specifically:

Control Network: The telemetry, tracking, and control (TT&C) functions. These functions are how data is uploaded to space vehicles for the navigation messages, as well as all the housekeeping functions that keep the space vehicle healthy.

Scheduling: A satnav system, especially a GNSS system with a score or more of space vehicles, has substantial scheduling needs, given that user performance is related to the frequency with which orbital elements are updated. One need often goes by the age of data (AoD) issue. All other things being equal, the more often the ephemeris is updated, the more accurate it will be (i.e., less extrapolation is better).

Engineering: Technical assessments of the space vehicle performance, including bus status such as power, thermal, and mission-related issues, which more or less depend on how the system is structured.

Database: This database is related to the mission database and possibly part of the same structure. For the control segment, the emphasis is on the administrative and engineering data needed to fly the space vehicles and the data to be uploaded as navigation data. It also contains information needed to perform scheduling operations, especially in systems that are not in constant contact with all of the space vehicles (e.g., GPS).

There are many nonpayload-related systems on space vehicles, such as propulsion, power, dynamics and controls, and thermal that require monitoring and care [15–17]. The control segment takes information from the space

vehicles and determines functions such as vehicle health and makes decisions about which actions are needed on the space vehicle.

Most critically, for our applications, it also takes information from the mission segment, and uploads the data for use in the navigation messages placed on the ranging signal. The frequency of uploads is one factor in the fidelity of navigation data, such as orbital predictions (the AoD).

Security Goals for Control Segment
The security goals for the control segment are similar to the space segment and consist of the cyber goals needed to protect the assets in the segment. In addition, the control channels to the satellite need to be protected for integrity and confidentiality.

2.3 Descriptions of Orbits

The fundamental characteristic of a satellite system are its orbits of the satellites or space vehicles (SVs); accurate information about orbits and thus satellite positions are crucial to determining PNT. Simple physics, as divined and then derived by Issac Newton, says that the orbit is determined by position, velocity, and time: seven parameters [18]. Classical orbital elements, the Keplerian description, also has six elements and time.[3]

Information on orbits can be found in [18], and particular information on each constellation is found in their interface specifications (ISs); for example, IS-GPS-200N [19] for GPS.

The position-velocity description is called the state or state vector description, and this is what is used for numerical orbit propagation. Another approach called classical or Keplerian is taken from the analysis of Tycho Brahe's data by Johannes Kepler, which showed that orbits are ellipses. Thus, the description is that of an ellipse. The simple description of an ellipse in the plane is

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$

and in the general setting, it is important to describe the plane that the ellipse is in with respect to the earth's equatorial plane. The last aspect is, at a given time, where the satellite is on the ellipse.

The Keplerian parameters are as follows (Figure 2.5):

- Shape
  1. Scale: Semimajor axis, a, of the ellipse.

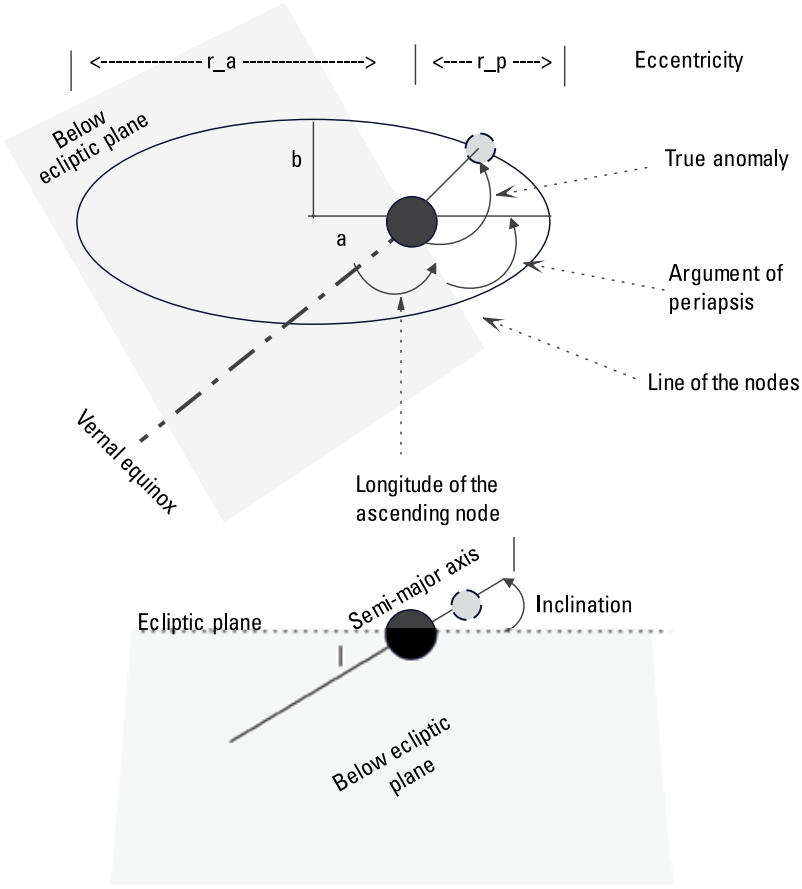

Figure 2.5 Classical orbital definitions.

2. Elongation: Eccentricity (e) is a measure of the elongation of the ellipse from a circle (which has $e = 0$), and all ellipses satisfy $0 \le e < 1$. The calculation of the eccentricity $e = \sqrt{1 - \frac{b^2}{a^2}}$ or alternatively $e = \frac{r_a - r_p}{r_a + r_p}$.

- The Orbital Plane
  1. Inclination (i): The tilt of the orbital plane with respect to the equatorial plane, measured at the ascending node.
  2. Longitude of the ascending node (Ω). The rotation of common line in the orbital plane, and the earth equatorial plane, called the *line of nodes*, measured from the the direction of the constellation vernal equinox (Aries).
- Orientation of the ellipse in the orbital plane, the argument of periapsis (i.e., how far the semi-major axis is rotated from the longitude of the ascending node).

- True anomaly ω (alternately, mean anomaly), where the satellite is on the ellipse, measured from the ascending node.

To solve for the position of the user, there need to be at least four satellites in view of the user, and the geometry of the viewing needs to be adequate to solve (2.2). As we have seen, the unknowns are the receiver's position and the offset to the GNSS system's clock. The name given to the geometric component of the solution is dilution of precision (DOP) measured by the trace of the covariance matrix in solving (2.2). The job of the constellation designer is to choose orbits that achieve good DOP for all users at all times.

Augmentation systems are similar, but they work in conjunction with one or more primary systems, and their orbits are designed to meet system-specific objectives. For instance, the U.S.-based WAAS provides navigation messages to receivers and the data is updated very quickly, with alerts within a few seconds. The WAAS system is not designed for ranging, but to update the parameters used in calculating a GPS solution to be very accurate.

In contrast, QZSS is designed to overcome issues with urban canyons in Japan and improve navigation and timing accuracy in Asia-Oceania. These satellites broadcast signals that are designed for ranging; they are, in fact, on the same frequencies and use the same code families as GPS. One of the orbits used, the quasi-zenith orbit (inclined geostationary orbit), spends about 16 hours of its 24-hour orbit at 20 degrees above the horizon in most of Japan. This is achieved by tilting the geosynchronous orbit inclination and making it eccentric[4] while retaining the geosynchronous aspect, which is not geostationary.

One constellation description that GPS initially used and is used by Galileo, GLONASS, and BeiDou, is the Walker constellation [20], see Figure 2.6.

2.4 Specific Constellations

We summarize a few of the specific details needed for satnav considered in this book in Table 2.1. The Galileo, GPS, GLONASS and a part of the BeiDou systems have many similarities. They are medium earth orbit (MEO) constellations, (e.g., orbits that are about 1/2 a day); they are significantly inclined from equatorial in the vicinity of 55° and they contain between 24 and 32 space vehicles.

2.5 Satnav Signals

We noted previously that the origin of satnav is in bistatic radar systems; as such, the structure of the signal is paramount. A satnav signal is composed of three layers, which can be differentiated by frequency:

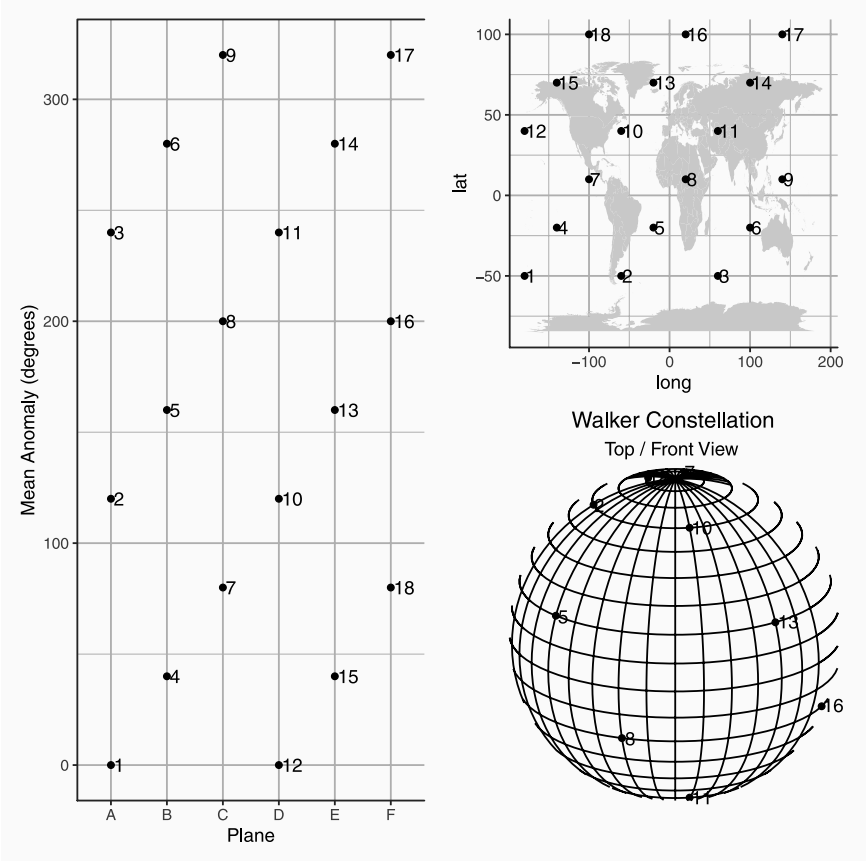

Figure 2.6 A typical Walker constellation denoted by i:/t/p/f, where i is the inclination, t = # SVs, p = # planes, and f = relative phasing between planes. Depicted is a 56:27/3/1(Galileo) (From: [21].)

Table 2.1
General Parameters

Name	Orbits	Altitude	Inclination	Typical ≠ SVs
GPS	Modified walker (24/6/2)	20,180	55°	30 (one spare per plane)
GLONASS	Walker (24/3/1)	19,100	54.8°	24
Galileo	Walker (34/3/1)	23,220	56°	30 (two spares per plane)
Beidou	Walker (24/3/1), IGSO 3 at (118E) GEO at 80E, 110.5E, 140E	21,528 35,786 35,786	55° 55° 0°	24 3 3

After: [22].

- The carrier, which is typically in the 1-2 GHz range;
- The code and modulation layer, which is typically 1 to 20 MHz;
- The data layer is typically in the low hundred Hz to occasionally a few KHz.

See the notional schematic in Figure 2.7. The cryptographic methods that we will show in this book apply to the code and data layers.

The carrier is the center frequency of the transmitted signal. Note that each source will arrive at the user equipment at a slightly different frequency due to the Doppler effect of the spacecraft's relative motion to the UE. The code layer contains the timing information and keeps the different sources separated in the receiver. The data signal contains the information needed to determine the position of the space vehicle and other data useful in interpreting the ranging solution (see Section 2.7).

The carrier and bandwidth of a satnav system (signal) are determined by treaty under the International Telegraphy Union [23]. The preference for L-band carriers (1–2 GHz) has to do with good transmission properties (i.e., low atmospheric loss), in this band that supports navigation from space. In general, it is preferable to have a wider bandwidth. However, a wider bandwidth means more energy consumption in the receiver and the space vehicle.

The core of the signal is the ranging code, and in most systems, that ranging code is given by a *code division multiple access (CDMA)* method. Each source is assigned a different spreading code; thus, it is possible to track each signal separately (see Section 2.6). The older GLONASS system is an exception; it used frequency division multiple access (FDMA).

CDMA codes were developed to allow efficient use of a channel, enabling several communication streams to operate in the same bandwidth with minimal interference [24–25]. The central idea is to develop sequences that are

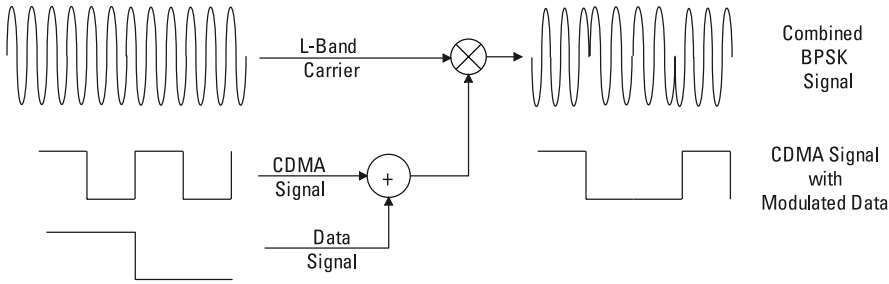

Figure 2.7 The composition of a PNT ranging signal. The carrier provides the pilot function and the CDMA carries the timing information and the data. GPS L1 C/A code has 1540 1.57524 GHz cycles per CDMA chip (1.023 MHz) and 20,460 chips per data bit (at 50 bits-per-second).

uncorrelated, either exactly or approximately; there are a myriad of other considerations, too. Since the function of a CDMA code is to separate the signals, any data is modulated onto the codes. For this reason, the bits that compose a CDMA code are called *chips* to distinguish them from the data bits that are being carried. Very often, but not always, in GNSS systems, the CDMA code can be a memory code; that is, the code is periodic and sufficiently short that simply storing in memory is efficient.

The code (CDMA) layer is composed of a modulation scheme and the code. The modulation scheme is used to shape the signal (e.g., the distribution of the energy around the carrier). For instance, it includes how the CDMA and data layers are impressed into the carrier. A typical scheme starts with binary phase shift keying (BPSK), in which the carrier is multiplied by ±1 at specific intervals. To keep the signal clean, the change in values is done at zero crossings of the carrier (e.g., at multiples of $2\pi\omega$), where ω is the carrier frequency for GPS L1 that would be 1.57542 GHz. This is how the CDMA signal is impressed onto the carrier. The values of BPSK, either plus one or minus one, encodes the CDMA signal and is further used to carry the data on the CDMA signal (see Figure 2.7). A more subtle point since the first set of signals on GPS, is that in the CDMA layer there can be an additional modulation. One of the easiest ways to think about the modulation is to let a chip be represented by something more complex than ±1. For instance in Manchester encoding (modulation) a +1 is represented as the sequence +1, –1 and –1 is the negative, –1, +1.

For instance, the L1C signal has two main components, L1CP (the pilot signal with 75% of the power) and L1CD (the data signal with 25% of the power). Both are modulated with a 10,230-bit Weil code (developed by one of the authors [25]). The signal is XORed with a binary offset carrier, *BOC(1,1)* signal, BOC(1,1) is a square-wave signal, but in the pilot channel, only 29 of 33 chips get the BOC(1,1) shape, and the remaining ones are modulated as BOC(6,1), where BOC(M,N) encodes a +1 chip as $k = \frac{2M}{N}$ symbols alternating +1, –1, +1... and encodes –1 chip as the –1, +1, –1.... Thus BOC(6,1) encodes each chip as 12 alternating symbols. This is done to push the power spectrum off band-center [26–28]. The combination of different BOCs is known as MBOC for multiplexed-BOC. In the context of this book, we usually will lump the modulation with the CDMA code.

The shape of the power spectrum for a BOC(1,1) signal moves the power away from L1 C/A so that while the signals have the same band center, and are very close in bandwidth (1.023 MHz) they occupy different spectrum [3, 24, 27, 28].

These signals have the following properties:

- Each signal is uncorrelated with the others. This is accomplished by the CDMA codes.
- Each signal is generated as a specific function of the time of transmission.

The important point is that the signal designs support distinguishing multiple signals at the same frequency in the UE. Since the detection technique is correlation, uncorrelated signals do not interfere with each other. If the CDMA code is periodic, then it is easy to acquire; if the code has a long period or is fully pseudorandom, then the search space for acquisition is much larger and initial time uncertainty is a serious consideration.

The technique that enables estimating the range is correlation. Inherent in the correlation process is the requirement that a local copy of the signal be generated; see the next section. This leads to the following security goals:

Security Goals for the Ranging Signal
The security goals for the ranging signal are:

- The received signal should verifiably come from the authentic source. This goal is the topic of Part III of this book.
- The timing of the received signal should be consistent with the origination from the authentic source. In particular, this goal requires that the UE search for any earlier version of the signal to avoid multipath or repeaters. Spoofers, including those arriving earlier than the true signal, will be eliminated by source authentication.
- Other features, like signal content, direction, and power should also be consistent with the authentic source. This goal includes reasonableness checks on data values; for instance, that the orbit is valid. This goal can depend on UE resources; for instance, the arrival angle depends on having enough resources (e.g., antenna elements, to discriminate direction).

In some cases, exclusivity of the signal may be desired. If that is the case, then the signal will have a private CDMA code, that is encrypted.

2.6 Correlation

GNSS systems broadcast relatively weak signals, that undergo significant loss. For example, there is path loss, which is more than 18 orders of magnitude for most systems (i.e., –180 dB of free space loss). The signal arrives at the UE below the thermal noise floor of the receiving circuit and is detected via correlation. This trait implies that space vehicle transmissions are replicated by each UE to give a reference against which to correlate. A fundamental fact about

current GNSS systems is that the fundamental measurement, the time of flight of the ranging signal, is made via an averaging process, correlation. The theory behind correlation can be found in [1, 4].

The fact that the UE must replicate the signal means that the signal is known to the receiver. The fact that the UE must generate the signal places significant constraints on the complexity of the signal. If the signal is widely known, then any attacker can then generate false signals.

2.6.1 An Instructive Example

We illustrate correlation with an example using GPS C/A code; see [1] or [29] for more precise details on how a GNSS receiver works. Especially note that the signal power numbers may vary a bit. The system interface specification (i.e., [19]) gives the *promised* performance.

- The data rate is 50 bits per second (bps) and the modulation is BPSK, in which {0, 1} is encoded as, $\pm 1 \times$ signal
- The chipping rate is 1.023 MHz, which means 20,460 chips per bit (20,460 = 1,023,000/50), and chips are also ± 1
- The carrier frequency is 1.57542 GHz; this link is called L1 (link 1), and there are 1,540 cycles of the carrier per chip
- Received power is 1,000X below the thermal noise floor (–30 dB)
  - Received power is –160 dBW, although the actual power currently might be a bit better
  - Thermal noise in a typical receiver circuit is -130 dBW. The value depends on the bandwidth of the receiving circuit, among other things

Let $l(t)$ be a local copy of the expected signal, and $r(t) = s(t) + n(t)$ be the received signal with noise. The expected power of a signal $m(t)$, over $[0, T]$ is

$$P_m - \langle m, m \rangle = \frac{1}{T} \int_0^T m(t) m(t) \; dt$$

We will take the power of $P_s = P_l = 1$ and $P_n = 1000 P_s$. The noise is 1,000 times the power of the signal; that is, the noise is 30 dB louder than the signal.

In a linear simulation then, the noise is multiplied by $\sqrt{1000} \approx 31.62$ with a change from power to energy units, if it is taken as initially having the same magnitude as the signal. The calculations are not particularly sensitive to the noise model, and we will use Gaussian noise for the calculations. A sample of the CDMA signal looks like Figure 2.8.

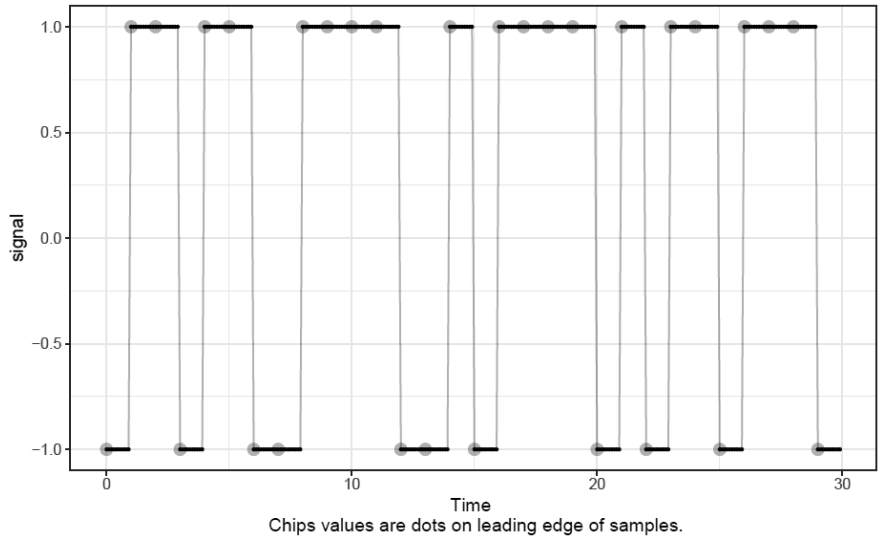

Figure 2.8 CDMA signal example. {1, 0} is encoded as {±1}.

The correlation process takes place at the CDMA chip level, and occurs over one data bit if data is present. The units of the calculations, therefore, have units of chips or samples of a chip. The expected number of counts from a correlator is the mean of the correlation process and can be calculated as the following, where a local replica l(t) of the CDMA code is correlated against the received signal $r(t)$:

$$
\begin{aligned}
E\left\{\int_0^{20,460} l(t)r(t)\,dt\right\} &= E\left\{\int_0^{20,460} l(t)\big(s(t)+n(t)\big)\,dt\right\} \\
&\approx E\left\{\sum_{i=1}^{20,460} l(t_i)\big(s(t_i)+n(t_i)\big)\right\} \\
&= \sum_{i=1}^{20,460} E\left\{l(t_i)s(t_i)\right\} + \sum_{i=0}^{20,460} E\left\{l(t_i)n(t_i)\right\} \\
&= \sum_{i=1}^{20,460} E\left\{l(t_i)s(t_i)\right\} + = \sum_{i=1}^{20,460} l(t_i)\underbrace{E\left\{n(t_i)\right\}}_{0} \\
&= 20,460E\left\{l(t_0)s(t_0)\right\} = 20,460
\end{aligned}
$$

So the expected number of counts is just the number of chips. This result is true only if the signal chips and the reference chips are roughly aligned. Since the chips are square (i.e., ± 1) the dilution of the count is the percentage overlap; if they are 1/2 a chip out, then the energy will be 50%, that is, the count

will have a mean of 10, 230 (see Figure 2.9 for a simulation of various chip offsets).

The variance is also of interest. It can be shown that

$$Var\left\{\int_0^{20,460} l(t)r(t)\,dt\right\} == E\left\{\left|\int_0^{20,460} n(t)\,dt\right|^2\right\} \approx \sum_{i=1}^{20,460} E\left\{n_i^2\right\} = 20,460\,\mathrm{Var}\left(n_i\right) \quad (2.3)$$

The noise terms dominate the calculation, and they are independent, so the variance is 20,460 times the variance of the noise term, and 20,460,000 for a standard deviation of 4,543 chips. If the signal were not present, then 95% of the time we would expect chip counts to be in the range of ±9,046, which is plus or minus two standard deviations.

Now, the correlation process is not perfect and the local signal chips and the received chips will not align perfectly. Thus, the correlator may track some signals even if not properly aligned. From the early-prompt-late figure (Figure 2.10), we see that being off 1/2 a chip yields about 1/2 the signal strength and we would likely see significant bit errors in the data. For C/A code, a half chip is about 150m, $0.5(c/1.023 \times 10^6) = 146.5$, so the receiver is not very accurate at that level either.

Taking 1/10 a chip error, then the expected count would be 18,414 ± 9,046 ≈ [9,367, 27,460], which is well away from the two standard deviations of noise-only counts: [–9,046, 9,046]. Thus one is very safe at 1/10 of a chip

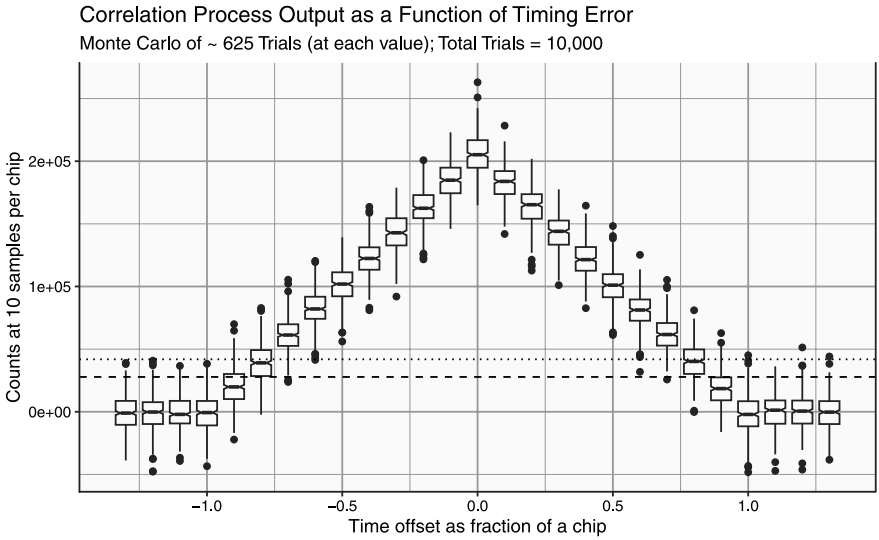

Figure 2.9 Chip offset statistics as a function of relative delay between the reference copy and the arriving signal.

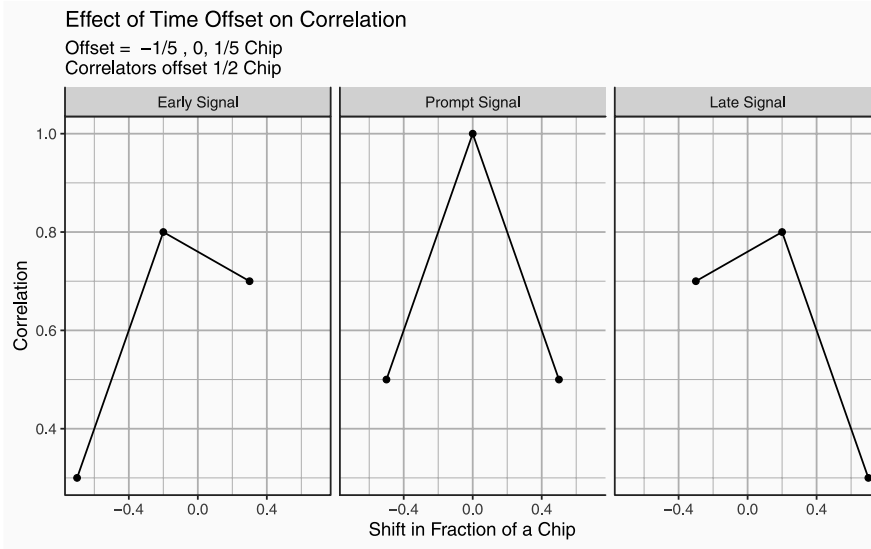

Figure 2.10 Early prompt late.

error or less from a detection point of view. This works out to be roughly 30m of error. The actual error rate would be calculated as a receiver operating curve [1, 3].

2.6.2 Misalignment Compensation

If the signals are not aligned perfectly, the integrals will be off. We would generate a signal, shift it, and then look at the result. In this case, take the local copy and look at what happens if the signal comes early.

The receiver takes advantage of this observation and executes three correlators, one nominally 1/4 of a chip early, one prompt, and one 1/4 of a chip late or some convenient fraction of a chip; see Figure 2.10. Using the relationships between early, prompt, and late, they can estimate at the subsample level accuracy for the actual arrival time and achieve the system accuracy we experience, while the intersample distance is relatively large:

- A nanosecond in timing is equivalent to ≈ 30 cm (or about a foot) in range;
- A C/A code data bit is 20 ms, so it represents roughly 6,000 km (3,726 miles);
- A C/A code chip at 1.023 MHz is about 300m (961 feet);
- A 1/10 of a C/A chip is about 30m (about 96 feet).

The balancing of the early-prompt-late correlators is the essence of the code tracking loop shown in Figure 2.3. The carrier tracking loop in that figure is usually accomplished with a nonlinear Costas loop, which is beyond our scope and is explained in [1, 4, 5].

2.6.3 The Importance of Correlation

We stress that the above calculations are how a receiver determines its timing offset, which leads to pseudorange. Pseudoranges in turn lead to the PNT solution. Thus, securing PNT ultimately comes down to finding ways to confirm correlation, which is the subject of Chapter 9.

2.7 Satnav Signal Data

The data transmitted on a satnav system is at least the minimum necessary data to process the signals. For a system that includes ranging, this data will include the ephemeris information, which is the detailed orbital information needed to calculate the broadcasting space vehicle's position and velocity. It would also include any timing information about the broadcasting vehicle's clock and usually conversions to other timing systems. For instance, GPS has GPS time to Coordinated Universal Time (UTC), the current number of leap seconds that keeps UTC coordinated with observed solar time, UT1 (see [19]). GPS time, like other satnav systems, is an atomic-clock-based time and is not steered to leap seconds. BeiDou, for example, uses a separate clock, and it is generally offset from GPS by 14 seconds. Receivers using both signals would need to estimate the relative time difference, so that adds another variable to be solved for, partially offsetting the advantage of tracking an additional system if only one SV is tracked.

The data on a navigation signal has security goals consistent with cybersecurity goals for information. Ways to protect navigation data is the subject of Chapter 8.

Security Goals for the Signal Data

The security goals for the navigation data are:

- The data should come from an authentic source, and the UE should be able to verify that.
- The data should have integrity verifiable by the UE. That is the data should be correct, and bound to the correct time of issue. In some cases, exclusivity of the data may be desired.

2.8 Takeaways

Our necessarily brief tour of satnav is meant to indicate the enterprise needed to achieve satnav and the specific items that may be vulnerable to threats. Much of the infrastructure, such as the satellites, the ground segment, and the UEs, have their specific security goals, and achieving those are fortunately not much different than what is needed by many other technological enterprises, as we will see in Chapters 6 and 7.

Unique to satnav are the security goals in navigation signals. These goals are needed to ensure that the PNT solution computed by the UE is trusted. The main items to be protected in the navigation signal mirror the two main uses of the signal:

- The timing via correlation needed from the signal to obtain pseudorange.
- The data modulated on the signal to obtain the information to compute PNT.

Part III of the book is devoted to the various ways achieving these protections have been conceived.

End Notes

1. The data needed to interpret the ranging measurements is the next consideration. We have heard it said that GPS, at least, is not a very good communication system, and we suppose that is true. GPS, and all of the other GNSS systems, are superbly designed ranging systems that are self-contained and not reliant on a communication system. The two functions, ranging and communications, are not the same and are competing.

2. This discussion is focused on a single satnav. If a receiver were to use multiple satnav systems, then one presumably would have a bias per system, since they are unlikely to be synchronized.

3. Kepler's observations were summarized in three laws when one body is massive and the other is much smaller (the satellite); for Kepler, these were the sun and the planets.

   (a) The orbit is an ellipse with the massive body at one focus;
   (b) The satellite sweeps an equal area in equal time at all points in the orbit;
   (c) The period is proportional to the semimajor axis of the orbit.

4. A fact worth knowing about orbits is that since energy is conserved in a noncircular orbit when the space vehicle is close to the earth, the potential energy is low, so the kinetic energy is high; that is, the space vehicle is moving faster than when it is further from the earth. This result follows from Kepler's second law: an orbit sweeps out an equal area in equal time. The result for the quasi-zenith orbit is that it spends more than half the time above the equator and less than half below. Another consequence of the conservation of

energy is any kinetic energy imparted at one point in the orbit leads to a change in the potential energy in other parts of the orbit. In particular, increasing the energy at the lowest point will increase the highest point, and increasing the energy at the highest point will raise the lowest point in the orbit [1, 18].

References

[1] Kaplan, E., and C. J. Hegarty, *Understanding GPS/GNSS: Principles and Applications,* Third Edition, Norwood, MA: Artech House, 2017.

[2] Pany, T., Navigation Signal Processing for GNSS Software Receivers, Norwood, MA: Artech House, 2010.

[3] Betz, J. W., *Engineering Satellite-Based Navigation & Timing: GNSS, Signals, & Receivers,* Piscataway, NJ: Wiley-IEEE, 2015, https://www.navtechgps.com.engineering_satellite_based_navigation_and_timing_gnss_signals_and_receivers/.

[4] Misra, P., and P. Enge, *Global Positioning System: Signals, Measurements, and Performance,* Second Edition, Lincoln, MA: Ganga-Jamuna Press, 2010.

[5] Lu, Y., *BDS/GPS Dual-Mode Software Receiver: Principles and Implementation Technology*, Volume 10, Navigation: Science and Technology, Singapore: Springer, 2021, https://link.springer.com/10.1007/978-981-16-1075-2.

[6] Spilker, J. J. Jr., and B. W. Parkinson, *Global Positioning System: Theory and Applications,* Volume I, Washington, DC: American Institute of Aeronautics and Astronautics, 1996.

[7] Van Diggelen, F., *A-GPS: Assisted GPS, GNSS, and SBAS*, Norwood, MA: Artech House, 2009, https://us.artechhouse.com/A-GPS-Assisted-GPS-GNSS-and-SBASP1729.aspx.

[8] Matas, A., "Radionavigation-Satellite Service (RNSS) and the ITU Radio Regulations," in *Proceedings of the ION 2013 Pacific PNT Meeting,* April 25, 2013, pp. 419–427, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=10998.

[9] eoPortal ESA, *QZSS (Quasi Zenith Satellite System) -eoPortal,* https://www.eoportal.org/satellite-missions/qzss#qzss-signals.

[10] QualComm, *When Mobile Apps Use Too Much Power: A Developer Guide for Android App Performance,* Qualcomm Developer Network, 2013, https://developer.qualcomm.com/download/trepn/trepn-whitepaper-power.pdf.

[11] Brown, R. G., and P. Y. C., Hwang, *Introduction to Random Signals and Applied Kalman Filtering with MATLAB Exercises,* Fourth Edition, Hoboken, NJ: Wiley, 2012.

[12] Gibbs, B. P., *Advanced Kalman Filtering, Least-Squares and Modeling: A Practical Handbook,* Hoboken, NJ: Wiley, 2011.

[13] Chui, C. K., and G. Chen, *Kalman Filtering: With Real-Time Applications,* Fifth Edition, 2017.

[14] Laurichesse, D., et al., "Real Time Precise GPS Constellation and Clocks Estimation by Means of a Kalman Filter," in *Proceedings of the 26th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS),* 2013, pp. 1155–1163.

[15] Fortescue, P., G. Swinerd, and J. Stark, *Spacecraft Systems Engineering,* Fourth Edition, Chichester, UK: Wiley, 2011.

[16] Uhlig, T., and F. Sellmaier, *Spacecraft Operations,* Cham, Switzerland: Springer, 2015.

[17] Wertz, J., D. Everett, and J. Puschell, *Space Mission Engineering: The New SMAD*, Vol. 28, Space Technology Library, 2011.

[18] Bate R. R., et al., *Fundamentals of Astrodynamics,* Second Edition, Mineola, NY: Dover Publications, 2020.

[19] GPS Program Office, NAVSTAR GPS Space Segment/Navigation User Segment Interfaces IS-GPS-200.

[20] Walker, J., *Continuous Whole-Earth Coverage by Circular-Orbit Satellite Patterns,* Technical Report 77044, United Kingdom: Royal Aircraft Establishment, March 24, 1977, p. 80, https://apps.dtic.mil/sti/citations/ADA044593.

[21] Piriz, R., Martin-Peiro, B., and M. Romay-Merino, "The Galileo Constellation Design: A Systematic Approach," in Proceedings of the 18th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2005), September 16, 2005, pp. 1296–1306, http://www.ion.org/publications/abstract. cfm?jp=p&articleID=6327.

[22] Davis, J. J., and D. Mortari, "Reducing Walker, Flower, and Streets-of-Coverage Constellations to a Single Constellation Design Framework," in *Advances in the Astronautical Sciences,* Vol. 143, 2012, pp. 697–712.

[23] Baumann, I., "RNSS and the ITU Radio Regulations," *Inside GNSS—Global Navigation Satellite Systems Engineering, Policy, and Design,* January 1, 2018, https://insidegnss.com/rnss-and-the-itu-radio-regulations/.

[24] Proakis, J., and M. Salehi, *Digital Communications,* Fifth Edition, Boston: McGraw-Hill, 2008.

[25] Rushanan, J. J., "The Spreading and Overlay Codes for the L1C Signal," *Navigation,* Vol. 54, 2007, pp. 43–51.

[26] Betz, J. W., "The Offset Carrier Modulation for GPS Modernization," in *Proceedings of the 1999 National Technical Meeting of The Institute of Navigation,* January 27, 1999, pp. 639–648, http://www.ion.org/publications/abstract.cfm?jp=p& articleID=716.

[27] Betz, J. W., et al., *L1C Signal Design Options:* Fort Belvoir, VA: Defense Technical Information Center, January 1, 2006, doi:10.21236/ADA456355, http://www.dtic.mil/docs/citations/ADA456355.

[28] Stansell, T., K. Hudnut, and R. Keegan, "Future Wave: L1C Signal Performance and Receiver Design," *GPS World*, April 1, 2011, pp. 30–36, https://www.gpsworld.com/gnss-systemgps-modernizationfuture-wave11401.

[29] Borre, K., et al., *GNSS Software Receivers,* Cambridge, UK: Cambridge University Press, 2023.

Part II
Cryptography

3

Symmetric Cryptography

Symmetric cryptography pertains to the use of *ciphers*, which is what most people think of when they hear the term cryptography. A cipher utilizes a shared secret, such as a pin, password, or key to conceal a piece of data so that only the intended parties can reverse the process to reveal the data. Hence the use of the term "symmetric." Without the shared secret, no one should be able to discern *any* information about the hidden data. A cipher can be used for more than just hiding data. For example, another use for a cipher can be to generate pseudo-random sequences, which are sequences that both act statistically random and are unpredictable.

The use of ciphers dates back at least 2,000 years, and many methods have been developed through the centuries to ensure that information is well-protected. A rich science has been developed for ciphers, and it is reasonable to say that creating a strong cipher has been essentially solved in many ways, even with the advent of quantum computers (see the Appendix). However, since ciphers can be resource-intensive, selecting a cipher requires a balance between performance and security. We discuss two types of ciphers, block ciphers and stream ciphers, and give examples of algorithms including important aspects of their implementations and some performance results.

While the study of algorithms is interesting, especially to us mathematicians, the algorithm is only a component in using ciphers. We need cipher modes, which define how to apply an algorithm to a larger set of data. We finish the chapter with two additional subjects. The first topic is a discussion of steganography, which also represents alternative methods to hide information. The second topic foreshadows how ciphers can be used in navigation.

3.1 Classic Ciphers

We begin with a brief discussion of the history of ciphers up to the digital age. While these ciphers would not be used in modern applications, they indicate what drove the evolution of modern ciphers and illustrate general principles.

3.1.1 The Ancient World and Child's Play

Several examples from BCE show the use of ciphers (i.e., a method used to obfuscate information).[1] We want to start by focusing on the Caesar cipher. Suetonius, in his biographies of the Roman emperors, in the chapter on Julius Caesar says [1]:

> If he [Julius Caesar] had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others.

To put this into modern usage, suppose we number the alphabet as A = 0 through Z = 25. Then the Caesar cipher changes an input letter p to an output letter c by

$$c = p + 3 \pmod{26} \tag{3.1}$$

In general, a Caesar cipher uses any fixed offset, such as $c = p + k \pmod{26}$. Interestingly, Suetonius also describes Augustus Caesar as using the same cipher with $k = 1$ [1].

The Caesar cipher is precisely the cipher of decoder rings, an example being the St.-Cyr disk. It consists of two rings with the alphabet in a circle, where moving one ring, say putting the D under the A, gives the complete mapping of the original Caesar cipher. As ciphers go, the Caesar cipher is trivial. The allowable keys (i.e., the offsets) are only 25 for a nontrivial mapping. It would be easy enough to try all possibilities by hand.

We can complicate this method in two ways. First, allow the mapping to be an arbitrary permutation of the alphabet; these are called monoalphabetic ciphers. That would yield a search space of 26! mappings. However, if English text is enciphered in such a manner, it is still relatively easy to find the mapping and decipher it. So easy, in fact, that such tasks are the purview of cryptogram puzzles.[2]

The other direction is to use multiple mappings, which are polyalphabetic ciphers. Such ciphers use a different mapping for each consecutive symbol. One famous example is the Vigenère cipher. Choose a keyword, say "CRYPTO."

Then the first symbol uses a Caesar cipher corresponding to the key C = 2; the second uses the key R = 17, and so on. After six symbols, the keyword is repeated. While this cipher is much stronger than a monoalphabetic one, techniques have been developed to break such ciphers efficiently when used on the actual text.

The development of ciphers led to the parallel development of breaking ciphers (i.e., cryptanalysis). There is the famous treatise by Al-Kindi (801–873 AD) and many books in the early twentieth century [2] on the topic of cryptanalysis.

3.1.2 To the Digital Age

As society advanced, so did the need for better ways to ensure secrecy. For example, the Vigenère cipher can be modified using *autokey*, where after the first keyword, the initial part of the original message serves as the continual key for the rest of the message. In the 1800s came the development of the Playfair cipher, where pairs of letters are changed to different pairs of letters. Playfair was widely used in World War I.

The above examples are *substitution* ciphers: the output symbol is a substitution of the input using a mapping defined by the key. Perhaps one of the most famous examples of a substitution cipher is the Enigma code used by the German military in World War II. A picture of an Enigma machine is shown in Figure 3.1. An operator encrypts a message by typing it into the machine after

Figure 3.1 An Enigma machine (author's photograph).

they have set the initial setting, which act as a key. After each letter is input, the machine changes its internal circuitry to alter the next mapping. The result is a very complicated polyalphabetic cipher.

The other common method for ciphers besides substitution is to use *transposition*; that is, permute the original letters. A simple example would be to write a message by rows and read off by columns. Transposition methods have the advantage that they do not alter statistics, and so some cryptanalytic methods may be hindered.

While the above examples are not digital in the sense of pertaining to information represented and acted on in more abstract binary settings, there is no reason why the methods of substitution and transposition should not apply in a digital setting. Indeed, as we see later the use of substitution and permutation functions is the basis for most modern ciphers, and the fundamental insight on this is due to Claude Shannon [3].

3.2 An Overview of Ciphers

The strategy in this chapter is to highlight the commonalities of ciphers and then drill down to specifics. Ciphers take in *plaintext* as input and produce the output *ciphertext*. The terms imply that the plaintext is obfuscated by the cipher in the ciphertext (i.e., that we do not want the plaintext revealed). But there are cases where everyone knows the plaintext, and what is desired is to produce a random-like ciphertext that may or may not be hidden by other means. For example, ciphers can be used to generate portions of navigation signal spreading codes, such in the Chimera protocols (see Chapter 10); such generation uses known plaintext. Thus, the terminology may not perfectly reflect actual usage; it is best to think of plaintext as input and ciphertext as output.

For notation, let P denote the plaintext and C denote the ciphertext. Then we denote *encryption* by the implied cipher as

$$E_K[P] = C \tag{3.2}$$

The value K represents the shared information known only to the participants; K denotes the key. The value K may be omitted if it is clear from the context. There are several synonyms for encryption, such as encoding or enciphering, and other terms for ciphers, such as code or codebook.

Conversely, we denote *decryption* by

$$D_K[C] = P \tag{3.3}$$

The encryption and decryption processes are related, of course, but the respective algorithms will usually have subtle differences. The notation for encryption and decryption is intended to be general to encompass various ciphers. In particular, there are two main groups of ciphers: block ciphers and stream ciphers.

3.2.1 Block Ciphers

Encryption takes a block of plaintext (say 128 bits) and outputs ciphertext of the same length (see Figure 3.2). The lengths must be the same to ensure that encryption can be reversed; that is, that the mapping is one-to-one (although not all cryptographic algorithms are necessarily one-to-one, as we will see with hashing in Chapter 4).

The key is given as a bit array. The length of the key is central to defining the strength of the algorithm and is not directly tied to the block length. Many algorithms may allow for different key sizes but keep the block length fixed. Other algorithms may allow for varying both, albeit with some restriction on allowable values. In practice, data is not just a single block of bits. How to extend the algorithm in the more general case is the subject of cipher modes in Section 3.7.

3.2.2 Stream Ciphers

The difference between stream ciphers and block ciphers is that stream ciphers act on a continuous stream of data such as, bits (see Figure 3.3). The left of the figure shows an example stream cipher process, where the input stream is changed to an output stream. The process uses a key, but unlike a block cipher, the cipher must maintain some state information to be updated when the next bit in the stream is generated.

The right side of Figure 3.3 shows how the encryption and decryption processes function. Encryption with a stream cipher exclusive-ors (XORs) the

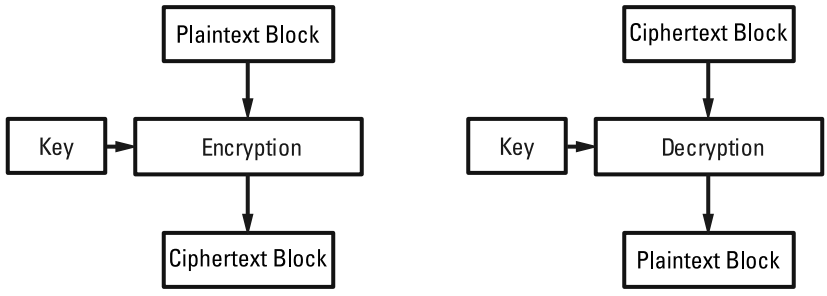

Figure 3.2 Block cipher.

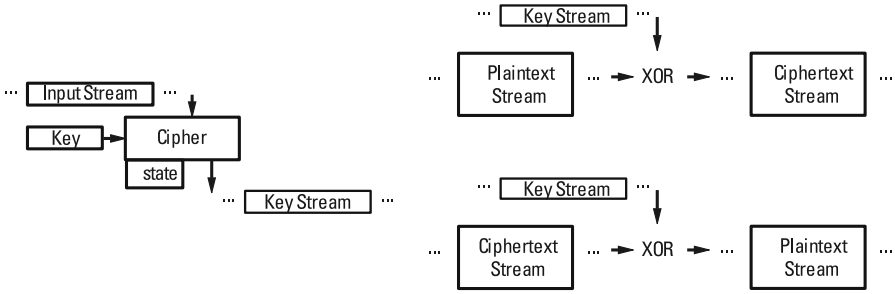

Figure 3.3 Stream cipher.

output stream from the stream cipher with the stream of plaintext data to produce the ciphertext stream. Decryption reverses this process by XORing the ciphertext stream with the output stream from the steam cipher to obtain the plaintext. In particular, the same cipher stream is used in both cases (i.e., encryption is the same as decryption), since the XOR function is self-inverse. There is no process to recover the input to the stream cipher from the output stream. This trait means the stream cipher can use noninvertible one-way functions.

3.2.3 Much of the Same

Stream ciphers are not as common as block ciphers, although their use is increasing. Note the distinction can be blurred. A stream cipher can run a certain number of steps to produce a block of output. Conversely, a block cipher could have a higher-level function that outputs a single symbol at a time. There are also modes in which a block cipher can produce a key stream similar to the output of a stream cipher (e.g., counter mode in Section 3.7.3). For our purposes, the distinction is what the algorithms entail from implementation, security, and performance. We define those goals next.

3.3 Desired Properties

The main goal for ciphers is to hide information. This means hiding the original information and the shared secret information used to protect it (key, password, etc.). This goal begs several questions as to what makes a good cipher:

- How does one measure how hidden the information is?
- What do we assume an adversary can do to reveal the information?
- Can we quantify a bound for how good a cipher is?

While the confidentiality of information is the primary goal, there are other desired properties that a good cipher should have. Ciphers are good for generating pseudorandom information, which we discuss further in this section. Ciphers can also be used for authentication and integrity, which we defer discussing to Chapter 4. These goals also require defining quantitative measures of attainment. However, be forewarned, such quantification will come with assumptions that one must be assured apply in practice.

3.3.1 Information Theory, or Shannon, Part 1

Measuring the degree for which information is hidden requires first understanding what we mean by information. That question gets us into the field of *information theory*.[3]

We assume some general knowledge of probability (also see the discussion in Chapter 1).[4] Most of the probabilities we use in this chapter are discrete probabilities, often over a space of equally likely events; for example, a fair coin toss, which has a 50% chance of coming up heads (and likewise tails), or a set of n-bit binary words that each have a probability of $1/2^n$ of occurring.

Shannon in his 1948 paper [4] introduced the concept of *entropy* to formalize what we mean by the uncertainty of information. For simplicity, we first define entropy for a single binary value (a bit). If one knew beforehand that the value was 0 or 1, then revealing the value yields no information, and so we expect the entropy to be 0. At the other extreme, if we only know it is equally likely that the value can be 0 or 1, like with a toss of a fair coin, then learning the value intuitively yields one bit of information.

To formalize, suppose that there is a chance p that the value is 1 and thus $1 - p$ that the value is 0. The entropy is defined as

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p) \tag{3.4}$$

The base of the logarithm is 2, which yields units as bits. For the coin toss, we have $H(0.5) = 1$ and $H(0) = H(1) = 0$, which can be found using limit properties. This definition can be extended to n values with probabilities p_i by

$$-\sum_{i=1}^{n} p_i \log_2 p_i \tag{3.5}$$

If there are 2^n equally likely outcomes, say from an n-bit array, then this yields n bits of information.

While the concept of entropy can be made much more precise[3] we will treat the calculations more with intuition. Consider these examples:

- Suppose you are given the parity check of a random n-bit array. Then there are now n–1 bits of entropy, rather than n, in the array. The reasoning is that not all 2^n arrays are equally likely; learning the single bit has cut the possibilities in half, or reduced the entropy by a bit. This concept extends naturally to knowing multiple parity checks. However, this reduction assumes that it is easy to exploit the new information. As we will see in Chapter 4, knowing bits that are derived from an n-bit array does not necessarily mean that one has practically reduced the entropy.
- Given an array of bytes, the entropy may be reduced if the nature of the bytes are known. If the bytes were to represent English text, then in fact one gets about 4.7 bits per symbol [5]. That value is less than the a priori expectation of 8 bits of entropy per byte.
- As another example, consider a common practice of using, say, four 5-letter English words as a password. There are about 2,000 such words, which is about 2^{11}; that yields about 4 times 11 or 44 bits of entropy. This value is much less than the four times 5 bytes (160) bits of entropy.
- Cryptographic algorithms like the Advanced Encryption Standard (AES) use a key of a certain length, say 128 bits. Another common practice would be to enter a passphrase to generate the key. If the pass phrase has fewer than 128 bits of entropy, the resulting derived key still only has that many bits of entropy, even though we are using a 128-bit key. Note that even padding out to the full 128-bit length would not increase the entropy if this additional padding was deterministic.

The overall rule of thumb is we desire the cryptographic algorithms (e.g., a cipher) to produce results that have maximal entropy. That is, the results act random, which means that each bit is acting like a fair coin with entropy equal to one bit of information.

3.3.2 Confidentiality

The primary goal of a cipher in most protocols is to provide confidentiality. As always, the word key means the secret information shared between the participants.

Goal: Confidentiality
Given the ciphertext, then no substantial information about the key or plaintext is revealed. Even if the plaintext is known, there still should be no information revealed about the key.

How does this goal relate to entropy? Knowing a length-n ciphertext should give no information about the plaintext or the key. That is, to someone who only possesses the ciphertext, their entropy should be their uncertainty in the plaintext and key. Even further, if someone knew both the plaintext and ciphertext, the uncertainty in the key should stay the same.

This informal definition has two importance nuances. First, knowing a plaintext/ciphertext pair clearly must eliminate many possible keys in theory. However, practically, the goal requires the features of the cipher algorithm to prevent someone from doing such an elimination in any feasible amount of time.

Which brings us to the second nuance: the uncertainties need to make assumptions about the capabilities of an adversary. When the definition says revealed, it means revealed when given some set of perceived attacks. The goal of modern ciphers is to mitigate as large a set as possible of these attacks, and when the cipher itself cannot so mitigate, then other methods can be applied, or the resources required for such an attack are infeasible to the adversary.

3.3.2.1 Attacks on Ciphers

Cryptanalysis is the study of attacks on ciphers that involve only aspects of the algorithm and not its implementation or infrastructure in use. There are many examples that have been created,[5] which are beyond our scope. Also not mentioned here but discussed briefly in Chapter 11 are attacks on the implementation of ciphers.

We mention a few overall themes for cryptanalysis; in all cases, the ciphertext is assumed known by the attacker:

- Known plaintext attacks allow the attacker to know both the plaintext and ciphertext, the goal then being to find the key.
- Chosen plaintext attacks allow the attacker to pick the plaintext and then receive the ciphertext; the cipher is treated as a black box.
- Chosen ciphertext is similar, except now the attacker picks the ciphertext and the cipher decrypts to produce the plaintext for the attacker.

We stress that all good modern ciphers are immune to these kinds of attacks. Indeed, even if the attacker possess many, many plaintext-ciphertext pairs

of their choosing, they still cannot produce the key better then brute force, which is searching through the key space.

Takeaway
Modern ciphers are by and large resistant to all these cryptanalytic attacks.

3.3.2.2 Brute Force and Kerckhoff's Principle

Brute force means searching through the whole space of possible keys. The assumption is that the rest of the details of the algorithm are known and only the key is unknown. If say the key were 128 bits, then brute force implies searching a space of size 2^{128}. This number implicitly assumes that all keys are equally likely to appear, which is usually true for uses of modern ciphers but often is not even close to being true for other shared secrets (i.e., the space of possible passwords contains items that are far more likely to appear in practice). There are methods to define the entropy in such cases.[6]

Brute force search is hampered by the exponential growth in the key space. For example, if a key length were to increase by 10 bits, the key space increases by a factor of 1,024. Table 3.1 show the amount of time to search a complete key space for various key lengths, assuming it takes only a nanosecond to try a single key. It is evident that key sizes of 112 or more are robust to brute force. Brute force is highly parallelizable: if one had 1,000 machines working at the key per nanosecond rate, all the times in Table 3.1 would be reduced by a 1,000.

A corollary to the desire to reduce possible attacks is that all the strength in a cipher should ultimately rely only on the key size. This goal is captured as Kerckhoff's principle:

The strength of the cipher should reside in the length and secrecy of the key.

Kerckhoff's principle implies that there is no loss if an attacker knows the details of the algorithm, an idea often captured as "no security through

Table 3.1
Time to Brute Force Key

Key Size	Time (Years)
64	585
112	1.6E17
128	1.1E22
192	2.0E22
256	3.7E60

1 nsec per attempt.

obscurity." However, while that sentiment is true with regard to the strength of the cipher, there may be other reasons not to reveal details about the cipher.

Finally, this discussion and Table 3.1 begs the question: How large should the key be? NIST has recommendations for key sizes given in [6, 7], but they come down to evaluating the brute force strengths just given. For enterprises that have a long life, such as satellite navigation, these recommendations need to consider the possible impact of quantum computing. We revisit this topic in the appendix.

3.3.2.3 Perfect Security and One-Time Pad

Brute force, and cryptanalytic attacks in general, require resources by the adversary, and thus security due to brute force depends on the limitations of adversarial resources.[7] Cryptographers have considered *perfect security*, which is the notion that an adversary cannot succeed in discovering the plaintext given the ciphertext even by brute force. The idea is that more than one plaintext will be consistent with the ciphertext (i.e., if there is no restriction on the space of possible plaintext, say, if a random key were encrypted, then any plaintext would be possible).

The common way that perfect security can be achieved is through a *one-time pad*. The idea is to use a random stream of bits as a key stream as in Figure 3.3; this is the "pad." Consider a bit b of plaintext. It will be XORed with a random binary value r from the key stream to produce $c = b \oplus r$. Then even if c is known, because r is equally likely to be a 0 or 1, one is forced to conclude that b also could be a 0 or 1 just as equally likely. This argument extends to the whole ciphertext stream. While we are using binary values as an example, one-time pads extend to any alphabet with appropriate combining; indeed, one can create one-time pads for use with Caesar ciphers.

There are three issues with one-time pads. First, the pad should only be used once. If the same r is used to produce c_1, c_2 respectively from p_1, p_2, then it follows that $c_1 \oplus c_2 = p_1 \oplus p_2$. That means knowing the two ciphertexts allows one to make conclusions about the plaintexts, namely where they are equal or different. Second, because the pad can only be used once, and it must be in length the same as the plaintext, there are inherent implementation issues, such as how to distribute the pad. The use of a modern cipher to generate a key stream is motivated by trying to mimic a one-time pad.

The third issue is that perfect security means in fact, perfect confidentiality. One-time pads do not by themselves give integrity. If an attacker were to modify the ciphertext, this would be undetectable in general. This point is important enough to stress:

> A *one-time pad* gives perfect security, but it does not achieve all security goals.

3.3.3 Pseudorandom Generation

Using ciphers to hide information is their fundamental but not only use. For navigation, a common use is to generate pseudorandom streams easily reproducible by multiple parties. An example would be the spreading code markers discussed in Chapter 9. The main goal is

> *Goal: Pseudorandomness*
> The outputs of the cipher should behave statistically random.

Statistical randomness means that the stream would pass as random against the usual randomness tests.[4] Sometimes, we also desire the stream to be exclusive (i.e., unpredictable). That way, someone could not generate their own copy of the stream in advance. Note that these streams are not being used as one-time pads; rather, the random stream itself is what is used.

It turns out that pseudorandom streams are easy to generate using modern ciphers. One example is the counter mode in Section 3.7. The intuition for why ciphers are good for this purpose is that if the ciphertext somehow failed the randomness tests, then the structure would be exploitable against the cipher.

3.3.4 Avalanche Property

There is one final desirable property for ciphers, and that is the *avalanche property*.

> *Goal: Avalanche Property*
> A change of a single bit of the input or the key will result in ciphertext that differs essentially randomly

The idea is that ciphers map separate plaintexts that are close to separate random locations in the ciphertext space. If the plaintexts differed in, say, a single bit, then the ciphertexts would agree and disagree in about half the bits.

Table 3.2 shows an example that uses AES with 128-bit plaintexts represented as 32 hex values, where we expect, on average, to see 64 bits differ. We

Table 3.2
Differences in Ciphertexts with Similar Plaintexts

Plaintext	Ciperhtext	Difference to Row 1
0x00000000000000000000000000000000	0x10e44eded4b02f04f3653050e1b3f7e7	N/A
0x10000000000000000000000000000000	0x725669c139e33e0630baeee5eca15d98	67 bits
0x20000000000000000000000000000000	0xb5387a3b8a20d5e8b55845ab541be54c	70 bits
0x30000000000000000000000000000000	0x916bbd3848fda9dbe8e9a25136e65e14	67 bits

start with the given plaintext and ciphertext, and then each row is the plaintext with a single bit changed and the resulting ciphertext (the key used is the 128-bit key consisting of all 1s). The last column shows how many bits are in common with the first row. The term avalanche comes from the fact that this property is desired in the cipher's internal workings (i.e., that divergence between two similar plaintexts increases during the cipher algorithm operation).

The avalanche property has a nice impact on navigation when the ciphers are used to generate parts of the spreading streams. If two different generations differ even in a single bit, the resulting stream will be random (i.e., uncorrelated). The avalanche property is also very relevant in the design of hash algorithms.[4]

3.4 General Principles for Creating Ciphers

The advent of computers and the digital age led to the effort to develop ciphers that met the above goals in those settings. Some general principles were developed, led by a seminal paper by Shannon. These principles evolved to the Feistel construction, which formed the bases for the Data Encryption Standard (DES) published in 1977. As the digital world advanced, the need for an updated standard resulted in the AES in 2001 (AES is discussed in the next section).

Before we begin the discussion on cipher constructions, we want to stress the following:

> *Hazard Alert*
> Almost no one should design their own cipher; only well-established ciphers should be used.

It is a truism in the cryptography world that everyone can make a cipher that they themselves can't break, but which will almost always have severe vulnerabilities (Schneier's Law, [8]).

3.4.1 Shannon, Part 2

Shannon introduced the ideas of confusion and diffusion in his 1945 classified (confidential) memo [9], which was declassified and published in 1949 [3]. Simply put, the concepts are

> *Confusion:* The trait of the output being an obfuscation of the key;
>
> *Diffusion:* The trait of dissipating the statistical structure of the information over the whole set of data.

These concepts led Shannon to the general notion of substitution/permutation networks, which is one common method to construct modern ciphers, such as AES, Speck, Simon, and Ascon, to name only a few. Since not every cipher is a network in the sense Shannon envisioned the term, we refer to this as an S-P construction and compare it to other common methods of building modern ciphers.

3.4.2 Substitution-Permutation Construction

Shannon's focus was on how to achieve an avalanche property, which is that a change of any one bit results in a 50% probability that each output bit is changed. The construction consists of a series of rounds, or iterations, composed of three steps:

Substitution: The round input is broken into groups, and each grouping is subjected to a nonlinear function for obfuscation.

Permutation: The output of the substitution step is permuted with the goal of spreading the changes from each group as widely as possible in the output of this step.

Key Mixing: The output of the permutation step is XORed (modulo 2 addition bitwise) with a transformed version of the key. This concept is also known as *round key addition*.

The above steps over several rounds are an encryption process. One round of this construction is not enough to achieve the avalanche property, but after several rounds, the effect cascades on itself. Decryption consists of running the process backward. Reversing the process means that somehow the substitution process is invertible (permutations and XOR are already invertible).

The transformed keys are known as a *key schedule*. In particular, modern ciphers do not use the key itself each round but instead use *subkeys* derived from the key. In some cases, the key transformation uses the same substitution and permutation as the S-P construction.

The substitution step is inspired by classical substitution ciphers as we saw in Section 3.1. The permutation step seeks to spread the effect of the substitution out, as with a transformation cipher. Figure 3.4 shows a generic schematic of an S-P construction for three rounds. The boxes that do substitution are commonly called *S-boxes*; those that perform permutation are *P-boxes*.

Since each transformation is invertible, the decryption process is the encryption process run backward (last step first, etc.). A desirable property found in many ciphers is to have the decryption process be very similar to encryption; such a trait aids in implementation, such as in software reuse.

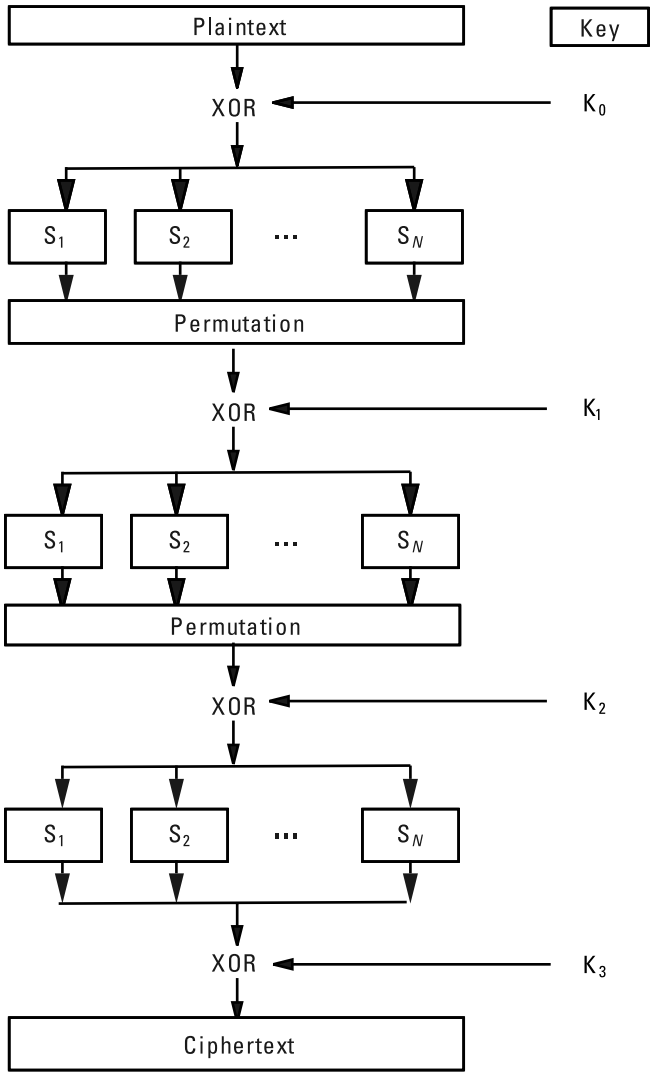

Figure 3.4 Substitution-permutation networks.

3.4.3 Feistel Construction

Horst Feistel and Donald Coppersmith created a structure known as a Feistel network or cipher; it is sometimes called a Luby-Rackoff (block) cipher after the researchers who gave the first proof of the security structure [10].

This specific implementation of the S-P construction begins by dividing the input data into two halves, called left *L* and right *R*. A given round does a computation on the right half, XORs with the left half, and that becomes the

new right half. The new left half is just the old right half. A round computation is summarized as is summarized as

$$(L_1, R_i) \leftarrow (R_{i-1}, L_{i-1} \oplus F(K_i R_{i-1}))$$

Cryptographic properties reside in the key schedule to produce the subkeys K_i and the F function that combines the subkey with the right half.

The construction is shown schematically in Figure 3.5. The right portion of the figure shows how every pair of rounds returns the left and right halves to their original positions.

Decryption just runs the process backwards. Notice that by construction $R_{i-1} = L_i$, and thus $L_{i-1} = R_i \oplus F(K_i R_{i-1})$. We stress that while encryption is reversed, the F function is never inverted, and indeed often is not invertible. The subkeys must be used in the opposite order (which typically means that the complete set is generated and used in reverse order).

The Feistel construction is the basis for many modern ciphers such as DES/3DES, Camellia, CAST-128, MISTY, Blowfish, and Twofish.

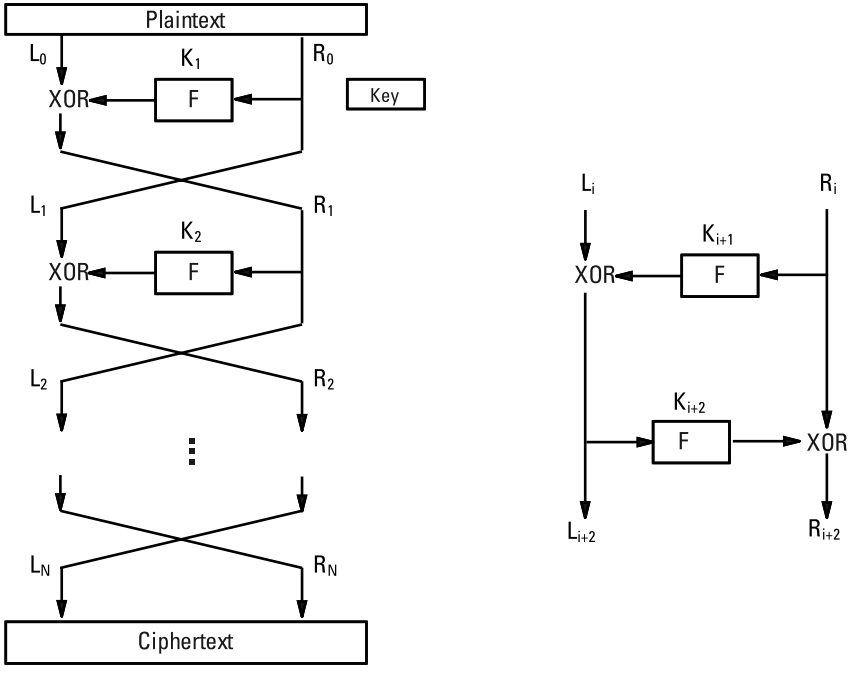

Figure 3.5 The Feistel construction.

3.4.4 Stream Ciphers

The above constructions pertain to block ciphers. We mention one general construction for a stream cipher shown in Figure 3.6. Several individual shift registers are combined to produce the output bits; the key will be used to set up and control the registers. At each clock step, some or all of the registers may advance.

3.4.5 Performance

While the above discussion focuses on the construction of ciphers from a security aspect, the performance aspects are just as important. For example, a Feistel network can be made more secure in general by increasing the rounds. But at some point, this has vanishingly small diminishing returns for security. Conversely, many excellent ciphers can be attacked in reduced-round modes. The emphasis then for cipher designers is how few rounds are needed while still maintaining security.

This performance goal is important for navigation applications, as we want cryptography to have as small an impact on the system as possible.

3.5 AES

The DES algorithm was found to have vulnerabilities by the late 1990s due to its small 56-bit key size. Even though 3DES provides extra protection with an effective key size of 112 bits, DES was designed with an emphasis on hardware, specifically binary operations. Thus, there was a desire by NIST in 1997 to create an *Advanced Encryption Standard* to supersede DES in 1997. The resulting competition chose a specific algorithm, Rijndael, which, after the choice was

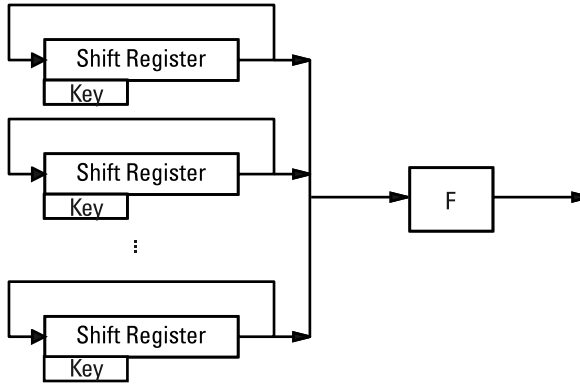

Figure 3.6 A Stream cipher construction.

made, became AES. AES is by far the universal standard for a modern cipher. This section covers the criteria of the competition (see [11]) and broad details of how AES works.

3.5.1 Choosing the AES Cipher

The competition to choose the AES cipher had several criteria. The obvious goal was security: the cipher should be robust against all known cryptanalytic attacks. The cost of the algorithm, in terms of execution complexity, was also a factor. The late 1990s saw the emergence of 32-bit architectures in computing and 8-bit architectures on smart cards; as such, the cipher needed to be implemented in both settings. Finally, the competition also considered general algorithm and implementation characteristics (i.e., implementation both in software and hardware). With regard to parameters, the candidates needed to support key sizes of 128, 192, and 256 bits with the plaintext block size of 128 bits.

After the initial call for possible candidates by NIST in 1997, 15 candidates passed through the first round in June 1998, and then that list was culled to the five AES finalists in August 1999. The finalists were MARS, RC6, Rijndael, Serpent, and Twofish [12]. The algorithm developed by Belgians Dr. Joan Daemen and Dr. Vincent Rijmen was Rijndael (a combination of their names), and it was selected as AES in October 2000. The parameters were standardized in the NIST November 2001 in the Federal Information Processing Standards (FIPS) publication 197 [13].

3.5.2 AES Structure

The general structure of AES is a substitution/permutation network shown in Figure 3.7 The number of rounds is a function of the key size and is given in Table 3.3. Figure 3.7 shows a 10-round AES.

While a thorough discussion of the AES operations is beyond our scope (see [12, 14]), we offer some brief descriptions. The round acts on 128 bits, which is taken to be 16 bytes. In fact, it is often convenient to consider these as a 4×4 array of bytes. The round function is given in four steps: byte substitution, shift rows, mix columns, and round key addition.

Byte substitution: A nonlinear function (S-box) is applied to each byte. This function has a simple definition, although in practice, it is usually implemented as a table. It is reversible.

Shift rows: Each row of the array is shifted by a different prescribed amount.

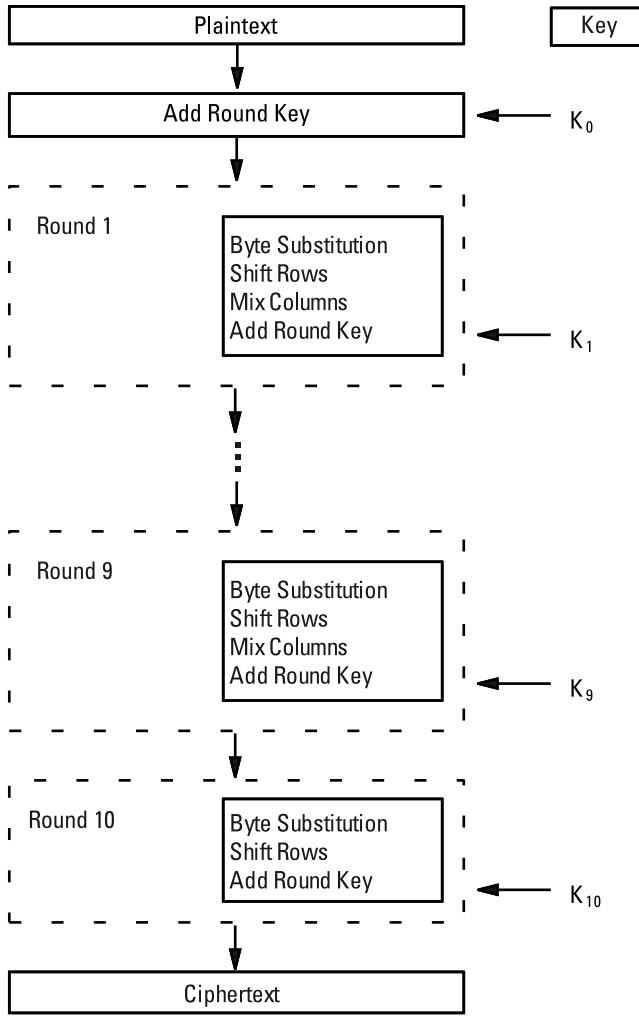

Figure 3.7 General AES structure.

Table 3.3
AES Key Size and Rounds

Name	Key Size	Number of Rounds	Number of Subkeys
AES-128	128	10	11
AES-192	192	12	13
AES-256	256	14	15

Mix columns: The array is considered as a 4×4 matrix, and it is premultiplied by a fixed prescribed 4×4 matrix.

Add round key: A 16-byte subkey generated for each round (and as a preaddition is XORed in bytewise.

The final round drops the mix-column operation. The rationale for this is that it adds no security in the last round.

The whole algorithm uses linear algebra over the *finite field* (or *Galois field*) GF(256). This field can be defined as the field of polynomials with binary coefficients modulo $m(x) = x^8 + x^4 + x^3 + x + 1$; see [14]. A very detailed explanation by the algorithm's authors is available [12]. See [15] for insights into the mathematical structure.

Reversing the process results in a set of functions that are very similar to those used for encryption. This structure means that encryption and decryption can reuse software or hardware blocks.

The AES key schedule uses the same AES S-box. The key is arranged as a byte matrix, with four rows and a variable number of columns, C_i, depending on the length of the key. The key state matrix is filled in column-major format (the first column is filled first, etc.). Then an iteration process is used to generate new columns from previous ones, using the S-box and some mixing of values. Note for decryption, the key schedule must be used in reverse order.

3.5.3 AES Performance

The ubiquity of AES means that many efficient implementations have been developed in both software and hardware. We show some results in Table 3.4. These are based on the speed function run in the widely used OpenSSL application for cryptographic functions. The results are based on one of our laptops and, of course, depend on details in the computing architecture. The results are for AES executed in cipher-block-chaining (CBC) mode (see Section 3.7). The columns show the size of the data being executed. The rows show approximate throughput in units of MB per second.

Table 3.4
AES Execution Time

Key Size	Data Size					
	16 bytes	**64 bytes**	**256 bytes**	**1024 bytes**	**8192 bytes**	**16384 bytes**
128	224 MB/s	191 MB/s	145 MB/s	183 MB/s	174 MB/s	179 MB/s
192	150 MB/s	157 MB/s	170 MB/s	168 MB/s	170 MB/s	176 MB/s
256	126 MB/s	69 MB/s	98 MB/s	97 MB/s	103 MB/s	101 MB/s

Looking across rows in Table 3.4, we see variation likely due to system architecture. On the whole, though, throughput is fairly stable. Looking down columns, we see the dependence on the key size due to the key size determining the number of rounds. The actual application should be considered if the added execution time cost warrants the extra security.

3.5.4 AES Takeaway

The ubiquity of AES also means that it is, by default, the first choice when a modern cipher is needed. Indeed, the onus is usually on giving a reason not to use AES.

3.6 Lightweight Crypto

Even with the wide use of AES in everyday protocols (see Chapter 6 such as Internet Protocol security (IPsec) and Transport Layer Security (TLS)), the desire in recent years has been to standardize cryptographic ciphers that can trade off strength with performance. One perceived use is for the Internet of Things (IoT), but a similar motivation would possibly apply to future navigation systems. These new ciphers are often described as lightweight due to their lower implementation complexity. We discuss the general construction for these ciphers and give some specific examples.

3.6.1 Add-Rotate-XOR Ciphers

One of the more interesting ideas in cryptographic design is the use of add-rotate-XOR (ARX) ciphers, a term coined around 2008. The add operation is modulo 2^N, and the rotate function is left circular rotation, on N bits; so $x \ll 5$ rotates left five places and $x \ll -5$ is a right circular shift by 5 (or $x \gg 5$). These operations are simple in software, and their uniformity helps in implementation security. An early block cipher that used an ARX construction was Feal [16], a Feistel cipher in the late 1980s, although every version has been badly broken. The NIST Hash competition (see Chapter 4) included a finalist cipher (Threefish [17]), which is an ARX cipher. We describe two ARX ciphers here: ChaCha20 and Speck.

3.6.2 ChaCha20

ChaCha20 was developed by Daniel Bernstein [18] in 2007–2008 as a modification to his previous work called Salsa. ChaCha20 has been selected by Google

for web security and has been expanded to include authentication. ChaCha20 is a stream cipher. It uses a state defined as a matrix of 16 32-bit words. The initial state S_{-1} comprises fixed four 32-bit constants, the 256-bit key in eight 32-bit words, a 64-bit counter, and a 64-bit nonce, a randomly chosen number that is used only once. The state update consists of several different functions that use only the operations of add modulo 2^{32}, XOR, and rotate. Encryption is performed by XOR of the plaintext with the state. Notice that the binary addition operation is the only non-linear operation, since linear means with respect to XOR.

3.6.3 Simon and Speck

Simon and Speck were designed by the National Security Agency (NSA) as block ciphers for IoT applications [19, 20]. We mention two features that make these algorithms worth considering. The first, which was articulated by Bruce Schneier is that NSA key schedules are *always* interesting [21]. The second is that the algorithms have been criticized for lower security margins than, say ChaCha20 (Section 3.6.2). The design team responded that the applications warranted the lower margins due to application constraints [22]. The lower margin was against an attack that is slightly better: a factor of $2^7 = 128$ in time with data complexity of $> 2^{127}$ in data than brute force [23] (this is a theoretical attack, not anywhere near practical in either data or time). The reduced margins show applicability to many IoT applications, and the design team specifically mentions that fractions of a penny matter in the cost of some IoT items. For many applications, this concern is true for satellite navigation user equipment, especially when interpreted as power limitations. In navigation signal design, this problem is sometimes known as the tyranny of the AA cell.

Simon and Speck have been adopted as Radio Frequency Identification (RFID) standards by the International Standards Organization (ISO) as (Simon) ISO/IEC 29167-21:2018 [24], (Speck) ISO/29167-22 [25]. They are simple algorithms; Simon is a Feistel design, and Speck is an ARX design with two Feistel-style loops per round. They were designed with a variety of data and key sizes, including some that are below the security threshold for preserving the confidentiality after a session ends.

In the design document [19], the authors note that key sizes under 80 bits will not provide much security and should be used only for transient applications. This is an interesting acknowledgment that long-term confidentiality may not be the main security goal in some applications. This observation applies to several of the methods for securing navigation ranging and data in this book (Chapters 9 and 10).

3.7 Cipher Modes

Our discussion of ciphers has focused naturally on the algorithms that take a block of plaintext and encrypt it to a block of ciphertext. However, actual data is almost never exactly a single block in length, which then begs the question: how do we encrypt a larger amount of data? The answer is to use *cipher modes,* which define the application of a series of calls to the encryption algorithm. This section discusses several examples of cipher modes with a focus on confidentiality-only goals; see [26, 27]. Some cipher modes are also used for authentication, which is discussed in Chapter 4. Excellent sources for a few more common modes are found in [14, 28].

3.7.1 Overview

All cipher modes consist of a procedure to take a sequence of plaintext blocks and produce a sequence of ciphertext blocks. How each ciphertext block depends on the key and plaintext is, of course, where the modes differ in security, complexity, and performance. The simplest and perhaps obvious choice of encrypting each block independently is called *electronic codebook (ECB)* and is one of the methods that we discuss. However, there are better methods that should be used besides ECB, specifically *counter mode (CTR)* and CBC. In all our examples, the specific cipher is subsumed by the encryption notation (e.g., one can assume AES as a default). While cipher modes have been around as long as modern ciphers, their implementation sometimes have flaws that must be acknowledged and mitigated when they matter. The NIST recommendations for cipher modes is found in [26, 27], which are being updated as of this writing.

Padding
Cipher modes do not necessarily define how to handle padding.

The definition of cipher modes may assume that the data is an integral number of blocks. When the data varies from that, it must be padded in some way. Some software implementations will pad automatically (e.g., with zero-value bytes). Since decryption will produce the original data plus the pad, adding arbitrary bytes at the end may or may not be an issue, depending on assumptions about the plaintext. A better solution is to prescribe a precise padding procedure. Such a solution is used, for example, in the hash function standards discussed in Chapter 4. An alternative to padding in some instances is ciphertext stealing, which uses previously computed ciphertext to avoid padding (see [14, 29]).

In the context of satellite navigation, and especially for applications for confidentiality in-band in a navigation signal, the confidentiality cipher modes

will likely play a significant role. That said, which cipher mode may vary depending on trading off complexity with other traits.

3.7.2 Electronic Codebook

As mentioned, ECB mode is the natural cipher mode to consider. The mode simply encrypts each plaintext block sequentially, as in Figure 3.2. If P_1, P_2, ... denote the sequence of plaintext blocks being encrypted with key K, then the ciphertext blocks C_1, C_2, ... are given as

$$E_K\left[P_i\right] = C_i, \quad i = 1,2,\ldots \tag{3.6}$$

Decryption follows straightforwardly as

$$D_K\left[C_i\right] = P_i, \quad i = 1,2,\ldots \tag{3.7}$$

Because blocks are independent, the operations are highly parallelizable, and there is no error propagation if, for example, a ciphertext block were received in error.

ECB suffers from a traffic analysis flaw. Because the same plaintext always encrypts to the same ciphertext, someone can discern when plaintext repeat even with a secure cipher and key. This flaw can be readily illustrated pictorially. Consider the images in Figure 3.8. The image on the left is encrypted using two different cipher modes. ECB mode still allows one to discern the image of the runner, even though the actual pixel data is ciphertext. The reason is that the same plaintext in the background encrypts to the same ciphertext. This flaw is a main reason why ECB is usually not recommended. As we see on the right side, another ciphermode (CBC) does not have this issue when properly implemented.

Our contention is that if plaintext does not repeat, say if it is random or timed information, then the use of ECB could be fine. An example would be

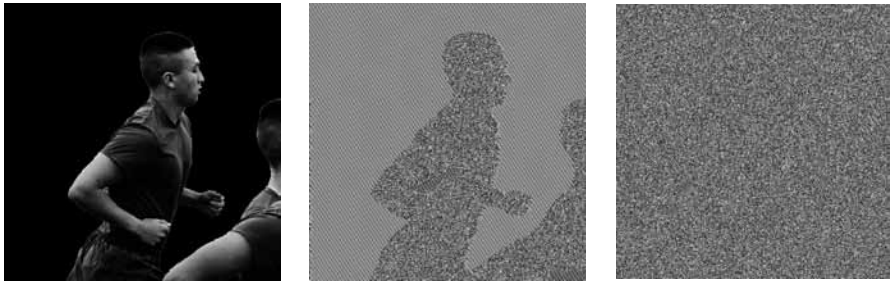

Figure 3.8 Information leakage in ECB mode compared to CBC mode.

key wrapping [30, 31] when a key is encrypted by a separate process. Similar examples include encrypting a nonce to obtain a one-time use pseudorandom block.

Pros for ECB:

- Errors do not propagate; an error in one ciphertext does not interfere with the decryption of the subsequent blocks.
- The size of the data is not changed.

Cons for ECB:

- It can expose repetitions in data.

3.7.3 Counter Mode

CTR mode is the emulation of a one-time pad using a cipher. The idea is that a counter is successively encrypted, and the result is XORed into the plaintext to produce the ciphertext. The counter often includes a nonce as part of the procedure. If the nonce-counter pair is agreed upon between the sender and the receiver, then the ciphertext is the same size as the plaintext. If an agreement is not made, then the nonce must be communicated to the receiver; generally, this would be encrypted using ECB.

Using CTR_i as the counter, we have

$$E_K\left[CTR_i\right] \oplus P_i = C_i, \quad i = 1, 2, \ldots \tag{3.8}$$

Decryption follows straightforwardly as

$$E_K\left[CTR_i\right] \oplus C_i = P_i, \quad i = 1, 2, \ldots \tag{3.9}$$

Notice that the decryption procedure still uses the encryption function of the cipher. As with ECB, the blocks are independent, which means the operations are highly parallelizable and there is no error propagation. The output of the encryption is often called the key stream. See Figure 3.9 for a schematic view of encryption for CTR; decryption just follows by XOR-ing the ciphertext with the key stream to obtain the plaintext.

For CTR mode to be an effective encryption, the combination of nonce and counter must *never be reused with the same key*. To reuse the nonce-counter would have the same issue as the reuse of a one-time pad. Because of this non-reuse, CTR does not have the traffic analysis problem that ECB has: the same plaintext block will not result in the same ciphertext block ever.

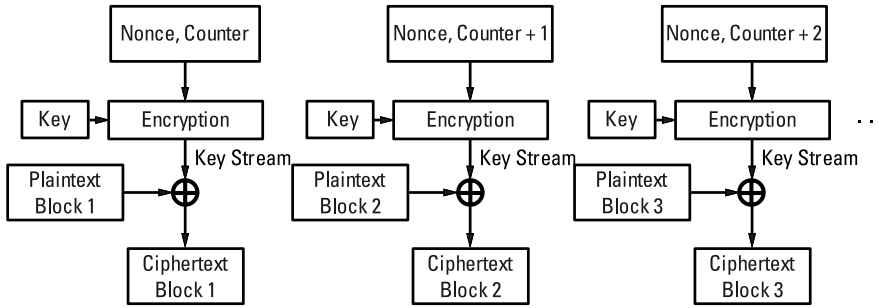

Figure 3.9 Encryption for counter mode.

A method similar to CTR is output feedback (OFB) mode, which also constructs a key stream as a cipher. However, unlike CTR, the key stream has dependencies in the key stream blocks. The process must be started by an *initialization vector (IV)*, and the IV must never be reused. The best choice for an IV would be a random nonce. If that cannot be agreed upon, it could again be sent encrypted along with the ciphertext,

For OFB, let the key stream blocks be denoted by S_1, S_2, …. Then E_K[IV] = S_1 and

$$E_K\left[S_{i-1}\right] = S_i, \quad S_i \oplus P_i = C_i, \quad i = 1,2,\ldots \tag{3.10}$$

Decryption follows straightforwardly as

$$S_i \oplus C_i = P_i, \quad i = 1,2,\ldots \tag{3.11}$$

using the same process to generate the S_i. Thus there is chaining in the key stream blocks, which means the key stream cannot be created in parallel. Since CTR is a simpler version of the same style of cipher mode, it is usually preferred.

Pros for CTR:

- Errors do not propagate;
- The size of the data is not changed;
- The key stream is not dependent on the data so it can be precomputed;
- There is no need for cipher-text-stealing, as CTR mode acts very much like a stream cipher;
- Encryption and decryption are precisely the same as the encryption mechanism is XOR;

• CTR mode encryption and decryption is parallel.

Cons for CTR:

• Coordination of the nonce;
• Reuse of the nonce/counter is disastrous.

3.7.4 Cipher Block Chaining

The final cipher mode we discuss in this chapter is CBC, which in many circumstances is the preferred method of encrypting data. The idea is to start encryption with an IV, feed it into the first block, and chain the subsequent blocks. If a nonrepeated IV is used, say a nonce, then encryption will produce ciphertext that is always unrelated to the plaintext. The cost is that some schemes for the IV will require an extra preencryption, that is, one extra ciphertext block to be transmitted. Generally, this is not a problem, but this can be a disadvantage in bandwidth-constrained situations and with short messages.

Let C_0 be the IV. Then CBC encrypts by chaining the ciphertexts by

$$E_K\left[C_{i-1} \oplus P_i\right] = C_i, \quad 1,2,\ldots \tag{3.12}$$

Decryption follows straightforwardly as

$$D_K\left[C_i\right] \oplus C_{i-1} = P_i, \quad 1,2,\ldots \tag{3.13}$$

See Figures 3.10 and 3.11 for schematics of CBC encryption and decryption.

Pros for CBC:

• Limited error propagation;
• Decryption can be done in parallel.

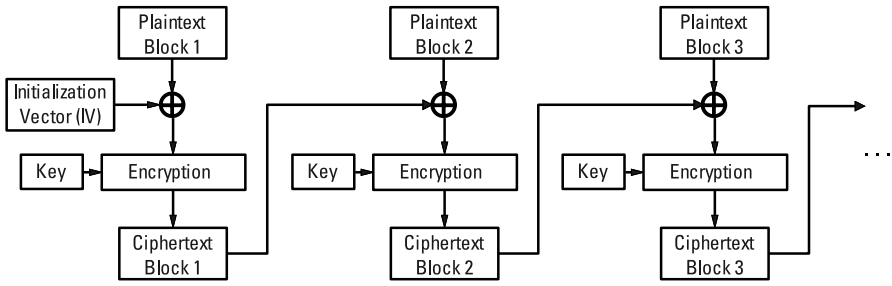

Figure 3.10 Cipher block chaining encryption.

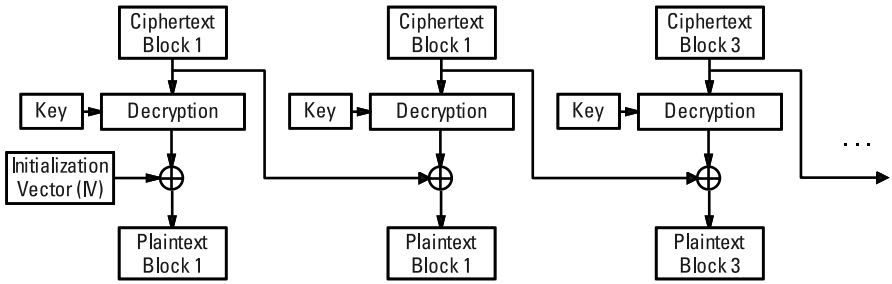

Figure 3.11 Cipher block chaining decryption.

Cons for CBC:

- Extends the message by one ciphertext (for the initialization vector);
- Sensitive to nonce reuse.

CBC encryption cannot be done in parallel, since the chaining depends on the plaintext. CBC decryption can be parallelized if all the ciphertexts are given, since the decryption for a given plaintext depends only on two consecutive ciphertexts. Error propagation is also contained: a single bit error in a ciphertext will affect the current plaintext block and the subsequent plaintext block.

3.7.5 Which Mode to Use?

It is difficult to give an answer to the question of which mode to use:

- CBC is often the recommendation choice using an encrypted IV. This mode has the advantage that there may be little coordination needed between the two parties.
- CTR is simpler, especially since any portion of the key stream can be computed independently of any other, and yet we do not have to worry about error propagation or traffic analysis.
- ECB suffers from the traffic analysis issue. But we feel it is a perfectly viable choice for applications that are not prone to traffic analysis.

3.8 Steganography

The usual goal of cryptography is to keep a message secret between two parties. Steganography has a similar goal of concealing a message between two parties.

Both words mean hidden or concealed writing using two different words from Greek. While cryptography obfuscates information, steganography hides the information in some other media. The media in which the message is placed is called the cover-work.

A related concept to steganography is watermarking. Various authors draw different distinctions between the two concepts. One useful distinction is that steganography hides a message in the cover-work that is unrelated to the cover-work. On the other hand, watermarking content is specific to the cover-work. Note that the definition of steganography requires that the message be hidden; that is, not easily detected. The definition of watermarking says nothing about the detection of the presence of the watermark in the cover-work we are following [32].

In steganography, messages can be layered. The idea of layering would be, for instance, to write an innocent letter, the cover-work, and then write in invisible ink a second letter in the spaces. Simple examples are using skim milk or lemon juice, which dry clear but are revealed with heat. Other examples of using physical methods include using Morse code to modulate natural features, such as blinking eyes or knots in fabric.

A more sophisticated example is the Bacon cipher, developed by Sir Francis Bacon. The idea was to use differences in the presentation to encode the message, such as two typefaces or written scripts. The text is then broken into groups of five letters, and shifts between the typefaces were used to encode the message. Using five letters allows for 32 symbols by switching between two typefaces, enough for English text. Other examples using writing include having the message be the third word in each sentence in a longer message, and so forth. Digital data allows greater possibilities for steganography. For example, one could encode information in the least significant bits in digital media, such as images or audio files.

These methods rely on only the intended party knowing of the existence of an encoding message. If instead everyone is aware of the technique, say for a Bacon-coded passage, then it is not steganographic but could act as a watermark. In this case, the watermark is asserting the authorship of the information. Occasionally, authors add requirements that the watermark be imperceptible or integral to the integrity of the cover-work. In trying to defeat intellectual property theft, these goals seem natural, but they add requirements to the application; an example is antipiracy of a movie.

The issue is that if a watermark is known, it can almost certainly be removed. Mitigating this risk is an important concern for using watermarks for navigation (i.e., in navigation signals). Since watermarks are intended to achieve the goals of authenticity and integrity, we defer the discussion to Chapter 4.

3.9 Foreshadowing: Navigation Examples

The two cipher uses, confidentiality and pseudorandomness, manifest in navigation examples. For confidentiality, most uses in a satnav enterprise that require exclusivity will require encryption at some stages. Often this desire will be accomplished using standard protocols (see Chapter 6). As such, the cryptography used is straightforward.

The use of ciphers for pseudorandomness is also useful in navigation applications. Cipher-based spreading sequences, by nature, are pseudorandom; this means uncorrelated and therefore they can be used in (or as) CDMA signals (see Section 2.6). The unpredictability of these sequences allows the spreading code to provide authentication of the spreading code and hence the signal to the extent that the key can be kept uncompromised. This observation leads to the usage of ciphers in spreading code markers (see Chapter 9).

3.10 Takeaways

A modern cipher, such as AES, transforms a plaintext block into a block of ciphertext using a shared secret called a key. This process achieves confidentiality: knowing the ciphertext reveals no substantial information about the key or plaintext even if the algorithm is known. Furthermore, knowing both the plaintext and ciphertext and multiple such pairs yields no information about the key. No information means that the only option is to brute force the key, which for reasonable key sizes is computationally infeasible.

Ciphers can also be used to generate pseudorandom information. Such information is both statistically random and unpredictable. This trait is useful in navigation, specifically in generating spreading code features.

Cipher algorithms must be used with a cipher mode when data is longer than a single block. The two main cipher modes to use are CTR or CBC; in some special cases ECB may be viable.

Finally, the most common choice for a cipher is AES, although there are several examples of lightweight ciphers that may be good choices in some settings.

End Notes

1. Examples date back to ancient Egypt and Mesopotamia. An interesting example from ancient Greece is the scytale, which is a tool used to implement a transposition cipher and dates from the seventh century BCE. Examples were prevalent throughout the middle ages and Renaissance, although often the methods were closer to steganography than ciphers (see Section 3.8). Substitution ciphers are prominent in two famous short stories: Sherlock

Holmes' "The Adventure of the Dancing Man" by Arthur Conan Doyle and "The Gold-Bug" by Edgar Allan Poe.

2. There are many puzzle books and websites for cryptograms (e.g., www.cryptograms.org).

3. There are several helpful books on information theory with different emphases. McEliece [33] introduces the subject with a focus on communication aspects, such as data compression and error correction. Cover and Thomas's book [34] is a good, rigorous treatment with callouts to many applications, including gambling and stock market investing. An advanced book on *information-theoretic cryptography*, with that title, is [35]. Finally, MacKay [36] looks at information theory with a focus on inference and machine learning applications.

4. Randomness is often defined formally in terms of probability spaces and random variables. A different but related method concerns *incompressibility*. Intuitively, a random string cannot be described smaller than its length. This concept can be made rigorous; see Li and Vitányi [37].

   Related to defining randomness is testing it. A variety of randomness testing suites exist, for example the NIST publication [38].

5. Textbooks on cryptography and block ciphers will discuss more sophisticated methods, for example, [39]. Linear cryptanalysis looks to leverage linearity biases between the key, plaintext, and ciphertexts. Slight biases can then be exploited. Differential cryptanalysis looks at biases in differences between pairs of plaintext and ciphertext. Modern ciphers will explicitly be designed to mitigate these attacks.

6. In a brute force of a key space of size 2^n, one expects on average to find the key in 2^{n-1} searches. This calculation assumes that the keys are equally likely. In the case when the search space contains elements that have different likelihoods of occurring, then a natural *guessing entropy* arises. The guessing entropy is just the expected search time, when the search strategy is to look first at the most likely item, then the second most, and so on. There are relationships between the guessing entropy and Shannon entropy.

7. The field of theoretical cryptography rigorously defines the complexity classes relevant to cryptanalysis similar to complexity classes like P, NP, and so on, used in computer science [40]. The relevance of this book is the relationship between classical and quantum computing, which we explore in the appendix.

8. Claude Shannon's importance in almost all aspects of the digital age cannot be overstated (we are continually perplexed that he is not better known). To name a few examples besides cryptography: Shannon did seminal work in showing Boolean algebra as the right tool for simplifying digital circuits. The Shannon sampling theorem gives the conditions for the equivalence of digital and analog signals. Shannon went on to develop a theory of communication, documented in [4], information theory, and the mathematical approach to the design of ciphers that we presented in this chapter [3].

References

[1] Suetonius, *The Lives of the Twelve Caesars, by C. Suetonius Tranquillus,* Project Gutenberg, https://www.gutenberg.org/files/6400/6400-h/6400h.htm.

[2] Singh, S., *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, in collaboration with Internet Archive, Anchor, 2000, http://archive.org/details/codebook00simo.

[3] Shannon, C. E., "Communication Theory of Secrecy Systems," *Bell System Technical Journal,* Vol. 28, No. 4, October 1949, pp. 656–715, doi:10.1002/j.1538–7305.1949.tb00928.x, https://ieeexplore.ieee.org/stamp/stamp.jsp? tp=&arnumber=6769090&isnumber=6769085.

[4] Shannon, C. E., "A Mathematical Theory of Communication," *ACM SIGMOBILE Mobile Computing and Communications Review,* Vol. 5, No. 1, 1948, pp. 3–55, doi:10.1145/584091.584093, https://doi.org/10.1145/584091.584093.

[5] Shannon, C. E., "Prediction and Entropy of Printed English," *The Bell System Technical Journal,* Vol. 30, No. 1, 1951, pp. 50–64, doi: 10.1002/j.1538-7305.1951.tb01366.

[6] Cryptographic Mechanisms BSI, *Recommendations and Key Lengths,* Version 2022-01, *Bundesamt Fur Sicherheit in Der Informationstechnik*, BSI TR-02102-1, 2022/01/28, 2022, https://www.bsi.bund.de/SharedDocs.

[7] Barker, E. B., and Q. H. Dang, *Recommendation for Key Management Part 3: Application-Specific Key Management Guidance*, NIST SP 800-57Pt3r1, National Institute of Standards and Technology, January 2015, NIST SP 800–57Pt3r1, doi:10.6028/NIST.SP.800-57Pt3r1, https://nvlpub.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf.

[8] https://www.schneier.com/blog/archives/2011/04/schneiers_law.html.

[9] Shannon, C. E., N. J. A. Sloane, and A. D. Wyner, *Claude Elwood Shannon: Collected Papers*, IEEE Press, 1993.

[10] Luby, M., and C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions," *SIAM Journal on Computing,* Vol. 17, No. 2, April 1988, pp. 373–386, doi:10.1137/0217022, https://epubs.siam. org/doi/10.1137/0217022.

[11] Nechvatal, J., et al., "Report on the Development of the Advanced Encryption Standard (AES)," *Journal of Research of the National Institute of Standards and Technology,* Vol. 106, January 2001, doi:10.6028/jres.106.023.

[12] Daemen, J., and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard,* Second Edition, 2020, doi:10.1007/978-3-66204722-4.

[13] Dworkin, M. J., E. B. Barker, J. R. Nechvatal, et al., *FIPS-197 Advanced Encryption Standard (AES)*, November 26, 2001, https://www.nist.gov/publications/advanced-encryption-standard-aes.

[14] Oppliger, R., *Cryptography 101: From Theory to Practice*, Norwood, MA: Artech House, 2021.

[15] Landau, S., "Polynomials in the Nation's Service: Using Algebra to Design the Advanced Encryption Standard," *The American Mathematical Monthly,* Vol. 111, No. 2, 2004, pp. 89–117, doi:10.2307/4145212. JSTOR: 4145212, http://www.jstor.org/stable/4145212.

[16] *NTT Encryption Archive List-NTT Social Informatics Laboratories*, NTT Cryptographic Primitive, https://info.isl.ntt.co.jp/crypt/eng/archive/.

[17] Information Technology Laboratory Computer Security Division, *SHA-3 Project-Hash Functions—CSRC,* CSRC-NIST, January 4, 2017, https://csrc.nist.gov/projects/hash-functions/sha-3-project.

[18] Bernstein, D. J., *ChaCha, a Variant of Salsa20*, January 28, 2008, https://cr.yp.to/chacha/chacha-20080128.pdf.

[19] Beaulieu, R., D. Shors, J. Smith, et al., *The SIMON and SPECK Families of Lightweight Block Ciphers*, June 19, 2013, https://eprint.iacr.org/2013/404.pdf (prepublished).

[20] Beaulieu, R., D. Shors, J. Smith, et al., *Simon and Speck: Block Ciphers for the Internet of Things*, 2015, https://csrc.nist.gov/csrc/media/events/lightweight-cryptography-workshop-2015/documents/papers/session1-shors-paper.pdf (prepublished).

[21] Schneier, B., *SIMON and SPECK: New NSA Encryption Algorithms-Schneier on Security,* July 1, 2013, https://www.schneier.com/blog/archives/2013/07/ simon_and_speck.html.

[22] Beaulieu, R., D. Shors, J. Smith, et al. *Simon and Speck: Block Ciphers for the Internet of Things*. Conf. Paper. NIST, July 15, 2015.

[23] Chen, H., and X. Wang, *Improved Linear Hull Attack on Round-Reduced \textsc{Simon} with Dynamic Key-Guessing Techniques*, 2015, https://eprint. iacr.org/undefined/undefined (prepublished).

[24] ISO/IEC, *ISO/IEC 29167-21:2018*, 2018, https://www.iso.org/standard/ 70388.html.

[25] ISO, 14:00-17:00, *ISO/IEC 29167-22:2018*, https://www.iso.org/standard/ 70389.html.

[26] Dworkin, M., *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, US National Institute of Standards and Technology (NIST), 2001.

[27] Dworkin, M., *Announcement of Proposal to Revise Special Publication 800-38A,* March 2022, https://csrc.nist.gov/news/2022/proposal-to-revise-sp-800-38a.

[28] Paar, C., and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*, Berlin: Springer, November 8, 2014.

[29] Dworkin, M., *Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*, US National Institute of Standards and Technology (NIST), 2011, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a-add.pdf.

[30] Rogaway, P., and T. Shrimpton, *Deterministic Authenticated-Encryption: A Provable-Security Treatment of the Key-Wrap Problem*, 2006, https://eprint.iacr.org/2006/221 (prepublished).

[31] Dworkin, M., *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping 800-38A*, December 2012, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf.

[32] Cox, I., M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography,* Second Edition, Burlington, MA: Morgan Kaufmann, 2007.

[33] McEliece, R., *The Theory of Information and Coding,* Second Edition, Cambridge, United Kingdom: Cambridge University Press, 2002, doi:10.1017/CBO9780511606267.

[34] Cover, T. M., and J. A. Thomas, *Elements of Information Theory,* Hoboken, NJ: Wiley-Interscience, 2006.

[35] Tyagi, H., and S. Watanabe, *Information-Theoretic Cryptography*, Cambridge, United Kingdom: Cambridge University Press, 2023.

[36] MacKay, D. J. C., *Information Theory, Inference, and Learning Algorithms*, Cambridge, United Kingdom: Cambridge University Press, 2003.

[37] Li, M., and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications,* Second Edition, Berlin: Springer-Verlag, 1997.

[38] Bassham, L. E., et al., *SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,* Gaithersburg, MD: National Institute of Standards & Technology, 2010.

[39] Menezes, A. J., P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton, FL: CRC Press, 2001, http://www.cacr.math.uwaterloo.ca/hac/.

[40] Sipser, M., *Introduction to the Theory of Computation,* International Thomson Publishing, 1996.

4

Hashing

The second main category of cryptographic algorithms is hashing. Hashes are the authors' favorite cryptographic methods because of the wide range of applications that utilize hashing to achieve the security goals of authentication and integrity. Indeed, for satellite navigation, arguably, authentication and integrity are the most important goals, and hashes are instrumental in many proposed and implemented methods.

A *hash function* reduces a large amount of information down to a small amount, also called a hash or fingerprint. There are many non-cryptographic examples, such as parity checks or hashes into a database table. Cryptographic hashes require that the functions obey rigid requirements with regard to their one-way-ness and robustness with regard to collisions, defined as incidences when items have the same hash. As with AES for ciphers, there are some well-established standards for hash functions, notably the standards for the SHA-2 family and the relatively new SHA-3 (SHA stands for secure hash algorithm).

Hash functions by themselves do not use a shared key as ciphers do. However, the presence of a shared key does allow for further security goals through the creation of keyed hashes, an example being message authentication codes (MACs). Examples of MACs include the hash-based message authentication code (HMAC) standard and the use of ciphers to create MACs. MACs can be used to establish both the integrity and authenticity of the data (i.e., that the data has not been modified and it originates from the correct source).

This chapter delves into several applications of hash functions. As mentioned, hash functions can be used as digital fingerprints, essentially distilling all the information in the input to a small snippet. The one-way notion of hash functions suggest chaining the function (i.e., repeated applications of hashing), which yields dependencies on previous hashes. Such hash chains have

important uses—perhaps most recognizable is their use in blockchains. The one-way nature of hash functions also yields a useful concept of *bit commitment*, which basically cements information before it is revealed. Hash chains and bit-commitment find their uses in satellite navigation, one example of which is the *Timed Efficient Stream Loss-Tolerant Authentication (TESLA)* protocol, which we describe in general terms in this chapter before detailing its specific uses in Chapters 8 and 10.

4.1 Hash Functions and Their Goals

A hash function reduces some amount of data to a smaller amount, also called the hash (see Figure 4.1). The hash function takes all the data to produce a relatively small hash. For example, a large software repository may be reduced to a 32-byte long hash. There is no notion of piecing together a single hash function operation to handle larger amounts of data, as is the case with ciphers and cipher modes; the piecing together is integral to the hash function. A variety of terms are used for the output besides hash, such as digest and fingerprint.

Our focus is on cryptographic hashes, which satisfy the specific goals listed below. Intuitively, we want the hash to be bound to the data so that any change in the hash indicates a change in the data. In addition, we want it to be difficult to find multiple data that produce the same hash. These traits mean that someone can verify that the information has not been modified if the hash of that information matches the given hash. In this way we achieve integrity. If the hash is given before the data is known, then we have authenticated the data when the hash is subsequently verified; that is called bit commitment and expanded on later. A common use of cryptographic hashes is as digests for software downloads, where the hash can be used to verify the integrity of the download.

Noncryptographic hashes include functions like parity checks and cyclic redundancy checks (CRCs).[1] Their purpose is to distill the information into a small hash in an efficient manner, often with other provable properties, like

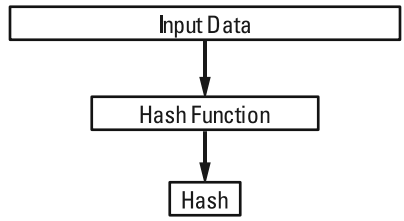

Figure 4.1 General hash function.

error detection. Interestingly, one can use these noncryptographic functions to create cryptographic hashes in a method called *universal hashing*.[2]

When a cryptographic hash is used as a fingerprint of the data, often called a *message digest*, it should have the following properties:

Distinguished: If two message digests are equal, then their inputs should be equal. Note that uniqueness is up to a random chance governed by the hash sizes.

Nonforgeable: This property is a corollary of uniqueness (i.e., different inputs cannot be found that produced the same message digest).

Integrity: Any changes in the input should be indicated by a mismatch in the message digest.

Efficiency: The goal of a fingerprint is to have a succinct representation of the input, which means it must be easy to compute and as small in size as the security goals allow.

The rest of subsection elaborates on these properties in terms of more formal definitions.

4.1.1 Preimage Resistance

The first goal for a cryptographic hash is that it should be one-way (i.e., it should not be easy to find anything that hashes to a given hash). This is the uniqueness property—that the hash essentially identifies the input. This goal is formalized as follow:

Preimage Resistance
Given a hash function $H(\cdot)$ and a hash value y, it should be computationally difficult to find a value x such that $H(x) = y$.

Suppose the hash y is n bits long. Then preimage resistance says that it is too difficult to find any input—of any length—that hashes to that n-bit value. If the data input is restricted to N bits, then on average we expect 2^{N-n} possible inputs, for example, $N = 1024$ and $n = 256$ implies 2^{768} possible inputs on average that would hash to the given value. A good cryptographic hash means that finding even one of these is practically impossible.

We could always brute force to find such an input. That would take on average 2^n computations, since each try has a $1/2^n$ possibility of succeeding. Thus the length of the hash would seemingly dictate its strength, similar to how the length of a key dictates the strength of a cipher. However, as we see next, in fact the strength of a hash is defined by half its length, $n/2$.

4.1.2 Collisions

We first define the other two main cryptographic goals for hashes. Both these goals relate to *collisions*, the idea of having two separate inputs that produce the same hash. Protecting against collisions relates to the properties of being nonforgeable.

The first goal pertains to finding a second input that matches a given input/hash pair. Violating this goal means we have a collision for the given hash.

Weak Collision Property
Given a hash function with $H(x_1) = y$, it is difficult to find another x_2 with $H(x_2) = y$

The second goal says that it is difficult to find any collision pair.

Strong Collision Property
It is difficult to find any pair x_1 and x_2 with $H(x_1) = H(x_2)$.

Note that it is possible to have the weak collision property without the strong collision. That is, suppose it is difficult given $H(x) = y$ to find a different preimage of y. It could conceivably be possible to find some pair that collide without their hashes being specified. In other words, the strong collision property implies the weak collision property: if we do not satisfy weak collision property, then we can find a collision, which means that we cannot satisfy the strong collision property.

This discussion implicitly requires a notion of difficulty, which ultimately is measured by how much work, like computer operations, are needed for a desired effect; this concept is similar to our notion of brute force for symmetric ciphers. If the hash is length n, then the question is how long does it take to find a collision pair? The answer turns out to be roughly $2^{n/2}$, which we derive next.

4.1.3 A Birthday Surprise

The problem of finding collisions relates to a well-used classroom example in probability and cryptography classes: how many people are needed to be queried before it is more likely than not that two people share the same birthday. The answer is surprisingly quite small, namely only 23 people are needed (small enough that this can be used as a classroom demonstration, although with varying success). The probability Prob(k) of a match among k random people is

$$\text{Prob}(k) = 1 - (364/365)(363/365)\ldots((365 - k + 1)/365) \tag{4.1}$$

Basically, we look at the probability of each new person not matching any of the previous people, and the desired probability is 1 minus that. Figure 4.2 shows a plot of the likelihood of obtaining a match. The solid line plots Prob(k). The circle is the point (23,0.5), which shows that it is more likely than not that among 23 people, two people will share a birthday.

The natural question is how to extend this simple calculation to the case of n-bit hashes. Equation (4.1) extends naturally to replacing 365 with 2^n. There is an easy-to-derive approximation[3] given by

$$\text{Prob}(k) \sim 1 - \exp\left(\frac{k(k-1)}{2N}\right) \tag{4.2}$$

where $N = 2^n$. Setting Prob(k) = 0.5 yields

$$k \sim \sqrt{2N\log(2)} \tag{4.3}$$

Thus, to get at least 50% success of finding a collision, about $\sqrt{N}$ items need to be checked. For an n-bit hash, this means searching $2^{n/2}$. This result yields hash strength, which is that the strength of an n-bit hash is taken to be $n/2$ bits.

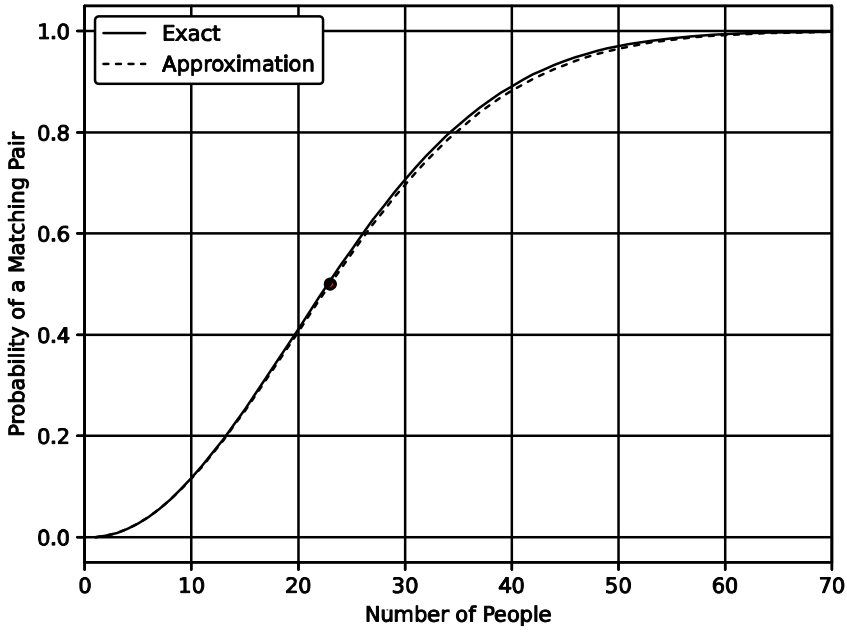

Figure 4.2 Probability of a birthday match.

Finally, notice that our calculation of Prob(k) is used to find the median of finding collisions. There are results about finding the mean of the searching, which is Ramanujan's Q-function; see [1].

4.1.4 Other Properties

There are two more properties of hash functions that we state for completeness. The first is that hash functions should exhibit the avalanche property, just as ciphers do. This property relates to the desired integrity of a message digest. Specifically, a single bit change in the input should yield hashes whose values are essentially independent of each other. The avalanche function is related to the collision properties, in the sense that if the avalanche property is violated (i.e., a small change yields hash values that are close), then that seemingly offers a more efficient path for finding collisions.

The second property is performance or efficiency. Hash functions form important building blocks of many protocols, and so their operation should be very efficient. As we see next with regard to constructions, many hash functions standards use repeated elementary operations that can exploit the usual instruction sets.

4.2 Construction

In the same spirit of the general constructions for ciphers in Chapter 3, there are some general constructions for cryptographic hashes. The first one we present is the Merkle-Damgård construction, shown in Figure 4.3, first suggested in [2]. The idea is that the input data, essentially of arbitrary length, is first padded to become an integral number of blocks for the hash functions. The pad is always added, even if the data was already an integral number of blocks. Each block serves as input to a compression function F. The first block gets an initialization vector defined by the algorithm; thereafter, each block receives the

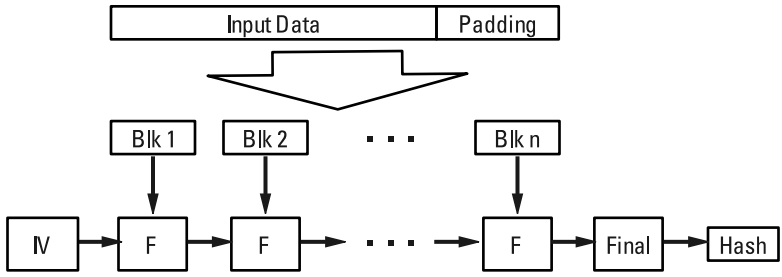

Figure 4.3 Merkle-Damgård hash function construction.

output of the proceeding call to F as its IV. After the last block, some finalization may be performed to yield the hash value.

Merkle [3] and Damgård [4] independently showed that if the compression function F is collision free, then so is the whole hash function. This trait thus reduces finding a good hash function to finding the function F. Contrast this approach with ciphers, which are defined for a single block and then a cipher mode must be chosen, with various choices and trade-offs, to encrypt a set of data. This construction is used in the SHA-2 family of hash functions discussed below.

The next general construction is the *sponge* construction, which is the basis of SHA-3 [5]. The idea of a sponge construction is that it takes in an arbitrary length input but can also produce an arbitrary length output. That trait means the construction can do more than just produce a hash. For example, a small input could be used to generate a long pseudorandom sequence using the same process that is used to produce a hash.

The general construction is shown in Figure 4.4. The input data is padded to create an integral number of r-bit blocks. The value r is called the *bit rate*; it relates to how much data is used at each step in the calculation. The value c is the *capacity*; this value is related to the strength of the process. Together the r-bit and c-bit quantities make up the state that is continually acted on. This action consists of XOR-ing the current block from the input data to the current r-bit portion of the state. The state is then transformed to a new $(r + c)$-bit state by the F function. The process begins with the all-zero state.

This process, called *absorbing* continues until all the input data blocks are used. At that point, the *squeezing* portion of the algorithm begins, hence the name "sponge construction." At each stage of the squeezing operation, the r-bit portion of the state serves as the output block. After the output, the state

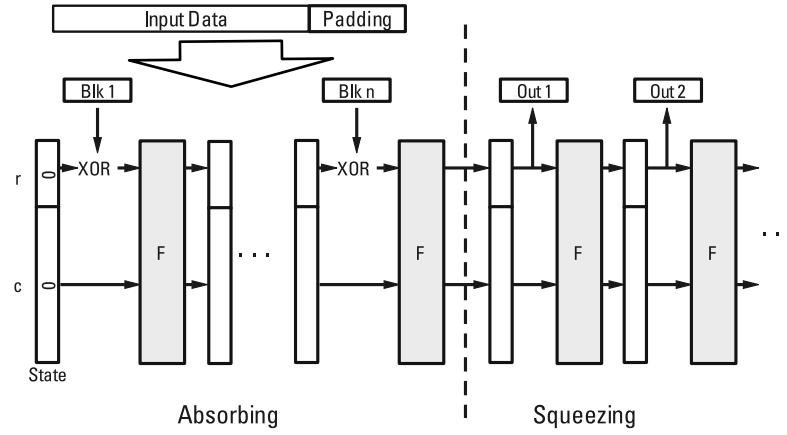

Figure 4.4 Sponge construction.

is updated again as before although there is no extra input. This process continues for as long as desired (i.e., a single hash value can be produced or a longer output stream). One virtue of this construction is the flexibility in the values of r and c, as we will see in the SHA-3 description.

Finally, the Merkle-Damgård construction does beg the question if the compression function can be made using ciphers. There have been several proposals, one of which is the Davies–Meyer compression function F that acts on a single block [6]. The construction is shown in Figure 4.5. The block represents the portion of the input to the overall hash function. It acts as the key to the single call of the encryption function (Enc). The values H_i are the outputs of encryption, with H_0 representing the IV of the process.

The interest in this construction is that it shows how one can leverage encryption functions to produce hashes. That said, there are some limitations for practical uses. Most modern ciphers have moderate-sized plaintext/ciphertext sizes, such as 128 bits for AES. A 128-bit hash is too small, since its level of protection is only 128/2 = 64 bits.

4.2.1 SHA-2 Family

Similar to the standardization of ciphers with first DES and then AES, NIST has standardized a family of SHA. The first example is SHA-0 (1993), which was quickly replaced with SHA-1 in 1995. SHA-1 uses a 160-bit hash and was the common choice of hash algorithms along with MD5 until both succumbed to serious flaws.[4] SHA-1 has been deprecated essentially since 2010 and should not be used.

The SHA-2 family is currently under use and recommended. It is a set of related algorithms with different parameter sizes, and the hash algorithms are denoted with the hash size (i.e., SHA-256, SHA-384, and SHA-512). They are defined together in NIST FIPS PUB 180-4 [7]. Since the strength of the hash is given as half the hash size, the three main SHA-2 algorithms have cryptographic strength equal to half the sizes of 256, 384, 512; that is, 128, 192, 256, respectively. Those strengths put them in line with the strength of the three flavors of AES.

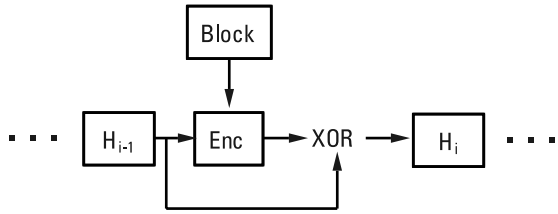

Figure 4.5 Davies-Meyer compression function construction.

The SHA-2 algorithms use a Merkle-Damgård construction in Figure 4.3. The block size varies by the algorithm. A padding is always provided even if the data is a multiple of the block size. The fundamental difference between the algorithms is with the compression function.

For concreteness we focus on SHA-512, which uses 1024-bit blocks. The compression function consists of the application of the procedure shown in Figure 4.6 applied over 80 rounds. The state is a set of eight registers, whose lengths depend on the hash size. For SHA-512, that means eight 64-bit registers. This state is updated through the 80 rounds.

Figure 4.6 shows several other variables and functions that use the registers:

- The *W*t 64-bit values are derived from the 1024-bit input for the block;
- The *K*t 64-bit values are the fractional parts of the cube roots of the first 80 prime numbers;
- Addition is regular 64-bit binary addition ignoring the carry; that is, modulo 2^{64};
- The Choose function uses the first input (*E*) to bit-wise choose either *F* or *G;*
- The Majority function is bitwise majority vote of the *A*, *B*, and *C* inputs;
- The two Sigma functions are the xor of three different circular shifts of the input register.

Notice that only two registers are changed in each round, while the other six registers are just shifts of the input registers. Our purpose in showing the compression function in some detail is to show how elementary the actual operations are.

As mentioned, the variation of the SHA-2 family allows one to tailor security and performance (see Table 4.1).

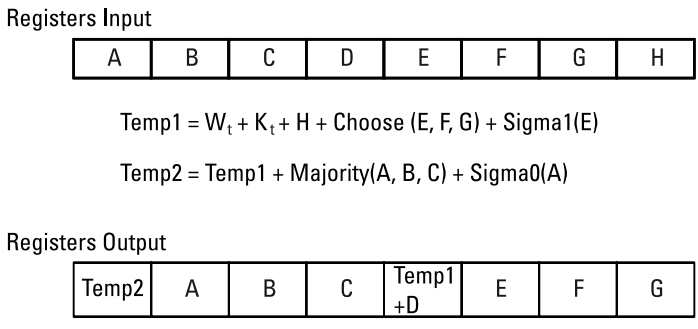

Figure 4.6 SHA-512 compression function.

Table 4.1
The SHA-2 Family of Hash Functions

Name	Hash Size	Block Size	State	Number of Rounds
SHA-1 (deprecated)	160	512	5×32	80
SHA-256	256	512	8×32	64
SHA-384	384	1024	8×64	80
SHA-512	512	1024	8×64	80

Table 4.2 shows a benchmark for the SHA-2 algorithms similar to the AES benchmarks in Table 3.4 using one of the author's laptop. Each algorithm is applied multiple times to an input given by the column headings. The table entries are the performance in terms of megabytes process per second. The important takeaway is how efficient the hash functions are, even for the larger hash sizes. In particular, the use of SHA-256 over SHA-1 is comparable and sometimes better.

4.2.2 SHA3

Because of the vulnerabilities in SHA-1, there were concerns in the early 2000s that the SHA-2 family may also have vulnerabilities. As such, NIST sponsored a hash function competition to create a hash function that was dissimilar to SHA-2. The initial submissions were accepted in 2008, and the winner, Keccak, was announced on October 2, 2012.

Keccak was designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche (note that Daemen also codesigned AES). As with Rijndael and AES, Keccak is synonymous with SHA-3. Note that no vulnerabilities have been found in SHA-2, and it is still recommended. SHA-3 provides an alternative hash function with tunable properties.

Keccak uses a sponge construction discussed above in Figure 4.4. The main permutation function is a little too involved to present here [5, 8]. Its computation only uses XOR, AND, and NOT operations. The main state is a

Table 4.2
SHA-2 Execution Time

	Data Size					
Hash	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes	16384 bytes
SHA-1 (deprecated)	304MB/s	886MB/s	1635MB/s	2187MB/s	2409MB/s	2455MB/s
SHA-256	307MB/s	753MB/s	1505MB/s	2469MB/s	3171MB/s	3395MB/s
SHA-512	75MB/s	368MB/s	505MB/s	842MB/s	961MB/s	986MB/s

$5 \times 5 \times 64$ array (1600 bits), arranged as a three-dimensional array of rows (x), columns (y), and lanes (z). An x-y section is a slice, an x-z section is a plane, and a y-z section is a sheet [5]. The permutation function, f, consists of various operations on subarrays. The only nonlinearity is an AND operation. The tunability comes from choosing the rate r and capacity c in Figure 4.4. Possibilities are shown in Table 4.3.

Table 4.4 shows a benchmark for some SHA-3 algorithms similar to the SHA-2 family in Table 4.2. Each algorithm is applied multiple times to an input given by the column headings. The table entries are the performance in terms of megabytes processed per second. The important takeaway is that SHA-3 is also efficient with less throughput variation due to input size. However, it is often slower than comparable SHA-2 hash functions.

4.3 Message Authentication Codes

Cryptographic hash functions are defined without using a key. Hence, anyone given the input data can compute the hash. This trait is an important feature, as it allows for using hashes as a fingerprint that is uniquely and securely bound to the data up to the computational strength of the hash, yet anyone can verify the fingerprint. In this way, hashes give *integrity* of the data, based on the various properties prescribed in Section 4.1.

Related to integrity is the notion of authentication, which is the desire to state that not only has the data not been changed, but in fact it was generated by a specific party. One approach for authentication is a MAC, which uses cryptographic hash functions along with a shared key (the other approach is to use

Table 4.3
SHA-3 Hash Function Options

Name	Hash Size	Rate r	Capacity c
SHA3-256	256	1088	512
SHA3-384	384	832	768
SHA3-512	512	576	1024

Table 4.4
SHA-3 Execution Time

	Data Size					
Hash	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes	16384 bytes
SHA3-256	31MB/s	125MB/s	323MB/s	415MB/s	5001MB/s	5105MB/s
SHA3-512	311MB/s	126MB/s	186MB/s	199MB/s	220MB/s	216MB/s

a digital signature, which we discuss in Chapter 5). Note the unfortunate use of the term MAC, which should not be confused with the widely used "media access control"; that latter concept will not be needed much in this book and will be called out explicitly when it is used.

The general view of a MAC is shown in Figure 4.7. The generation is similar to the generation of a cryptographic hash as in Figure 4.1, except that a shared key is used in generation. To verify a MAC, the exact same generation procedure needs to be performed to obtain a version of the MAC that is compared against the received version. This generation needs to have access to the same key. If the two MACs match (completely), then one has confirmed not only integrity of the data but that someone in possession of the key generated the MAC. Thus, we obtain authentication up to those parties that share the key.

One virtue of using a MAC is that only a portion of the MAC needs to be used for the procedure. For example, one could use only the lower 128 bits of a 512-bit hash as the MAC. The verifier computes the whole 512-bit MAC, and then just checks the requisite 128 bits. This flexibility is useful in various protocol settings where bit lengths need to be tunable. This trait is not possible with digital signatures. Of course, smaller MACs would be more vulnerable to birthday-type attacks.

4.3.1 What Doesn't Work

We mention two approaches for using a key with a hash that may not work. First, consider the procedure

$$\text{MAC} = \text{Hash}\left(\text{key} \| \text{data}\right) \tag{4.4}$$

The issue is that depending on the hash function, it will be possible to extend the data and obtain a legitimate MAC without knowing the key. This possibility then violates the authentication goal that we want the MAC to achieve.

The second approach that does not work is

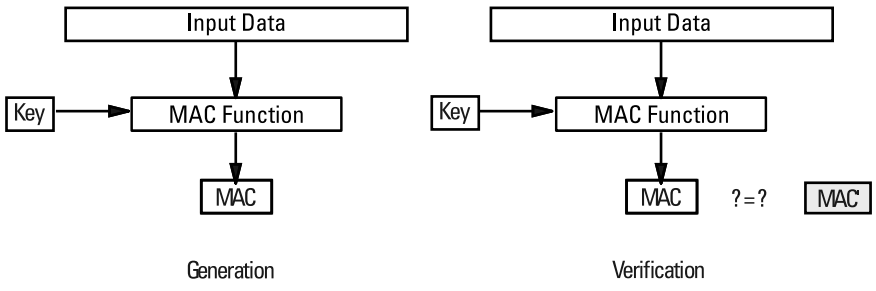

Figure 4.7 General message authentication code.

$$\text{MAC} = \text{Hash}\left(\text{data} \| \text{key}\right) \tag{4.5}$$

This example is not prone to length extension, because the key is used last. The issue now is one of collisions. If data_1 and data_2 yield collisions when hashed, then they may yield collisions when the MACs are computed, again depending on the hash function. That trait again violates the desired security goals.

We mention these two examples to show that sometimes seemingly obvious methods to extend cryptographic functions may be very vulnerable.

4.3.2 Hash-Based Message Authentication Code

Fortunately, HMAC is a method that is almost as easy to state as the previous two failed attempts. It is defined for any cryptographic hash function. This algorithm dates from 1996 and is standardized in [9]. It is very secure as long as appropriate key sizes and hash algorithms are used.

The algorithm is simply given as

$$\text{HMAC} = \text{Hash}\left(\left(K^{+} \oplus opad\right) \| \text{Hash}\left(\left(K^{+} \oplus ipad\right) \| \text{data}\right)\right) \tag{4.6}$$

Here K^{+} is the shared key extended with zeros (or hashed and truncated) to get to the hash function block size. The values ipad or opad are, respectively, the bytes 0x5c and 0x36 repeated to the block size. The hash function is versatile, but in practice is SHA-2 or SHA-3.

HMAC only requires that the hash function be called three more times on an input of the block size than computing just Hash(data). That extra computation is not meaningful for large data sets, and even for small data sets, the efficiency of the hash functions means that HMAC can be used without worrying about performance.

4.4 Ciphers and Authentication

We have seen ciphers as a possible way to create a hash function, basically by implementing the necessary compression function. In this section, we further explore the relationship of ciphers and hash by using ciphers to create MACs and achieving authentication. We first look at cipher modes that are designed to create MACs; this field is perhaps surprisingly dynamic. Second, we discuss the concept of an authenticated cipher, which combines both confidentiality and authentication and offers a brief look at Ascon, the new standard for authenticated ciphers.

4.4.1 Cipher Modes

Recall the CBC cipher mode from Section 3.7. A natural thought for obtaining a MAC is to use the last block of the cipher calculation. This process is shown in Figure 4.8. The input is broken up into a set number of blocks of a size equal to the plaintext/ciphertext block sizes. Starting with an IV of all zeros, each successive ciphertext is xor-ed into the next block of input. The MAC is the last block, and this process is called CBC-MAC. The motivation for CBC-MAC is that the properties of a cipher should ensure the cryptographic hash goals. For example, preimage resistance is precisely the goal of not being able to infer anything about the plaintext from the ciphertext.

The issue is that authentication goals for the MAC may be violated without breaking the cipher. For example, consider data that is one block long, so that

$$\text{CBCMAC}_K\left[\text{msg}\right] = E_K\left[\text{msg}\right] = M \tag{4.7}$$

Then it is easy to show that we can create a 2-block long message with the same MAC, namely

$$\text{CBCMAC}_K\left[\text{msg} \| M \oplus \text{msg}\right] = M \tag{4.8}$$

This fact shows that one can forge the MACs for some messages without knowing the key.

Methods that are based on CBC-MAC can be made to work, and we next show two cipher mode methods that do work: CTR with CBC-MAC (CCM) and Galois counter mode (GCM). These are examples of authenticated

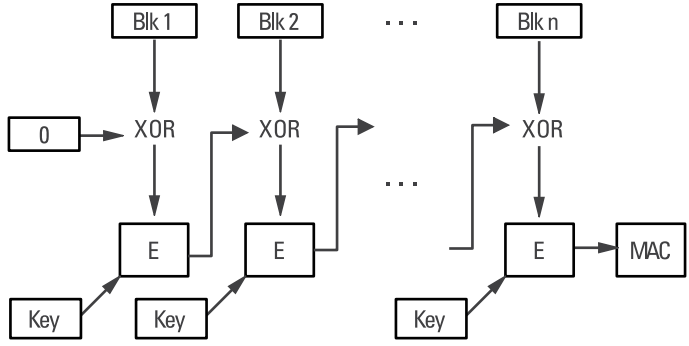

Figure 4.8 CBC cipher mode and a possible MAC.

encryption with associated data (AEAD), which allows for checking the authentication of both the ciphertext and associated data. That said, the above examples show that creating cipher modes that will also yield authentication is subtle, and second, that one should refrain from creating such solutions themselves.

4.4.2 Counter Mode with CBC-MAC

The first cipher mode geared to authentication is counter mode with cipher-block-chaining message authentication code (CTR with CBC-MAC, known as CCM). This mode was developed for wireless standards (IEEE 802.11–Wi-Fi) and submitted to the IEEE and NIST in 2002/2003 [10, 11]. It requires 128-bit plaintext blocks, and so is defined mainly for AES. The total number of uses under the same key is limited to 2^{61} bytes of associated data and plaintext. See the specification [12].

The mode both encrypts and authenticates. The plaintext is encrypted with CTR mode. The blocks for the key stream use various flags, a nonce, and a counter. The resulting cipher text is obtained by xor with the key stream.

For authentication, an optional amount of associated data is prefixed to the input. This associated data is sent in the clear. The combined associated data and input are then encrypted using CBC mode. The final block is the MAC, which can be truncated to a desired length smaller than 128 if need be. Decryption and verifying the MAC follow from CTR-mode decryption and regenerating the MAC and comparing.

The algorithm requires two encryptions for each block of ciphertext generated; it is not parallelizable in either direction. It is provably secure under the restriction of the number of blocks used [13]. Note that CCM does not share the vulnerability that we mentioned with CBC-MAC above because of the nonce prepending that is required.

Pros for CCM:

- Demonstrable security proof.

Cons for CCM:

- Not parallelizable;
- Inefficient, two ciphertexts are generated per plaintext;
- Complex details involving bit manipulation for the lengths of fields.

4.4.3 Galois Counter Mode

Galois counter mode is another cipher mode that combines encryption and authentication. GCM works by encrypting the data using CTR. In addition, the ciphertext blocks from the CTR encryption along with possibly some prepended associated data are chained using a function that treats the input as an element in GF(2^{128}) (i.e., as a 128-bit binary polynomial). The output for that process is the MAC. If only the MAC is desired, the method is called GMAC. For a detailed explanation, see [8], and for the NIST specification, see [14]. GCM is thus similar to CCM, except for the addition of the Galois field operation.

There is a concrete security proof for AES-GCM [15]. The lower bounds of the security proof have been demonstrated [16]. These bounds suggest issues when smaller tags are used. The restriction on smaller tags is a problem for applications such as IoT or satellite navigation, where data packets tend to be small, and large tags represent increased overhead.

The NIST recommended algorithm is AES-GCM [14]. Careful consideration should be used in the choice of the IV/Nonce, and NIST seems to be revising specifications to provide guidance involving IV/Nonces selections [17]. One effort to reduce the sensitivity to the initialization vector is called GCM-SIV for the *synthetic initialization vector*; the idea is to protect against possible defects in the random nonce generation. In the long run, it does seem like GCM-SIV should be preferred over GCM.

Pros for GCM:

- Parallelizable for both encryption and decryption;
- Fast operations for the authentication portion;
- There is a concrete security argument for AES-GCM [15];
- NIST standard available [14];
- Implementations are available in Internet Protocols (e.g., TLS 1.3).

Cons for GCM:

- Weak keys, if $E_{Key}(0^{128}) = 0$ then the authentication breaks down;
- Nonce reuse allows for various attacks;
- Size limitation of 64 GB;
- The lower bounds of the security proof have been demonstrated by Ferguson [16], and Ferguson's argument indicates that smaller tags are not good for AES-GCM;

- The security argument limits the number of blocks to be processed to 2^{32} if random values are used for the nonce.

4.4.4 Authenticated Ciphers

There has been a strong desire to have ciphers that have authentication essentially built in authenticated encryption. There are many approaches; see [18] for a comparison of six different modes for authenticated encryption. An international competition was created to find such ciphers, called Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR). Their focus is on three categories: lightweight applications, high-performance applications, and defense-in-depth. This section discusses *Ascon*, which has been selected by NIST [19, 20] as a lightweight cipher with AEAD.

Ascon is a substitution-permutation network based on the sponge construction introduced by the SHA-3 family of hashes. Ascon acts on a 320-bit state; when the state is represented as a column vector S, it has two parts, S_r (the "rate" portion) and S_c (the capacity portion).

The rate portion interacts with the input blocks: plaintext or associated data. For this reason, the rate portion is often referred to as the outer state, since it faces out to the input. The capacity portion holds the residual randomness, and is often referred to as the inner state. The Ascon's authors' recommendation for blocksize (S_r) is either 64 or 128 bits.

Pros for Ascon (adapted from [21]):

- Authenticated encryption and hashing using a single lightweight permutation;
- Sponge-based modes of operation with a custom-tailored permutation;
- Provably secure mode with keyed finalization for additional robustness;
- Easy to implement in software and hardware;
- Lightweight for constrained devices: small state, simple permutation, robust mode;
- Fast in hardware and software;
- Scalable for more conservative security or higher throughput;
- Key size equals tag size equal the security level: 128 bits recommended.

Cons for Ascon:

- There are not many cons to this new algorithm. As it is implemented, some may emerge.

4.5 Application: Digital Fingerprints

The rest of this chapter is devoted to applications. These applications illustrate the usefulness of the various properties of cryptographic hash functions. The first application is using the hash as a digital fingerprint and related concepts. As Figure 4.1 indicates, the hash value somehow distills all the information in the input.

Noncryptographic hashes can also perform this distillation. For example, consider a CRC or a series of parity checks.[1] The result can identify the input in the sense that say a single bit error is detected. A similar example is hash functions used to insert a longer record into a database. There, the hash value distills the record enough that it is hoped it is unlikely that two records hash to the same value. These examples show how even noncryptographic hashes may be able give some indication of modification in the input (e.g., an error in the case of some error detection method). The issue is that the methods won't indicate all such modifications. If we desire more absolute integrity goals, we need a cryptographic hash.

4.5.1 Message Digest

When a cryptographic hash function is used as in Figure 4.1, the result is often called a message digest of the input. The first widely used message digest was MD5 designed by Rivest [22] (MD stands for "message digest").[4] SHA-1 was widely used, but that hash function likewise has problems. Currently, the SHA-2 family is commonly used, with SHA-256, a 32-byte hash a specific choice.

To use a message digest to attest integrity means that one trusts the source of the message digest. For example, the integrity of a software repository can be determined by checking the message digest if one is sure of the source of the message digest. Often, message digests from trusted sources are used to verify the integrity of the information on other sites.

4.5.2 Secure Verification

The verification allowed from using cryptographic hash functions has an important application in the protection of passwords. A system needs to store the shared secret—the password—of its users. Clearly it would be bad to store this information in the clear. Instead, early on it was decided to store a hash of the password, since knowing the hash would not yield information about the password from the preimage property.

But this naïve approach has problems. If two users share a password, then their hashes would be the same, and thus each would learn the password of the other, especially since password files may not be otherwise protected. In

addition, if one started computing the hashes of all possible passwords, then one could just look up the hash value in the constructed list. This is a *dictionary attack*; these attacks can be made more efficient using *rainbow tables.*[5]

The approach is improved by adding salt. Salt is information combined with the password, say prepended. Random salt would mean that a dictionary attack becomes problematic, assuming the salt is known only to the storage system. In addition, a separate piece of salt should be used for each user to ensure that no two stored hashes are the same. Note the implicit reliance on the hash properties (i.e., the collision and avalanche properties).

4.5.3 Watermarking

The concept of fingerprint extends to other media. It is this extension that is most relevant to the goals of navigation. In this case, the fingerprints are commonly called *watermarks*. The idea of the watermark is to embed the fingerprint in the information. Sometimes we desire the watermark to be discernible, while at other times the watermark may only be available to someone with hidden knowledge; recall the discussion of steganography in Section 3.8. As with fingerprints, the main goal is to obtain integrity. Authenticity is also achieved if there is assurance as to who could have inserted the watermark.

The most familiar watermark to most people is on currency. For instance, the U.S. $5 bill has a portrait of Abraham Lincoln, and to the right, on the face of the bill, a ghostly image impressed into the bill of the same portrait. U.S. currency, since before 2000, has had these watermarks on bills higher than $5 and in many denominations earlier. One presumes that this makes it difficult to bleach the currency fabric and reprint it with a new, higher, denomination or as a simple detection device, as this feature is hard to reproduce. Authenticity is derived because the material is U.S. currency paper, and integrity is that the denomination is correct.[6]

As mentioned, some authors add requirements that the watermark be imperceptible or integral to the integrity of the cover-work. In trying to defeat intellectual property theft, these goals seem natural, but the most basic example (e.g., money) shows that these are added requirements of an application rather than integral to the application. For example, there are watermarks in digital media to prevent DVDs from being pirated and sold outside legitimate channels. These are almost always detectable, and the cat and mouse game against pirates is ongoing (a nice discussion of this can be found in [23]).

The challenge is that often if a watermark is known, it is vulnerable to removal. Overcoming this challenge is central for watermarking navigation signals. That said, in the context of navigation signals, detecting tampering with a signal is a goal of watermarking, and so a watermark that is removed has still done its job.

4.6 Application: Chains, Trees, and Blockchain

A cryptographic hash function abstractly is a function $H(x) = y$, where x is the input (or arbitrary length) and y is the hash or a truncation of the hash. A very useful building block in many protocols is the repeated application of this function, such as, $H(H(x)),H(H(H(x)))$. Such an application creates a *hash chain*. Furthermore, this concept can be extended to consider applications like $H(H(x_1) \parallel H(x_2))$, and so forth. This idea leads to *Merkle trees* or *hash trees*. Finally, we give a brief introduction to *blockchain* and the roles hash chains play.

4.6.1 Hash Chains

As stated, a hash chain is formed by repeated application of the hash function H. Suppose we begin with a value x_0, and form a chain of values with $x_i = H(x_{i-1})$, for $1 \leq i \leq n$. This chain can be represented graphically as

$$x_0 \xrightarrow{H} x_1 \xrightarrow{H} \ldots \xrightarrow{H} x_n \tag{4.9}$$

The arrows represent the application of the hash function. The arrows are necessarily one-way, by the preimage resistance property of the hash function. That is, no information can be gained backwards. Often the H is omitted from the notation if it is clear from context. Hash chains have some notable properties:

- Knowing any x_i allows the computation of any x_j with $j > i$.
- If we are assured of the validity of any x_i, then we are assured of the validity of the whole chain. For $j > i$, this assurance follows from the first bullet. For $j < i$, note that the collision resistance properties of the hash function means that it is highly unlikely that another possible value for the antecedent could have been found, and thus by induction all of the previous x_j are validated.

One example of how hash chains could be used is as one-time-use passwords. Suppose that a user generates a hash chain as in (4.9). The user gives x_n to a system that they wish to access. To access, they provide x_{n-1}, and the system confirms that $H(x_{n-1}) = x_n$. Once validated, the system discards x_n and stores x_{n-1} for the next usage, which will require the user to furnish x_{n-2}, and so on. Thus the user has n uses of the system using a credential that is good for only one time. This example also is another illustration of bit commitment, discussed in Section 4.7.

4.6.2 Merkle Trees

The chain idea extends to a treelike structure as shown in Figure 4.9. Merkle patented the concept in 1979 (U.S. patent number 4309569A). The central idea is to divide the data of interest into separate chunks (i.e., $X_0,...X_7$ in the figure and in general 2^n). The leaf nodes of the trees get the value $H(x_0)$ through $H(x_7)$. The remaining nodes are defined using

$$\text{parent} = H\left(H\left(\text{child}\right)_L \middle\| H\left(\text{child}\right)_R\right) \tag{4.10}$$

That is, in the figure, H_{01} is the hash of $H(x_0)$ and $H(x_1)$ concatenated together. The process continues until the root of the tree is reached. The usefulness of a Merkle tree lies in the fact that the hash of the root node ensures the integrity of all the data: any change in an x_i changes all the hashes of its parents up the tree.

To create the hash tree, there is the overhead of having to compute double the number of hashes. However, that overhead yields a useful property, namely that if one of the x_i values changes, only the hashes in the tree that it influences need be updated. That number is only the depth of the tree (i.e., the log of the number of leaf nodes). In addition, the hash tree can be computed in a distributed manner. That is, if two separate trees need to be combined, only the respective root nodes need be hashed together to yield the root of the new larger tree.

4.6.3 Blockchain and Cryptocurrency

One important use of Merkle trees is in blockchains. Indeed, for cryptographers, blockchain is basically the use of Merkle trees, which have been around

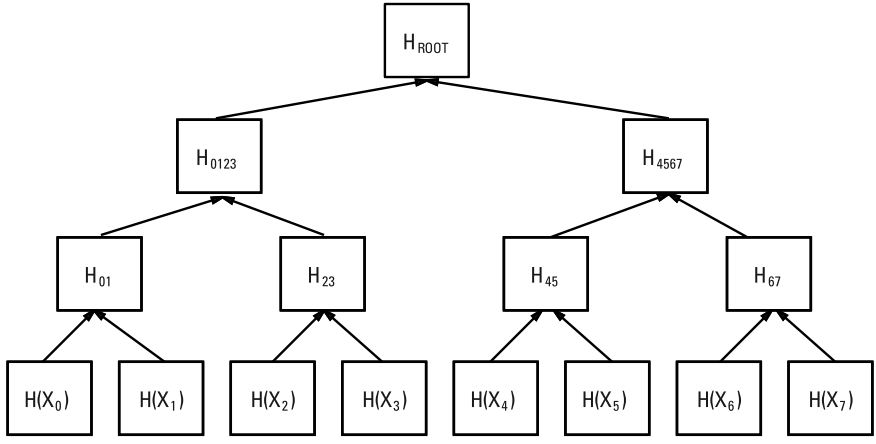

Figure 4.9 Merkle tree construction.

since 1979. Their current fame is their use in cryptocurrency applications, but the core cryptographic feature is straightforward. Our discussion focuses on that use.

One goal of any cryptocurrency is to be able to record all transactions in a manner that can be verified while still obfuscating information. That is precisely the use of a hash function. This property is the creation of a secure ledger, secure in the sense that once a transaction is recorded, it can't be changed. The ledger is often distributed, in that records of separate (families of) transactions can be combined together. In bitcoin [24], the ledger for a given block of transactions is frozen, and then combined with the next block of transactions. The Merkle tree allows all of the transactions in a given block to be summarized as the root of the tree, which is then combined with information like the nonce mentioned below.

The above discussion explains blockchain, but does not touch all that is needed for cryptocurrency. A full discussion is out of the scope of this book, but we mention two important aspects. First, bitcoin introduced the concept of *mining* to allow for decentralized creation of the blocks in the ledger that are provably correct. This trait is accomplished by forcing the final hash value to have a special property, say ending in m 0s. By random chance, it would take on average 2^m applications of a hash function to find such a hash value. Those applications are done by modifying a nonce that is combined with the hashes of the ledger. If m is large enough, this computation requires a lot of work, and furthermore, it becomes unlikely that two entities will produce the requisite hash. Thus, mining is a way to show proof-of work, and the entity that performs the work and expends the necessary energy doing so receives a reward. But we stress that one can have blockchains (i.e., secure ledgers) without the need for mining.

Second, our discussion does not talk about what a transaction is: how one transfers an asset. That process involves using public key information (see Chapter 5) to indicate ownership.

Does blockchain have an application to satellite navigation? There may be uses in the enterprise, but probably not in the navigation signals themselves.

4.7 Application: Bit Commitment

The preimage property of cryptographic hashes enables a cryptographic protocol primitive of importance called *bit commitment*. We saw an example in the previous section, where knowing the current stored hash value allowed one to furnish the input to the hash as a password. This example shows the basic concept: knowing the hash of data reveals nothing about the data. Furthermore,

because of collision properties, knowing the hash also commits the data, because any change in the data would result in a different hash. Bit commitment then is the idea of revealing some information that is later confirmed when more information is revealed (see[7] for a noncryptographic example).

Our first example is a literal bit commitment, namely a way to digitally toss a coin. One immediate idea would be for Alice to send Bob a hash of either 1 or 0 (heads or tails). Alice can then tell Bob which she picked, and Bob can confirm. The issue with this procedure, or protocol, is that there are only two possible hash values, and so Bob could just precompute them. A better procedure is for Alice to choose a nonce and compute either Hash(nonce || 0) or Hash(nonce || 1). Afterward she reveals both the nonce and her choice. This procedure can be refined easily by having both parties choose values and commit to each other, and so forth.

This simple example can be extended in a few ways. For example, suppose a file needs to be submitted by a certain time, but there is not enough bandwidth to submit the whole file. Then submitting the hash of the file commits the full file while being easier to transmit. Relatedly, bit commitment could be used in secret auctions, say for a contract bid. Here, the file would consist of the bid. As with the coin example, nonces could be used so that someone can't precompute all possible bid values.

The use of bit commitment is important in some navigation protocols, such as Chimera and Timed Efficient Stream Loss-Tolerant Authentication (TESLA). TESLA as a general protocol is discussed next.

4.8 Application: TESLA

The last application that we present is the *TESLA* protocol. TESLA is a method for authenticating a stream of data messages. TESLA dates from the early 2000s [25, 26]. Its motivation comes from the desire to authenticate a stream of information like network packets (video stream, stock ticker, etc.), where some packets may fail to arrive. Additionally, this authentication should be as efficient as possible.

The main cryptographic components of TESLA are hash-based, which is why we present the protocol here. The protocol also uses digital signatures to achieve complete authentication. Digital signatures are described in the next chapter; for our purposes here we just use the fact that a digital signature can be verified by anyone with the right information, similar to a MAC, but only one entity can create the signature, unlike a MAC. Digital signatures are much more computationally inefficient than MACs, which is why TESLA was created as opposed to just digitally signing each message.

4.8.1 Main Idea

The overall structure of TESLA is shown in Figure 4.10. TESLA comprises four main parts:

- *A Stream of Data Messages.* We assume that the data messages are in a well-defined sequence.
- *A Hash Chain of Keys.* A sequence of keys is generated using a hash chain.
- *Message Authentication through MACs.* Each message is authenticated by a MAC, where the key used is revealed in the next message.
- *Occasional Digital Signature.* A digital signature is used to sign some message or messages.

The following sections elaborate on these components. The protocol is described initially from the perspective of the creator of the TESLA protocol (e.g., the transmitter). The actions of the recipient are mentioned where appropriate.

4.8.2 Data Stream

The stream of data messages is well-defined, in the sense the protocol needs to know the number of data messages. The content does not need to be known until the messages are transmitted. Let N be the number of messages.

No assumption is made about the formatting, except that it should be clear where a received message falls in the sequence of messages. Our illustration will imply that the messages are the same sizes, but there is nothing to prescribe

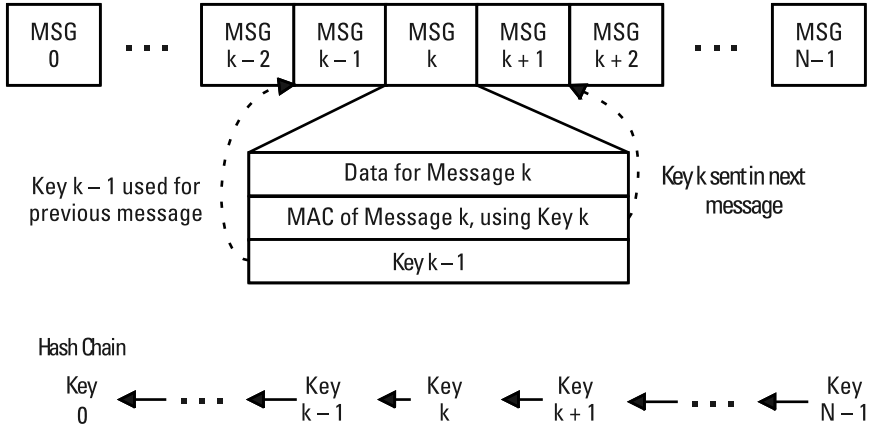

Figure 4.10 Overview of the TESLA protocol.

that. There should be room in the message to accommodate the MAC and key information shown in Figure 4.10.

4.8.3 Hash Chain of Keys

A sequence of keys is generated using a cryptographic hash method H. The keys are denoted by

$$K_{N-1} \xrightarrow{H} K_{N-2} \xrightarrow{H} \ldots \xrightarrow{H} K_0 \tag{4.11}$$

The key K_0 is called the *TESLA root key*. The property that any K_i can be derived from K_j if $j \geq i$ will be important. In particular, knowing any key allows one to compute the TESLA root key. The key K_{N-1} is the *TESLA start key*, since it is the initial key in the chain.

4.8.4 Message Stream and MACs

The heart of the TESLA protocol is in the authentication of the messages using a MAC. We present the general construction first and then handle the edge cases. In Figure 4.10, a generic message with index i gets two pieces of information:

- A MAC of the message using the key K_i. At this point, only the transmitter knows K_i.
- The key K_{i-1}, which was used for the MAC in the previous message.

On reception of a message, a receiver can use the added information as the following:

- Verify the contents of message indexed $i - 1$ using key K_{i-1}.
- Save the contents of message index i until K_i is sent in the next message.

There are two edge cases that need to be defined further. For message index 0, it will contain the MAC using the key K_0, but the key that it includes needs to be the key K_{N-1} for the previous TESLA chain. For the last message index N, the key K_{N-1} it uses for its MAC must be sent in the first message index 0 in the next TESLA chain.

4.8.5 Authentication via Digital Signatures

The chain of MACs with later-revealed keys that form the TESLA protocol serve to authenticate all the data messages, with the slight delay of waiting for

the next message. However, that does not prevent someone from creating a whole fake TESLA chain and message stream. To combat that issue, TESLA requires that some K_i be digitally signed by the entity who is creating the TESLA chain. It does not matter which K_i is signed, since that will authenticate the whole hash chain, and thus all the messages, by the discussion on hash chains above.

4.8.6 Properties

We outline the properties of TESLA in terms of its name:

- *Timed:* The security of the TESLA protocol depends on the bit commitment inherent in the later reveal of the key used for a MAC. If this key were to appear before the message, then anyone who learned that key could create a fake message with a valid MAC. Thus the security proof in [24] is based on the prescribed delay between messages. To use this protocol in practice, such a constraint on messages needs to be observed or otherwise dealt with, for example, by ignoring a message if it arrives after another.

- *Efficient:* The use of MACs instead of digital signatures allows for a better use of data for authentication. Not only are fewer bits needed, often at least by a factor of two, but MAC algorithms are much more computationally efficient than digital signatures. We show benchmark results in Chapter 5.

- *Stream:* TESLA authenticates a whole stream of messages, not just a single one. The interdependencies means that an adversary cannot focus on a single message for spoofing, or repeat messages.

- *Loss-Tolerant:* Authenticating a whole stream of messages would be problematic if, say, we were to send a single MAC for the whole stream, because the loss of any one message would deter the verification of the MAC. Instead, TESLA tolerates the loss of a message by allowing for later messages to still yield the keying information needed to authenticate through the hash chain of keys.

- *Authentication:* TESLA lives up to its purpose: every message is potentially authenticated, both by the MACs and ultimately by the signature of the transmitter that validates the whole stream.

4.9 Foreshadowing: Navigation

Hashes and similar constructs have widespread use in satellite navigation because of their emphasis on integrity and authentication. We discuss these uses more thoroughly in Part III. For example, any authentication of data ultimately uses a cryptographic hash. In some cases, this is through the use of MACs, as in using TESLA. At other times, authentication comes from using digital signatures, and cryptographic hashes are a core piece of signing algorithms, as we'll see in the next chapter.

Signal authentication is not as straightforward as data authentication. When the transmitter and receiver share a key, then ciphers can be used as mentioned in Chapter 3 to create encrypted spreading codes for CDMA (see Chapter 9). But when no shared key is practical, as is the case with open services, then methods based on bit commitment are useful. One example is Chimera (Chapter 10), where the spreading code features are added to a signal using a post application revealed key. Such revealing is, of course, how TESLA also functions.

Our final example is more in the spirit of how hashes are used as digital fingerprints. The goal is to have something analogous to a identifying a navigation radio transmitter (radio fingerprinting), which is called specific emitter identification (SEI). Most of the techniques for identifying a particular radio transmitter, are based on the peculiarities of the analog portions of the transmitter. These methods date at least to WWII. Some of these include the identification of Morse code operators by their "style."

The techniques are based on the subtle and occasionally not-so-subtle variations of each transmitter [27, 28]. The issue with this technique for satellite navigation is not that the transmitters don't have these features, but as was discussed in Chapter 2, the reception of navigation signals uses correlation (see Section 2.6) and this averaging tends to wipe out the fine features needed for fingerprinting. Furthermore, SEI techniques tend to be computationally intense [29], which is not a useful for low power user equipment. The goal of SEI leaves open the design of artifacts that can be injected into the signal that could be easily detected, yet at the same time, they must not be easy to replicate. We discuss one possibility in Section 9.4.3.

4.10 Takeaways

Cryptographic hashes are an important workhorse for cryptography because they are secure one-way functions that are computationally efficient. Such functions are secure in the sense that no preimage can be efficiently found given the hash value, and that finding collisions—two inputs that produce the same

hash value—is likewise difficult. The strength of an n-bit hash is $n/2$ bits, which serves as the analogy to the brute force strength of an n-bit cipher. A hash of the input data gives the integrity of that data; any modification of the input will yield a hash value that only agrees randomly with the original hash. As such, hashes can be used as digital fingerprints.

Cryptographic hash functions can be used to create MACs when it is possible to share a key. MACs give the same integrity benefits as hashes but also give authentication, because only someone who possesses the shared key could have created the MAC. Even when shared keys are problematic from a user base perspective, methods that use bit commitment, such as TESLA, remain viable.

Repeated application of a hash function, in a chain or a tree, allow for succinct integrity measures of a whole collection of data. Examples include the use within cryptocurrency as the secure ledger and in TESLA, where the hash chain allows for authenticating the stream of data.

End Notes

1. A single parity bit of a binary input is a simple way to reduce all the data to a small value. Collecting several parity bits could serve to actually identify the input data. The issue is that such a collection of parities is not like a fingerprint, it is more like a label. It is relatively easy to construct different inputs with the same collection of parities. Indeed, forward error correcting codes can correct some errors, but if too many errors occur, there is ambiguity as to which input produced the parities.

2. The idea behind universal hashing [30] is to have simple, nonsecure functions such as linear functions, that is, parity checks be leveraged in a secure manner. The way that is accomplished is to choose the set of simple functions randomly, say using a nonce. In that way, a new hash function could be used for each message, while getting the benefit of easy-to-apply functions. Universal hashing has an advantageous feature in that the amount of entropy leakage can be rigorously proved, which was used in the early development of quantum key distribution algorithms [31].

3. We use the easy-to-see approximation that $1 - x \le e^{-x}$. Therefore $1 - k/365 \le \exp(-k/365)$. Equation (4.1) becomes

$$\text{Prob}(k) = 1 - \prod_{j=1}^{k-1} (1 - j/365) \tag{4.12}$$

$$\ge 1 - \prod_{j=1}^{k-1} \exp(-j/365) \tag{4.13}$$

$$= 1 - \exp\left(-\left(k(k-1)/(2 \cdot 365)\right)\right) \tag{4.14}$$

   Using $k = 23$ yields $\text{Prob}(k) \ge 0.5$. The result generalizes by changing 365 to any N, from which it follows that the point where there is a 50% likelihood of having a match goes as $\sqrt{N}$.

4. The MD5 algorithm is a message-digest (MD) algorithm designed by Ron Rivest in 1991 [21]. It creates a 128-bit hash and was widely used until collision attacks were made ef-

ficient in the mid-2000s. Indeed, finding MD5 collisions is so easy that it is a homework problem in [32].

5. A rainbow table, in general, is a precomputation of the outputs of a cryptographic process, usually a hash table. Consider the problem of guessing passwords from their hashes. Rather than just compute all possibilities, a table can be constructed using a space-time trade-off, where a secondary function is used to reduce a hash value to a possible password, which is then hashed, and reduced, and so on, resulting in a chain. If the start and endpoint of the chains are stored, then any given hash can be looked up by recreating the chain it is in. The "rainbows" aspect comes from having different colors of chains to be more efficient in avoiding collisions in the search. See [33] for the initial take on rainbow tables.

6. Currencies often have many security devices in them, for example, the United States [34]. One interesting trade-off is that the $1 and $2 U.S. bills don't have many of the more sophisticated security features in them, such as, metallic strips, metallic printing, and anticopy patterns.

7. Another example of bit-commitment is the Guy Fawkes protocol given in [35]. The idea is that a criminal could make sure to get credit for a crime by asserting that they committed the crime in a message and publishing the hash of the message ahead a time. Then, when the crime is committed, the message is revealed. In this way, no information of what crime is going to occur happens ahead of time, but the claim is still verified.

References

[1] Flajolet, P., et al., "On Ramanujan's Q-Function, *Journal of Computational and Applied Mathematics,* Vol. 58, No. 1, 1995, pp. 103–116, doi: https://doi.org/10.1016/0377-0427(93)E0258-N, https://www.sciencedirect. com/science/article/pii/0377042793E0258N.

[2] Merkle, R. C., "Secrecy, Authentication, and Public Key Systems," PhD thesis, Stanford, CA: Stanford University, 1979.

[3] Merkle, R. C., "A Certified Digital Signature," in *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '89),* Berlin: Springer-Verlag, 1989, pp. 218–238.

[4] Ivan Bjerre Damgård, "A Design Principle for Hash Functions," in *Advances in Cryptology — CRYPTO' 89 Proceedings,* Gilles Brassard (ed.), New York, NY: Springer, 1990, pp. 416–427.

[5] National Institute of Standards and Technology, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions,* Federal Information Processing Standard (FIPS) 202, U.S. Department of Commerce, August 4, 2015, doi:10.6028/NIST.FIPS.202, https://csrc.nist.gov/publications/detail/fips/202/final.

[6] Menezes, A. J., P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography,* Boca Raton, FL: CRC Press, 2001, http://www.cacr.math.uwaterloo.ca/hac/.

[7] Information Technology Laboratory Computer Security Division, *Decision to Revise FIPS 180-4, Secure Hash Standard—CSRC*, CSRC—NIST, March 7, 2023, https://csrc.nist.gov/news/2023/decision-to-revise-fips-180-4.

[8] Oppliger, R., *Cryptography 101: From Theory to Practice,* Norwood, MA: Artech House, 2021.

[9] Krawczyk, H., M. Bellare, and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, RFC 2104, February 1997, doi:10.17487/RFC2104, https://www.rfc-editor.org/info/rfc2104.

[10] Whiting, D., R. Housley, and N. Ferguson, "AES Encryption & Authentication Using CTR Mode & CBC-MAC, *IEEE P802* 11 2002, pp. 802–811.

[11] Whiting, D., R. Housley, and N. Ferguson, *Counter with CBC-MAC (CCM)*, RFC Editor, September 2003, RFC3610, doi:10.17487/rfc3610, https://www.rfceditor.org/info/rfc3610.

[12] Dworkin, M., *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, NIST Special Publication (SP) 800-38C, National Institute of Standards and Technology, July 20, 2007, doi:10.6028/NIST.SP.800-38C, https://csrc.nist.gov/publications/detail/sp/80038c/final.

[13] Jonsson, J., "On the Security of CTR + CBC-MAC," *Selected Areas in Cryptography* (K. Nyberg and H. Heys, eds.), Lecture Notes in Computer Science, Berlin: Springer, 2003, pp. 76–93, doi:10.1007/ 3-540-36492-7_7.

[14] NIST and M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, NIST Special Publication (SP) 800-38D, National Institute of Standards and Technology, November 28, 2007, doi:10.6028/NIST.SP.800-38D, https://csrc.nist.gov/publications/detail/sp/80038d/final.

[15] McGrew, D. A., and J. Viega, *The Security and Performance of the Galois/Counter Mode of Operation (Full Version)*, 2004, https://eprint.iacr.org/ 2004/193.

[16] Ferguson, N., "Authentication Weaknesses in GCM," 2005, https://www.semanticscholar.org/paper/Authentication-weaknesses-in-GCM-Ferguson/d85fa090978d3a257cbd9422c29b67987d27f500.

[17] Dworkin, M., "Announcement of Proposal to Revise Special Publication 800-38A," March 2022.

[18] Svenda, P., *Basic Comparison of Modes for Authenticated-Encryption (IAPM, XCBC, OCB, CCM, EAX, CWC, GCM, PCFB, CS)*, self-published, 2005, https://www.fi.muni.cz/xsvenda/docs/AE_comparison_ipics04.pd.

[19] NIST, *Announcing Lightweight Cryptography Selection—CSRC*, CSRC-NIST, February 6, 2023, https://csrc.nist.gov/News/2023/lightweight-cryptography-nist-selects-ascon.

[20] NIST, "NIST Selects 'Lightweight Cryptography' Algorithms to Protect Small Devices," NIST, February 2, 2023, https://www.nist.gov/news-events/news/2023/02/nist-selects-lightweight-cryptography-algorithms-protect-small-devices.

[21] Information Technology Laboratory, NIST, "Lightweight Cryptography Standardization Process: NIST Selections ASCON," CSRC-NIST, February 6, 2023, https://csrc.nist.gov/News/ 2023/lightweight-cryptography-nist-selects-ascon.

[22] Rivest, R. L., *The MD5 Message-Digest Algorithm*, RFC 1321, April 1992, doi:10.17487/RFC1321, https://www.rfc-editor.org/info/rfc1321.

[23] Cox I., et al., *Digital Watermarking and Steganography,* Second Edition, Burlington, MA: Morgan Kaufmann, 2007.

[24] Nakamoto, S., "Bitcoin: A Peer-to-Peer Electronic Cash System," *Bitcoin,* May 2009, https://bitcoin.org/bitcoin.Pdf.

[25] Perrig, A., et al., "The TESLA Broadcast Authentication Protocol," *RSA CryptoBytes* 5, November 2002, doi:10.1007/978-1-4615-0229-6_3.

[26] Perrig, A., et al., *Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction*, RFC 4082, June 2005, doi:10.17487/RFC4082, https://www.rfc-editor.org/info/rfc4082.

[27] AoC, *Specific Emitter Identification (SEI)-AOC Virtual Series*, 2023, https: //www.crows.org/page/SEI-ondemand.

[28] Matuszewski, J., "Specific Emitter Identification," in *2008 International Radar Symposium*, 2008, pp. 1–4, doi:10.1109/IRS.2008.4585772.

[29] Zha, X., et al., *Specific Emitter Identification Based on Paired Sample and Complex Fourier Neural Operator*, May 31, 2022, doi:10.36227/techrxiv.19902835.v1, https://www.techrxiv.org/articles/preprint/Specific_Emitter_Identification_Based_on_Paired_Sample_and_Complex_Fourier_Neural_Operator/19902835/1 (prepublished).

[30] Carter, J. L., and M. N. Wegman. "Universal Classes of Hash Functions," *Journal of Computer and System Sciences,* Vol. 18, No. 2, 1979, pp. 143–154, doi:https://doi.org/10.1016/0022-0000(79)90044-8, https://www.sciencedirect.com/science/article/pii/0022000079900448.

[31] Bennett, C. H., G. Brassard, and J.- M. Robert, "Privacy Amplification by Public Discussion," *SIAM Journal on Computing,* Vol. 17, No. 2, 1988, pp. 210–229, doi:10.1137/0217014, https://doi.org/10.1137/0217014.

[32] Ferguson, N., B. Schneier, and T. Kohno, *Cryptography Engineering: Design Principles and Practical Applications,* Indianapolis, IN: Wiley Publishing, 2010.

[33] Oechslin, P., "Making a Faster Cryptanalytic Time-Memory Trade-Off," in *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Proceedings 23*, Santa Barbara, California, August 17–21, 2003, pp. 617–630.

[34] U.S. Treasury Department, *Dollars in Detail: Your Guide to U.S. Currency*, https://www.uscurrency.gov/sites/default/files/download-materials/en/CEP_Dollars_In_Detail_Brochure.pdf.

[35] Anderson, R., et al. "A New Family of Authentication Protocols," *ACM SIGOPS Operating Systems Review,* Vol. 32, No. 4, 1998, pp. 9–20, doi:10.1145/302350.302353, https://doi.org/10.1145/302350.302353.

5

Public Key Cryptography

This chapter focuses on the third type of cryptography, *public key cryptography*, also known as *asymmetric cryptography*. Public key cryptography applies to the similar security goals of confidentiality, integrity, and authentication. It does so in methods that contrast with what we have seen for symmetric ciphers and hashes, namely by separating the cryptographic material into two parts: one private part, usually for a single entity, and one public part that is generally available to anyone. This asymmetry is extremely useful, and public key cryptography applies widely to situations where the participants cannot be assumed to be trusted with shared information.

Public key cryptography has three main uses. The first use is encryption, where anyone can encrypt information so that only a specific person can decrypt it. The second use is the dual idea: one person can sign information so that anyone can verify the authenticity. The third use is key exchange: public-key cryptography enables two entities without shared secret information to agree on a secret key for use with traditional cryptography methods. Such a use is important when initiating protocols, which we will see in Chapter 6. These three categories of uses will be contrasted with shared-key methods when appropriate.

The different uses of public key cryptography have spawned several well-known algorithms; we give a few examples. Perhaps most familiar is Rivest–Shamir–Adleman (RSA), based on the difficult problem of factoring integers. Other methods use the difficult discrete logarithm problem, notably in the digital signature algorithm (DSA). The discrete logarithm problem also forms the basis of the Diffie-Hellman key exchange algorithm. Public key cryptography relies on computationally difficult problems. We defer a more general discussion of what "computationally difficult" means to the appendix on quantum

computing concepts because quantum computing has the potential of breaking all of the previously mentioned algorithms, and thus, other methods based on other hard problems are needed.

We discuss a few applications specific to public key algorithms, such as novel uses of digital signatures. We describe some of the many aspects of public key infrastructure (PKI), which supplies the enterprise support needed for many protocols in Chapter 6. We do offer one protocol example in this chapter, namely how secure email works, because secure email techniques yield a nice way to unify the three main families of cryptographic methods. Finally, as always, we foreshadow how public key cryptography applies to satellite navigation.

5.1 Motivation and History

The motivation of public key cryptography is to remove the symmetry needed in other methods. Symmetric ciphers require that all participant share the same secret material (i.e., the key). That requirement puts a large trust burden in the system; for example, it seemingly makes problematic the use of such methods for navigation systems with billions of users, all of whom would need to share and protect the same key. This point is not to say that ciphers and hashes are not needed. Indeed their efficiencies make them the first choice when possible. The protocols in Chapter 6 and the subsequent chapters will often have public key methods as enablers to use shared key methods.

5.1.1 Physical Analogs

It may be helpful to think of physical analogs that cryptography works to enable in the digital world. Symmetric methods are analogous to having a single lock with distributed keys or combination. That analogy shows the good security aspects but also exemplifies the explicit trust in all who possess the key. Hashing has the physical analogue of the literal fingerprint: a single observable that can be used for identification. Integrity manifests in the real world like seals that indicate the presence of tampering. Finally, bit commitment likewise is used in real life, in the form of sealed envelopes that are opened at a later date.

The physical analogs that public key solves are twofold. The first item is the use of signatures to authenticate the signer, and by extension, such things as notary stamps. Signatures have the strict property that only the signer should be able to produce the signature, yet we want anyone to be able to verify it. The second item is a locked box, like a suggestion box, where anyone can submit something such that it is hidden to everyone else except the person who can open the box. This chapter explores how to enable these two items in the digital world.

5.1.2 History

Declassified documents from Great Britain's Government Communications Headquarters (GCHQ) indicate that public key methods were known in the 1970s. For example, see the discussion in [1]. The main contributors to this effort, Clifford Cocks, James Ellis, and Malcolm Williamson, were honored by the NSA as 2021 Hall of Honor Inductees. The main algorithms that are being used today came soon after, such as RSA, DSA, and Diffie-Hellman, which we discuss as needed below.

5.2 Public Key Cryptography Goals

The goals for public key cryptography depend naturally on its uses, which we define in the three main sections below. Public key cryptography uses two separate pieces of cryptomaterial: a *public key* and a *private key*. In practice, these items consist of more than just a key value but will include other metadata; for simplicity we stick to the use of the word "key." We emphasize two traits that apply to all public key methods:

- *Private Key:* The private key material should be kept secret and only known (nominally) to a single entity.
- *Public Key:* The public key material can be made known to anyone. No information about the private key should be discerned from knowing the public key. We stress that the word "key" in "public key" does not imply that it is secret.

Our discussion of public key methods will use the usual two protagonists Alice and Bob. They will often have different capabilities and goals.

5.2.1 Encryption

Using a public key for encryption is meant to establish the same confidentiality goals as encrypting with symmetric key methods. The difference is that public key cryptography is used to encrypt something to the holder of the private key such that they and no one else can decrypt it. For example, suppose Bob wants to encrypt something to Alice (Alice and Bob are the ubiquitous entities doing cryptography). Using public key methods, Bob will not be able to decrypt (i.e., reverse the encryption process, once he encrypts it to Alice).

Consider Figure 5.1. Bob desires to encrypt plaintext to Alice. He does this by using Alice's public key along with a specific encryption procedure to produce the ciphertext. Alice, on reception of the ciphertext, uses a decryption

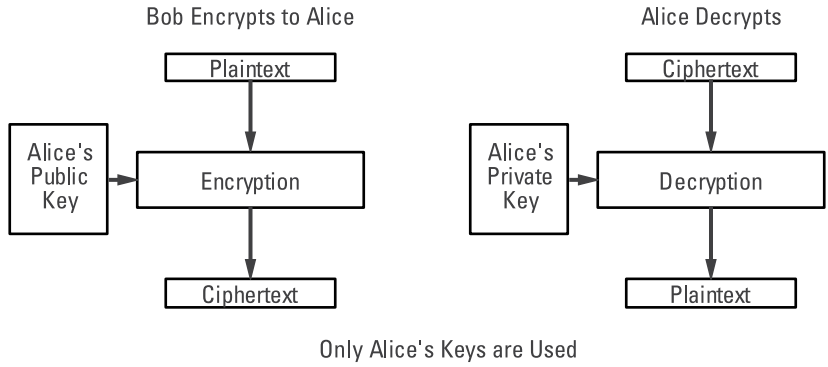

Figure 5.1 Public key encryption: Bob encrypting to Alice.

procedure along with her private key to recover the plaintext. Note that only Alice's keys are used in this process and that the only information Bob needs is usually readily available, which is Alice's public key. One example of this usage is in secure email, where someone can encrypt email so that only the recipient can read it (see Section 5.9).

Public key for encryption has the same goals as using ciphers:

- No information about the plaintext or private key should be discernible from the ciphertext;
- Knowing the plaintext and ciphertext should reveal no information about the private key even if many plaintext-ciphertext pairs are known.

These goals apply even with knowing the public key and algorithms involved. In fact, as a general comment, algorithms are generally published, which allows for vulnerabilities to be discovered more easily and fixed.

5.2.2 Signatures

Suppose that Bob wants to sign some input data, basically to ensure its integrity and that it originated with him. Creating signatures are in some sense the opposite of the encryption procedure, in that the cryptographic material is used in the opposite order.

Consider Figure 5.2, where Bob creates a signature. He first takes the input data and hashes it using some cryptographic hash function. The resulting hash is then used with Bob's private key to create a signature using some signing algorithm. That signature is now bound to the data, similar to a physical signature on a document. A hash is used to make the usually more computationally expensive signature algorithm work on a smaller piece of data.

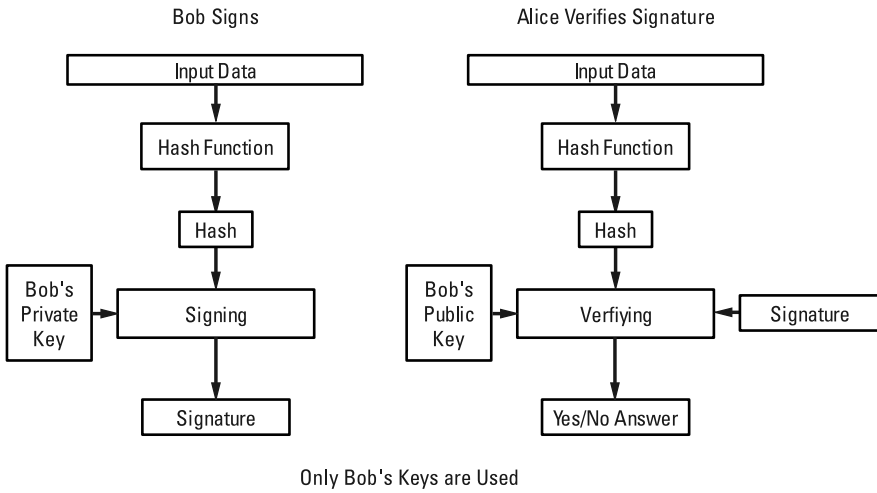

Figure 5.2 Public key signing: Bob signs and anyone verifies.

When Alice receives the input data and signature, she verifies the signature by first repeating the hash generation from the input data. This hash is used with a verification algorithm, Bob's public key, *and* the signature to produce a yes/no answer. Note also that this whole procedure only used Bob's keys.

Compare this verification to verifying a MAC as in Figure 4.7. In that latter case, since both parties share the secret information, verification consist of regenerating the MAC and then comparing the received MAC with the newly generated MAC. That process cannot work for signatures, since Alice does not know how to generate Bob's signature. Thus, we need to create asymmetric algorithms that permit only one person to generate the signature yet allow anyone else to verify it. This point is important to stress: for shared key methods like MACs, verification means regenerating the MACs; for signatures, verification needs a procedure that is separate from generation.

The goals of a digital signatures are authentication and integrity:

- *Authentication* means that one can identify the creator of the signature if the signature is valid;
- *Integrity* means that the signature attests that the signed data has not been modified.

Signing only the hash allows attesting to the integrity of the input data using only the smaller hash.

Some sources may talk about signing in terms of "encrypting" with one's private key. We avoid that term in this book and will use the verb "signing"

instead. The reason is that encrypt usually connotes keeping something secret, and creating a signature does not produce something that is kept secret. On the contrary, the point of the signature is so that anyone can verify it.

Finally, it is worth comparing signatures with MACs. Both are ways to confirm authenticity and integrity of data, yet they have different uses.

MACs and Signatures

- A MAC is more computationally efficient as a rule, and has the added benefit that a truncated portion of the MAC can be used instead of the full value.
- However, MACs require that all participants share the MAC-generating key.
- Signatures remove the trust concerns of sharing keys, since only the signing entity has the key. In this way, signatures give *nonrepudiation*: the signer cannot deny the signature once it is verified.
- However, digital signatures are computationally more burdensome, especially because they cannot be truncated and must be sent and used as a whole.
- Both MACs and signatures also need to take into account the birthday analysis for obtaining collisions. That means that the strength is related to the size of the MAC or signature.

5.2.3 Key Exchange

The final use for public key cryptography is in key exchange. Here the idea is for two parties to use each other's public information to generate a shared key. The purpose is that once a shared key is established, more efficient methods like ciphers and MACs can be used. Indeed, a major trait of many cryptographic protocols is to leverage public key methods just enough to then switch to shared key methods. As an example, the TESLA protocol in Section 4.8 uses shared key methods but also uses digital signatures.

Figure 5.3 shows a generic picture of the key exchange process. The process assumes that Alice and Bob have private information that only they know. This information is used to create public information. This information is not the only public information: other information is assumed to be known to both parties as needed, such as underlying algorithms and open parameters. After each party receives the other's public information, they combine it with their public and private information in a way where both sides create the same shared key. Clearly there is something special going on, since Alice and Bob obtain the same shared key, even though neither knows the other's private information.

We stress that in key exchange protocols, the confidentiality of the shared key is a fundamental goal that is achieved by the methods even if the public information is known. However, we likewise stress even more that the process shown in Figure 5.3 does not in itself have authentication. That is, nothing ensures that Alice and Bob are actually interacting with each other. Such

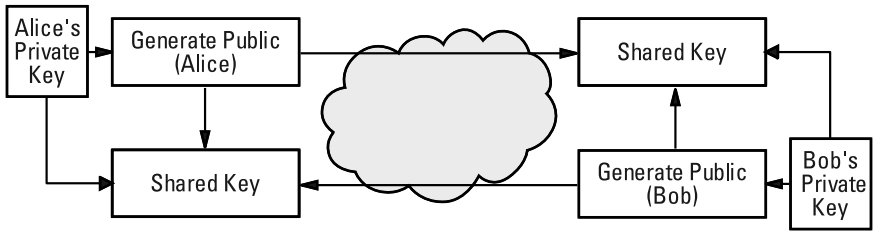

Figure 5.3 Key exchange generic process.

authentication can also be obtained using public key cryptography (e.g., using digital signatures).

This observation shows that several methods are needed to obtain all security goals, which will be pertinent in Chapter 6.

5.3 Math Foundations

This section describes the mathematical foundations for some public key methods. The mathematical techniques in the last two chapters show alternative ways to achieve the methods. For ciphers, the techniques show alternatives to creating confusion and diffusion by substitutions and permutations. For hashing, the techniques create one-way functions that yield essentially no information about the input. The crux to obtaining these techniques is to find mathematical problems that are difficult to solve, while at the same time permit an easy solution if additional information is known.

In some cases, what is desired is a *trapdoor one-way function*, which is a function that is difficult to invert unless extra information is known. Some public key methods are based on such functions. For example, RSA is based on the difficulty of integer factoring, and knowing the pertinent factors is the trapdoor. On the other hand, discrete logarithm based schemes do not evidently have a trapdoor.[1]

In this section, we present enough math background for both the factoring and discrete logarithm based methods. The mathematics needed is basically at the level of arithmetic, and we necessarily limit the discussion. The focus in this section is on number theory; a good reference is [2].

5.3.1 Number Theory 101

We will work with integers *modulo* some integer m called the *modulus*. Modulo in this context means the remainder of the integer when divided by the modulus m, and it is written as mod m, for example, $31 = 5 \pmod{13}$. This means

that 31 has a remainder of 5 when divided by 13. We note that we are not going to discuss the notion of equivalence classes usually used to make modulo arithmetic precise; for our purposes we just a convenient representation of the value modulo the number.

Arithmetic operations of addition, subtraction, and multiplication all behave as one would expect with modulo arithmetic. For example, adding and subtracting multiples of the modulus does not change the relationship, so, for example, $-1 = 12 \pmod{13}$. There are m possible remainders when dividing by m, which we denote as

$$\mathbb{Z}_m = \{0,1,\ldots,m-1\} \tag{5.1}$$

When we talk about $\mathbb{Z}_m$, we assume the standard operations modulo m.

The concept of division is a little trickier with modulo arithmetic. For example, in $\mathbb{Z}_{15}$, $3 \cdot 5 = 0$. That means that neither 3 nor 5 could have a reciprocal in $\mathbb{Z}_{15}$. However, there are numbers in $\mathbb{Z}_{15}$ that have reciprocals, for example $2 \cdot 8 = 1$. Such numbers are called *units*, and the set of units modulo m is denoted by $\mathbb{Z}_m^*$. In the example,

$$\mathbb{Z}_{15}^* = \{1,2,4,7,8,11,13,14\} \tag{5.2}$$

Euclid's algorithm is an efficient method to find the greatest common divisor (gcd) of two numbers and can also be used to find reciprocals modulo a number.[2]

The units are just those numbers that are relatively prime to the modulus; that is, the gcd between the number and the modulus is 1. The total number of units is *Euler's totient function* denoted by $\varphi(m)$. In the example, $\varphi(15) = 8$. Calculating $\varphi(m)$ is easy if the factorization of m is known.[3]

When m is a prime p, then every number between 1 and $p - 1$ inclusive is relatively prime, and thus

$$\mathbb{Z}_p^* = \{1,\ldots,p-1\} \tag{5.3}$$

To emphasize: every nonzero number in $\mathbb{Z}_p$ has a multiplicative inverse or reciprocal. That means $\mathbb{Z}_p$ is a *finite field*, like the Galois field we saw in Chapters 3 and 4.

We finish with three main facts we will use to prove some of the public key methods.

Generator of a Prime

For any prime p, there is (at least) one element of $\mathbb{Z}_p^*$, g, called a *generator* such that the powers of g generate $\mathbb{Z}_p^*$.

For example,

$$\mathbb{Z}_7^* = \{3, 3^2 = 2, 3^3 = 6, 3^4 = 4, 3^5 = 5, 3^6 = 1\} \tag{5.4}$$

A prime can have multiple generators; in the example, 5 is also a generator. Generators are also called *primitive roots*. The existence of a generator says that $\mathbb{Z}_p^*$ is *cyclic*.[4]

Fermat's Little Theorem
Given a prime p and generator g, then $g^{p-1} = 1 \pmod{p}$. In fact, $x^{p-1} = 1 \pmod{p}$ for all nonzero x.

Euler's Totient Theorem
Given an integer m, then $x^{\varphi(m)} = 1 \pmod{m}$ for all $x \in \mathbb{Z}_m^*$.

These two theorems mean that we can readily surmise the results for certain exponential calculations.

5.3.2 Hard Number Theory Problems

The factoring problem is to find the factors of an integer; the factors are unique. For small numbers, the factorization is easily done in many mathematical packages. Once the number gets large, the time to factor increases, and the best algorithms take more than polynomial time as a function of the bit-length of the number. There are several methods listed in [2]. For example, the general number sieve methods have the time to factor an integer n on the order of

$$\exp\left(c\left(\ln n\right)^{1/3}\left(\ln \ln n\right)^{2/3}\right) \tag{5.5}$$

(c a constant; observe that ln n is just a multiple of the number of bits to represent n). This function is much less than exponential, but still much larger than polynomial.

The other hard problem of interest is the discrete logarithm problem. This problem is defined in a finite field; for concreteness, consider $\mathbb{Z}_p$ for a prime p. Calculating $y = x^k \pmod{p}$ is easy to do even for large integers. The efficient method is to do repeated squaring and multiplications based on the binary representation of k. The discrete logarithm problem is to find k given x and y. If we were working with real numbers, one would just take the logarithm of both sides of the equation. No simple method exists for a finite field, although algorithms have been developed.

The discrete logarithm problem extends to the general case of a large cyclic structure defined by a generator, that is, $\{g, g^2, g^3, \ldots\}$. Given an element,

the question is to find the exponent of g. This setting is used in elliptic curves (Section 5.5). The complexity of the discrete logarithm problem is also more than polynomial but subexponential. One thing that makes discrete logarithm methods attractive is that the parameter sizes can be smaller than factoring based methods for the same cryptographic strength, where strength is defined by the amount of effort to break the cryptography.

5.4 RSA

RSA takes its name from its developers, Ron Rivest, Adi Shamir, and Leonard Adleman, and dates back to 1977 [3]. As such it is one of the earliest public key algorithms, although the method was known a few years before based on declassified data. A U.S. patent was granted to RSA for the algorithm in 1983, and persisted for the prescribed 17 years until 2020. RSA is widely used in many protocols.

5.4.1 Definition

RSA is based on the difficulty of factoring. The definition of the algorithm comprises five integers, which are commonly specified as

- Two prime numbers p and q.
- Their product $n = pq$. The length of p and q in bits determines the strength of the RSA algorithm. For example, if each is 1024 bits, then n is 2048 bits, and one speaks of 2048-bit RSA.
- A number e, which is relatively prime to $\varphi(n) = (p - 1)(q - 1)$.
- The number d such that $ed = 1 \pmod{\varphi(n)}$.

There are efficient methods to create all of these parameters, (i.e., the large primes and the value d). The value e is often chosen to be a simple form, such as $2^{16} + 1$, if that works.

RSA Parameters: The public values in RSA are the values n and e. The private values are p, q, and d.

Given those parameters, encryption and decryption are defined by:

RSA Encryption: Choose a plaintext message M considered as an integer modulo n. Then the ciphertext integer C is defined by

$$C = M^e \pmod{n} \tag{5.6}$$

RSA Decryption: Given the ciphertext integer C, the plaintext message M is computed as

$$M = C^d \pmod{n} \tag{5.7}$$

The exponents e and d in the RSA algorithm stand for encrypt and decrypt, respectively.

We offer a numerical example. Let the primes be $p = 10223$ and $q = 10243$. Compute $n = pq = 104714189$ and $\varphi(n) = 10222 \cdot 10242 = 104693724$. Choose $e = 2^{16} + 1$. Then $d = 12754241$. To encrypt a message, we would convert the message into a number less than n. For example, if the message is $M = 2024$, then

$$C = M^e = 2024^{65537} \pmod{104714189} = 21769398$$

Checking decryption

$$C = M^e = 21769398^{12754241} \pmod{104714189} = 2024$$

5.4.2 Justification

The proof that RSA works comes down to showing that decryption recovers the plaintext. First, assume that M is relatively prime to n. Then modulo n:

$$C^d = M^{ed} \tag{5.8}$$

$$= M^{1+k\varphi(n)} \quad \text{Definition of } e \text{ and } d, \text{ for some } k \tag{5.9}$$

$$= M \cdot \left(M^{\varphi(n)}\right)^k \tag{5.10}$$

$$= M \quad \text{Euler's totient theorem} \tag{5.11}$$

Notice that there are two different modulo arithmetic operations being used. In the exponents, arithmetic is modulo $\varphi(n)$, while the exponential computations are module n.

If M is not relatively prime to n, then M and n share either p or q as a factor, and thus all the private information would be exposed easily using Euclid's algorithm. The probability of that occurring for random M is very, very small.

5.4.3 Implications

The strength of RSA depends on not being able to discern the private information, similar to learning the key in a symmetric cipher. Since n is public, the strength depends on not being able to factor n. Various challenges have existed and are ongoing to factor an RSA integer. For example, 512-bit RSA has been factored using a lot of computing power and the current state of the art is approaching 1024-bit RSA.

If e and d are known, it is possible to factor n efficiently [2]. This property means that RSA moduli cannot be shared: if two people share n, then either would be able to factor and learn the private keys of the other.

Since e is the public exponent, there is a strong desire to have e be simple. Because efficient exponentiation depends on the binary representation of the exponent, $2^{16}+1$ is common for its sparse binary representation and the small likelihood that it would share a factor with $\varphi(n)$. However, one should be careful about using a very small e like $e = 3$, because then some plaintext could be recovered by just taking the cube root over the reals.

5.4.4 Signatures

RSA can be used to create digital signatures. The natural procedure is to sign an integer M by using the private d exponent, that is,

$$S = M^d \pmod{n} \tag{5.12}$$

Then given M, S can be varied by checking

$$M \stackrel{?}{=} S^e \pmod{n} \tag{5.13}$$

This procedure works for the same reason as the proof for encryption above. Since e is the public exponent, anyone can verify the signature.

However, this natural approach is problematic because any M can serve as the signature of M^e, which means anyone can create fake signatures. The solution is not to sign M but say hash(M) for some cryptographic hash; in fact, the message here can then be of an arbitrarily length. On receipt the message must be hashed first and then the signature is verified using e.

5.4.5 Practicalities

Some pathological cases can arise based on the fact that the plaintext in RSA are just numbers. Examples are the problem that could arise if e is small, or the problem of forgeries in the first signature example. The solution is to have well-defined standards for padding and the use of hashes; see, for example, [4].

5.4.6 Attacks

The main attack on RSA is to factor the modulus n, and that can be prevented if n is large enough and the primes p and q are chosen correctly. For example, if $p - q$ is small, then one can use *Fermat's factorization* to efficiently factor n. "Small" means less than $2n^{1/4}$. There are also issues if $p - 1$ or $q - 1$ have many small factors. Modern software will create prime pairs taking into account these restrictions. Of course, the prime generation needs to have good randomness to ensure that one is generating random, independent primes.

Kocher in 1995 developed one of the first well-publicized timing attacks against RSA [5]. The idea is that exponentiation takes time depending on the bit representation of the exponents, and so monitoring execution time can yield information about the exponents. The solution is have implementations that always take the same amount of execution time.

Finally, consider the following scenario. An attacker Mal wishes to decrypt $C = M^e$, which was encrypted to Bob. Mal picks a random integer r and asks Bob to sign $C \cdot r^e$. Bob does so, yielding $S = (C \cdot r^e)^d$, which is $M \cdot r$. Mal can now divide by r, since it is almost surely relatively prime to n and recover M. This example shows that care needs to be taken in what exactly is being signed.

5.4.7 Strengths

A natural question is how strong is RSA as a crypto-system compared to say using AES. Table 5.1 is taken from [6]. It shows the strength of RSA by the modulus n in terms of an equivalent symmetric cipher brute force. These values do not take into account quantum computing, which we discuss in the appendix.

5.4.8 Performance

The strengths given in Table 5.1 come with a computational burden. Table 5.2 shows benchmarks for signing and verifying for RSA; the benchmarks are using the OpenSSL **speed** command on one of the author's computers. Here "sign" means using the private exponent d and "verify" means using e. For each RSA

Table 5.1
RSA Strength versus Size

Security Strength	RSA Size (n)
80	1024
112	2048
128	3072
192	7680
256	15360

Table 5.2
RSA Performance

RSA Size (*n*)	Sign (*d*)	Verify (*e*)	Sign/Sec	Verify/Sec
512	0.000047s	0.000005s	21462.6	199198.6
1024	0.000173s	0.000017s	5765.7	57685.5
2048	0.000946s	0.000041s	1057.0	24666.0
3072	0.003444s	0.000073s	290.4	13623.7
7680	0.140899s	0.000583s	7.1	1715.7
15360	0.369213s	0.002225s	2.7	449.5

size, the time for signing and verifying are given along with the reciprocal for the rate of these operations.

Consider the benchmarks for AES given in Table 3.4. Using a key size of 128 bits, we see around 180 MB/sec; the results were from the same computer. To get strength of 128 bits for RSA requires 3072-bit RSA. From Table 5.2, that corresponds to 13K encryptions per second and 290 decryptions per second (using *e* is for encryption and *d* for decryption). Putting these numbers in terms of bits per second by multiplying by 3072 or 384B), we get encryption of about 4 MB/s and decryption of only about 100 KB/s. Thus AES is much, much faster.

5.5 Digital Signature Algorithm

After soliciting proposals for a public key signature standard separate from RSA, NIST proposed the DSA to be the Digital Signature Standard (DSS). DSA is defined in FIPS 186, with the latest revision being FIPS 186-5 [7]. This latest version deprecates DSA over prime numbers for signing, instead recommending elliptic curve based methods discussed later. That said, we describe the algorithm using prime numbers, since that will show the underlying structure more readily. The overall structure of the algorithms are similar using different underlying mathematical frameworks.

DSA uses modulo arithmetic and relies on the hardness of the discrete logarithm problem.[5] Even though DSA uses modulo exponentiation like RSA, it can use smaller moduli for the same strength, and thus has the benefit of being more efficient. DSA is purely a digital signature scheme, which addresses authentication, integrity, and nonrepudiation. As such, we need to specify the public and private information and the procedures for generating and verifying the signatures.

5.5.1 Definition

DSA is defined in three parts: the setup, signing, and verifying. Setup first requires creating a common set of parameters that anyone can use:

- An approved cryptographic hash function is assumed. If needed, the hash can be truncated.
- Choose the *key length* value L. For concreteness, we will use $L = 2048$.
- Choose the modulus length N. Again, for concreteness, pick $N = 224$.
- Pick an N-bit prime q.
- Pick an L-bit prime p such that q divides $p - 1$.
- Compute g such that $g^q = 1 \pmod{p}$ and $g^\ell \neq 1$ for smaller powers $1 \leq \ell < q$. One can find such a g by using some generator h modulo p and computing $h^{(p-1)/q}$.

The parameters p, q, and g are public and usable by any user. Next, each individual user creates their own public and private parameters,

- Choose a random $x \in \{1,\ldots,q-1\}$.
- Compute $y = g^x \pmod{p}$.

The value x is the private key for the user, and the value y is their public key.

Signature generation creates two separate numbers using a one-time signer-generated random value. This random value serves to obfuscate the private information. The verifier performs a calculation that essentially cancels out the random information if and only if the signature is valid. The message to be signed is M, and $z = \text{Hash}(M)$ is the hash of M truncated to N bits if need be.

Signature Generation

Choose a random value k with $0 < k < q$. Let k^{-1} be the multiplicative inverse of k modulo q, that is $k \cdot k^{-1} = 1 \pmod{q}$. Compute

$$r = \left(g^k \left(\text{mod } p\right)\right)\left(\text{mod } q\right) \tag{5.14}$$

$$s = \left(k^{-1}\left(z + xr\right)\right)\left(\text{mod } q\right) \tag{5.15}$$

The signature is (r, s), where $0 < r < q$ and $0 < s < q$.

The verifier computes a quantity using (r, s) to vet the signature. As a first check, one needs to confirm that both r and s are strictly between 0 and q.

Signature Verification
Compute the following assuming $0 < r < q$ and $0 < s < q$

$$w = s^{-1} \left(\mathrm{mod}\, q\right) \tag{5.16}$$

$$u_1 = z \cdot w \left(\mathrm{mod}\, q\right) \tag{5.17}$$

$$u_2 = r \cdot w \left(\mathrm{mod}\, q\right) \tag{5.18}$$

$$v = \left(g^{u_1} y^{u_2} \left(\mathrm{mod}\, p\right)\right)\left(\mathrm{mod}\, q\right) \tag{5.19}$$

The signature is valid if and only if $v = r$.

5.5.2 Justification

First observe that (5.15) can be rewritten as

$$k = \left(zw + xrw\right) = \left(u_1 + xu_2\right)\left(\mathrm{mod}\, q\right) \tag{5.20}$$

That means raising g to both sides should yield

$$r = g^k \overset{?}{=} g^{u_1} g^{xu_2} = g^{u_1} y^{u_2} \left(\mathrm{mod}\, p\right) \tag{5.21}$$

The nuance is that the exponent arithmetic is all modulo q, because we are only working with powers of g.

5.5.3 Implications and Attacks

The strength of DSA is based on being difficult to discern x. Since $y = g^x$ is public, finding x is precisely the discrete logarithm problem modulo p. This observation is why L, the bit length of p, is the key length. It is also why DSA has been deprecated and the use of elliptic curves is preferred, since the equivalent strength can be achieved with fewer bits (see below).

From (5.15), notice that since r, s, and z are known, learning k would reveal x. That observation means that great care must be used in choosing k. The

value k should be random and unpredictable, nor should it be revealed. Furthermore, one should never reuse the value k. The reason is that if k were used twice for two different messages yielding two different (r, s) pairs, then one can easily construct two equations with the unknown k and x and easily solve for x.

A consequence of the nonreuse is that it is possible that signing the same data a second time would yield a different signature. Thus, unlike with a MAC, one cannot necessarily compare the signatures to verify a signature of data even if the data had not changed; instead, one would indeed need to do the verification process. A related trait is that the whole signature must be given; one cannot use a truncated portion. Attacks on DSA have been due to violating the requirements on k.

5.5.4 Elliptic Curve Methods

Elliptic curves over finite fields are a more efficient way to do discrete-log based cryptography methods; they are much more efficient than RSA and DSA over prime moduli for the same strength. Our discussion of elliptic curves is necessarily brief but should give a sense of the underlying computations and parameters. A good reference is [8].

To give some visualization, we first consider elliptic curves over the real numbers. A typical curve is shown in Figure 5.4. The curve is $y^2 = x^3 - x + 1$, which is an example of the general form $y^2 = x^3 + ax + b$. Because of the y^2 term, the curve is a relation and not a strict function of x.

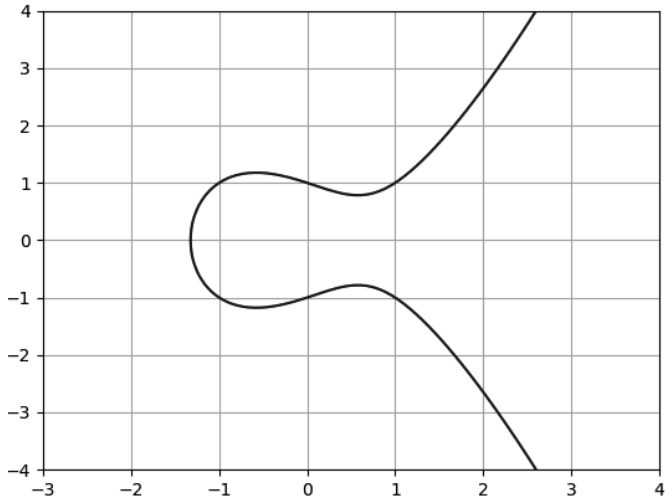

Figure 5.4 An example elliptic curve: $y^2 = x^3 - x + 1$ idea of creating an integer multiple of a point.

It is surprising that an arithmetic can be defined on the points of elliptic curve. That is, the points on the curve form a mathematical group using an operation called addition. The definition of adding points must have a point that acts like 0. There needs to be the notion of subtracting points (i.e., the additive inverse). And finally, there needs to be the idea of creating an integer multiple of a point.

These concepts are shown graphically in Figure 5.5.

- The 0-element, denoted as O, is the point at infinity represented by vertical lines.
- The additive inverse, (i.e., the negation of a point P) is given by reflecting the point across the x-axis, that is, if P =(x, y), then –P =(–x, y).
- Doubling a point (i.e., 2P = P + P), involves drawing the line tangent to the point, seeing where it intersects the curve, and reflecting the result.
- In general, adding two points P + Q involves drawing the line that connects them, look for the third point where that line intersects the curve, and then reflecting the result.

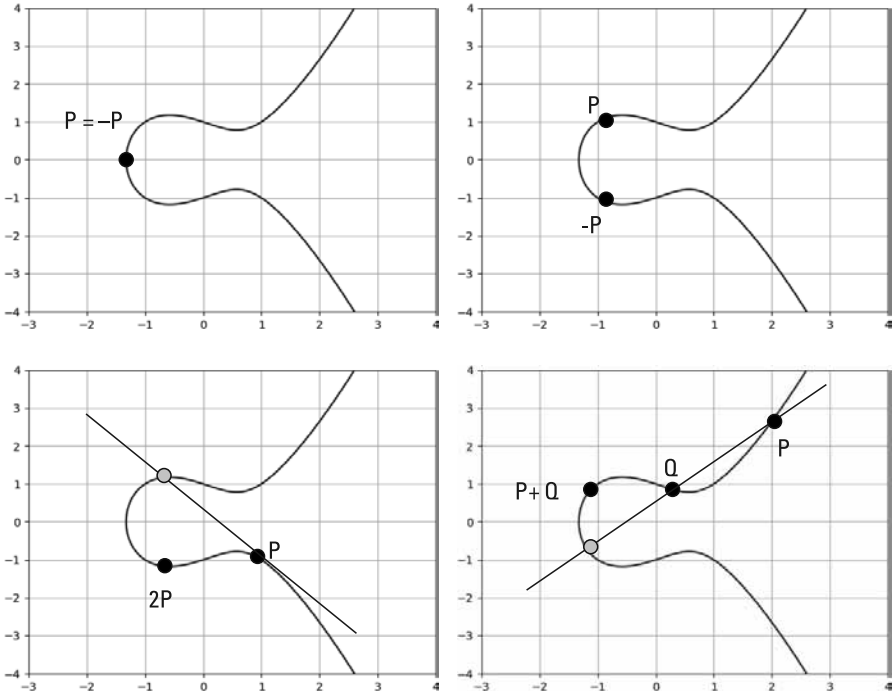

Figure 5.5 Adding points on $y^2 = x^3 - x + 1$.

The fact that these operations have all the desired properties, specifically commutativity and associativity, is straightforward although tedious.

Now consider the equations modulo a prime, that is, points are pairs x, y that satisfy $y^2 = x^3 + ax + b \pmod{p}$ (the curves can also be defined over more general finite fields). This defines a finite set of points with the same addition operation. Moreover, there is fundamental correspondence, a mathematical isomorphism, between addition operating on points and exponentiation modulo a prime p.

$$O \text{ point} \Leftrightarrow 1 \pmod{p} \tag{5.22}$$

$$P + Q \Leftrightarrow a \cdot b \pmod{p} \tag{5.23}$$

$$-P \Leftrightarrow a^{-1} \pmod{p} \tag{5.24}$$

$$P, 2P, 3P \ldots \Leftrightarrow a, a^2, a^3, \ldots \pmod{p} \tag{5.25}$$

This correspondence means that the DSA equations can be defined analogously over an elliptic curve. The results of this is called elliptic curve DSA (ECDSA) defined in FIPS 186-5 [7]. For example, one would create the elliptic curve and a generating point G, like the generator g. We need a prime q with $qG = O$. A private key is an integer x, and the public key is the point xG. Equation (5.14) then involves choosing a value k with the usual provisos and defining the first part of the ECDSA signature as $R = kG$. The second part S is defined similarly, since it uses modular arithmetic modulo q.

5.5.5 Strengths

The strengths for DSA and ECDSA are defined similar to the strength of a hash function, because digital signatures are also subject to birthday-like analysis (see [6]). Table 5.3 shows the strength. For prime-based DSA, the strengths yield moduli similar to RSA. Note the size of the signature is actually two times the size of q in bits. Similarly, the size of the elliptic curve needed for ECDSA is the same as q, and also four times the strength; the size of the larger p is comparable to the RSA moduli. The efficiency of the elliptic curve methods comes in the fact that DSA needs to do arithmetic modulo p, a much bigger number, while ECDSA does all computations in the size of the curve.

Table 5.3
DSA Strengths versus Sizes

Security Strength	DSA (q)	DSA (p)	ECDSA Curve
80	160	1024	160
112	224	2048	224
128	256	3072	256
192	384	7680	384
256	512	15360	512

5.5.6 Performance

Table 5.4 shows the benchmark for signing and verifying signatures using ECDSA; again, the benchmark use one of the author's computer. The entries are the times to perform the operation and the rates of doing those operations. In most cases, signing and verification are comparable; contrast that with the benchmarks for RSA in Table 5.2. RSA verification (using e) is generally much faster than ECDSA verification because the form of e is usually chosen to be efficient, while all parameters in ECDSA are essentially random.

It is also worth comparing the efficiency of hashes given in Table 4.2 for SHA2. The overhead for HMAC is only three addition hashes, but these hashes are hashes of the 256-bit hashes. Thus consider the hash speeds from Table 4.2 of 16 bytes of data, which is only around 307MB/s, or about 15 million hashes a second. That means on the order of say 3 million HMACs per second. Contrast that with the P256 ECDSA numbers, which are on the order of 10,000s signatures per second, which is about 300 times slower. This calculation is why, for example, TESLA in Section 4.8 uses MACs.

5.6 Diffie-Hellman Key Exchange Protocol

This section describes the Diffie-Hellman (DH) key exchange protocol. It is named after the two creators Whitfield Diffie and Martin Hellman and dates

Table 5.4
DSA Performance

NIST ECDSA Curve	Sign (d)	Verify (e)	Sign/Sec	Verify/Sec
P192	0.0003s	0.0003s	3774.8	3353.9
P224	0.0004s	0.0004s	2253.6	2466.9
P256	< 0.0001s	0.0001s	40608.2	15049.6
P384	0.0011s	0.0012s	874.0	835.5
P521	0.0029s	0.0019s	349.0	514.0

from 1976. The procedure is based on the key exchange ideas discussed in Section 5.2.3. Two participants, Alice and Bob, desire to agree on a shared key. They agree on some upfront parameters and algorithms, which in fact can be shared broadly. Then they each create some public information, sometimes called public keys, from private information or a private key. The combination of one's private key with the other's public key yields, perhaps surprisingly, the shared key.

5.6.1 Protocol

We give the details for DH first in terms of modular arithmetic. We discuss the changes later for elliptic curves, which form the basis for modern DH procedures. The procedure requires:

- A prime numbers p;
- A generator g modulo p.

Both p and g are made publicly available.

Alice and Bob establish that they wish to perform the key exchange. They execute the following steps, which are shown in Figure 5.6.

Diffie-Hellman Protocol

The following steps are performed:

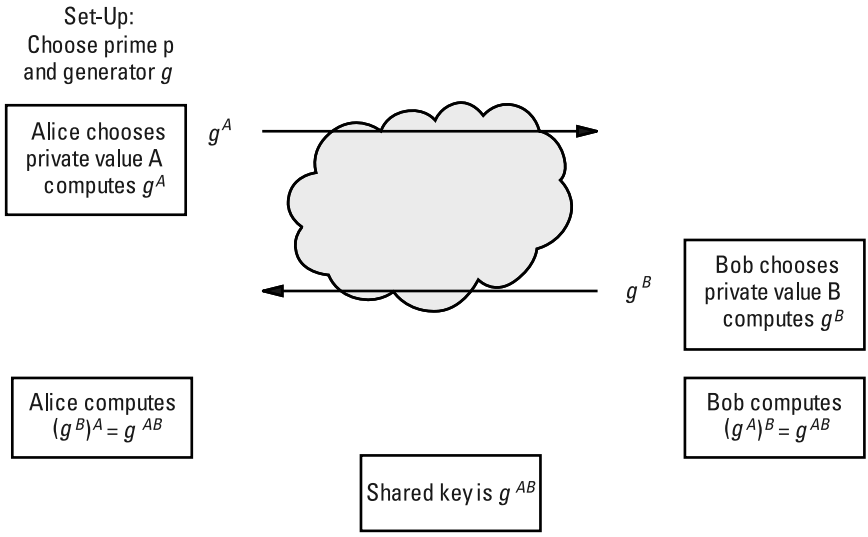

Figure 5.6 The Diffie-Hellman key exchange protocol.

1. Alice creates a one-time-use random private value A and computes $g^A \pmod{p}$.
2. Alice sends g^A to Bob;
3. Bob creates a one-time-use random private value B and computes $g^B \pmod{p}$.
4. Bob sends g^B to Alice;
5. Alice and Bob each compute g using their private value and the public information sent by the other;
6. The shared key is g^{AB}; it is essentially random with the same bit length as p.

Steps 1, 2 and 3, 4 can be done in parallel. The values g^A and g^B are public.

5.6.2 Implications and Attacks

The security of DH depends on the difficulty of the discrete logarithm problem. For example, if one can find A from the public g^A, then the shared key is also obtained by observing g^B. The difficulty depends on the size of p. The value of g does not matter.

The usual application, such as in protocols like TLS (Chapter 6), is for A and B to be one time use. However, there are other possibilities discussed in [9]. The one-time-use is the *ephemeral* case. The other main case is the *static* case, where the private/public information of the entity is used for a long time. Such use may be attractive for efficiency when it is combined with other methods to ensure security goals. For example, using the same key for a long time may allow replay attacks of previously created messages using that key.

By itself, DH does not given authentication. Indeed, the protocol described above is subject to a man-in-the-middle attack:

1. Eve creates private E and public $g^E \pmod{p}$;
2. Eve intercepts Alice's message g^A to Bob and instead sends g^E to Bob;
3. Eve intercepts Bob's message g^B to Alice and instead sends g^E to Alice;
4. Alice and Eve compute g^{AE};
5. Bob and Eve compute g^{BE}.

At this point, Eve has a separate shared key with Alice and Bob. Alice and Bob, on the other hand, think they have a shared key with each other. Thus, all future communication will go through Eve.

This attack is mitigated by first adapting some authentication procedure (i.e, Alice and Bob use digital signatures). Such a solution requires separate

public key methodology. How to obtain the necessary enterprise to support such methodology is the subject of Section 5.8.

5.6.3 Elliptic Curve Methods

Recall from Section 5.5.4 that discrete logarithm based problems can translate to elliptic curve calculations for more efficiency for a given strength. The idea is that one has an elliptic curve and a generating point G, which will act like g. Alice and Bob each create their random numbers a and b, and compute the points $a \cdot G$ and $b \cdot G$. The exchange key is then derived from $a \cdot b \cdot G$.

5.6.4 Strengths

The recommended strengths can be found in [9] and some values are summarized in Table 5.5. The values are very similar to those in Table 5.3.

5.7 Applications: More on Signatures

The existence of secure digital signature methods allows for the creation of digital analogs of several related processes found in the physical world. Some of these methods rely on a trusted third party or trusted authority, which we denote by TA for short. As the name suggests, this entity has public key information that everyone has authenticated. That means if something that the TA signs is verified, one can be sure that the TA did indeed sign the material under whatever circumstances are prescribed. Other methods will require that two participants have authenticated each others' public key information. There are protocols and infrastructure to support that, as we discuss in the next section and in Chapter 6.

5.7.1 Time Stamps

An easy example of using a TA is as a time stamp authority. The idea is that if someone has a document they wish to bind to a certain time, they send a

Table 5.5
Diffie-Hellman Sizes

Security Strength	Modulo p	Elliptic Curve Size
112	2048	224
128	3072	256
192	8192	384
256	(no entry)	521

hash of the document to the TA. At the desired time, the TA signs the hash and the time stamp (see [10]). Anyone can verify the original document was signed at that time by both verifying the signature and verifying the hash of the document. Using the hash is efficient and reminiscent of the bit commitment schemes we discussed in Chapter 4.

The above scheme begs the question of the accuracy of the time of the TA. The protocols do not address that, and, of course, assuring the time estimate is part of the role of the cryptographic methods we present later in this book for satellite navigation.

5.7.2 Blinding and Blind Signatures

The concept of *blinding* in cryptography applies when one entity, Alice, desires for another, Bob, to compute some function for Alice without knowing the input. The general way this is accomplished is that Alice blinds or obscures her input to the function, Bob computes on this blinded input, and Alice then takes the results and removes the blinding. Accomplishing this goal in practice requires the function and blinding procedure to commute in a way such that Alice can remove the blinding.

A common example is in blind signatures, where the idea is that Alice wants Bob to sign a message without being allowed to see the message. Presumably Alice would then share the message and signatures with others who value Bob's signature. One way to accomplish this goal is to send a hash of the message to Bob, Bob signs the hash, and then Alice can present the signed hash along with the message. This procedure is precisely the same as discussed in the previous subsection. A physical analogy would be someone affixing their seal to an already closed envelope. That would attest that they had possession of the sealed envelope, say at a given time, without necessarily knowing the contents.

Another blind signature method takes advantage of RSA. Consider an RSA scheme with Bob's parameters being the usual $\{n, e, d\}$. Alice would like to obtain M^d from Bob; this is essentially Bob signing M. She blinds M by choosing a random value r and sending Mr^e to Bob. Bob then signs this quantity by computing $(Mr^e)^d = rM^d$, which he sends back to Alice. Alice can then remove r by multiplying by $r^{-1} (\text{mod } n)$, because it is highly unlikely that r does not have an inverse modulo n. Thus, she obtains the desired M^d.

5.7.3 Double Signatures

Recall that a message digest provides a way to bind information. One excellent example that shows how message digests can be used with digital signatures is the example from the *secure electronic transaction* protocol [11]. While this protocol is not in active use, the core cryptographic primitives are worth considering.

Alice desires to purchase something from Bob using funds from her credit card company, say Charlotte for short. She forms the purchase order information (OI) and the payment information (PI) separately, and computes the message digest of both, respectively MD(OI) and MD(PI). She then creates the digital signature S of MD(MD(OI), MD(PI)), which is called a double signature. This one signature S serves to validate and bind together both the OI and PI. She then sends the following:

- Bob receives the OI, MD(PI), and S;
- Charlotte receives the PI, MD(OI), S.

When Bob requests funds, he sends MD(OI) to Charlotte, who can then verify the signature by computing MD(MD(OI), MD(PI)). Similarly, Bob is protected if someone else tries to use different payment information. On the other hand, Charlotte has no idea what Alice is purchasing.

The virtue of this protocol is that Alice can sign two pieces of information (order and payment) that must be linked together and yet need to be shared separately. This example also shows how using properties of different cryptographic primitives can achieve more goals. The discussion of email security later in this chapter is another example, as are the various protocols throughout this book.

5.8 Applications: PKI

Public key cryptography relies on sharing public key information while keeping private key information closely held. The first goal is accomplished by having *certificates* that store public information in some standardized way, usually with associated metadata. The owner of the private key must use some method to protect it; this private key will likewise have some standardized format. However, it is not enough to create and hand out a certificate to someone. First, the certificates need to be bound to the associated party. Often this binding comes from having the owner physically proving their identity before receiving the certificate. Second, there must be a way to revoke a certificate if it is expired or if the party should no longer be trusted.

These goals imply that some enterprise is needed to manage the public information. This enterprise is commonly called a public key infrastructure. Describing how PKI works fully is beyond the scope of this book; see the discussion and references in [12]. Instead, we give a broad description of the goals, processes, and elements for PKI.

The development of PKI began concurrently with the development of public key cryptography and its use in protocols. The ad hoc nature of these

protocols (i.e., letting individuals without shared secret information communicate) demanded a way to coordinate the necessary information. That situation grew more so with the advent of the internet, web browsers, and ubiquitous ad hoc sessions (see the protocols in Chapter 6).

5.8.1 PKI Structure

An example of the overall structure of a PKI is shown in Figure 5.7. The enterprise begins with the end user, who desires a public key certificate. This information is created by the *certificate authority (CA)*. The CA is also the signing authority: they attest to the validity that they in fact created the certificate. Creation implies a process that vets the user to the CA; this process is executed by the *registration authority (RA)*. The RA is to also responsible for revocation and renewal requests.

In practice, there are many PKIs. Each large organization may have its own CA to issue certificates to its users. CAs may themselves be associated with a larger CA. Such an association would enable a user associated with one PKI to trust certificates from another PKI, as long as, for example, the certificate of the other PKI's CA, which is used to sign the certificates of the other PKI's users, is itself signed by that user's CA. This hierarchy ends with the *root CA*, which is self-signed: it is considered trusted. We will also speak of this root CA as a *trust anchor* or a *root of trust*, since other trusts derive from it. See Figure 5.8.

5.8.2 PKI Operations

We now discuss the fundamental operations of a PKI. First is registration. A user desiring a certificate needs to present their identity to the RA. This process almost always requires some physical validation to bootstrap the process.

The RA would then pass on the request to the CA. The certificate would be created, which includes the specific keying information needed for the specific algorithms, such as RSA or ECDSA. Because of the various choices involved, standards have been developed to capture the metadata needed (see the

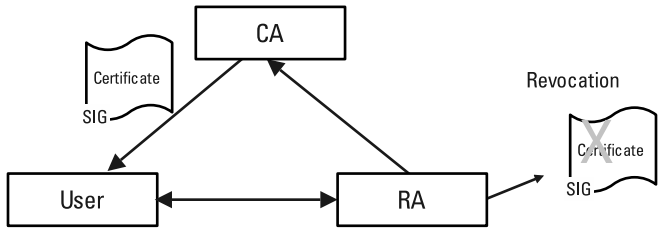

Figure 5.7 PKI elements.

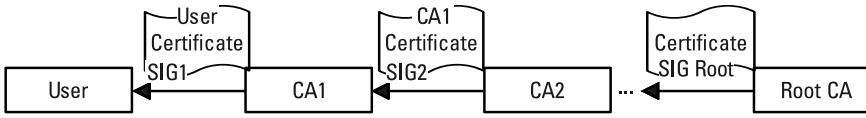

Figure 5.8 PKI hierarchy.

next section). The CA signs the certificate with its own information. Additional signatures from other CAs may be added, as in Figure 5.8.

Once obtained, the user can use their certificate in various protocols and applications. Such uses almost always need to use the public information from another user. That in turn requires that the one has obtained the certificate of the other user and verified it. Having a common hierarchy of certificates facilitates this process, as does methods for storage and query.

Certificates have to be renewed, and that requires interacting with the RA again. Current certificate lifetime are often measured in years, but some applications can use shorter lifetimes.

Finally, certificates need to be revoked on occasion. That implies that users need a way to check if another's certificate has been revoked and there are two main methods that are used. A *certificate revocation list (CRL)* is a list of those certificates no longer valid to use. CRLs require the continual update and bandwidth to send the list. The alternative is to use the *Online Certificate Status Protocol*, which checks the validity of the certificate using the internet. This is, of course, more timely than a CRL, but requires connectivity to the server since CRLs can be downloaded periodically to allow nonconnectivity.

5.8.3 Certificates

Fundamental to a PKI is a standardized way to represent information and in particular the public certificate. This certificate serves to give anyone the information they need to use that person's public key. The certificate needs to contain more than just the algorithm parameters. It should contain such items as the identity information of the individual (name, organization, etc.), validity period, and issuer's information. All this information should be digitally signed by the issuer, which requires knowing what algorithm was used for the signature. It also requires—separately, of course—the certificate of the issuer, which is signed, and so on.

One ubiquitous standard for certificates is the X.509 standard (see [13]). The standard dates from 1988, although several versions have been published; the latest as of this writing was in 2021. There are basically two types of certificates, in the spirit of the above discussion. The first type is the end user certificate. The second type is the issuer or CA certificate, which is used to sign other certificates. Eventually, a CA certificate will go up to a root certificate,

which is self-signed. The elements in an X.509 certificate include those in the previous paragraph [13]. In addition, the certificate can optionally have extensions, which can specify constraints on the usage of the certificate. The X.509 certificates can be put into a variety of standard formats.

The process that we mentioned of verifying a certificate using a signature by the CA, which in turn is verified by its CA, and so on, is called a *certificate chain*. Such chains end with the *trust anchor* (e.g., the self-signed root CA certificates). Trust anchor is an important concept that we revisit in discussing protocols in the second half of the book; basically trust in the complete process derives from trust in the trust anchor. There is nothing that prevents entities from having multiple certificates from different CAs (i.e., different PKIs). In fact, the two different chains can be linked, allowing for cross validation between the PKIs.

5.8.4 Important Points

This section only presents a very brief overview of PKIs. We stress a few points:

- The purpose of the PKI is to create and manage the information needed for public key cryptography including revocation;
- Those processes are accomplished by RAs to register users and CAs to issue the requisite keys;
- Registration requires independent verification of the user's identity, for example, by physical means;
- Besides issuing, the CA validates the user's information;
- Different CAs can validate other CAs, eventually ending with a root CA, which is fully trusted;
- The information in the PKI is encapsulated in certificates, of which X.509 is a popular standard.

5.9 Applications: Secure Email

The last application of public key cryptography we mention here is email security. While the protocols discussed in Chapter 6 highlight ad hoc sessions between participants that by and large rely on public key information, email security is in some ways simpler due to its one-way communication. This section also serves to tie together the three families of cryptography—symmetric ciphers, hashing, and public key cryptography—into a single application.

Email dates back to 1980 and the *Simple Mail Transfer Protocol (SMTP)*, although other message schemes existed before that. Indeed, once one has a

network, sending messages on that network is arguably an obvious first application. We do not need specific details about SMTP and focus instead on applying security goals to the email.

One of the first attempts at securing email was *Privacy-Enhanced Mail (PEM)*. PEM relied on a public key hierarchy with a single root with some fairly rigid requirements. As such, it was never adopted, although PEM as a file format still exists as a standard way to represent public key information.

Better and more common approaches are *Secure/Multipurpose Internet Mail Extensions (S/MIME)* [14] and *Pretty Good Privacy (PGP)* [15]. Both offer various cryptographic methods for email messages. They differ in the underlying infrastructure.

5.9.1 Goals

Here is the large set of security goals for secure email:

Confidentiality/privacy: The contents of the message should only be readable by the intended participant(s).

Authentication: The recipient should be able to trust that message came from the stated sender.

Integrity: The message contents should not be modified.

Nonrepudiation: The sender should not be able to deny that they sent the message.

Proof of Transmission: The recipient should have assurance that the message was sent.

Proof of Delivery: The sender should have assurance that the message was received.

Anonymity: It should be possible to send anonymous messages.

Revocability: It should be possible to revoke a message from being sent.

Some of these goals are clearly addressable by cryptography, while some rely more on the enterprise, for example, methods residing in the mail servers.

5.9.2 General Framework

We describe a general process for secure email, shown in Figure 5.9.

Suppose first that Alice has an email that she wishes to send to Bob. She wants the email encrypted so that the contents are private between her and Bob. And she wants Bob to be able to verify that the contents come from Alice and have not been modified. The process assumes that Alice and Bob have public/

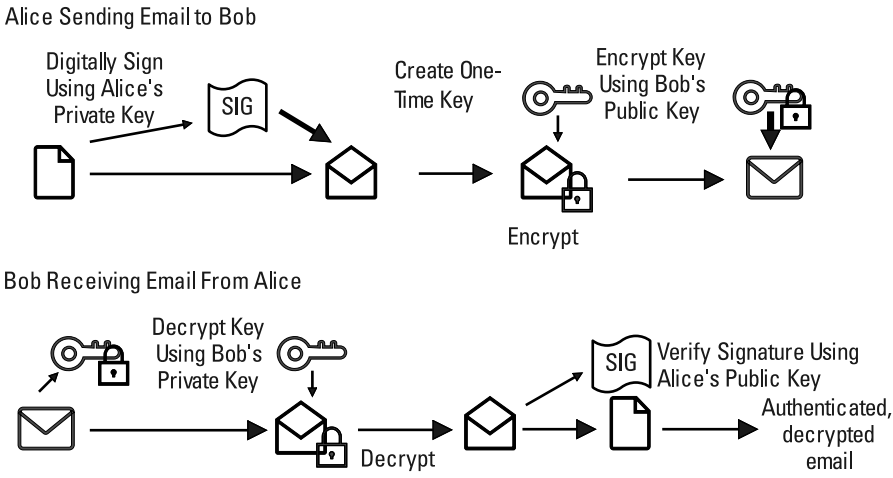

Figure 5.9 Generic secure email process.

private key information, and that they have shared the public key information with each other (e.g., through certificates).

The steps for Alice are as follows:

1. Alice digitally signs the email using her private key. We assume that the signature algorithm uses a hash (not shown) of the document for efficiency.
2. Alice creates a random one-time use key, often called a *session key*. This key is only used to encrypt the email.
3. Alice encrypts the email and signature using some symmetric cipher like AES and the one-time key.
4. Alice encrypts the one-time key using Bob's public key.
5. The encrypted key and encrypted message are bundled together as the final email sent to Bob.

When Bob receives the email, he executes the following steps, essentially reversing Alice's procedure:

1. Bob separates out the encrypted one-time key and the encrypted email.
2. He decrypts the one-time key using his private key.
3. Bob then uses the one-time key to decrypt the email and the signature.
4. To ensure the integrity and authentication, Bob verifies the signature using the email contents and Alice's public key.

The above procedure shows how all three branches of cryptography—ciphers, hashes, and public key methods—blended together to achieve a final result that gives confidentiality, integrity, and authentication. This process is inherently the same for modern secure email systems with perhaps differences in the algorithms and which specific parts are implemented.

For example, often only signed email is desired. In that case, the one-time key and encryption steps are skipped, and Alice does not need any of Bob's information. Note digitally signing also give nonrepudiation: Alice cannot deny having sent the email, since only she possesses the private key that generated the signature.

5.9.3 Nuances

There are two topics that we want to emphasize in the above email procedure. The first topic is the order of signing and encrypting. Figure 5.9 shows signing first, then encrypting. This order has the advantage that the signature authenticates the messages itself and ensures integrity of the message. In addition, if the message is stored later, the signature is still bound to that message, and thus still authenticates it.

Signing after encryption has the slight advantage that the sender's identity is confirmed before decryption. However, that does not necessarily authenticate the creator of the message: anyone could sign an encrypted message, perhaps forwarding. As such, the usual recommendation is to sign first, then encrypt as we have presented.

The second topic relates to trust. Alice and Bob need to share each other's public information. In practice, that often relies on sharing certificates, which in turn will leverage some PKI. Most large organizations implement secure email this way, and indeed depending on the trust in the various root certificates, interorganizational secure email is possible.

PGP takes a different approach in its *web of trust* protocol. The idea is that a given individual will trust certain parties enough that if they in turn trust someone—that is, vouch for their certificates—then so should that individual. In this manner, certificates can be verified as they are signed by more people, building up that web. This decentralized approach is useful when there is not a formal PKI to provide the supporting enterprise.

5.10 Foreshadowing: Navigation

Public key cryptography provides the cornerstone of many satellite navigation methods. The reason is the inherent asymmetry in satellite navigation: there is one service provider (e.g., the transmitters, and millions of users). To provide

for security services, the enterprise would keep its own secure private information and distribute public information to every user.

A main example would be to digitally sign information, such as the the messages on the navigation signals. That application is the subject of Chapter 8. Protecting the timing of the signal (i.e., the spreading code) is more subtle. One example is the Chimera protocol in Chapter 10.

5.11 Takeaways

Public key cryptography provides important primitives that cannot be met by symmetric key cryptography. It requires that an individual possesses a public key and a separate private key. The main application areas are:

- Encrypting information to someone using their public key, such that only they will be able to decrypt the information;
- Digital sign information using one's private key, such that anyone can verify the signature using one's corresponding public key;
- Key exchange, where two parties use each other's public information to create a shared random key that no one else knows.

There are several standard algorithms to achieve these goals, such as RSA, DSA and ECDSA, and the Diffie-Hellman key exchange. These algorithms are in the process of being replaced due to quantum computing concerns, as we discuss in the appendix.

Many applications use public key cryptography. In this chapter, we explored some novel uses of signatures and secure email, which blends all aspects of cryptography in one procedure. Finally, the use of public key cryptography requires a supporting enterprise to handle the participants public/private information: registration, creation, storage, distribution, and revocation. Such an enterprise is termed a PKI.

End Notes

1. The notion of a trapdoor function and one-way functions in general can be formalized using the machinery of complexity theory. See the discussion in the appendix. In particular, while some examples such as using factoring are assumed to be hard using classical computing, the existence of a theoretical one-way function is unknown and would imply $P \neq NP$ (see the appendix).

2. Euclid's algorithm (e.g., [2]) is a very efficient algorithm to find the gcd of two integers. Its running time is logarithmic in the bit length of the numbers. Euclid's algorithm can be

generalized to also compute the inverse of one number modulo the other number if the numbers are relatively prime.

3. Euler's totient function $\varphi(n)$ can be computed using knowledge of the prime factorization of n. Suppose $n = p_1^{k_1} \ldots p_\ell^{k_\ell}$, where the p_j are prime numbers. Then $\varphi(n)$ is computed using these two rules

$$\varphi(m_1 \cdot m_2) = \varphi(m_1)\varphi(m_2) \quad \text{if } \gcd(m_1, m_2) = 1 \tag{5.26}$$

$$\varphi(p^k) = (p-1)p^{k-1} \quad p \text{ a prime.} \tag{5.27}$$

   For example, since $91 = 7 \cdot 13$, $\varphi(91) = \varphi(6) \cdot \varphi(13) = 6 \cdot 12 = 72$.

4. For generators modulo a prime, the common practice is to choose the smallest value to use, such as 2, 3, and so on. It is unknown if 2, as an example, is a generator for an infinite number of primes. To find a generator for a given prime p, there are somewhat efficient methods given the prime factorization of $p - 1$. The concept of generator extends to other mathematical structures, such as the elliptic curves and the non-zero elements of any finite field.

5. The ElGamal cryptosystem uses a similar idea for public key based encryption to the procedure in this section for digital signatures [2]. The main commonality is the use of a one-time random number in encryption that serves to obfuscate the ciphertext, which consists of two parts. The two part ciphertext are combined in decryption in such a way that the randomness is canceled. This randomness is generated unilaterally by the encrypter and is not shared with anyone.

References

[1] Singh, S., *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography,* New York: Doubleday, 1999.

[2] Menezes, A. J., P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography,* Boca Raton, FL: CRC Press, 2001, http://www.cacr.math.uwaterloo.ca/hac/.

[3] Rivest, R. L., A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, Vol. 21, No. 2, 1978, pp. 120–126, doi:10.1145/359340.359342, https://doi.org/10.1145/359340. 359342.

[4] Moriarty, K., et al, PKCS #1: RSA Cryptography Specifications Version 2.2, RFC 8017, November 2016, doi:10.17487/RFC8017, https://www.rfc-editor.org/info/rfc8017.

[5] Kocher, P. C., "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," Advances in Cryptology-CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, August 18–22, 1996, Proceedings, Vol. 1109, Lecture Notes in Computer Science, Springer, 1996, pp. 104–113, doi:10.1007/3-540-68697-5_9.

[6] National Institute of Standards and Technology, Recommendation for Key Management: Part 1–General, Technical Report, NIST Special Publication 800-57 Part 1 Re-

vision 5, Washington, DC: U.S. Department of Commerce, 2020, doi:10.6028/NIST.SP.80057pt1r5.

[7] National Institute of Standards and Technology, Digital Signature Standard (DSS), Technical Report, NIST Special Publication 800-57 Part 1 Revision 5. Washington, DC: U.S. Department of Commerce, 2023, doi:10.6028/NIST.FIPS.186-5.

[8] Koblitz, N., *A Course in Number Theory and Cryptography,* Berlin: Springer-Verlag, 1987.

[9] National Institute of Standards and Technology, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, Technical Report, NIST Special Publication 800-56A, Washington, DC: U.S. Department of Commerce, 2018, doi:10.6028/NIST.SP.800-56Ar3.

[10] Zuccherato, R., et al., Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), RFC 3161, August 2001, doi:10.17487/RFC3161, https://www.rfceditor.org/info/rfc3161.

[11] Kawatsura, Y., Secure Electronic Transaction (SET) Supplement for the v1.0 Internet Open Trading Protocol (IOTP), RFC 3538, June 2003, doi:10.17487/RFC3538, https://www.rfc-editor.org/info/rfc3538.

[12] Oppliger, R., *Cryptography 101: From Theory to Practice,* Norwood, MA: Artech House, 2021.

[13] Housley, R., et al., Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC 2459, January 1999, doi:10.17487/RFC2459, https://www.rfc-editor.org/info/rfc2459.

[14] Roach, A., RTP Payload Format Restrictions, RFC 8851, January 2021, doi:10.17487/RFC8851, https://www.rfc-editor.org/info/rfc8851.

[15] Heinrich, C., "Pretty Good Privacy (PGP)," in Encyclopedia of Cryptography and Security (H. C. A. van Tilborg, ed.), Boston: Springer, 2005, pp. 466–470, doi:10.1007/0-387-23483-7_310.

6

Cryptographic Protocols

We finish Part II of the book devoted to cryptography by examining cryptographic protocols. Just as cryptographic algorithms are designed to compute a specific function to achieve various security goals, cryptographic protocols use algorithms to accomplish an assured mission between one or more participants. Protocols involve multiple steps, each one of which may involve some algorithm or process. Protocols are subject to their own type of threats. Indeed, we stress that while the constituent cryptographic algorithms may be secure, the protocols may fail.

We have already seen some examples of protocols when illustrating algorithms, such as using hashing for auctions, the Diffie-Hellman key exchange, and using TESLA to authenticate data streams. In this chapter, we delve into what it takes to make protocols: what are the properties to be defined and what is the supporting infrastructure? The latter has two important pieces. The first are methods to establish the proved identities of the participants. The second is to manage the cryptographic key material.

Once we present the infrastructure, we look at some specific protocols. The main focus is on the well-established network layer model [1]. This model is a modular separation of functionality in a network, from the physical medium of the network through establishing link between network nodes, routing information, and then protocols that transport the data needed by the higher-level applications. Considering these layers, their various and different security goals, and their cryptographic protocols to achieve those goals show the kind of thinking that is needed when designing protocols for other purposes. We finish the chapter looking at nonnetwork protocols for the same reasons.

The relevance of this chapter to satnav is twofold. First, the protocols that will be needed in the last part of this book often assume the use of protocols

here; for example, for the establishment of user identities. Second, these specific protocols are important to protect the satnav enterprise. The control segment is highly dependent on networks, and thus dependent of the network protocols and their security. We will point out these connections to satnav throughout the chapter.

6.1 Protocol Principles

This section describes the main features required for protocols. Unlike algorithms, protocols rely usually on multiple entities performing specific steps. As such, several assumptions for the protocols need to be met:

- Foremost, the protocol needs to address the specific task.
- Everyone using the protocol needs to understand their roles and all the steps being performed.
- As a corollary, everyone must agree on the steps being performed;
- These steps are well-defined;
- The protocol must adhere to outside constraints, such as policy;
- The protocol needs to cover all possible contingencies, and as such, have steps defined for those contingencies;
- In particular, the protocol needs to consider possible threats.

Given these assumptions, the ultimate goal of a protocol is to be robust in the sense that when these assumptions fail, the impact of that failure is known and minimized.

The steps involved can use cryptographic algorithms and primitives. Our underlying assumption is that the cryptography is strong; that is, such things as brute force and cryptanalysis are a nonissue. Even in the cases where the cryptography may be weak, it is possible for the protocol to meet its needs. Some of the navigation methods may be examples of this case; when that is so, we will show why the protocol is sufficient. As we will see throughout the discussion of protocols, many threats are possible that defeat the protocol's mission, even when the cryptography is strong.

Consider as an example the Diffie-Hellman protocol from Chapter 5. Alice and Bob know the steps they need to perform: creating a private random number, using that number to send a public value to the other party, then computing the key. These steps are well-defined. But as described, this protocol is subject to a man-in-the-middle attack. What steps should be in place to mitigate that risk? Perhaps Alice and Bob authenticate each other first

using public key certificates? Perhaps their communication channel precludes an eavesdropper? Perhaps there is a separate challenge response that can be performed? Accomplishing any mitigation will be through separate steps that need to be specified.

6.1.1 The Need for Trust

The entities in the protocol have a need to trust, either implicitly or explicitly, the other participants involved. They need to trust that the others perform the steps correctly. For example, the need to generate a random number to use in the protocol assumes the random number has required entropy and is unpredictable. Participants are assumed not to cause security goals to fail by their actions, for example by reusing nonces, although most protocols will have built-in methods to check for deviations. Handling failures of such trust is a difficult problem.[1]

Where does trust come from? Depending on the protocols, it may reside in the ethics of the parties involved with some safeguards put in place. Much of e-commerce relies on such trust foundations, along with the use of reputation or legal mechanisms to enforce trust if need be. The dual concept of trust is risk, which we discuss below: what is lost when trust is violated?

What are the implications for satnav? In public satnav enterprises, users will trust that the various segments do what there are supposed to based on the reputations involved, and implicit trust from government entities. The same is true to some extent in trust in the UE, which the user presumably has more control over, since they can always acquire a different UE. These trust assumptions allow one to infer that when behaviors deviate from expected results (i.e., when PNT solutions are incorrect) the problem lies not with the protocols in place in the enterprise but threats against those protocols.

6.1.2 Ensuring Trust

There are several general methods that protocols can use to ensure the trust requirement; see Figure 6.1. The first example is self-ensuring protocols. In these cases, trust failures, whether accidental or intentional, are detected by the protocol itself. For example, a key exchange protocol may include a challenge response to ensure that all parties agree on the key. If a failure is detected, the protocol should halt or reset. Self-ensuring protocols are easy to perform but difficult to enable handling the contingencies. They usually have higher communication and computation burdens, and such protocols need to protect against man in-middle attacks.

Some protocols may function as if they are self-ensuring but allow for an arbitrator if there is a dispute. E-commerce transactions fall under this type of

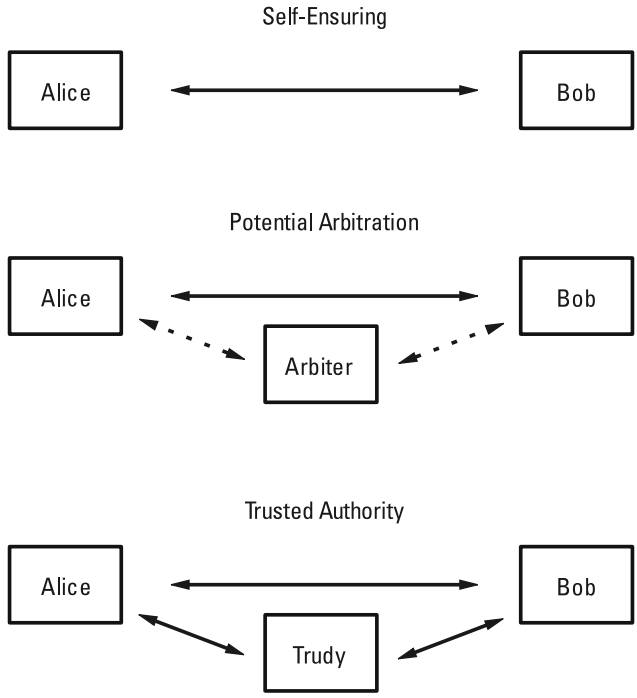

Figure 6.1 Ensuring trust in protocols.

protocol. The steps for online purchase and receptions of goods by and large happens between the two entities. When there is an issue, a third person is involved, say the hosting e-commerce site, which adjudicates. Of course, the entities have trust in that third party.

Finally, the notion of an arbitrator can be extended to a much more involved trusted authority, which we often call Trudy. An example of this idea is the certificate authority discussed in Section 5.8. All parties rely on such an authority to enable cryptographic features, such as certificates or shared keys, that they can then use. The many benefits of the trusted authority come at the cost of the expenses to maintain it, the possibility of it being a bottleneck, and the risk that it is a single point of failure.

6.1.3 There's Always Risk

What happens when a protocol fails? Risk depends on the likelihood of it failing and the resulting consequences. The likelihood will depend on failures and threats occurring that can't be mitigated. In most cases, the hope is that the protocol fails safely; for example, it just needs to be redone.

6.1.4 Trust in the Protocol

A fundamental question with protocols is how to be assured that the protocol steps do not leave something out. For example, can we trust that there is no information leakage or no vulnerability to replay? Such questions go beyond trust in the cryptography algorithms. As we saw in the discussion with cipher modes in Section 3.7, the underlying algorithms may be rock-solid, but there still may exist vulnerabilities from how they are used. There has been a lot of work in developing techniques to prove protocol security; see the essays in [2].

6.2 Identity Methods

Protocols consist of entities performing the prescribed steps. As a corollary, the entities must be assured that the participants are who they say they are. Trust in the identity of the participants may not be explicitly part of the protocol but subsumed implicitly by external methods. This section delves into what it means to establish identity, both noncryptographic methods that highlight the concepts and cryptographic methods.

There are close ties between the concepts of authentication and identity, which we mentioned in Section 1.3. Establishing identity uses authentication methods, and once trust in an identity is established, other trust follows. This idea will manifest in the latter part of the book, where achieving authentication of signals and data is the central goal of the PNT cryptographic protocols.

6.2.1 General Traits

Methods to establish identity have several goals:

- Foremost, of course, they should have a high probability of correctly identifying the true entity and a small probability of verifying someone who is not the true entity. This goal implies the standard trade-offs between detection and false alarm. Those trade-offs are challenging to do in practice for physical systems, such as various biometric scanners. Fortunately for cryptographic methods, we can bound the false alarm rate at the cost of some added computational complexity.
- The method should be resistant to attacks in the form of deceit (e.g., counterfeiting). Similar trade-offs that relate to false alarm rates apply.
- The method should be efficient in time and resources needed. Many complaints about secure identification methods are due to the extra time needed. As an example, consider captcha schemes.

- If the method is being used to establish the identifies of multiple users, then the usual system-level goals apply such as reliability, maintainability, and upgradability.
- Lastly, the method may need to accommodate transferring identity from one entity to another.

Noncryptographic methods are usually based on three factors: something an entity *has*, *knows*, or *is*. The "has" relates to something an entity possesses, such as keys, smart cards, or tokens. The "knows" pertains to information the entity, and presumably only the entity, has, such as passwords or pass phrases, or security questions. Finally, the "is" refers to traits the entities has as part of themselves, not only such things as various biometrics (fingerprints, retina scans, etc.), but also perhaps things only an entity can do, such as methods using keyboard typing.

Multifactor authentication derives from the goal to use more than one of these methods. Such use mitigates certain threats. For example, stealing a smart token will not help if a secret pass phrase is needed. Clearly additional factors provide greater security, but at the coast of increased time and complexity.

Noncryptographic identification methods have their uses in the satnav enterprise. Users within the various segments will almost surely require multifactor authentication even if cryptography is used. There has been research in fingerprinting navigation signals to achieve authentication of the signal analogous to regular fingerprints. Such attempts are a motivation of the Chimera protocol in Chapter 10.

6.2.2 Cryptographic Identification

Cryptographic identification, which is sometimes referred to as entity authentication, uses cryptography to establish the identity of someone or something. For example, recall digital signatures from Chapter 5. If Alice signs some document, like "I am Alice," then that identifies her in some sense, in that only she could have signed it assuming she alone possesses the signing key, and that the signing method is such that someone cannot fake (i.e., spoof) the signature. But her signature does not identify her with freshness: someone could replay the signature. What we desire instead is an interactive protocol to establish identity. Such protocols have the trait of *challenge-response*.

In these cases, Bob desires to establish Alice's identity but offering a challenge only Alice can respond to. We assume that Alice and Bob register beforehand, such as creating some cryptographic material, say shared keys or public keys. Consider Figure 6.2, where Bob issues the challenge to Alice, who then

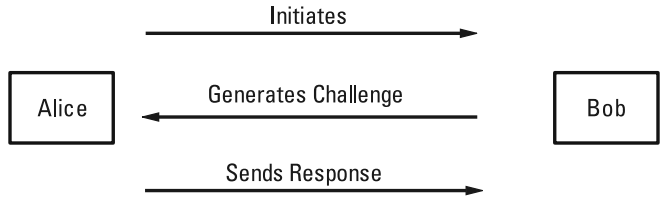

Figure 6.2 Generic challenge response.

offers her response. Bob verifies the response and concludes that it is Alice's identity. Notice that in this example, Alice never establishes Bob's identity. That asymmetry is perfectly acceptable in many settings; for example, a server may want to authenticate a user but the user may not need to authenticate the server.

We first consider symmetric methods, such as those shown in Figure 6.3. Assume that Alice and Bob share a key K. The steps are:

1. Alice shares a key K with Bob;
2. Bob generates a challenge random bit string c and sends it to Alice;
3. Alice computes the response $r = E_K[c]$;
4. Bob computes $D_K[r]$ and compares it with c.

The protocol verifies that Alice knows K, and thus verifies Alice's identity, assuming that K is unique to her and Bob.

This protocol relies on the properties of the encryption algorithm. A similar protocol can be made instead if Alice in the response computes a keyed-hash of the challenge string, say using HMAC. Bob would then confirm the response by computing the same keyed hash.

Observe that both of these protocols assume *freshness*: Bob is querying Alice to respond to a challenge at that moment in time. The protocols also assume that the challenge is random and only used once, to prevent replays by someone else observing the response. Failure of these assumptions would allow some threats to occur, namely someone impersonating Alice.

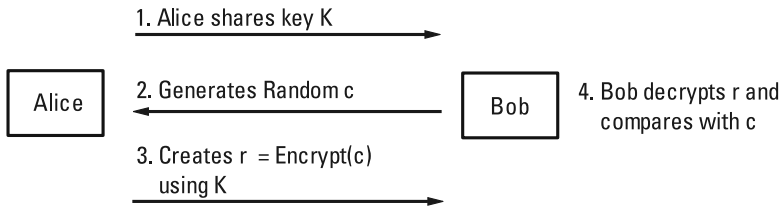

Figure 6.3 Specific challenge response using symmetric cryptography.

Asymmetric protocols are similar, except we assume that Alice and Bob share public key information, while protecting their private keys. One example is:

1. Bob generates a challenge random bit string c and encrypts it with Alice's public key PU_A before sending to Alice.
2. Alice receives $E_{PU_A}[c]$, decrypts it with her private key PR_A, and sends c to Bob.
3. Bob compares the received value of c to his original.

A similar protocol can be made with Bob sending c to Alice and asking her to sign it using PR_A; Bob would then verify the signature. Why would one choose one method over the other? Perhaps the public key method only allows for signatures, such as ECDSA.

Both the symmetric and asymmetric methods require that cryptographic material be registered beforehand. How that registration is accomplished requires additional protocols, which we discuss in the next section. Such protocols will often use trusted authorities.

The application of these methods for navigation is problematic because satnav is one-way broadcast. That is, there is no obvious way to do a challenge response. Some protocols, such as TESLA (Section 4.8) and signing the hash chain of keys do what amounts to a challenge response: the challenge from the UE to the SV is the receipt of the hash chain, which the SV must eventually sign.

6.2.3 Zero-Knowledge Proofs

The topic of *zero-knowledge proofs* consist of methods where one entity tries to prove another entity that they know a certain piece of information. In some sense, this question is what a digital signature does: proves that Alice knows the private key associated with the public key. However, zero-knowledge proofs are much broader in application; for example, Alice can prove to Bob that she knows a random string without revealing any information to Bob about the string (see [3, 4]).

Zero-knowledge proofs involve two entities: the *prover* and the *verifier* (the subject uses these terms rather than the ubiquitous Alice and Bob). The prover desires to convince the verifier that they know some information. The verifier desires to confirm that is the case. These are the goals for zero-knowledge proofs:

- *Completeness:* The property that the verifier will be convinced if the prover possesses the information;

- *Soundness:* The property that a dishonest prover cannot fool the verifier;
- *Zero-knowledge:* After confirmation, the verifier should gain no knowledge of the information.

In practice, that limitation on knowledge is probabilistic and can be made as small as desired.

There are many noncryptographic examples of zero-knowledge proofs. One popular example is "Where's Waldo?" In the well-known puzzle book, Waldo is obscured in a very busy picture. If the Prover wished to convince that they know where Waldo is, they could cover more than the whole picture except for a small cutout revealing Waldo's picture. This procedure would show that they know where Waldo is, and yet no information is revealed to his actual location.

For a cryptographic example, we give the *Schnorr identification scheme*; see Figure 6.4. Suppose that the prover knows a private key x and makes public $y = g^x$ modulo a large prime p. To convince the verifier that they know x, they do the following:

1. The prover generates a one-time random value r and computes $a = g^r \pmod p$, which they give to the verifier;
2. The verifier picks the challenge value c and sends it to the prover;
3. The prover computes $z = r + x \cdot c \pmod{p-1}$ and sends it to the verifier;
4. The verifier checks that $g^z = a \cdot y^c$.

Completeness is assured in that if the prover knows x, then the check will be correct. Soundness is achieved with sufficient entropy in the challenge, or perhaps repeating the procedure several times.

This scheme is reminiscent of the digital signature methods like ECDSA, but there is an important difference. Since the values r and thus a can be precomputed, the real-time computational requirements on the prover are

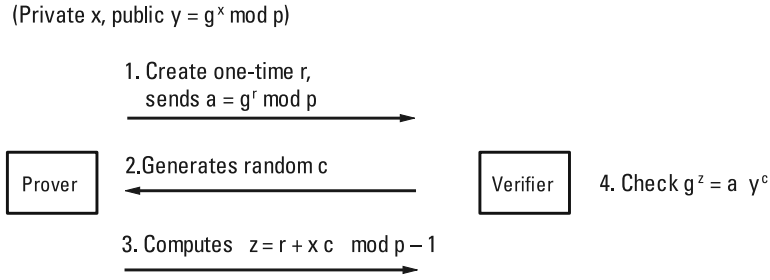

Figure 6.4 Schnorr identification scheme, a zero-knowledge proof example.

minimized. That means a constrained device with lower computational power can still provide a real-time identity proof. Such asymmetry in computational requirements is useful in many settings.

6.3 Managing Keys

Fundamental to any cryptographic protocols is the key material used by the participants. This material needs to be created, distributed, stored, maintained, and destroyed in secure manners. Several different frameworks have been established to accomplish these goals. This section delves into some for both symmetric and asymmetric cryptography.

There is a close relationship between the identity procedures in the last section and the keying methods in this section. Often this leads to a seeming chicken/egg conundrum: how can keys be established between parties if they have not verified identities, and how can identities be verified without exploiting keys? The answer is hierarchical: using some established infrastructure, such as PKI in Section 5.8, the parties can have some foundational material to then bootstrap the keying. This example highlights the other important feature of the protocols in this section: the keying material is usually ephemeral; long-term material is leveraged purely to create the short-term material, which is what is used for actual applications.

The relevance to navigation for the methods in this section is in a support role. Cryptographic methods used in navigation signals use keys that will be shared in ways that differ from ephemeral ad hoc sessions. However, enabling that sharing may involve methods outside the navigation enterprise. For example, if users need to use a system public key, the question of how they get that public key, and why they should trust it, will use procedures from other infrastructures.

Finally, we mention that we are considering key establishment between two parties. The problem of establishing a key among multiple users is known as *group keying* and is more challenging to do efficiently, as opposed to many one-to-one exchanges.[2]

6.3.1 Key Management Ingredients

We define *key management* as the set of techniques for supporting establishing and maintenance of cryptographic keying relationships between various parties. These techniques should encompass:

- *Initialization and registration of the parties.* These procedures are a precursor to the actual key generation, although they may involve creating long-term key material.

- *Generation, distribution, and installation of keying material for the parties.* These techniques are the focus of this section: given some foundational key material, how can the parties establish temporary keys?
- *Controlling the use of keying material.* Often this goal is in the hands of the parties (i.e., safeguarding the material). But some methods may have built-in time limits where the keys expire after a certain time.
- *Update, revocation, and destruction of keying material.* These techniques are the mirror of the initialization step; that is, the ability to revoke parties, and as a by-product, make their de-registration known.
- *System-level maintenance of key material, like storage, backups, archive when needed, and so forth.* Maintenance also applies to the necessary software and hardware consid erations. Clearly this goal depends on the nature of the use of the key material.

These techniques have associated security goals, goals that are assumed by the subsequent protocols. Key material, shared keys and private keys, need to be confidential and maintain integrity. Confidentiality may extend to other parameters, such as identifications of the parties. There should be trust in key management, which extends to trust in the processes and the identities of the parties. Unauthorized use of key material should be preventive. This goal goes beyond confidentiality and includes such things as preventing replay. Finally, there is the underlying need for authentication of all the entities involved.

At the system level, key management includes the usual processes and procedures, such as well-defined practices and roles/responsibilities, accounting/auditing, and periodic reviews. Such considerations are beyond the scope of this book, although we will point out where they will need to be specified when appropriate. It is important to remember that key management considerations can be crucial for the difference between the success and failure of protocols and systems that use them.

6.3.2 Public Key Methods

In some ways, the key exchange problem for parties that share public key information is straightforward from what we saw in Section 5.8. If Alice and Bob are in the same PKI and have certificates that they each can verify, then they can establish their identities and proceed to use something like Diffie-Hellman to establish a key. The issue is one of timeliness: how long should they trust the other's certificates? That, of course, is the problem of certificate revocation, which we discussed in Section 5.8.

Another approach would be to have a trusted third party in real time authenticate Alice and Bob by verifying their public keys, which would allow

them to then establish a key. Consider as an example the protocol in Figure 6.5 due to Denning and Sacco [5]. Here we denote the trusted third party as Trudy (also known as an authentication server). The steps are:

1. Alice sends her and Bob's IDs to Trudy.
2. Trudy signs Alice's ID, public key, and a time stamp. She also signs Bob's ID, public key, and time stamp.
3. Alice then forwards that information to Bob along with the new shared session key K_S signed with the time stamp by Alice, all of which is encrypted to Bob using his public key.

At this point, Alice and Bob have the key. Not shown are follow-on steps that can ensure they agree on the key; for example, Bob could send a challenge encrypted with K_S, which Alice can then decrypt and send back.

The use of time stamps requires some level of time synchronization. That requirement can be removed with the cost of extra steps and nonces. An example is the Needham-Schroeder-Lowe protocol, first proposed by Needham-Schroder [6] and then improved by Lowe [7]. The complete protocol is shown in Figure 6.6. The steps are:

1. Alice sends her and Bob's IDs to Trudy.
2. Trudy signs Bob's ID and public key and sends back to Alice.
3. Alice now contacts Bob, sending her ID and a nonce N_A encrypted with his public key.
4. Bob now contacts Trudy sending Alice and Bob's IDs along with N_A encrypted with Trudy's public key. This latter item protects N_A from ever being exposed.

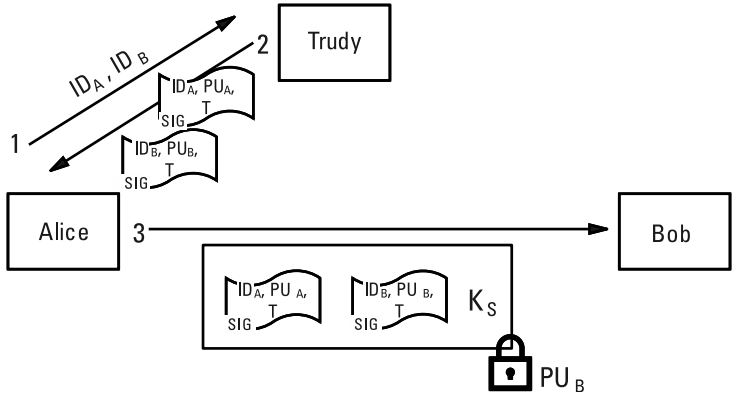

Figure 6.5 Public key verification using time stamps.

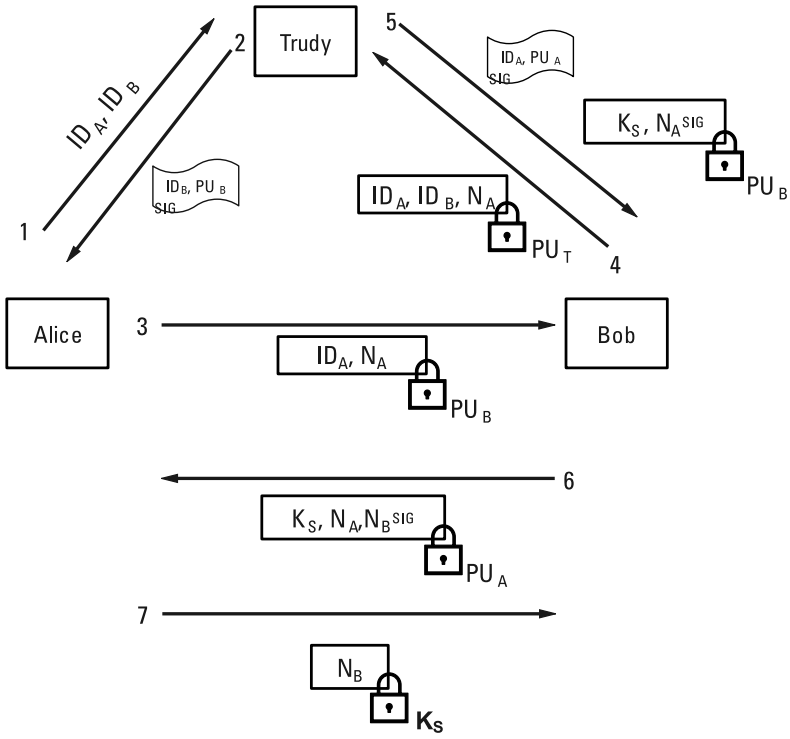

Figure 6.6 Public key Needham-Schroeder.

5. Trudy now returns to Bob a signature of Alice's ID and public key. She also sends encrypted with Bob's public key N_A, a session key K_S, and Bob's ID along with Trudy's signature of those items.
6. Now Bob can get back to Alice by encrypting with Alice's public key those items signed by Trudy, Trudy's signature, and a nonce that Bob creates, N_B.
7. Finally, Alice sends to Bob N_B encrypted with K_S.

We point out a few subtleties in this protocol. First, note the use of Bob's ID in step 5. It would be naturally to think this item is not needed; after all, Alice initiated the protocol to Bob. However, if Alice were to start this protocol with someone else, say Eve, then Eve in theory could send the messages to Bob pretending to be Alice. That fails to work because of Step 5 and 6, which would cause Alice to realize that she is not talking to Bob. This was the fix found by Lowe [7]. Second, the protocol is designed to share K_S, but the protocol works fine if all that is desired is to authenticate each other's public key. The final challenge response can be used using the public key information.

6.3.3 Symmetric Methods

The methods in this section also use a trusted third party, often denoted as a *key distribution center (KDC)* (although we will still use "Trudy" to show parallels to the last section). The premise is that Alice and Bob share a relationship with the KDC (e.g., long-lasting shared keys). The goal is for Alice and Bob to establish a shared session key.

We first give the Needham-Schroeder protocol for this case; see Figure 6.7. The keys shared by Alice and Bob, respectively, are K_{TA} and K_{TB}. The steps are:

1. Alice sends her and Bob's IDs to Trudy along with a nonce N_A.
2. Trudy sends the following encrypted with K_{TA}: Bob's ID, N_A, and session key K_S. She also sends K_S and Alice's ID encrypted with K_{TB}; Alice, of course, cannot decrypt this.
3. Alice forwards the items encrypted with K_{TB} to Bob.
4. Bob creates the challenge nonce N_B and sends it to Alice encrypted by K_S.

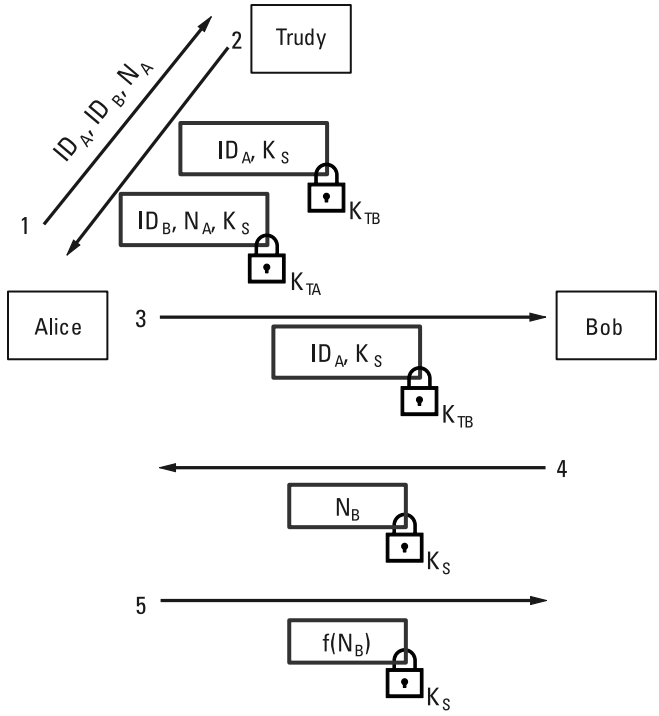

Figure 6.7 Symmetric key Needham-Schroeder.

5. In response, Alice performs some simple function on N_B, like incrementing by 1, and encrypts it with K_S to send to Bob.

After the protocol, Alice and Bob have a key that they know is shared with each other and that no one else knows.

As written, this protocol has the vulnerability that if K_S were ever compromised, then someone could replay the steps to get Bob to assume that it is a fresh session key and use it. One way to fix the protocol is to add yet another nonce. Basically have Alice first contact Bob, have Bob reach out to Trudy with another nonce, and then use the information from Trudy to establish the key with Alice.

Another fix, which we focus on, was proposed by Denning [5] and uses time stamps. Basically, instead of nonces in the above protocol, one uses synchronized time stamps to protect against any replay attacks. Synchronization of clocks require some bound that takes into account such events as delays in travel; the synchronization requirement bounds which replay attacks could be possible. Since some protocols have a very course granularity, such as generating a session key good for a day, the synchronization requirement is usually not that stringent. Of course, the reliance on timing of any sort shows the need for trusted PNT.

An important use of Needham-Schroeder using time stamps is *Kerberos*. The Kerberos infrastructure is shown in Figure 6.8. It comprises an *authentication server (AS)*, a *ticket granting server (TGS)*, and users. Users register and have a shared key with the AS. The system is set up so that the user can connect with any service that is under the purview of the TGS, for example, like a client connecting to a server. The virtue is that the client does not share a key with the server, and yet they will agree on a session key that secures their session. Kerberos was developed by the Massachusetts Institute of Technology (MIT) in 1994 [8].

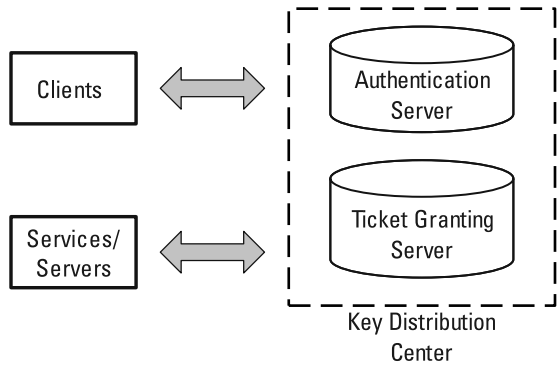

Figure 6.8 Kerberos system.

The steps that make Kerberos work is the merging of two separate Needham-Schroeder protocols using time stamps. The first protocol establishes a shared key between the user and TGS using the AS as the trusted third party. The resulting key is bundled with other information, such as the time limit, identities, time stamp, to yield what is called a *ticket*. This ticket is now available during its validity time to connect with any service associated with the TGS. For example, if the user wants to connect to a server, it uses the TGS as the trusted third party in a Needham-Schroeder protocol to establish a key between them and the server. In this manner, a user could, for example, easily log into any server on a network without sharing keys ahead of time. These steps are summarized in Figure 6.9, where the numbers 1,2,3 parallel the steps in Figure 6.7.

6.3.4 Comparisons

Both public key methods and shared key methods can work to create a framework to let users establish session keys. PKI-based methods are more versatile in that they can be readily extended. However, they are somewhat more computationally intensive, and certificate validation needs to be considered. Shared-key methods using a KDC, like Kerberos, are more efficient but the KDC or AS represent single points of failure and could be a bottleneck. As with most things with security, these are trade-offs to be considered that go beyond the assurance of the protocols.

6.4 The Network Stack

Cryptographic protocols can be used to achieve a large variety of security goals. This section illustrates that breadth by looking at the standard layered architecture for communication networks and show how protocols manifest at the

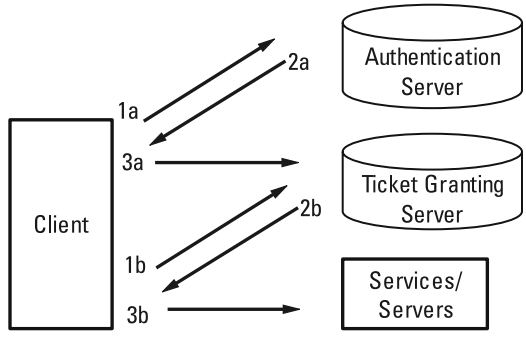

Figure 6.9 Kerberos steps.

different layers. We discuss the security goals inherent at each layer and give examples of protocols that achieve them. These protocols are also applicable to satnav, in that they are vital for securing aspects of the satnav enterprise.

6.4.1 The Network Stack

A typical communication network is shown in Figure 6.10. The various nodes share information with any other nodes in the network. The network must support establishing links between neighboring nodes and the ability to route information between nodes on the networks. Each network could be interconnected (internet) with other networks. All the information is ultimately transmitted over some physical channels. A good reference on networking is [9].

A layered architectures for network divides functionality into separate layer, such as the physical medium or routing. We will focus on a simple version the Open Systems Interconnection (OSI) model, shown in Figure 6.11 [1]. The simplicity comes from grouping the upper layers, such as presentation and session layers, into a single application layer. Each of the layers is described separately below.

The advantages of a layered approach is it achieves modularity. Solutions, whether for performance or security, used at one layer can be updated without usually affecting other layers. A simple example is the lowest physical layer: Internet traffic moves seamlessly over wired and wireless backbones without any change to the higher-layer network protocols themselves. The disadvantage of a layered approach is efficiency. While performance may be optimized at each layer, there is no guarantee that the overall result is optimized. Similarly, security goals at one layer may also imply security goals at another layer, but regardless, protocols for security may be in place at every layer. Indeed, it is common to have any given bit of information at the application layer to be ultimately encrypted three or four times in succession through the network stack.

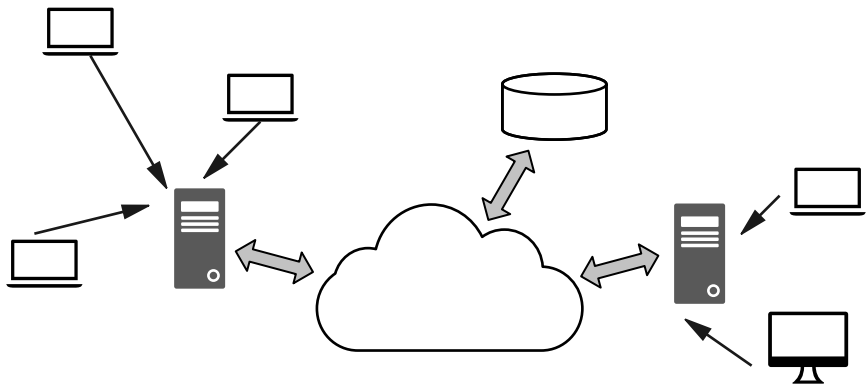

Figure 6.10 Communication networks.

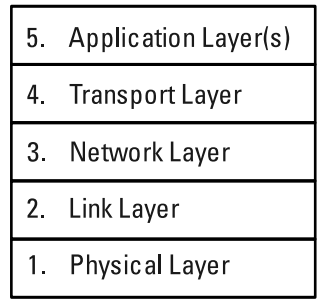

Figure 6.11 OSI network layer model (simplified).

The layered approach to defining the architecture is useful for satnav. For example, navigation signals have the signal and data layers, as we saw in Chapter 2. While design at each layer is often specified in a single interface specification, there can be reuse, such as using the same data format on different signals.

6.4.2 Layer 1: The Physical Layer

Communication networks send information between network nodes, and ultimately this in formation needs to be sent over some physical medium. There are many, many choices for the media: RF signals, fiber, copper wires, acoustic, laser pulses, and so on. Essentially, any medium that allows some way to modulate information could serve as the physical layer.

In a network, multiple entities are using the same physical medium, which then begs the question of how they can use it simultaneously without interfering each other. This question is the *multiple access problem*, and several methods have been developed. We highlight three of them:

- The simplest method to allow multiple access is to force entities to participate only during preassigned times. This method is called *time-division-multiple-access (TDMA)*. It was used in cellular system Global System for Mobile Communications (GSM).
- RF signals occupy some portion of the RF spectrum, and one method to control access is to assign specific frequencies. This method is called FDMA. An example is Wi-Fi, which assigns a specific channel to a user.
- The final example is called CDMA. The idea behind CDMA is to assign to each user a random-like variation of the physical medium with the goal that each user looks like random noise to the others. Thus they are in fact allowed access the same resources at the same time, but have a bounded affect on each other.

The multiple access problem suggests the fundamental assurance goal for the physical layer: availability. How can users ensure that they have access? In some ways, this problem is difficult in that an adversary could always physically impede the medium: cut the wire, block a laser, or jam RF. Cryptography can't do anything against such attacks.

However, cryptography can help prevent other attacks on access. The idea is to cause variation in the medium that is cryptographically controlled. For example, one form of FDMA is *frequency hopping*, where the signal changes its frequency in time. That change could be controlled by cryptography, for example, using the ciphertext generated from AES to select the hop frequencies. Similarly, cryptography could be used to create the noiselike signals in CDMA.

What about other goals like confidentiality, integrity, and authentication? These goals pertain to the information being modulated on the signal and thus belong to the higher layers in the protocol stack. Likewise, the goal of limiting access to the physical medium itself is based on control protocols at the higher layers.

As we saw in Chapter 2, the physical medium for satnav is the RF signals broadcast mostly in L-band. The issues we mentioned apply (i.e., that the signal is easily jammed). Access is for the most part given by CDMA using spreading codes, although GLONASS did use a form of FDMA. Cryptography applies to this layer for satnav as we discuss in Chapter 9.

6.4.3 Layer 2: The Link Layer

Layer 2 of the network protocol stack concerns sending data directly between network nodes; that is, establishing the link between two nodes on the network. There are a host of various standards under *IEEE 802* that define how to make this link, from Ethernet to wireless local area networks (LANs) and personal area networks (PANs). These standards consider this layer divided into two sublayers:

- The *Media Access Control* sublayer, which is responsible for modulating the *data frames* onto the physical media. (We will not use MAC as the abbreviation for "media access control" since we already use MAC for "message authentication code.")
- The *Logical Link Control* sublayer, which is responsible for handling the flow of the data frames.

To illustrate how cryptography can be used in this layer, we focus on the wireless LAN standard 802.11 known as Wi-Fi. The overview of 802.11 is shown in Figure 6.12. We consider the case where there is an access point (AP)

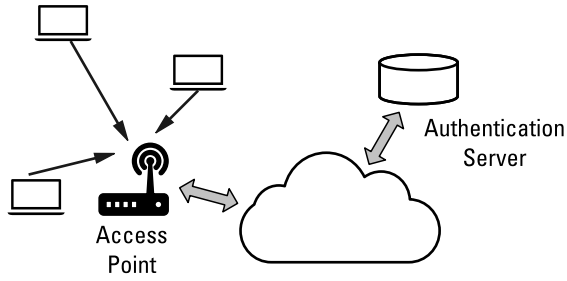

Figure 6.12 Overview of Wi-Fi.

that several users want to access. The AP will be used to connect to other parts of the network. Part of that connection will possibly be to reach back to an authentication server to validate the users. A user wants to establish a link to the AP, which would allow them to likewise access the network.

Cryptography will be used to achieve the following security goals:

- *Authentication of the AP.* A user desires to connect only to the legitimate AP.
- *Authentication of the user.* The network may desire to control access of who may connect. Note that open access networks, which would allow anyone to connect, may be desirable in some contexts.
- *Confidentiality and integrity of the data frames.* Only the user and AP should be able to read the data frame and ensure it has not been modified. Methods used to achieve this goal can also ensure that data frames are not replayed.
- *Availability.* Adversaries should not be able, for example, to disassociate a user from the AP by mimicking their data frames.

Authentication between the entities uses some type of identity protocol similar to what we saw in Section 6.2. See the entities in Figure 6.12. The user initiates the session with the AP, who in turn queries the authentication server. The process will involve some challenge response from the authentication server, likely using some shared secret information such as passwords. Once the user is authenticated, they will be allowed to associate to the AP. The authentication server also allows the user to ensure that they are not connecting to a rogue AP. The complete process establishes shared keys between the user and the AP.

Once the keys are established, the data frames can be encrypted and authenticated. The current protocol is called *Wi-Fi Protected Access 3 (WPA3)*. AES is used in different modes depending on the desired strength. Strongest would be 256-bit AES in GCM mode using SHA384 for HMAC. Not only

are the data frame payloads encrypted and authenticated, but the media access control address is also authenticated, which prevents data being hijacked. There is a long history of Wi-Fi encryption, with earlier methods being egregiously broken.[3]

6.4.4 Layer 3: The Network Layer

Communication networks exist to transfer information between nodes, which may not be directly connected by a specific link. That purpose requires that information can be put into packets and routed from the source to the destination, going through several links along the way. The purpose of layer 3, the network layer, is to provide this routing. It serves as the interface between receiving the information from higher layers to be routed and sending that information through each link.

We focus on the most used protocol for the network layer, the *Internet Protocol (IP)* [1]. IP defines the network packets to be routed along with the header information needed for the routing. The IP packet for IPv4 is indicated in Figure 6.13. It comprises header information, the source and destination address, which are 4 bytes each, and then the payload data. The IPv6 packet is similar, with the important feature of larger fields devoted to addressing (16 bytes).

Similar to layer 2, there are several security goals:

- *Correctness of the routing.* Packets should go where they are intended.
- *Confidentiality and integrity of the packets.* One should be able to ensure the packet contents have not been observed or modified.
- *Availability.* Packets should get to their destination.

There are also security goals related to the network routing.[4]

Achieving these goals can be done using good practices and correct implementation, such as to mitigate threats against routing and denial of service. Security of the packet content is achieved by the IPsec protocol. IPsec considers two protections on the packet and two uses, so four possibilities in all. The two protections are called *Authentication Header (AH)*, which provides authentication of the header and payload contents, and *Encapsulating Security Payload (ESP)*, which provides confidentiality through encryption along with au thentication. The two uses are first *transport* mode, which protects the packet

Figure 6.13 IP packet.

contents but not the outer address. *Tunnel* mode protects the entire packet, basically treating it as the payload in another packet.

The four possibilities are summarized in Figure 6.14. The cryptographic functions are applied to select parts of the original packet, depending on the purpose and use; basically authentication or encryption is applied to all the information to the right of the indicated field. The specific algorithms have a large variety of choices, but in practice commonly used algorithms are AES and HMAC using SHA2 hashes.

The use of cryptography implies keys, and IPsec has as part of its architecture key exchange methods under the *Internet Security Association and Key Management Protocol (ISAKMP)*. There are several possible methods that can used, notably *Internet Key Exchange (IKE)*, which uses certificates to establish identities and then Diffie-Hellman for the key exchange. Before ISAKMP can be used, security associations need to exist between the participants, which specify which algorithms are permitted, their certificates, and so on.

The use of IPsec allows for creating *virtual private networks (VPNs)*. For example, using IPsec between a remote user and an enclave network gateway would make that user for all intents and purposes be as if they were within the enclave. In particular, network traffic between the user and anyone else within the enclave could be made unobservable outside. IPsec also allows for control of the packets at firewalls, permitting only allowed information to go past network boundaries.

As with the link layer, the network layer impacts satnav in its ubiquity in communication networks, and thus its vital importance to the satnav enterprise.

6.4.5 Layer 4: The Transport Layer

The transport layer serves as a bridge between user applications and the network. The purpose is to take the data generated from applications and set up

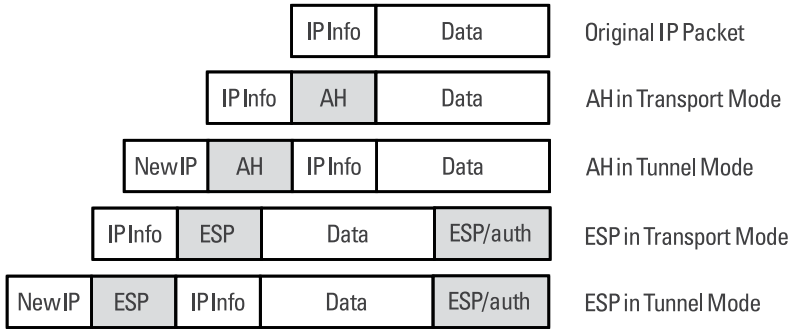

Figure 6.14 IPsec uses.

sessions with information that then is transported through the network. There are two main protocols that we consider:

- The *Transport Control Protocol (TCP)* is a connection-based service between two entities, denoted client and server. TCP is designed for reliable transport: data will arrive eventually error-free and can be put in the correct order. Since it is connection-based, there is an initial handshake to establish the connection.
- The *User Datagram Protocol (UDP)* is connectionless, which means there is no handshake and reliability is not guaranteed. UDP is useful for applications where it is better to lose information then have information be out of order or arrive late, for example many real-time functions.

We first focus on securing TCP and then look at UDP. A useful illustration for TCP is web traffic on the internet. The security goals for TCP are similar to those for the lower layers. Data should have both confidentiality and integrity. The connection should be able to be established and not subject to denial of service. Finally, the parties in the connection should be able to authenticate each other. Not all of these goals need be true, that is, sometimes authentication is only one-way, say a client authenticating a server.

The original protocol to secure TCP was called *Secure Sockets Layer (SSL)* [1]. It has now been replaced by TLS, developed by the Internet Engineering Task Force (IETF). TLS is a protocol that rides on top of TCP. TLS's protocol stack is shown in Figure 6.15. The main component is the record protocol, which bundles the information to be sent after cryptographic processing. The record protocol establishes the secure connection and provides the confidentiality and data integrity.

The record protocol operates on *fragments* of data; the action on fragment is shown in Figure 6.16. The main points are that the data is first fragmented. Each fragment is then possibly compressed, a MAC is created for authentication, and then encryption is done before transmission with its header. These

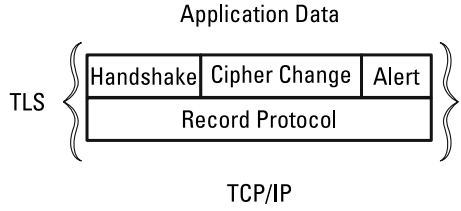

Figure 6.15 TLS protocol stack.

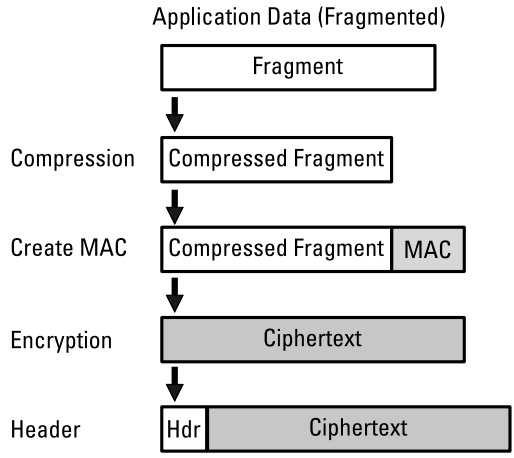

Figure 6.16 TLS record protocol.

cryptographic operations use symmetric keys generated in each session. The specific algorithms are highly tunable. Typical algorithms are AES/GCM, RSA, and elliptic curved Diffie-Hellman.

The other three subprotocols are specific to controlling the overall protocol and are within the record protocol. The choices for the algorithms and the required keys are part of the negotiation in the handshake subprotocol. The change cipher subprotocol is used to notify the overall protocol to updates its cipher suite. The alert subprotocol sends either fatal or warning alerts when something problematic is found in the protocol.

The handshake protocol begins with mutual "hello" messages. After that certificates are exchanged. Those can be two-sided or one-sided (e.g., often a client just desires to authenticate the server). Once the certificates are verified, the key exchange occurs, for example using Diffie-Hellman. The handshake process can also be performed using preplaced shared keys. In such a case, there is still mutual authentication using challenge-response methods, and a session key is established.

Note how TLS relies on TCP (i.e., a reliable transport of the data). How then can connection-less UDP traffic be secured? There are protocols in place, such as *Datagram TLS (DTLS)* and *Secure Real-time Transport Protocol (SRTP)*. They establish keys first, and then the data is protected using usual cryptographic methods. The latter can handle the loss of data.

As with the other protocols in the network stack, the importance of TLS to satnav is to the satnav enterprise; see Chapter 7. The main usefulness is the ability to create secure both ad hoc sessions for established relationships to exchange data in the enterprises and users.

6.4.6 Upper Layers: Applications

The upper layers take the data from applications and bundle it for, for example, TCP/IP. Perhaps the most widely used example is HTTPS, which is the secure protocol for *Hypertext Transfer Protocol (HTTP)* (i.e., almost all web traffic). HTTPS is just HTTP riding on top of TLS; that is, the TLS record protocol is used for the transport of the HTTP information, with TLS also establishing the connection.

Another example is secure email. We saw how to secure email in Section 5.9. However, if those methods are not available, it turns out the email may still be secured over most of the transport. The method is called *opportunistic TLS* (sometimes called STARTTLS), and the idea is that TLS sessions are established between the email servers when needed. Since it was proposed, opportunistic TLS has become widely used.

6.4.7 Observations

We stress a few things from the discussion on the network stack:

Network Stack Important Points

- The network stack secure protocols are modular. For example, both secure UDP and TCP traffic can ride on IPsec protections.
- The protocols have commonalities, like establishing a session and then applying cryptography.
- Security goals vary by the layer. For example, the network access goals of layer 2 are not relevant at layer 4.
- Attacks also vary by layer. One example is that denial-of-service attacks are different depending on the layer.
- Finally, while cryptographic methods may seem redundant—encrypted email over TLS over IPsec over a WPA3 link encrypts each bit four times—the performance is usually not an issue. In turn, one gains the benefit of protections from attacks that could be focused on specific layers.

Summary

The suite of protocols used in network communications is far-reaching and omnipresent. They give examples for how all of the cryptographic primitives we described earlier can be applied.

These protocols also are good examples of where the impact of quantum computing described in the appendix will come in. Basically, any protocol that uses nonquantum-resistant public key methods would need to be updated; indeed, TLS has already started adapting new algorithms.

6.5 Other Protocols

Since cryptography is designed to protect information, cryptographic protocols are used in other settings besides the networking. We highlight a few and offer some insights.

6.5.1 Other Communication Protocols

Devices and users communicate in other ways besides using TCP/IP. One example used to be the cellular networks. Specialized cryptographic techniques and algorithms were developed in 2G and 3G. The cellular system security goals are interesting for their asymmetry. From the cellular enterprise perspective, primary goals are availability, so that the use of the system is attractive to consumers, and authentication of the end devices, to ensure a revenue stream. From the user's perspective, besides availability, they desire privacy of their communication. Methods to accomplish these goals included challenge response using shared keys, keys stored in the user's device, and encryption of the voice streams and data, although data services were just emerging in these generations of cellular. As the world has moved to 4G, 5G, and beyond, cellular has gone from circuit-switched to packet-switched networks, and as such there has been a convergence with methods in the network stack.[5]

More local communication involves PANs. One prime example of a PAN is Bluetooth (formerly 802.15.1, now managed by the Bluetooth Special Interest Group). The Bluetooth standard is its own protocol stack, from defining a physical layer that uses frequency hopping to data transfer between connected nodes. Cryptography manifests in an initial session key generation based on shared master keys between the two devices, such as a PIN. The resulting key is used to encrypt the data stream. The Bluetooth security architecture can be found in [10].

Communication at the application layer often has its own suite of protocols regardless of the underlying transport mechanisms. One example is email security, which was presented in Section 5.9. In recent years, applications designed to share messages between users have arisen along with accompanying security. One example is Whatsapp, which provides private messaging. It uses elliptic curve methods to authenticate users and key exchange, with AES and HMAC used for the actual private messaging [11, 12].

6.5.2 Internet of Things

The notion of IoTs is the emergence of having almost any electronic devices, such as sensors and control objects, connected to a network. The network, despite the name, does not have to be the worldwide internet, but rather it represents some connectivity between the remote device and the users of the device.

IoT devices have security goals besides the expected confidentially, integrity, and availability. Access and control is paramount. Only legitimate parties should be able to access these remote devices, which are often not under direct control of the users. Failure would mean having a remote device with access to one's network. Similarly, if the IoT devices serve as controls, such as a thermostat, then protections need to be in place to make sure that false commands are not executed.[6]

On the one hand, the protocols discussed in the previous section for the network stack apply to these devices. In particular, TCP/IP-based protocols and security are relevant. There are nuances, though, in that IoT devices are often very resource constrained in power and data bandwidth. As such, while the cryptographic protocols may be similar, the desire is to use lightweight algorithms when possible, and indeed the development of lightweight cryptography is motivated by IoT; see Section 3.6. On the other hand, some specific protocols have been developed, such as the 802.15.4 family, which includes Zigbee [13] and 6LoWPAN [14].

The IoT security problem is relevant to satnav since these devices will want secure PNT, which will be provided by satnav. Thus, protections in these devices could possibly be exploited to achieve PNT security goals, for example as in the transfer of data for assisted GNSS; see Chapter 8.

6.5.3 Zero Trust

We finish this chapter on cryptographic protocols to mention one important trend in assurance philosophy called *zero trust*. The idea behind zero trust is not to trust end devices by default, but rather force regular verification of these devices, which must be (re)established for every transaction. This notion is in opposition to the Cadbury egg or M&M models of perimeter security.[7]

The operational concept for perimeter security is to screen at perimeters, keep bad things out, and then allow validated processes inside a perimeter fairly wide access. If additional privileges are needed then a new perimeter is encountered and validation is performed. This model reflects the idea that once an entity has validated itself to a perimeter defense, it is trusted. Thus, if a perimeter is breached the damage can be very significant. A reflection of this is that most attacks proceed by escalation of privilege, and this escalation is enabled by the fact that the threat is allowed to be persistent in the perimeter that they have breached.

Zero-trust counters this philosophy with several changes. One change is to assume that breaches will occurs and limits the damage once a breach has occurred. There are also the concepts of least privilege access and explicit verification. Several important technologies enable zero-trust, such as (distributed) identity management and the ability to detect abnormal patterns of activity, a

data-science approach to activity monitoring [15]. Zero-trust methods put a slight burden on the user in terms of the regular authentication, but it mitigates having rogue devices/end points going undetected. See [16] for NIST's zero trust architecture. The applicability to satnav depends on whether the mutual authentication can take place, since such authentication presupposes connectivity; we discuss the use of communication for satnav in the next chapter.

6.6 Takeaways

This chapter serves as a summary of Part II on cryptography. Cryptographic protocols use the three families of algorithms—symmetric ciphers, hashing, and public key methods—and combine them to accomplish various security goals. The protocols comprise a series of steps and interactions between participants, and as such rely on some fundamental assumptions. The participants must agree on the steps and trust that everyone adheres to them or have the protocol account for deviations.

We showed various ways that participants can perform mutual authentication to verify their identities. These methods often use a variation of challenge response, where one party offers a challenge that only someone with the requisite cryptographic material will be able to respond to. At times, these protocols may use a trusted third party. Related to the fundamental problem of verifying identities is to establish the key material needed for the protocol. This key material can be established using symmetric cryptography, such as with Kerberos or with a PKI. It is good practice that the protocol key material is used for that session, while longer-lasting keys are used just to create the short-term material.

We focused on the network stack to illustrate the use of the protocol sessions. Each of the layers has its own security goals, which cryptography can ensure. The commonality is to create a connection and session at the layer, which uses identity, establishes keys, and then protects the data being exchanged for confidentiality and privacy. Other protocols work similarly, and indeed essentially any communication protocol will be secure using the same sort of methods.

End Notes

1. A fundamental assumption in the cryptographic protocols is that the parties share the same security goals, specifically the desire to achieve those goals. When one of the parties violates this assumption, for example by actively working to prevent those goals, we call that an *insider threat*. Examples include leaking confidential information, falsifying records, and so on. Preventing insider threat is a difficult problem, and most techniques involve detection, whether by monitoring actions or the flow of information, such as on a network.

2. The group key problem has two main aspects. The first is how to create a key that is shared by multiple users. While such sharing is problematic from a trust standpoint— all users must trust that the key is protected by everyone—efficient methods have been developed in the various protocols. The second aspects is related in that we want an infrastructure that allows such sharing to be updated as needed. The survey paper [17] discusses several different methods. The idea is to give users a separate key material that is only used to protect the distribution of keys. The structure of this auxiliary key material is where the various methods differ in their use and efficiency.

3. The first attempt to define security in Wi-Fi was with *Wired Equivalent Privacy (WEP)* in 1999. It uses the stream cipher RC4 for encryption and only a 32-bit CRC for integrity, although of course the CRC does not give cryptographic integrity. The initial key length of 40 bits with a 24-bit IV was not strong, and security was worsened by the fact that implementations could easily reveal bits of the key. WEP was so broken that apps could be downloaded to find keys, and the methods were deprecated by 2004. WEP was replaced by *Wi-Fi Protected Access (WPA)* and implements *Temporal Key Integrity Protocol (TKIP)*, which prevents the attacks on WEP. That said, WPA was not as strong algorithm-wise as the currently widely used WPA2 and WPA3.

4. The *Domain Name System (DNS)* is used to translate internet domain names as strings to the actual IP addresses. DNS has been around since the beginning of the internet in the 1980s. IETF has developed a set of security extensions known as *Domain Name System Security Extensions (DNSSEC)*. The basic idea is to sign DNS lookup records; the signatures are validated using certificates that trace back to trust roots. Several different public key algorithms are supported; see [1].

5. The original phone system over a century ago was *circuit-switched* in that any given call was an actual complete physical circuit between the two end points (originally shared by multiple parties, but eventually party-to-party). The early cellular standards followed this model, and cryptographic methods were developed accordingly. For example, see the suite of algorithms used by GSM. Today's cellular networks have moved to *packet-switched*, where calls and other data are just packets in the networks. Such a migration means that the ubiquitous network security methods can be used.

6. Recent interest in *cyberphysical systems* has arisen because of cyberattacks on control systems that cause a physical response, as opposed to a cyber response. Satnav is a prime example of a cyberphysical system: cyber is used in all three segments, with the fundamental feature produced and used being a physical RF signal. More importantly for our purposes is the emphasis on how cyber, namely cryptography, can be used to protect these physical aspects.

7. Using zero trust is not the first time that M&M security has been declared dead. In 2011 there was a blog post titled: "M&M Security Bound to Be Eaten without Least Privilege" touting a book on least privilege [18], a concept that has become part of zero trust. M&M is a reference for being hard on the outside and soft on the inside.

   At this point it might be apropos to discuss the Gardner hype model [19, 20]. Innovation ramps into the peak of hype, followed by a trough of disillusionment finally recovering to a more nuanced view. Zero trust is a security concept, but it is *also* a marketing phrase.

References

[1] Stallings, W., *Cryptography and Network Security: Principles and Practice,* Eighth Edition, Harlow, UK: Pearson, 2020.

[2] Dougherty, D., J. Meseguer, S. A. Mödersheim, and Paul Rowe (eds.), *Protocols, Strands, and Logic: Essays Dedicated to Joshua Guttman on the Occasion of His 66.66th Birthday,* Lecture Notes in Computer Science, Springer, 2021, doi:10.1007/978-3-030-91631-2.

[3] Goldwasser, S., S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," *SIAM Journal on Computing,* Vol. 18, No. 1, 1989, pp. 186–208, doi:10.1137/0218012.

[4] Oppliger, R., *Cryptography 101: From Theory to Practice,* Norwood, MA: Artech House, 2021.

[5] Denning, D. E., and G. M. Sacco, "Timestamps in Key Distribution Protocols," *Communications of the ACM,* Vol. 24, No. 8, 1981, pp. 533–536, doi:10.1145/ 358722.358740, https://doi.org/10.1145/358722.358740.

[6] Needham, R. M., and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM,* Vol. 21, No. 12, 1978, pp. 993–999, doi:10.1145/359657.359659, https://doi.org/10.1145/359657. 359659.

[7] Lowe, G., "Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR," in *Tools and Algorithms for the Construction and Analysis of Systems* (T. Margaria and B. Steffen, eds.), Berlin: Springer, 1996, pp. 147–166.

[8] Neuman, B., and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications Magazine,* Vol. 32, October 1994, pp. 33–38, doi:10.1109/35.312841.

[9] Peterson, L. L., and B. S. Davie, *Computer Networks: A Systems Approach,* Fifth Edition, San Francisco, CA: Morgan Kaufmann Publishers, 2021.

[10] Padgette, J., et al., *Guide to Bluetooth Security,* January 2022, NIST Special Publication 800-121 Revision 2, Guide to Bluetooth Security, doi:https://doi.org/10.6028/NIST.SP.800-121r2-upd1, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=934038.

[11] WhatsApp, *WhatsApp Encryption Overview*, 2023, https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf.

[12] Rastogi, N., and J. A. Hendler, "WhatsApp Security and Role of Metadata in Preserving Privacy," *Computer Research Repository (CoRR),* abs/1701.06817, 2017, arXiv:1701.06817, http://arxiv.org/abs/1701.06817.

[13] Connectivity Standards Alliance (CSA), Zigbee Home Page, https://csaiot.org/all-solutions/zigbee/.

[14] Mulligan, G., "The 6LoWPAN Architecture," in *Proceedings of the 4th Workshop on Embedded Networked Sensors*, EmNets '07, Cork, Ireland: Association for Computing Machinery, 2007, pp. 78–82.

[15] Rais, R., C. Morillo, E. Gilman, and D. Barth, *Zero Trust Networks: Building Secure Systems in Untrusted Networks,* Second Edition, Beijing: O'Reilly Media, 2024.

[16] Rose, S., O. Borchert, S. Mitchell, and S. Connelly, *Zero Trust Architecture*, August 2020, doi:https://doi.org/10. 6028/NIST.SP.800-207, https://tsapps.nist.gov/publication/get_pdf. cfm?pub_id=930420.

[17] Pande, A. S., and R. C. Thool, "Survey on Logical Key Hierarchy for Secure Group Communication," in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT),* 2016, pp. 1131–1136, doi:10.1109/ICACDOT.2016.7877763.

[18] Lang, S., "M&M Security Bound to Be Eaten without Least Privilege—BeyondTrust," *Beyond Trust,* November 28, 2011, https ://www.beyondtrust.com/blog/entry/mm-security-bound-to-be-eaten-without-least-privilege.

[19] Gartner Hype Cycle Research Methodology, *Gartner Hype Cycle,* June 1, 2024, https://www.gartner.com/en/research/methodologies/gartner-hype-cycle.

[20] Fenn, J., and M. Raskino, *Mastering the Hype Cycle: How to Choose the Right Innovation at the Right Time,* Boston: Harvard Business Review Press, 2008.

Part III
Securing Satellite Navigation

7

Cryptography and the Satnav Enterprise

The third part of this book applies the cryptographic algorithms and protocols from Part II to satnav, and in particular the navigation signals. This chapter will look at the uses of cryptography in securing the satnav enterprise, and the following chapters will focus on securing the satnav signals.

We begin by reviewing protocols and how they can manifest in satnav. The breadth of a satnav enterprise forces us to limit our scope. A common reference for space enterprise architecture is [1]. The security goals of the satnav enterprise have a lot in common with other infrastructures, notably communication networks, and as such, are widely covered in other places [2–4]. General security is not our focus. Instead, we focus on where cryptography is used.

7.1 Space and Security

The need for securing commercial space systems has been evident for a while. Commercial attacks on satellite downlinks date at least to the 1990s, with satellite TV being one of the first widely publicized targets [5]. DirectTV and other providers had to roll out several generations of user sets before the economic impact of pirated signals was abated. At the start of the war in Ukraine, Viasat satellite internet modems were attacked and required factory resets or replacement to operate [6, 7]. It has been more than a decade since the first attacks on satellite commanding was authoritatively documented [8]. There are other well-documented attacks on space infrastructure since then [9, 10].

There are standards for security in space systems from the Consultative Committee for Space Data Systems (CCSDS) [11], the "Blue Books." These

standards are developed to be very general and cover a wide variety of situations. But sometimes the contents may be counter-intuitive results, such as that the TC (telecommand) Space Data Link Protocol has authentication and encryption as optional. There are many documents that focus on security from the space community. These documents include the NIST report "Introduction to Cybersecurity for Commercial Satellite Operations" [12]. The Cybersecurity and Infrastructure Agency (CSIA) has advice for satellite communication (SATCOM) providers [13]. There is explicit direction for U.S. government systems, Space Policy Directive 5, "Cybersecurity Principles for Space Systems" [14], and the language is such that it is also applicable to private space systems.

The U.S. government offers general advice on securing any enterprise. In particular, the CSIA has a resource center [15]. The National Aeronautics and Space Administration (NASA) Office of the Inspector General has at least two reports on cybersecurity: the first is a cybersecurity assessment of problems at the Jet Propulsion Laboratory (JPL) [9] and the second is a report on cybersecurity readiness at NASA in general [10].

It also makes sense to follow a holistic view of the security of an organization, and in particular, software is a regular topic in the discussions of security issues. A holistic view of enterprise architecture is in [16], and while security is mentioned, a more modern view would incorporate security and cloud architectures more fully. There is literature on secure software development [17]. However, most of their focus is naturally on network aspects, as applications move into cloud services [18, 19]. For a more general view of security, the current edition of Ross Anderson's *Security Engineering* is a good place to start [3].

7.2 Protocols and Satnav

Chapter 6 gave an overview of the general principles of cryptographic protocols. This section applies those lessons to satnav, namely in the areas of trust, identity, and key management.

7.2.1 Trust

From the user's perspective, trust in satnav is similar to the trust in any public utility. For example, users trust that the satnav enterprise will produce the signals they need and that these signals will adhere to the various interface specs. When there is a deviation, the users trust it is most likely to happen by accident and not deliberately from the enterprise. When we say "most likely," we are ignoring the possibility of the threats that we describe later in this chapter.

The user's trust is based on the usual trust in a government-provided service (i.e., failures would be fixable through the usual appeal channels). As

private satnav services begin to proliferate, trust must manifest the same way as trust in any private enterprise: through reputation, independent assessments, trusted authorities, and so forth. Most of the time, the need for trust is one-directional; the satnav enterprise does not as a rule have trust requirements of its users. The exception is when users are entrusted to protect information, such as keys. In that case, lessons from similar enterprises may be relevant.[1]

Finally, a user trusts in the various methods used, which comes from trust in the signal and the signal processing needed for navigation. This trust includes trusting any time a cryptographic protocol is used, that the procedures are implemented correctly, and that the enterprise protects its material.

Often trust is delegated, in that a user will trust certain features that in turn anchor the trust. One example is in the chain of X509 certificates; a user will implicitly trust that this chain is valid without direct interaction. Another example is trust in the device. such as in the use of a Trusted Platform Module (TPM), which often will have something like NIST FIPS 140 certification.

7.2.2 Identity

Recall the discussions in Chapters 5 and 6 on various ways to achieve identity, such as PKI and a web of trust. For current users of a satnav system, there is no chain of trust, although this may be changing. The Galileo system does provide for digital signing of navigation messages (discussed in Chapter 8). Private companies are going to offer more complete services and it is likely that those services will have cryptographically traceable chains of trust.

In the public GPS and other legacy providers, the signals serve as the proxies for the enterprise: if the signal is verified, then implicitly so is the enterprise. This idea is complicated by the fact that the signals have to be sufficiently easy to generate that every receiver can generate a local copy and perform correlation.

The satnav enterprise is a broadcast system, so that the usual methods of establishing authenticity are inadequate as they involve bidirectional communication. For ordinary user equipment that is not fully network enabled, the methods of Chapter 6 (such as TLS) will not apply. Similarly, challenge-response methods cannot directly apply to disconnected user equipment. That does not mean that methods analogous to challenge-response don't apply, in that the methods such as TESLA and Chimera will have a user verify a cryptographic-generated value that only the system could have produced.[2]

Although completely disconnected user equipment may not be able to completely verify that they are receiving authentic signals, that does not mean that they cannot detect problems. It will just be more difficult to have final certainty, which some applications require. Some of what they can do is discussed in [20].

7.2.3 Key Management

Key management requirements and associated methods in the satnav enterprise usually follow common patterns. For example, securing communication to the satellites will use shared keys that are generated and distributed by trusted organizations. The ground segment, both control and monitoring, take advantage of network security methods and other standard information security (INFOSEC) practices. Where methods specialized come into play is for the key management required to enable any cryptography in the satnav signals.

Cryptographic keys used by satnav signals can be both symmetric and asymmetric. For shared keys, the first issue is one of distribution: how do we get the keys to both the satellites and UE. Figure 7.1 shows a generic key distribution center being used to share the keys, although in practice it is a single key being used by the satnav signal. For the satellites, this sharing uses a similar, likely the same, secure communication link that is used for TT&C. Shared key systems are used when exclusivity in the users is desired, such as in a private system or certain government systems.

For the users, we need similar secure links to all the users. This need is where the difficulties arise: satnav will have millions of users. The other difficulty for shared key management is one of trust. As mentioned previously, the enterprise would require that all users protect the shared key material. That trait makes some solutions, such as disperse ad hoc distribution centers, harder to realize.

These difficulties point to using asymmetric cryptography, which is much more amenable to a large, ad hoc user base. In this case, the framework is as shown in Figure 7.2. The central distribution is some PKI. Again, the satellites will receive their material, namely the private key, through a secure channel. The users, who need the public key material, do not need a secure channel, in the sense of confidentiality, but just an authenticated channel. That need is indicated by the SIG denoting a signature from the enterprise. Some existing methods, that we explore in Chapter 8, may have a hierarchical PKI, with different enterprise certificates being used for different purposes and with different validity times.

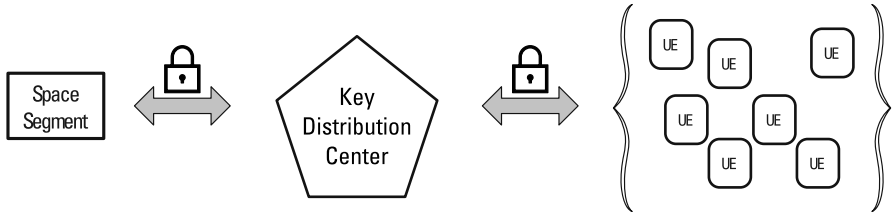

Figure 7.1 Distribution of shared keys in satnav.

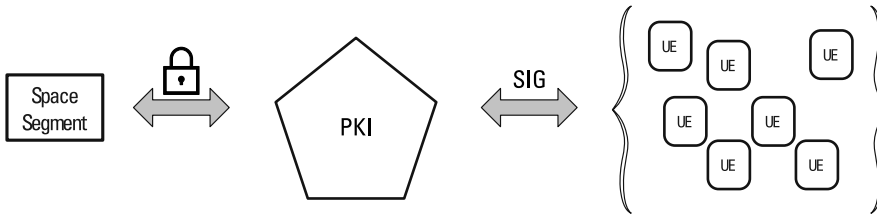

Figure 7.2 Distribution of public keys in satnav.

These are not the same trust issues for asymmetric cryptography as for symmetric. Users could share their public key information, so long as the public keys can be authenticated, such as having signatures that anyone can verify. Still there is the usual problem of authenticating the certificates.

Finally, we note that there does not seem to be the need for ad hoc session security, such as TLS provides, for PNT. This observation stems from the broadcast nature of satnav. That said, ad hoc sessions could possibly be used to exchange information like public keys or ephemeris.

7.3 Satnav Infrastructure

The main emphasis of this book is on cryptographic protocols used in satnav signals, but of course cryptography is used everywhere in a satnav enterprise. This section touches on those nonsignal applications; these applications serve to secure the infrastructure needed to enable the signals. We build on the framework discussed in Chapter 2.

In space enterprises, it makes sense to balance many factors with security, because resources are very constrained in space. One framework for balancing the various factors is NIST's risk management framework [21]. These steps in this framework can be summarized informally as follows:

1. Prepare (i.e., commit to the process);
2. Categorize the desired security goal and look at the impact of failure;
3. Select controls; look at NIST SP 800-53 [22] for general and NIST IR 8270 [12] for the orbital component;
4. Implement the controls and document;
5. Assess and evaluate how the controls are implemented and if they are functioning as intended;
6. Authorize the specific entities that need to accept risk and document the risks that are accepted;

7. Monitor and update as needed using the data from the selected controls.

There are certainly other approaches to risk management, and we are generally of the opinion that while flexibility is very important in security, it is discipline and consistency that are necessary and are likely a larger factor in achieving security. *In other words, good processes can enable good results.*

7.3.1 Space Segment

We first look at the space segment; recall Figure 2.1. The traditional design of a spacecraft has been to put all the security on the perimeter of the spacecraft at the external interfaces and to allow the spacecraft to be free of security controls. We feel this era is over for spacecraft as it is for other systems. That said, internal security controls are subject to many constraints on a space vehicle.

As the complexity of the orbital components of space systems grows, the need for security inevitably will also grow. Architectural complexity, like crosslink commanding and hosted payloads, especially affects proliferated constellations. In fact, there is evidence that hosted payloads are under consideration in some sectors [23]. Further, low earth orbit (LEO) constellations that favor small satellites are under development [24–26]. All these will have their pertinent security architectures.

As an example, the networks currently used in spacecraft design, such as the archaic 1553 bus [27] and the slightly more modern SpaceWire [28], don't have any explicit provisions for security at the physical layer. SpaceWire is compatible with the CCSDS Packet Transfer Protocol [29], which optionally allows encrypted packets.

At this juncture, it is likely that Ethernet would be the best option for a secured bus network at the physical layer, and this is being developed for NASA's Orion program [30], European Space Agency (ESA) projects, and commercially available [31, 32]. Security in these networks can be done using the methods from Chapter 6.

The control segment controls the SVs operations, which include uploading messages and commands in the TT&C link. There are the usual security goals of confidentiality, integrity, authentication, and availability of the TT&C links. Cryptographic protocols to establish these are straightforward, although the specific methods are likely kept private. These methods ultimately use a key shared between the control segment and SVs.

How is this shared key established? It could use keys that are prepositioned and updated as needed. Or a key exchange algorithm could be used, which would leverage a long-lasting private key on the SV. In both cases, observe that the link is bidirectional, which enables two-way protocols to establish identity.

The SV needs to execute secure protocols, both for the links to the control segment and in generating signals (e.g., digitally signing messages). Those functions require protections of the computations via good computer security methods. These methods are challenging for two reasons. First is that the computation itself is challenging, since radiation-hardening is needed. That requirement may affect the processing speeds involved, which could impact how much can actually be done on the SV.

Second, there is a growing move toward reprogrammability in spacecraft, namely, the ability to future-proof the transmitter and allow algorithms and protocols to upgrade. The ability to reprogram puts a burden on protections (e.g., to make sure that the spacecraft cannot be adversely upgraded).

7.3.2 Control Segment

Some of the security architecture of the ground segment was just mentioned with regard to the space segment. Other parts of the architecture as they relate to protections within the control and monitoring networks are shown in Figure 7.3. There is a set of communications links between the monitoring stations and the control networks; these links are likely dedicated. There are also communication links from the control segment and distribution nodes. Examples of the latter include perhaps distribution of ephemeris information for out-of-band uses and any key material. These links are not dedicated but instead will use the internet.

As such, there is some protection that could be to ensure confidentiality but often just needs to be authenticated.

The individual locations must have good INFOSEC protections. There is nothing special about this need with regard to satnav other than some information that needs to be protected in the short term, such as the integrity of ephemeris, probably does not need long term protection in storage. An exception would be if logs need to be kept for audit purposes.

7.3.3 Mission Segment

In a satnav system, the monitoring sensor network is one of the most critical systems. Most of the protections for the actual sensor are the same as for any high-value user equipment; for examples, see the next section. The other issue for sensors is basing, and the most common is ground site basing.

Secure, reliable communication links are critical, and sensor sites must be sufficiently plentiful to allow observability of the system in the face of all potential failures, including mischief. The net result is that planning for sensor placement and design of secure, redundant, geometrically appropriate sensing is not a trivial task.

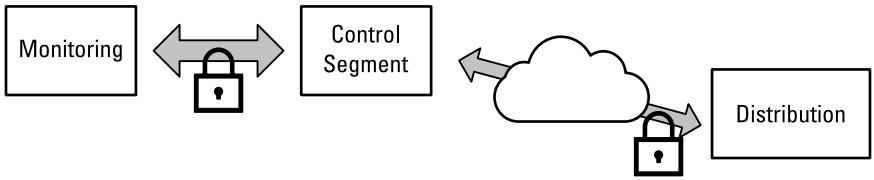

Figure 7.3 Security architecture in ground segment.

One way to reduce the dependence on terrestrial sensing would be via cross-link (X-link) ranging, which has been under consideration for GPS since the mid-1980s [33].[3] Cross-links have their own additional security concerns.

Sensors in a complex system, such as a GNSS system or any other control system must have specific properties, including availability, integrity (nonrepudiation and authenticity) among them. Notice that confidentiality is not among the *required* security properties. It may be that availability and integrity are achieved using confidentiality, but confidentiality is not usually a primary focus. Indeed, the nonemphasis of confidentiality in GNSS systems is one of the very interesting aspects of working on GNSS security problems, since in the design of most security systems, confidentiality is a main goal.

The importance of quality measurements cannot be overemphasized. The heart of a GNSS system is to solve the estimation problem that minimizes the user's errors. Data includes almanac, ephemeris, and clock synchronization' There is a basic system engineering tension between accuracy and performance [34, 35]. The usual method for the estimation of information when there are system dynamics and time updates is a Kalman filter, which is a time recursive estimator for a dynamic system.

As with any least squares estimator, the problem with bad measurements naturally occurring due to sensor failure, or malicious intent is easily understood. Since least squares is the minimum of the sum of the squares of the residual errors, a large measurement error will produce a large effect on the estimate.

Consider Figure 7.4. Four points were modified, two at each end of the data by swapping their values. The result seriously affects the accuracy of the estimation, specifically the estimate of the slope parameter.

The figure also indicates that data integrity is hugely important in a control system such as satnav. No new data was generated in the attack; it was accomplished by manipulating the existing data. Data stores and pipelines generally need atomic, consistent, isolated, and durable (ACID) properties and user interfaces are particularly susceptible to attack surfaces [36]. These examples motivate that cryptographic methods, in conjunction with physical security, and statistical measures are needed to maintain the integrity of the data at the core of a satnav system.

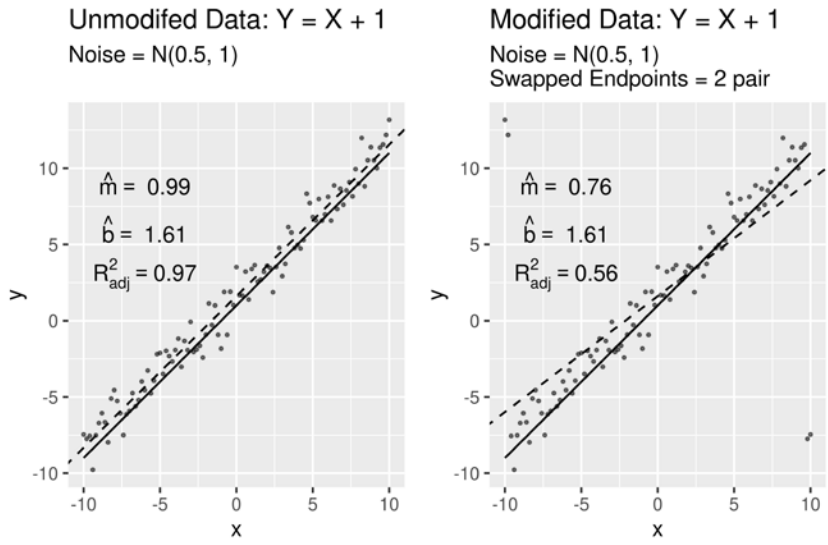

Figure 7.4 Illustration of data manipulation in least-squares estimation.

7.3.4 User Segment

The user segment consists of the UEs and any distribution system they need. There are three possibilities: the UE is not network capable, the UE is fully network assisted, and the UE has limited or occasional network access. These different network capabilities dictate possible security solutions.

For the nonnetwork-capable user equipment, some strategies may be limited. Some security goals, especially against some threats against the satnav signals, can be done with good antenna systems and signal processing, such as angle of arrival or phase measurements. See [20, 37] and many papers at Institute of Navigation conferences as well as hundreds more at IEEE conferences.

The fully network connected UE is in a position to leverage a lot of resources. For example, the hope is that in the long run, information from the satnav augmentation services will be available over the internet. The first suggestion of using GPS data over a "network" was in 2002 by S. Lo [38] in his PhD thesis, to broadcast GPS WAAS data over Loran-C. There are several options for network-assisted GNSS users, such as decent time hacks like the NTP protocol time distribution [39], which can improve time-to-first fix and also to gate time in antispoofing algorithms. Another is to get authenticated data such as almanac and ephemeris. Currently most cell phones are capable of getting data from the cell network, but the data is not assured beyond that provided by the cellular link protections.

Out-of-band data distribution for GPS that is digitally signed from the control segment has been proposed for some time. The U.S. federal government already has an extensive PKI initiative [40]; so, establishing a trust path to the U.S. government should not be difficult. In addition, in Chapter 10, we discuss fast channel Chimera, an out-of-band enabled version of Chimera to allow network users to avoid the long delays associated with bit-commitment over the navigation message.

The user who has network access that is limited is an interesting case. Occasional network access allows participation in PKI/X509-based trust networks, with the proviso that revocations will not take place until an update from the network. Operational strategies must address the incurred risks. There are several cases of infrastructure that fit in this category, such as receivers for commercial aviation. Protocols like NMA (Chapter 8) and (slow channel) Chimera (Chapter 10) are targeted at these user cases.

The SBAS augmentation systems can be considered a network access. While not two-way, they do provide additional data at a significantly higher rate (250 bits per second) and the signal supports a time-to-alert of GPS anomalies of less than 7 seconds. The U.S. Department of Transportation has data and graphics on performance of GPS, WAAS, Galileo, and so on [41]; EGNOS has similar reporting [42]. The capabilities of these augmentation based receivers is a preview of the capabilities of a fully connected receiver. It should be noted that given their role in time to alert, these systems are additional attack surfaces for the PNT service. Thus either errors or spoofed signals represent significant issues for the users of these services. At this time there are no trust mechanisms built into these augmentations.

The UE itself will need INFOSEC to protect itself from cyberattacks (i.e., designed to cause it to fail in its PNT calculations). A possibly amusing example of this need is an application on cell phones that spoof GPS to allow cheating in the location based Pokémon Go game; more than one app advertises itself as the best spoofer. This problem has been around for a while, but it is a mostly self-inflicted problem.

Additionally, the UE will need to protect information that the satnav enterprise deems private, such as keys that allow access to non-public signals. If the protection provided by commercial methods, such as. a TPM or other such solutions, then the cost might not be very much. However, these protections can be burdensome, as already mentioned.

7.3.5 Threats

As has been pointed out, satnav infrastructure shares features common to most large technical systems involving networks, such as the cellular system, and the

power grid. A lot of effort has been put into protecting such critical infrastructure. Examples include:

- The PNT conformance framework and the subsequent IEEE P1952 standard for resilient PNT platforms [43];
- The DHS program for PNT [44];
- NIST recommendations for cybersecurity in an enterprise [45].

These recommendations include discussions of the wide range of possible threats. It is out of our scope to list these, again because they are not unique to satnav. For example, vulnerabilities have been collected in a taxonomy for years called the Common Vulnerabilities and Exposures (CVE) [46]. Related work is MITRE's ATT&ACK® [47], and specific to space systems, and nascent, Aerospace's Space Attack Research and Tactic Analysis (SPARTA) [48].

7.3.6 What about Satnav Signals?

Section 7.5 will look at the general construct, security goals, and the main types of threats against satnav signals. However, we first illustrate cryptographic examples for the rest of the satnav enterprise.

7.4 A Summarizing Sample Enterprise

In this section, we outline a fictional GNSS service at an operational view, or as a signal flow diagram, and indicate the protocols that might be considered on various communication paths. In addition, there is a sample of cryptographic algorithms that might be appropriate for those protocols. The algorithms are notional. This discussion is not intended to be comprehensive, but more to provide a concrete flavor the discussions in this chapter.

Figure 7.5 shows the components of the sample enterprise. We list the components with their functions, security responsibilities, and interfaces to other components. All the components *except UE* are internal to the system to some extent. The specifications for UE are usually generated by the system and its operators, but in many cases the UE's implementations are not prescribed by the system.

The *mission segment* computes almanac and ephemeris, payload health, and various performance measurements. It is responsible for aspects of enterprise security and data integrity. It interfaces with the control segment (I), key management (C), and the ephemeris service (A).

The *control segment* is responsible for the SV's health and to upload information such as payload data. It is also responsible for aspects of enterprise

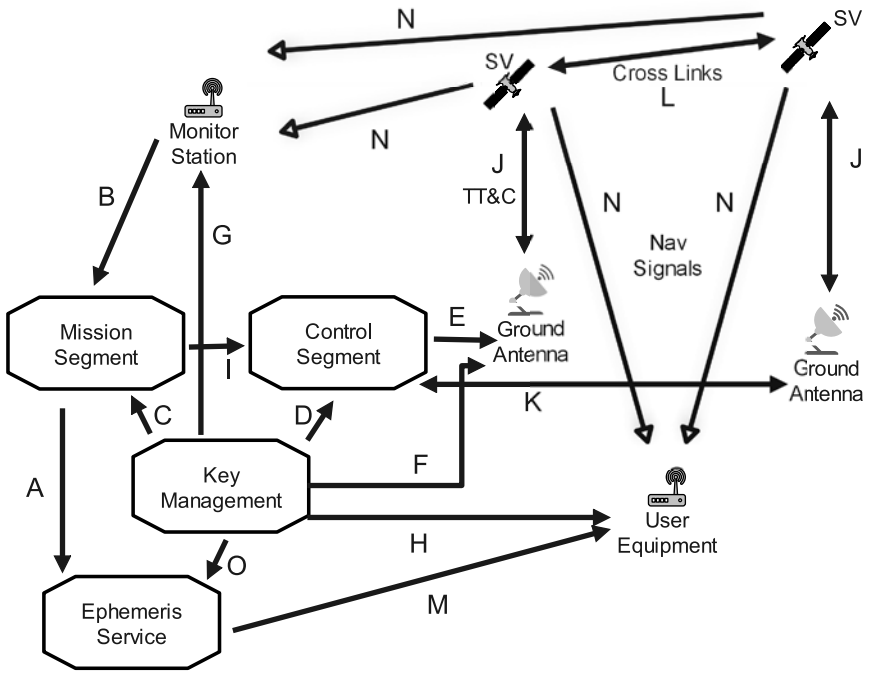

Figure 7.5 Overview of the components of an abstracted satellite navigation system, showing data flows.

security, process control security, and data integrity on the TT&C link. It interfaces with the mission segment (I), key management (D), the ground antennas (E and K). Indirectly it interfaces to the SVs through interfaces J and L.

The *ground antenna* contacts the SVs using the TT&C link. It is responsible for secure communication, including data integrity and confidentiality. It interfaces with the SV (J), the control segment (E), and key management (F).

The *monitor station(s)* observes and calibrates SV trajectories and health. It is responsible for its data integrity and observing signal integrity. It interfaces with the navigation signals (N), the mission segment (B), and key management (G).

The space vehicle (SV) broadcasts the navigation signal, which includes ranging and data features. It is responsible for its bus integrity, payload integrity, its portion of the TT&C communication, and cross-link security. It interfaces with the UE and monitor segment through the signals (N), the control segment through E via the ground antenna (J), and other SVs through the cross links (L).

The *key management* provides all of the keys needed for the cryptographic functions. The specific operations include key generation, key wrapping/protection, key distribution, and key archiving. Specific interfaces include to the

mission segment (C), the control segment (D), the SV via the control segment (E, J, L), the monitor station (G), the ephemeris segment (O), and the user equipment (H).

The *ephemeris service* provides SV position data for transmission and archival. The security function it uses is secure distribution of this information. It interfaces with the mission segment (A), key management (O), possibly user equipment and the public in general (M).

The *user equipment* provides PNT service. Security functions include device integrity and various processing functions to counter threats: cryptographic protections, RAIM methods, antijam, and antispoof. The UE interfaces with the SVs through the navigation signals (N), key management (H), and perhaps the ephemeris segment (M).

7.4.1 Common Elements

Here are the general methods that could be used throughout the enterprise, even if some are not in use today; see Chapter 6 for descriptions.

- Zero-trust methods (NIST SP 700-207, SP 800-207A);
- Two-factor authentication methods;
- Identity management (IM) of everyone;
- TLS with mutual authentication.

One might speculate that, for instance, the ephemeris service need not use TLS, but mutual authentication cuts down the attack surface on denial-of-service attacks and opens up options in dealing with flooding of requests. The downside is that users would have to authenticate, and one still has to deal with the problem that a large number of compromised peers would pose.

One way to proceed would be to host TLS on IPsec. In some sense, the combination of TLS and IPsec are akin to the use of TRANSEC (IPsec) and COMSEC (TLS), where TRANSEC prevents traffic analysis, and patterns of use, and COMSEC provides security for the confidentiality for each of the constitute flows.

When we discuss internet or intranet traffic, we will choose the newer postquantum alternatives whenever they are available. For example, Dilithium could eventually be used for digital signature, but until then, RSA and ECDSA are still secure; see the appendix. Similarly, it may make more sense to look forward and use key encapsulation methods like Kyber than to discuss Diffie-Hellman in this context. We note that for symmetric algorithms, the standard AES-256 *is a postquantum algorithm*. AES would be used in modes like AES-CBC or AES-GCM. Hash functions would likely be the SHA-2 family.

7.4.2 Mission Segment

The mission segment is discussed in Section 2.2.3. The mission segment's primary job is to compute the almanac and ephemerides, and ancillary data, such as clock steering, for the constellation. It is also responsible for performance data. There are two critical interfaces: the control segment and the monitoring segment. The monitoring segment provides the raw data to the mission segment's calculation (i.e., data checks followed by a Kalman filter). The integrity of the data is paramount, so the links need to be encrypted. The transport of the data needs to fit the model; if the communication is soft-real-time, then one might use a Kalman filter in a smoothing mode to allow for delays and unordered data. Securing real-time systems and databases is well-understood and can use existing internet protocols and the associated security models [19, 49]. The second critical interface is to the control segment, and can be conceptualized as a database interface. The rub here is that if the mission and control segment use the same database, then appropriate tables can be coordinated [50]. Otherwise, the coordination is more complex (see Table 7.1).

7.4.3 Control Segment

The control segment is responsible for TT&C for the constellation. We will assume that the ground antennas (G/A) are responsible for space vehicle contact sessions, with keys specified in the control segment and retrieved from key management.

The control segment tracks SV state information, such as physical state (thermal, power, propulsion, attitude, etc.) and security state information, such as what is needed for the antireplay protocol. The CCSDS recommends that a counter field, antireplay sequence number be used to prevent replay attacks [51]. A good version of this protocol will also stop operators from repeating commands (see Table 7.2).

Table 7.1
Summary of Methods in the Mission Segment

Function	Link	Protocols
Data Storage	NA	TLS, ACLs, IM, Database Security
Data Transport (from GA, to Control Segment)	E, I	IPsec, TLS, cross-domain, REST, IM
Data Transport (to Ephemeris Server)	A	FTP, IPsec, TLS, cross-domain, REST, IM
Key Management Service	C	IPsec, TLS, IM

Table 7.2
Summary of Methods in the Control Segment

Function	Link	Protocols
GA	E	IM, Real-time
Key Management	D	TLS, IM, Two-factor
TT&C to SV's	J	See SV (CCSDS)

7.4.4 Ground Antenna

The ground antennas are responsible for the mechanics of contacting the spacecraft. This includes TRANSEC generation. Data flows from the control segment thru the ground antennas to the spacecraft and back. The contact pattern may be a few times a day, as in normal with GPS, or it may be more frequent, as is with QZSS and Galileo.

See Table 7.3 for ground antenna interfaces. The TT&C interface to the SV is governed by CCSDS standards. Some of this is tunneled through to the control segment, since the GAs don't unwrap noncommunication-related telemetry.

7.4.5 Monitoring Stations

As discussed in Section 2.2, monitoring stations are usually a network of privileged UE that sends measurements with some time lag back to the mission segment that will be used to calibrate the system and to provide input to the quality of service functions.

If there are encrypted signals, then there is an interface to key management. If there is processing done on authenticated signals, the monitoring station might have interfaces to the ephemeris service and the key management to predetect issues with the measurements (see Table 7.4).

7.4.6 Space Segment

The main job of a space vehicle in a GNSS system is to provide the signal-in-space (see Section 2.2.1). Realistically, there are many other details to be

Table 7.3
Summary of Methods in the Ground Antenna

Function	Link	Protocols
Key Management	F	TLS, IPsec
TT&C Traffic to Control Segment	E	TLS, IPsec
TT&C to SV	J	See SV (CCSDS)

Table 7.4
Summary of Methods in Monitoring

Function	Link	Protocols
SIS	N	ICD's, TRANSEC, or Authentication
Sensor Measurement to Control Segment	N	REST/FTP IM, IPsec, TLS
Key Management	G	TLS, IPsec
Ephemeris Service	M	TLS, IPsec

attended to, such as the health of the navigation payload and the space vehicle itself, which are paramount.

The security literature from NIST and elsewhere indicates that a shift to zero trust mechanism with the spacecraft is desirable (note that this shift will necessitate major re-thinking from current system designs). Achieving good security on TT&C links and cross-links is well understood, even if it is evolving, to meet the zero-trust paradigm but is optional in the CCSDS standards.

See Table 7.5 which details space vehicle interfaces; the most important is the actual SIS, while the TT&C are vital to the system's long-term health. Cross-links, especially ranging cross-links, can lighten the burden of links with the ground.

7.4.7 Ephemeris Service

The ephemeris service exists to provide out-of-band navigation data, such as almanac and ephemeris, from an authoritative source. This is for real-time applications, and for the processing of historical data. The data needs to be presented in a way that allows further distribution of the data and if transmitted intact the authenticity of the data should remain intact (see Table 7.6).

7.4.8 Key Management

The job of the key management service is to orchestrate keys used by the system. This includes

Table 7.5
Summary of Methods in the Space Segment

Function	Link	Protocols
TT&C	J	CCSDS Space Data Link Security Protocol [52] CCSDS Authentication Credentials [53] CCSDS Cryptographic Algorithms [54]
SIS to users	N	See UE
X-Link	L	Same as TT&C

Table 7.6
Summary of Methods in the Ephemeris Service

Function	Link	Protocols
Data at rest	NA	Database, IM
Serving data requests	M	IPsec, TLS
Receiving data from the mission segment	A	IPsec, TLS

- Key generation;
- Administration of keys;
- Key distribution including revocation.

Keys are generated using random bits, but it is important to make sure that a key is never used twice and have correct entropy.

It might make sense to partition some functions, for instance to provide cryptographic information that is public (e.g., public keys) in the ephemeris service, since it is already public facing and scalable.

Compromised user keys, from a user view, have the same effect as the compromise of keys elsewhere in the system. That is, the system keys and the user keys need to synchronized, and it does not matter to the UE which is incorrect.

See Table 7.7; all the services are network based and work the same.

7.4.9 User Equipment

The job of user equipment is to receive the signal-in-space and process it into PNT information as described in Section 2.2.2.

If a private service is being used, then protection of symmetric keys becomes a concern. Modern TPMs do provide for secure key storage and in some cases this may be adequate. Subscription services for video services and video games use a combination of protection, identity management, and just-in-time delivery of symmetric keys to protect their products. We would expect that private satnav services would use similar technologies.

Table 7.7
Summary of Methods for Key Management Service

Function	Link	Protocols
Key generation	NA	Random Bit Generation
Key distribution	C, D, F, G, H, O	IM, two-factor
Data at rest	NA	Database, IM

For instance, John Deere, a company that manufactures precision agricultural equipment, uses PIN codes, with time and geographic restriction devices; Starfire receivers are integrated into the equipment [55]. They are also equipped, at least in the United Kingdom, with the construction equipment security and registration (CESAR) data tagging system, which includes overt and covert features and tamper-evident devices. Ownership is transferable (for a small fee) [56].

See Table 7.8; the UE has one mandatory interface defined by ICDs to receive the SIS. Any other interfaces will be internet (or intranet) and the device will be considered net-assisted. An example of an intranet assisted device would be a cell phone that gets GPS navigation message data from a cell tower.

7.5 Satnav Signals

The signal-in-space is a unique entity in GNSS. Part communication and part bistatic radar**,** one-way radar, the signal is broadcast and there is nominally no other communication between the broadcast and the receiver. The signals are tracked in the receiver using correlation tracking loops. Correlation implies that the signals can be generated, at least if all the information is known.

The use of TRANSEC for GNSS is to prevent *electronic attack (EA)*:

> Electronic attack (EA): Transmission of hostile signals (jamming) to interfere with ethe reception of legitimate signals, consume bandwidth in order to compete with legitimate signals (denial of service), and/or transmit signals intended to achieve imitative or manipulative communications deception based on signal parameters. The intentions of these EA efforts are to deny *signal availability* or *defeat signal integrity* [57] (emphasis ours).

The goals for designing signals and protocols for GNSS systems are to allow efficient and accurate determination of PVT *and* to allow for robust understanding of the availability and integrity of the signal.

Antijam, roughly, is achieved by processing gain, and the use of signal processing and antenna-based methods. Processing gain can be used if the signal generation method is not known by the attacker. In a public GNSS system,

Table 7.8
Summary of Methods in User Equipment

Function	Link	Protocols
SIS	N	ICD's, Authentication, TRANSEC
Key Management	H	TLS, IM, IPsec
Ephemeris Service (optional)	M	TLS

the signal generation methods are known. There are many things that can be done in signal processing, see [20], which is a good starting place to understand the vast amount of literature in this area.

A moment of philosophy is in order. There are very good methods to defeat electronic attacks, and some of them are very simple to implement, such as white listing parameters as recommended by the U.S. DHS [58] for GPS. These measures make it less likely that an attacker will be successful. But, for instance, whitelisting cannot be used to assert the data is correct, only that it *could* be correct. As a simple example, a GPS ephemeris data set for yesterday with an adjusted time must pass a whitelist, and yet not be correct for today. Instead, our main thesis is that cryptographic methods, if applied correctly, act as proofs that the signal *is* correct.

Cryptographic and noncryptographic approaches complement each other. One way to view this is that noncryptographic tests start with a prior assumption that the signal might be correct and each test adjusts that prior assumption that it is a correct signal. The probability that the signal is correct is a function of the tests and the assumed capability of the attacker. If the attack is sufficiently sophisticated the test will fail to detect the attack, leading to a false positive.

Cryptographic techniques start with the prior assumption that the signal is correct and attempt to authenticate that assumption. The test is independent of the capabilities of the attacker. Under the assumption that the required cryptographic materials have not been compromised, there is no possibility of a false positive.

But let us be very clear, cryptographic materials can prove the authenticity of the signal and bind it to time, but in a ranging system, they cannot be used to prove the path between the receiver and the space vehicle was minimal. The authentic signal arriving at a receiver 100 nano-seconds late via multipath or a repeater, is not correct. The only solution to this is to look for earlier arriving signals and use the earliest one. This is squarely in the domain of the signal processing techniques. The hazard of taking the earliest signal is that if a spoofer is present and understands that preference, then their strategy is clear: make sure the malicious signal is the first signal to arrive at the user equipment.

Cryptographic methods used in satnav signals are the unique part of the satnav security architecture, and as such the last chapters of this book are devoted to describing them and their current, planned, and proposed implementations. This section offers an overview of these methods.

7.5.1 Satnav Signal Layer and Security Goals

Figure 7.6 recaps what was presented in Section 2.5 with an emphasis on security goals. We divide a satnav signal into three layers:

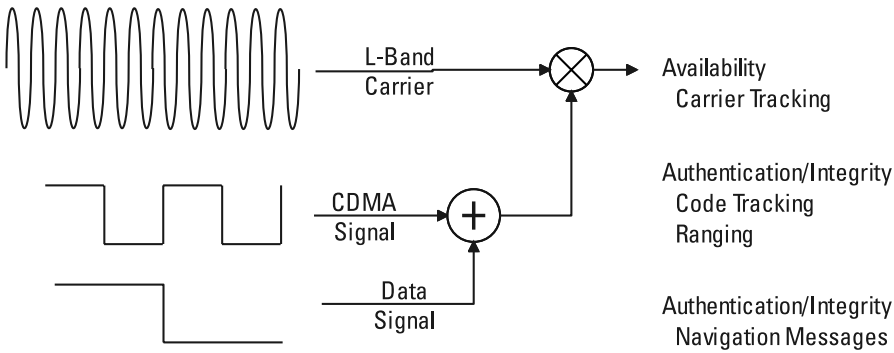

Figure 7.6 Signal layers and goals.

- Carrier layer, which captures such things as power, center frequency, and power spectrum;
- Code layer, which is the CDMA spreading code and modulation;
- Data layer, which contains the navigation messages.

Each layer has its own security goals, and while there may be commonalities in the names of the goals (e.g., availability), how they manifest will be different.

7.5.1.1 The Carrier and Modulation Layers

The primary goal perhaps for the carrier and modulation layers is availability. A UE should be able to acquire the signal if it is visible. In addition, the UE should be able to continue to track such a signal. For authentication, which includes integrity, the UE should be confident that the signal that is acquired and tracked is the legitimate signal. That means it comes from a valid source and its parameters, such as direction and timing have not been changed. Finally, sometimes exclusivity is desired, which means only authorized users can acquire and track. This goal is found in certain private services.

While the goals for the carrier and modulation layers are similar, the cryptographic methods differ. Usually the carrier is a fixed frequency, although certain spread spectrum methods could use frequency hopping or FDMA, as early GLONASS did [59]. Instead, cryptographic methods are more common as part of the spreading modulation, which is the subject of Chapter 9.

7.5.1.2 The Data Layer

Availability means that the UE should be able to demodulate the data and obtain the messages without error. Note that the data could be unavailable even though the signal is being tracked. This situation occurs if the integrate-

and-dump of the correlation process is good enough to provide feedback to the tracking loops but not good enough for data demodulation. See Section 2.2.2 and Figure 2.9 chip offsets of around |0.6| would result in data errors, but be sufficient to keep tracking loops working. Jamming will cause a similar effect to chip offset, especially when the electronics start to saturate.

The goal of digital signatures is to authenticate data (i.e., show the data comes from the valid source and has not been altered in content or timing). The various methods explored to apply cryptography toward these goals is the subject of Chapter 8.

7.5.1.3 The Complete Signal

Cryptographic methods can be applied to each layer separately without interaction, similar to what we observed for the different protocols in the network layers in Chapter 6. However, it turns out that recent methods have been developed that merge the layers together, essentially binding their security. These methods are explored in Chapter 10.

Indeed, the Chimera protocol combines spreading code layer and data layer cryptographic methods. This book presents each in isolation (Chapter 8 and Chapter 9 and then uses Chapter 10 to show their combination).

7.5.2 Taxonomy of Threats

The threats to the satnav signals divide naturally into three parts:

- Jamming, which is a denial of service of some aspect(s) of the signal;
- Meaconing, which repeats a valid signal with the intent that it be interpreted falsely;
- Spoofing, which is the creation of an invalid version of the signal.

A good reference on GNSS signal threats is [20].

In addition, threats against the satnav infrastructure could manifest in the signal. Our focus is on threats to the UE generated by the falsification of the signal-in-space.

7.5.2.1 Jamming

Jamming for a satnav signal encompasses any type of denial of service: the prevention of a UE from using the signal. Jamming could mean preventing acquiring or tracking the signal. It also includes preventing the demodulation of data or causing a non-correctable number of data bit errors. A general picture is provided in Figure 7.7, which shows the UE being subject to both the actual signals and whatever the jammer produces.

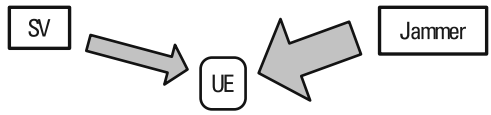

Figure 7.7 Jamming.

Because satnav signals are so low power, they are easily jammed, and jammers can be purchased.[4]

There are well-documented cases of interference in the GPS bands, interference of commercial aircraft being "routine" around Ukraine and Syria as of this writing, as measured by Automatic Dependent Surveillance-Broadcast (ADS-B) messages on commercial flights. McGurn of the GPS PNT advisory board commented on U.S. interference events in 2009 [60] and several prominent events have occurred since then, some of which are understood [61] and some that are not [62].

We do not elaborate on the jamming threat in this document, because cryptography cannot mitigate being overwhelmed by power. There are solutions, such as having a good antenna system that can null jammers, using automatic gain control, and having good depth of the analog-to-digital converters. Of course, a good option is moving away from the source of jamming if possible.

Not shown is the possibility of threats to the satnav infrastructure that could manifest as denial of the signals. For example, cyberattacks on the control segment cause all signals to be set unhealthily. Mitigating such attacks falls under the protections required in the infrastructure.

7.5.2.2 Meaconing

Meaconing, also called repeating, is the situation where the true satnav signals are captured and then rebroadcast as is. The end result is that another copy of the true signal is received, delayed in time. In some ways the effect of the meaconer is like multipath (i.e., what can occur when the true signal reflects off of an object and arrives later in time) that often occurs in urban settings. The nominal situation is shown in Figure 7.8.

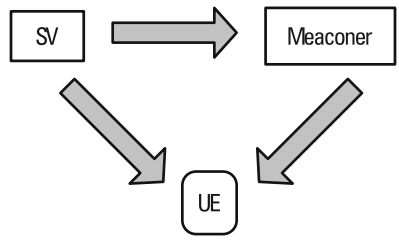

Figure 7.8 Meaconing.

There are methods to account for multipath. Basically, the earliest arriving signal in the UE is assumed to be the true signal. This assumption may be false, for example if the only signal present is the reflected signal, which can again often happen in an urban setting if the true signal is blocked by a building. Another observation concerns polarization. For example, GPS and Galileo signals are right-hand circularly polarized, and every reflection reverses the signal polarity and usually attenuates the signal. However, since a meaconed signal is intentional, the polarization and power loss traits can be corrected by the attacker, although the late arrival cannot.

Again, cryptography cannot help against meaconing; any signal can be rebroadcasted. The same noncryptographic solutions used for jamming can help meaconing, such as antenna nulling and power-monitoring. The best mitigation is determining that the problem is meaconing and choosing the first arriving signal. The choice of using the first arriving signal has two issues. First, as already mentioned, there is the corner case when the actual signal is not available. But there is the more problematic case if the adversary can generate a copy of the actual signal and broadcast it in advance of the arrival of the signal. That case is no longer meaconing, of course, but spoofing, but it does show the danger with blindly accepting the earliest arriving signal.

7.5.2.3 Spoofing

The final threat on the signals is spoofing, where the adversary creates copies of the signals and broadcasts them. The nominal situation is shown in Figure 7.9. As indicated, the UE is receiving two sets of signals, and there is perhaps nothing a priori that indicates which are the true signals.

There are many, many methods developed for anti-spoofing that do not involve cryptography. Good surveys in include the papers by Psiaki/Humphreys [63] and Günther [64]. There are many related good ideas suggested for receiver developers/manufacturers to help prevent attacks on receivers [20, 65].

Cryptography has a lot to offer for preventing spoofing, which is the focus of the last chapters of this book. Basically, as Figure 7.9 indicates, the spoofer needs to generate the signal. If there are cryptographic features in the signal, then that should prevent such generation. The nuances will be in protecting all layers in the signals.

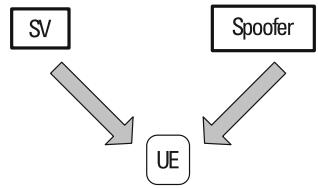

Figure 7.9 Spoofing.

7.6 A Brief Relevant History

This section gives a timeline for the rise of cryptographic methods for protecting the satnav signals. For each, we indicate where they apply. More details will be given in the following chapters where appropriate.

7.6.1 The Need: Pre-2000

Between 1978 and 1985, the GPS Joint Program Office launched the Block I GPS spacecraft, which did not have security measures; that is, the L1-L2 signals used P-Code, a short cycled linear feedback shift register code that recycled at the end of a week.

In 1989, the first Block II spacecraft was launched and in 1990 there was a modification, and Block IIA was launched. The GPS system met the requirements for initial operating capability on December 8 1993 [66]. The Block IIA (II Advanced) spacecraft could generate the P(Y) encrypted signal and continued launching until 1997.

With the launch of the Block IIA spacecraft, two cryptographic functions were implemented, selective availability (SA) and the antispoofing signal P(Y). The navigation signals were divided into two classes, Standard Positioning Service (SPS) and Precise Positioning Service (PPS). The SPS signals were degraded [66] starting on July 4, 1991; the PPS signal P-Code was replaced with an encrypted version P(Y) [66] in 1994. The GPS system was declared operational and it achieved full operational capability in 1995. In March 1996, the United States announced that SA would be terminated [67]. In 2000, a U.S. Presidential Directive committed the United States to turning SA off, and the next day, the signal was visibly improved. In 2007, the United States committed to never use SA again [68].

The Gulf War in 1991 was a major turning point for GPS, demonstrating it as an enabling technology for maneuver warfare [69]. There was a lack of GPS military receivers reported, and civil receivers were said to have been used, with reports of SA being turned off for some time during the conflict [70]. GPS jammers were reportedly deployed by the Iraqis [69].

The initial user equipment to deal with SA and AS was the Precision Pointing Security Module (PPS-SM), and these were generally available in 1994 [71]. However, the next generation of user equipment security modules was under development and in 1996, the SAASM Card Integration Program (SCIP) selected Alliant Tech to develop the first selective availability antispoofing module [72], and there was a Chairman of the Joint Chiefs of Staff instruction mandating the use of SAASM for PPS user equipment. SAASM provides for enhanced security, the use of encrypted black keys and over-the-

air-rekey [73, 74]. The stated use of an encrypted signal was to prevent signal spoofing [71].

As part of the 1996 Presidential Decision Directive, the Department of Defense (DoD) initiated a review of the GPS system. Starting in 1998 the first results were announced [75] and by 2000–2001 a plethora of additional results were announced [76, 77]. The upshot was new military and civil signals, the military signals were retrofitted into the last GPS IIR (replenishment) spacecraft and are designated as the IIR-Ms. The next block, IIF ("follow-on"), was the first of the modernized vehicles, with the L5 and M-Code signals. Twelve GPS IIFs launched between 2010 and 2016. The GPS III spacecraft, with launches starting in 2018, also has new civil signals L2C and L5, and M-Code the modernized military code.

7.6.2 2000s

In 2001, the Department of Transportation's Volpe Center issued *Vulnerability Assessment of the Transportation Infrastructure Relying on Global Positioning System* [65], a comprehensive look at GPS as a system and how it is used and a frank assessment of the potential problems that could occur. This influential report spurred a lot of investigation into how GPS is used. Fairly soon after this report, academic interest shifted to look seriously at GPS vulnerabilities. Shortly afterward, in 2004, the U.S. Department of Defense established the Joint Navigation Warfare Center in New Mexico and their responsibilities include PNT (position, navigation, and timing) testing for the DoD [78].[5]

Academic papers on GPS threats started coming out of Mark Psiaki's group at Cornell University after the Volpe report [79], and one of his students, Todd Humphreys, took a position at the University of Texas and published more than a score of papers on attacks and countermeasures for GPS starting in 2009 [80].

More germane was the beginning of papers to counter the threat of spoofed signals, notably one by Logan Scott [81], who proposed combining digitally signing the navigation message and inserting spread spectrum security codes (SSSC). Scott's paper begins with a quote from the Volpe report. He proposed using the digital signature as a key to generate a cryptographically generated puncture (the SSSC). This idea is possible since the spacecraft knows the messages well in advance of the broadcast (e.g., dozens of messages in advance). He also notes that the substantial penalty is a delay in authentication; for critical applications, he proposes using SSSCs with a preplaced key for authorized users. There is a pretty cogent analysis of the issues associated with trying to protect preplaced keys and proposes using the FIPS-140-2 process at level 3 to protect the keys. A paper with similar ideas was written by M. Kuhn [82].

A Galileo-inclined group proposed NMA [83] for GPS and Galileo. The Galileo team picked up on this and a few years later, and there was a serious proposal [84, 85]. The current vision of the system as of 2023 is described in [86], and the Open Service Navigation Message Authentication is documented in the signal-in-space Interface Control Document [87].

The use of the TESLA protocol (see Section 4.8) to get around the size of digital signatures and to increase loss tolerance seems to have occurred in several places, with it first appearing in general publication in [84].

7.7 Takeaways

For the satnav enterprise, cryptography is used in similar protocols as with other enterprises, such as communication and networking. These protocols require good information security between the various segments in the enterprise, such as in the TT&C signals, the monitoring station networks, and distribution of information to the users.

For satnav signals, cryptography applies to the different layers in different ways. These protocols are specialized because of the nature of the signals and the specifics of the user base. Creating cryptographic methods that protect signals buried in the noise is challenging. Enabling cryptography among millions of users even more so. Exploring such methods is the subject of the next few chapters.

End Notes

1. Consider satellite radio services. These are broadcast systems, where a single signal is transmitted and then received by many users, but only authorized users should be able to receive the signal. This requirement puts some trust in the end user, or more accurately, in the end user's device. There is a trade-off between how well the security goal is achieved versus burden on the user. Note the security goal is asymmetric in that the user likely is not concerned about the exclusivity of the signal to other people.

2. Some schemes that use bit commitment, such as TESLA, are similar to challenge-response in that the sender commits to a challenge that they then must respond to. However, these methods differ, of course, from true challenge-response because the verifier (the user) did not generate the challenge.

3. Cross-links are also sometimes called intersatellite links (ILS), especially in non-U.S. contexts [88]. The U.S./GPS community likely avoids the use of ILS for intersatellite communication because of the use of ILS as an acronym for instrument landing system in the aviation community The use of cross-link for intersatellite communication has been in use since it was first demonstrated in the mid-1970s by Lincoln Labs' LES 8/9.

4. In 2013, a truck driver was fined $32,000 for using a device to defeat GPS tracking of his work vehicle in 2009; he was reportedly also fired [89]. His actions had a significant impact on the Newark Airport, which was severely disrupted.

5. There are references to NAVWAR (navigation warfare, differentiated from naval warfare, which gets similarly abbreviated), in DoD solicitations in the mid-1990s [90, 91]. An excellent history of the GPS military program from the origin of the modernization program, in a Presidential Decision Directive in 1996 through 2013 is [91], written by two U.S. Air Force officers who were in the GPS Joint Program Office in the 1990s and returned to the GPS program as GPS program senior leadership around 2010.

References

[1] Wertz, J., D. Everett, and J. Puschell (eds.), *Space Mission Engineering: The New SMAD,* Volume 28, Microcosm Press, 2011.

[2] Sherwood, N., A. Clark, and D. Lynas, *Enterprise Security Architecture: A Business-Driven Approach,* Boca Raton, FL: CRC Press, 2021.

[3] Anderson, R., *Security Engineering: A Guide to Building Dependable Distributed Systems,* Third Edition, Indianapolis, IN: John Wiley & Sons, 2020.

[4] Moyle, E., and D. Kelley, *Practical Cybersecurity Architecture: A Guide to Creating and Implementing Robust Designs for Cybersecurity Architects,* Birmingham, UK: Packt Publishing, 2020.

[5] "Pirate Decryption," Wikipedia, May 10, 2024, https://en.wikipedia.org/w/index.php?title=Pirate_decryption&oldid=1223131546.

[6] Pearson, J., "Russia Hacked Ukrainian Satellite Communications, Officials Believe," *Reuters,* March 25, 2022, https://www.bbc.com/news/technology-60796079.

[7] "Case Study: Viasat Attack," *CyberPeace Institute*, 2023, https://cyberconflicts.cyberpeaceinstitute.org/law-and-policy/cases/viasat.

[8] Reinsch, W. A., et al., *Report to Congress of the U.S.-China Economic and Security Review Commission,* November 2011.

[9] Office of the Inspector General NASA, *Cybersecurity Management and Oversight at the Jet Propulsion Laboratory*, Report IG-19-022, NASA, June 19, 2019.

[10] Office of the Inspector General NASA, *NASA's Cybersecurity Readiness,* Report No. IG-21-019, May 18, 2021.

[11] CCSDS, *CCSDS.Org-Blue Books: Recommended Standards*, 2024, https://public.ccsds.org/Publications/BlueBooks.aspx.

[12] Scholl, M., and T. Suloway, *Introduction to Cybersecurity for Commercial Satellite Operations,* NIST Internal or Interagency Report (NISTIR) 8270, National Institute of Standards and Technology, July 25, 2023, doi:10.6028/NIST.IR.8270, https://csrc.nist.gov/pubs/ir/8270/final.

[13] CISA, *Strengthening Cybersecurity of SATCOM Network Providers and Customers—CISA*, May 10, 2022, https://www.cisa.gov/news-events/cybersecurityadvisories/aa22-076a.

[14] "Cybersecurity Principles for Space Systems," *Federal Register,* September 10, 2020, https://www.federalregister.gov/documents/2020/09/10/2020-20150/cybersecurity-principles-for-space-systems.

[15] CISA, *All Resources & Tools—CISA*, May 2, 2024, https://www.cisa.gov/resources-tools/all-resources-tools.

[16] Taylor, R. N., N. Medvidovic, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice,* Hoboken, NJ: John Wiley & Sons, 2009.

[17] Deogun, D., D. Bergh Johnsson, and D. Sawano, *Secure by Design,* Shelter Island, NY: Manning, 2019.

[18] Adkins, H., et al., *Building Secure and Reliable Systems: Best Practices for Designing, Implementing, and Maintaining Systems,* Beijing: O'Reilly Media, 2020.

[19] Rais, R., C. Morillo, E. Gilman, and D. Barth, *Zero Trust Networks: Building Secure Systems in Untrusted Networks*, Second Edition, Beijing: O'Reilly Media, 2024.

[20] Dovis, F., *GNSS Interference Threats and Countermeasures,* Norwood, MA: Artech, 2015.

[21] Information Technology Laboratory Computer Security Division, *About the RMF NIST Risk Management Framework—CSRC,* CSRC-NIST. November 30, 2016, https://csrc.nist.gov/Projects/ risk-management/about-rmf.

[22] Joint Task Force NIST, *Security and Privacy Controls for Information Systems and Organizations,* NIST Special Publication (SP) 800-53 Rev. 5, National Institute of Standards and Technology, December 10, 2020, doi:10.6028/NIST.SP.800-53r5, https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final.

[23] SDA, *SDA Issues Request for Information for Tranche 2 PNT Service Payload–Space Development Agency*, November 21, 2022, https://www.sda.mil/sda-issues-request-for-information-for-tranche-2-pnt-service-payload/.

[24] Jewett, R., "Satellite Navigation Startup Xona Raises $19M in Series A Round," *Via Satellite,* May 8, 2024, https://www.satellitetoday.com/finance/2024/05/08/satellite-navigation-startup-xona-raises-19m-in-series-a-round/.

[25] "Aerospacelab and Xona Space Systems Partner to Revolutionize Navigation Satellite Systems,"*Aerospacelab,* March 19, 2024, https://www.aerospacelab.com/blog/press-releases-1/aerospacelab-and-xona-space-systems-partner-to-revolutionize-navigation-satellite-systems-16.

[26] Xona Space Systems—Precision Navigation—United States, 2024, https://www.xonaspace.com.

[27] U. S. Department of Defense, *MIL-STD-1553C*, 2018.

[28] ESA-ECSS, "ECSS-E-ST-50-51C–SpaceWire Protocol Identification (5 February 2010)," European Cooperation for Space Standardization, 2010, https://ecss.nl/standard/ecss-e-st-50-51c-spacewire-protocol-identification-5-february-2010/.

[29] CCSDS, "Space Packet Protocol," 2020.

[30] "NASA Helps Make Connections in Air and Space," *NASA Spinoff*, 2024, https://spinoff.nasa.gov/NASA_Helps_Make_Connections_in_Air_and_Space.

[31] "Microchip Announces Industry's First Space-Qualified COTS-Based Radiation-Tolerant Ethernet Transceiver and Embedded Microcontroller," *Microchip,* 2024, https://www.microchip.com/en-us/about/news-releases/products/microchip-space-qualified-cots-based-radiationtolerant-transceiver-and-microcontroller.

[32] Miles, M., "Space-Conformant Gigabit Ethernet Switch IP for RTG4™ FPGAs," *DornerWorks,* August 19, 2022, https://www.dornerworks.com/ blog/space-switch-rtg4/.

[33] Ananda, M. P., H. Bernstein, W. A. Feess, and T. C. Paugstat, "Global Positioning System (GPS) Autonomous User System," *Navigation,* Vol. 35, No. 2, June 1, 1988, pp. 197–216.

[34] Boyd, S., and L. Vandenberghe, *Convex Optimization,* Cambridge, UK: Cambridge University Press, 2004.

[35] Jin, Y., and B. Sendhoff, "Trade-Off between Performance and Robustness: An Evolutionary Multiobjective Approach," in *Evolutionary Multi-Criterion Optimization* (C. M. Fonseca, P. J. Fleming, E. Zitzler, L. Thiele, and K. Deb, eds.), Volume 2632, Berlin: Springer, 2003, pp. 237–251, doi:10.1007/3-540-36970-8_17, http://link.springer.com/10.1007/3-540-36970-8_17.

[36] Keppelmann, M., *Designing Data-Intensive Applications,* Beijing: Beijing: O'Reilly, 2017, https://www.oreilly.com/library/view/designing-dataintensive-applications/9781491903063/.

[37] *Resilient Positioning, Navigation, and Timing (PNT) Reference Architecture,* Department of Homeland Security, June 9, 2023, https://www.dhs.gov/science-and-technology/publication/resilient-pnt-reference-architecture.

[38] Lo, S. C., "Broadcasting GPS Integrity Information Using Loran-C," PhD thesis, Stanford University, July 2002.

[39] Martin, J. et al., *Network Time Protocol Version 4: Protocol and Algorithms Specification*, RFC 5905, June 2010, doi:10.17487/RFC5905, https://www.rfceditor.org/info/rfc5905.

[40] Federal PKI, *Modernize Federal Identities*, June 6, 2024, https://www.idmanagement.gov.

[41] FAA, Welcome to the WAAS Test Team Website, June 6, 2024, https://www.nstb.tc.faa.gov/.

[42] EGNOS, Realtime—EGNOS User Support Website, June 2024, https://egnos.gsc-europa.eu/egnos-system/realtime.

[43] P1952 Standard for Resilient Positioning, Navigation and Timing (PNT) User Equipment, IEEE, https://sagroups.ieee.org/p1952/.

[44] Positioning, Navigation, and Timing (PNT) Program, DHS, https://www.dhs.gov/science-and-technology/pnt-program.

[45] NIST Cybersecurity, NIST, https://www.nist.gov/cybersecurity.

[46] CVE, CVE Program, https://www.cve.org/.

[47] MITRE, *MITRE ATT&CK,* June 2024, https://attack.mitre.org.

[48] Aerospace Corporation, "SPARTA," February 29, 2024, https://sparta.aerospace.org/.

[49] Garbis, J., and J. W. Chapman, *Zero Trust Security: An Enterprise Guide,* Berkeley: Apress, 2021.

[50] Beaulieu, A., *Learning SQL: Generate, Manipulate, and Retrieve Data,* Third Edition, Beijing: O'Reilly Media, 2020.

[51] CCSDS, Space Data Link Security Protocol—Extended Procedures, CCSDS 355.1-B-1, Blue Book, February 2020.

[52] CCSDS, Space Data Link Security Protocol, CCSDS 355.0-B-2, Blue Book, July 2022.

[53] CCSDS, CCSDS Authentication Credentials, CCSDS 357.0-B-1, Blue Book, July 2019.

[54] CCSDS, CCSDS Cryptographic Algorithms, CCSDS 352.0-B-2, Blue Book, August 2019.

[55] "John Deere's New Security Feature Locks Thieves Out," *Farmers Guide,* March 25, 2019, https://www.farmersguide.co.uk/machinery/john-deeres-new-security-feature-locks-thieves-out/.

[56] CESAR—The Construction and Agricultural Equipment Security and Registration Scheme, CESAR, 2024, https://www.cesarscheme.org/index.php.

[57] Committee on National Security Systems, CNSSI No. 1200 National Information Assurance Instruction for Space Systems used to Support National Security Missions, 2014.

[58] *GPS Receiver Whitelist Development Guide,* Department of Homeland Security, July 12, 2021.

[59] GLONASS Signal Plan GLONASS Signal Plan, EAS Navipedia, 2011, https://gssc.esa.int/navipedia/index.php/GLONASS_Signal_Plan.

[60] McGurn, T., Space-Based Navigation and Timing Advisory Board, https://www.gps.gov/governance/advisory/meetings/2009-11/.

[61] Goward, D., "What Happened to GPS in Denver?" *GPS World,* September 21, 2022, https://www.gpsworld.com/what-happened-to-gps-in-denver/.

[62] "The Unsolved Mystery of the 2022 Texas Interference," *Inside GNSS,* September 7, 2023, https://insidegnss.com/the-unsolved-mystery-of-the-2022-texas-interference/.

[63] Psiaki, M. L., and T. E. Humphreys, "GNSS Spoofing and Detection," in *Proceedings of the IEEE,* Vol. 104, No. 6, 2016, pp. 1258–1270, doi:10.1109/JPROC.2016.2526658.

[64] Günther, C., "A Survey of Spoofing and Counter-Measures," *Navigation,* Vol. 61, No. 3, September 2014, pp. 159–177, doi:10.1002/navi.65.

[65] Volpe Transportation Center and United States. Department of Transportation, *Vulnerability Assessment of the Transportation Infrastructure Relying on Global Positioning System,* August 29, 2001, https://rosap.ntl.bts.gov/view/dot/8435.

[66] USNO, *Global Positioning System Overview,* 2024, https://www.cnmoc.usff.navy.mil/Our-Commands/United-States-Naval-Observatory/Precise-TimeDepartment/Global-Positioning-System/Global-Positioning-System-Overview/

[67] Office of Science and Technology National Security Council, Press Release, "U.S. Global Positioning System Policy," March 1996, https://clintonwhitehouse4.archives.gov/WH/EOP/OSTP/html/gps-factsheet.html.

[68] The White House, Statement by the Press Secretary, 2007, https://georgewbush-whitehouse.archives.gov/news/releases/2007/09/20070918-2.html.

[69] Greenmeier, L., "GPS and the World's First 'Space War'," *Scientific American,* February 8, 2016, https://www.scientificamerican.com/article/gps-and-the-worlds-first-space-war/.

[70] McNamara, J., *GPS for Dummies,* Second Edition, Hoboken, NJ: John Wiley & Sons, 2008.

[71] Cox, T., Jr., *PPS GPS: What Is It? And How Do I Get It,* https://apps.dtic.mil/sti/citations/ADA281345.

[72] Goussak, K., T. Kusserow, and B. Goblish, "Review and Analysis of the Selective Availability Anti-Spoofing Module (SAASM) Card Integration Program (SCIP)," in *Proceedings of the 54th Annual Meeting of The Institute of Navigation,* June 3, 1998, pp. 585–592, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=1282.

[73] Operational Test and Evaluation Office of the Director, *2011 Annual Report,* 2011, https://www.dote.osd.mil/Annual-Reports/2011-Annual-Report/.

[74] Holm, R., "Why Convert to a SAASM-based Global Positioning System?" *Military Embedded Systems,* 2005.

[75] Lucia, D. J., and J. Anderson, "Analysis and Recommendation for the Reuse of the L1 and L2 GPS Spectrum," in *Proceedings of the 11th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 1998),* September 18, 1998, pp. 1877–1886, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=3125.

[76] Loverro, D., "Plenary Session – GPS Modernization," in *Proceedings of the 57th Annual Meeting of The Institute of Navigation,* June 13, 2001, pp. 1–32, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=930.

[77] Barker, B. C., "GPS Military Signal Modernization: Updates to Design and Characteristics," in *Proceedings of the 14th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 2001),* September 14, 2001, pp. 2716–2721, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=1951.

[78] JNWC, Kirtland Air Force Base > Units > Joint Navigation Warfare Center, 2024, https://www.kirtland.af.mil/Units/Joint-Navigation-Warfare-Center/.

[79] Humphreys, T. E., et al., "Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer," in *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008),* September 19, 2008, pp. 2314–2325, http://www.ion.org/publications/abstract.cfm?jp=p& articleID=8132.

[80] Humphreys, T. E., et al., "Assessing the Spoofing Threat," *GPS World,* January 1, 2009, https://www.gpsworld.com/defensesecurity-surveillanceassessing-spoofing-threat-3171.

[81] Scott, L., "Anti-Spoofing & Authenticated Signal Architectures for Civil Navigation Systems," in *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003),* September 12, 2003, pp. 1543–1552, http://www.ion.org/publications/abstract.cfm?jp=p& articleID=5339.

[82] Kuhn, M. G., "An Asymmetric Security Mechanism for Navigation Signals," in *Information Hiding,* (J. Fridrich, ed.), Berlin: Springer, 2005, pp. 239–252, doi:10.1007/978-3-540-30114-1_17.

[83] Wullems, C., O. Pozzobon, and K. Kubik, "Signal Authentication and Integrity Schemes for Next Generation Global Navigation Satellite Systems," in *European Navigation Conference (ENC-GNSS 2005)*, 2005, https://eprints.qut.edu.au/38275.

[84] Fernandez-Hernandez, I., et al., "Design Drivers, Solutions and Robustness Assessment of Navigation Message Authentication for the Galileo Open Service," in *Proceedings of the 27th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2014)*, September 12, 2014, pp. 2810–2827, https://www.semanticscholar.org/paper/Design-Drivers%2C-Solutions-and-Robustness-Assessment-Fern%C3%A1ndez%E2%80%90Hern%C3%A1ndezRijmen/9d9c6a4cc33b7e1ae3bbf0d58c5ff3343ce6c9c7.

[85] Fernandez-Hernandez, I., et al., "A Navigation Message Authentication Proposal for the Galileo Open Service: NMA Proposal for Galileo OS," *Navigation,* Vol. 63, No. 1, March 2016, pp. 85–102, doi:10.1002/navi.125.

[86] Fernandez-Hernandez, I., et al., "Galileo Authentication and High Accuracy: Getting to the Truth," *Inside GNSS,* February 13, 2023, https://insidegnss.com/galileo-authentication-and-high-accuracy-getting-to-the-truth/.

[87] European GNSS Supervisory Authority, *Galileo Open Service Navigation Message Authentication (OSNMA): Signal in Space Interface Control Document (SIS ICD),* Issue 1.0, December 2022, https://data.europa.eu/doi/10.2878/594840.

[88] "BeiDou Completes Inter-Satellite Link Testing; Only GNSS with This Accuracy-Improving Feature," *Inside GNSS,* https://insidegnss.com/beidou-completes-inter-satellite-link-testing-only-gnss-with-this-accuracyimproving-feature/.

[89] "FCC Fines Operator of GPS Jammer That Affected Newark Airport GBAS, *Inside GNSS,* August 31, 2013, https://insidegnss.com/fcc-fines-operator-ofgps-jammer-that-affected-newark-airport-gbas/.

[90] NAVWAR ACTD, InitiativeWriteup-new, Navigational Warfare (NAVWAR) Advanced Concept Technology Demonstration (ACTD), September 10, 1997, http://www.wslfweb.org/docs/roadmap/irm/internet/nav/init/html/navwar.htm.

[91] Gruber, B. J., and J. Anderson, "Space Superiority, Down to the Nanosecond," *Air and Space Power Journal*, September 2013.

8

Navigation Message Authentication

Cryptography for navigation signals can be divided into three cases: methods to protect the data layer, methods to protect spreading code and the modulation layer, and methods that link those two parts together. We begin this chapter with a discussion about protecting the navigation data messages. We generally call this protection *navigation message authentication* because the focus is on the integrity and authentication of the data messages. Confidentiality of the messages is rarely a goal for messages on open navigation signals (i.e., signals that are available to everyone). The next chapter will look at cryptography applied to signal protections; we start here with NMA because historically this was the first proposal for navigation protections and remains the most mature.

We begin with a look at the various goals and constraints for NMA followed by the history and a tour of possible protection methods. We then delve into specific methods: Galileo's *Open Service NMA (OSNMA)* and methods in the Chimera signal. Our discussions of real systems must necessarily be brief on specific details, but we will indicate the main concepts and tie them to the cryptographic methods from Part II of this book.

8.1 Protecting Navigation Data

On the one hand, protecting navigation messages should be very straightforward. Chapters 4 and 5 gave methods for authenticating data based, respectively, on shared keyed and public key cryptography. On the other hand, the nature of satnav adds some nuances that constrain and often prevent such methods.

8.1.1 Goals

Messages on navigation signals have two important traits:

- *Messages are ephemeral.* They indicate things like the current health of the system, almanac and ephemeris for the SVs, and possibly other status information. A UE needs this information to compute PNT information, but once the lifetime of the information is over, there is little reason for the UE to save it. (The enterprise may wish to save the information for logging and auditing.)
- *Messages are universally available.* Our focus is on open satnav systems, which are available to all users. That means the messages need to be accessible for all users.

Given these two traits, the security goals are first, the integrity of the messages—they should not be modifiable, and second, the UE should be able to establish the authentication of the messages. Confidentiality is not a goal, and while availability is a goal, it falls under the more widespread goal of availability of the signal.

There are specific performance goals tied to the security goals. Because navigation devices are often constrained by resources such as power, any methods to achieve the security goals should be conservative in resource utilization. Resources include the complexity of the cryptographic processing, along with any required storage. System-level goals also relate to the complexity of implementation (e.g., any infrastructure required to support the cryptographic methods such as distribution of cryptographic material).

8.1.2 Constraints

NMA has some constraints due to the nature of satnav. Perhaps foremost is the limited data bandwidth in satnav signals. Data rates can be on the order of 50 bits per second before error correction. At such rates, a digital signature of 500 bits would take 10 seconds to receive, which is both a large latency and a large use of the data channel just for authentication. This limitation is even more pronounced when one considers the ultimate need for quantum-resistant digital signatures (see the appendix).

This limitation has pushed the use for more bandwidth-efficient methods such as TESLA (Section 4.8). In addition, since the data messages are ephemeral, the authentication needs are also ephemeral, which could lower the requirement of the size of the authentication information. This trait is again pertinent with the trade-offs for using TESLA, as we see later in this chapter.

Satnav is an open broadcast system with the system sending messages to millions of users. That characteristic means that shared key cryptographic methods are off the table: there would be too much of a trust issue with millions of users having to share the same key. That suggests that public key methods must serve at least in part with any NMA solution.

8.1.3 Measures

The fundamental measure for NMA is how well the authentication material indicates a violation of the security goals. That measure is a function of the strength of the cryptographic methods, as we discussed in Part II of this book. Again, the strength needs to be measured taking into account what the threat can actually accomplish: Can the data messages be spoofed or modified in the narrow window of time that the UE cares about?

Related to strength is latency. How long does it take for a UE to verify the authentication information? For example, a data message that takes 10 seconds to receive followed by authentication material that takes an extra 5 seconds means that it is at least 5 seconds, and up to 15 seconds for the bits in the data message to be pronounced verified after reception. Such latency means that threats may only be detected after they have occurred.

The latency measure leads to the notion of *time to first authenticated fix (TTFAF)*, which is the time it takes to authenticate all the information needed for the PNT solution. TTFAF could be the same as latency if only some messages are needed and they arrive in parallel.

Latency and TTFAF are a function of the message error rates in the channel called the *authentication error rate (AER)*. Messages that arrive in error mean the UE needs to wait for a retransmission before it can be authenticated. Thus, lower SNR scenarios will naturally have higher AER, latencies and TTFAFs.

There is the notion of *time between authentication (TBA)*. This quantity will depend on the spacing of the signal between authentication information. Note the fundamental trade-off that a lower TBA means a greater portion of the data channel is only being used for authentication and not to send navigation information.

Finally, authentication methods have an opportunity cost on the data layer. Namely, bits used for authentication are not being used for other services. That loss of bits may impact other functions in the UE.

8.2 History

Before describing the various methods for NMA, we give a brief overview of the history behind NMA. This history serves to motivate the evolution of the

methods. In some ways, the actual methods are straightforward; the challenges are usually more from a system perspective.

8.2.1 Original Thoughts

Since digital signature methods have existed since the early 1980s, it is natural to apply them to navigation information, especially given the rise of security concerns outlined in the previous chapter. Various ideas were being explored by the end of the 1990s because of the Volpe report [1].[1] Explorations were aided by emerging protocols like TESLA from 2000.

The first open literature discussion is perhaps that of the paper by Scott [2] in 2003. Scott's paper is seminal for not only NMA but also for the techniques in Chapters 9 and 10. Specifically, the Level 1 of [2] outlines NMA. A few years later, there are the survey articles in *InsideGNSS* by Hein et al. [3, 4]. These articles provide a very brief overview of cryptography and then offer possible applications. One section is devoted to NMA. Finally, we mention the work by Wesson et al. [5].

8.2.2 Recent Trends

The modern take on NMA begins with the 2016 paper by Fernández-Hernández et al. [6]. This paper was a precursor to the emergence of NMA on Galileo signals. The results are presented later in this chapter. The experimental satellite NTS-3 began defining its authentication around this time. The Chimera protocol defined in [7] occurred in parallel and includes in part an NMA solution. Chimera is defined in Chapter 10, although this chapter presents the details on its NMA methods.

8.3 General Methods for NMA

There are two current techniques to provide authentication/integrity protection for navigation messages due to the broadcast nature of satnav.[2] The first is the use of digital signatures; the reader should recall the discussions in Chapter 5, such as in Section 5.2. The second is to use the more data bandwidth efficient technique TESLA (Section 4.8). While TESLA uses shared key MACs to provide authentication, the complete scheme still relies on the use of the occasional digital signature.

Both of these schemes can be either in-band our out-of-band. In-band means the UE receives all the information it needs through the navigation signal. Out-of-band uses a non-GNSS channel to transfer authentication information. This use removes the constraint on the data bandwidth, but, of course,

requires connectivity to that channel for the UE. We discuss each of these ideas in more detail first for in-band and then describe the changes for out-of-band.

One important fact should be stressed for any method: their use by a UE is optional. That is, a UE could ignore the authentication information, seemingly with the accompanying risk, and still be able to obtain position and time.

8.3.1 Using Only Digital Signatures

The use of digital signatures to sign navigation messages is often what is referred to when people say "NMA." The main concept is to take a message or group of messages and then send the digital signature of those message(s). The digital signature is sent in another message after the signed messages. A general framework is shown in Figure 8.1.

The steps associated with Figure 8.1 are as follows:

1. The satnav system creates all of the necessary information for the NMA scheme. This information includes the scheduling for the digital signatures ("DigSig" in the figure), the parameters and algorithm to be used, and the public/private keys.
2. The information required by the SVs is uploaded. Assuming that the SV creates the digital signatures, this information must include the private key, which is shared by every SV.
3. The public information, such as algorithm and public key, needs to be distributed to all UEs.
4. On the SV, the digital signature is computed using one or more messages according to the schedule. Each SV does this separately.

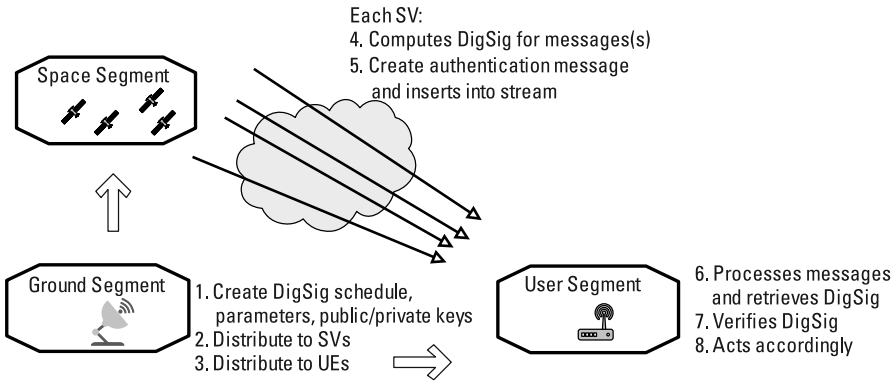

Figure 8.1 General system view of using digital signatures for NMA.

5. The digital signature needs to be bundled into its own message and inserted into the message stream.
6. In the UE, the messages are received and processed, which will include any error detection and correction.
7. The pertinent messages and digital signature are extracted and the digital signature is verified.
8. The UE acts accordingly based on the verification procedure.

A general view of the signing procedure is shown in Figure 8.2. The transmitter gathers one or more messages to be signed, signs them, and then puts the signature in another message. On the receiving end, the messages must be gathered and bundled together until all are received. Then, once the signature is also received, the whole collection can be verified.

Note that the process cannot begin until everything is received. While the NMA steps are straightforward, there are some variations that need to be noted. First, it is possible that the digital signatures are generated on the ground. This possibility has the advantage of flexibility, but of course not all information is then protected (e.g., any data generated by the SV, such as health information). There is also the issue of how often information can be uploaded. On the other hand, if the SV computes the digital signatures, then it needs to be able to handle the more intensive computation required in the usually limited on-board processor.

Second, there are trade-offs on the actual scheduling of the digital signatures. If the signatures are created too often, then a larger portion of the data bits are devoted just for authentication. If instead the signatures are generated

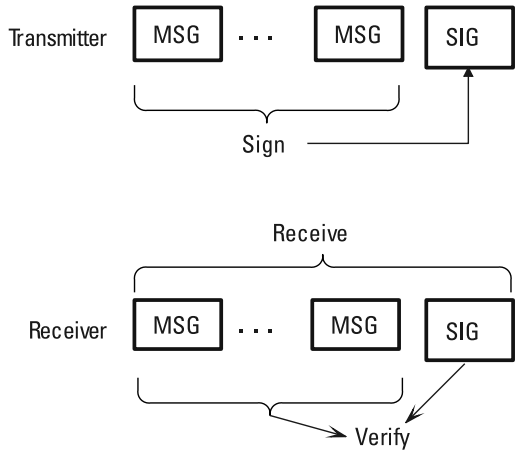

Figure 8.2 Signing and verifying NMA digital signatures.

more infrequently, then there is the issue that all signed material must be received error-free for the digital signature to verify.

Finally, we consider the performance metrics.

- The strength of the scheme depends on the strength of the digital signature. Since the information being protected is ephemeral, the signatures could have less cryptographic strength. Less strength means smaller digital signatures, which are needed for the navigation messages.
- Latency will depend on the length of time that the signed message(s) take to be received and the additional time for the digital signature. Signing more than one message makes for efficiency with regard to the use of data bits for the digital signatures, but at the cost of greater latency and the risk of a message being received in error.
- TTFAF comes from the time to receive signatures for all pertinent messages on at least four SVs. This latency will be comparable if the signatures are synched.

8.3.2 Using TESLA

Recall from Section 4.8 the TESLA protocol, which is designed to authenticate a stream of data. TESLA works by using shared key MACS for the authentication whose size can be adapted via truncation, unlike digital signatures. The protocol uses bit commitment, in that the shared key used for the MACs is sent in a later message. This later transmission adds latency, but usually the latency is improved overusing just digital signatures.

The keys used for the MACs form a chain, where repeated application of a one-way function like a hash function can be used to derive keys that were used earlier. However, the whole hash chain needs to be vetted via some digital signature. That requirement has two notable features:

- If the hash chain is divided into subchains among all the SVs (see below), then a digital signature on any SV serves to verify the whole chain;
- Because of the multiple subchains, the digital signatures can be staggered, which makes it more likely a UE can verify some signature more quickly.

Figure 8.3 shows the general framework for using TESLA. The steps are:

1. The satnav system creates all of the necessary information for TESLA. First, it periodically—say once a week—creates a long chain of

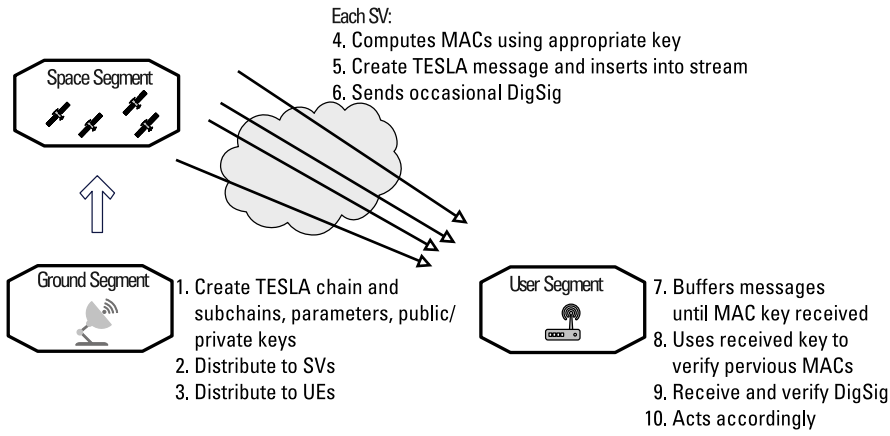

Figure 8.3 Creating and verifying TESLA MACs for NMA.

TESLA keys using a one-way function. It divides the chain into subchains for each SV. Associated parameters need to be defined, like MAC sizes and algorithm. Additionally, the algorithm and keys are defined for the digital signatures.

2. The information required by the SVs is uploaded. In particular, each SV receives the information to generate its subchain (i.e., because of the way the chain is generated, the SV could regenerate its subchain from some specific starting key). If needed, the SV will receive the private key, although it is possible that a digital signature is used to vet the chain and all subchains, and thus could be generated on the ground.
3. The public information, such as algorithm and public key, needs to be distributed somehow to all UEs. This information needs to include the specifics on the TESLA chains (i.e., when a UE can expect a switchover to a new TESLA chain).
4. On the SV, the SV computes the MACs for the messages using the current key, which is not broadcast until a later time.
5. The SV creates the TESLA message that has the previously used key along with the MACs and sends it.
6. Occasionally, the SV broadcasts a digital signature that is used to authenticate the TESLA chain. For example, it could sign one of the keys in the chain.
7. The UE receives the messages with their MACs and buffers them as it awaits the key used, which is sent later.

8. The UE uses the current received key to verify the MACs in the previous buffer.
9. If a digital signature is received, the UE verifies it.
10. The UE acts accordingly based on the various procedures.

A general view of the MAC procedure is shown in Figure 8.4. The transmitter gathers one or more messages to be protected and creates the MAC or possibly separate MACs. The current key from the TESLA key chain is used for the MAC. The MACs along with the previous key in the chain are bundled into a new message. On the receiving end, the messages must be gathered and bundled together until all are received; the MACs are also bundled. These MACs cannot be verified until the following set of messages and MAC/key message. Instead, the received key is used to verify the MACs in the messages received before.

A main advantage to using TESLA is that shared-key MACs are more efficient from both a computational perspective and number of bits used. Not only are MACs smaller for a given cryptographic strength compared to digital signatures, but MACs can be truncated, which means the size can be tuned to the cryptographic strength. Since the MAC is being used to verify information only for a few minutes, there can be significant savings here. See, for example, [8].

In addition, the efficiency in MAC sizes allows for flexibility in what the MAC applies to. While a digital signature may need to apply to several messages

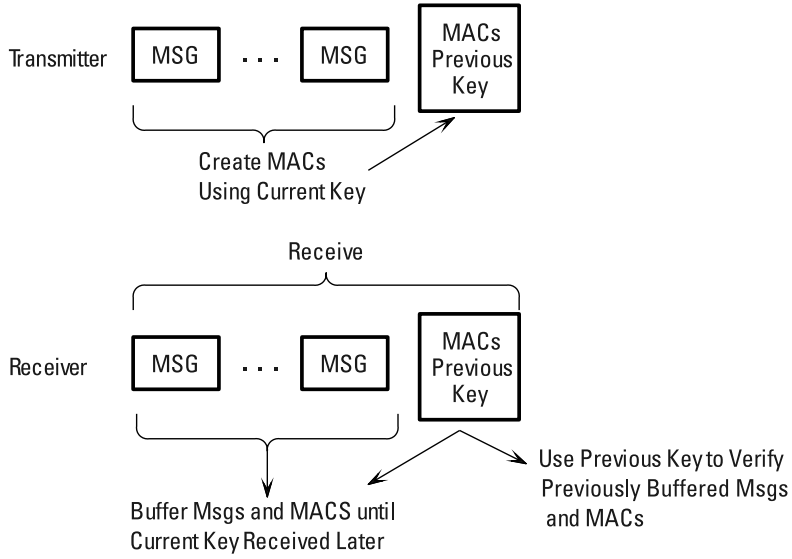

Figure 8.4 General system view of using MACs for NMA.

for data bandwidth efficiency, one could instead have a single smaller MAC apply to each individual message.

That allows for perhaps only a single message not being verified if errors occur.

Another advantage of using TESLA is that the TESLA chain can be broken into smaller subchains. Figure 8.5 shows the idea. The overall chain is length N with indexing starting at 0. If nominally we desire four subchains, then they can be formed by basically a raster scan of the overall chains. Of course, four would be larger in practice, and this construction assumes that N is divisible by the number of subchains.

From a scheduling standpoint, note that the MACs will need to be generated on the SV. The main reason is that it is unlikely all the required information could be uploaded in a timely fashion, and not being able to authenticate more often defeats the purpose of using TESLA. Fortunately, generating MACS is more computationally efficient, as is the generation of the keys in the subchain. Of course, these observations depend on the satnav system architecture.

There are similar trade-offs with regard to scheduling. As we mentioned, smaller MACs could occur more often. But the key that generated these MACs needs to appear (delayed) at some rate. Too infrequent is more efficient but increases latency.

Finally, we again consider the performance metrics.

- The strength of the scheme depends on the strength of the MACS. Again, since the information being protected is ephemeral, these MACs

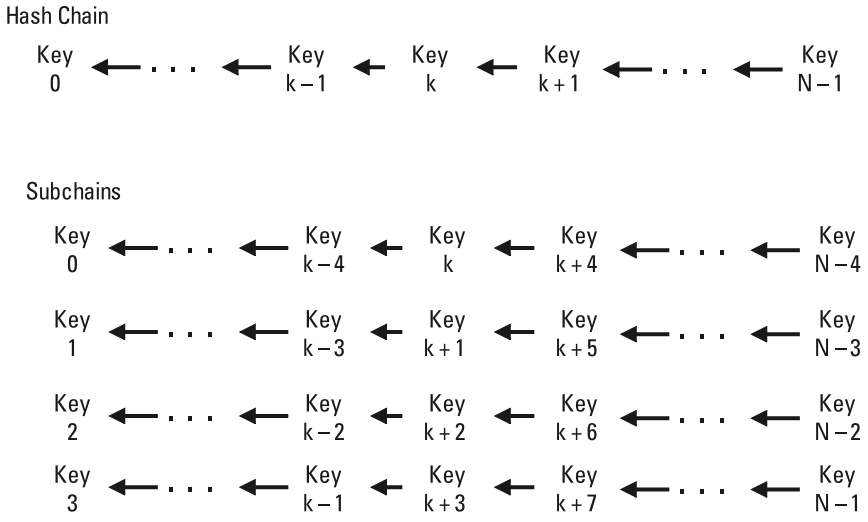

Figure 8.5 Creating TESLA subchains.

could have less cryptographic strength. The digital signature likewise could have trade-offs, since it is being used to authenticate the TESLA chain.

- Latency depends on the frequency of the TESLA keys being sent. That frequency also affects the size of the message buffers in the UE, although that is likely not a storage issue.
- The time for a full authentication solution would be compared to the time to receive signature for all pertinent messages on all SVs. Because of the subchains, this time is reduced if at least one TESLA key is received and the TESLA chain verified.

Finally, and very important, the security for a given receiver depends on its time uncertainty. Since TESLA is a bit commitment scheme, security is lost if the keys used for MACs are known to an adversary so that they can send messages with legitimate MACs. Fortunately, this time uncertainty is usually not an issue, being on the order of tens of seconds.

8.3.3 Out-of-Band

The above two methods were described in terms of in-band solutions, where all the authentication information is within the navigation signal. On the other hand, either of the methods could use an out-of-band channel to transfer the authentication information to the UE. This out-of-band channel is likely over the internet or cellular system, but dedicated transport is also possible. A general system view is shown in Figure 8.6.

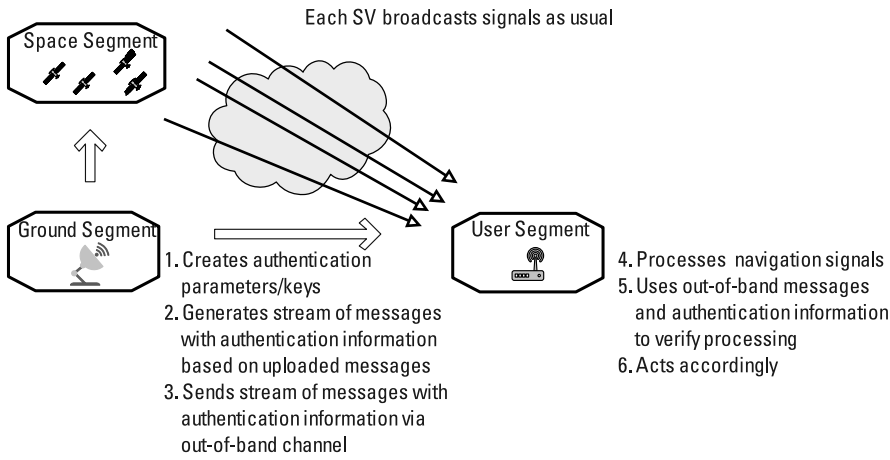

Figure 8.6 General system view using out-of-band methods for NMA.

In this method, the SVs act as usual, broadcasting the satnav signals. The steps for the other segments are as follows:

1. The ground segment begins by generating all the parameters and keys that will be used for authentication. This information also includes things like the schedule for authentication messages and the TESLA chain.
2. For the ongoing satnav signals, the control segment generates in parallel the pertinent messages and authentication information. This stream of messages is consistent with the messages that were uploaded, but the stream can contain more information. For example, the bandwidth in the out-of-band channel could permit sending the full almanac more often.
3. The ground segment then continually sends the stream of messages and authentication messages to the various UEs using some out-of-band channel. More than one channel could be used to accommodate different users.
4. On the UE end, it will process the satnav signals.
5. It will receive and verify the out-of-band stream of messages. This information should be consistent with its normal processing.
6. Finally, it will act accordingly. Besides deciding what to do if it does not authenticate, the UE may have the opportunity to indicate any issues using the same out-of-band link (i.e., to give situational awareness).

Since this method inherits the cryptography of the other methods, their system issues are comparable. Note that latency and frequency of information is likely as small as possible, because the data bandwidth in the communication channels is presumably much, much larger. That trait means MACs and digital signatures could be at full-strength. Indeed, if the transport is over the internet, then using TLS (Section 6.4.5) would enable the transmission of full strength material. That low latency also means that the time to a full authenticated solution is likewise low.

Finally, out-of-band methods have the important trait that they are more easily modified (e.g., changing algorithms, schedules, and parameters). It is much more difficult to change methods on the SVs.

8.3.4 Summary

All of these methods are viable in their own way, but also come with system constraints:

- Using just digital signatures is perhaps the simplest but has data bandwidth inefficiencies;
- Using TESLA is more bandwidth efficient, although the cryptographic protocol is more complicated;
- If an out-of-band channel is available, then either of these methods provides more timely authentication.

In Chapter 10, we show how considering implementations across the whole constellation can amplify the bandwidth efficiencies of using TESLA.

We stress that these methods provide data authentication and not authentication of the signal timing. That trait means that while there is protection from some threats, there is not protection against all threats. In addition, all of the example procedures had the last step of the UE "acting accordingly." We do not say in this book what a UE should do. Should it stop processing, raise a flag, call the authorities? There are many choices that all depend on the specific nature of the UE and its mission.

8.4 Galileo OSNMA

Galileo is the first GNSS to implement NMA in its enterprise with its OSNMA. The initial Galileo NMA concepts date from the origins of Galileo in such papers as [9] and the articles [3, 4]. This foreshadowing led to the paper in 2016 [6], which has many of the methods that were eventually adopted. This section provides an overview of OSNMA and discusses the cryptographic methods and rationales. Details are necessarily limited, and so the reader is referred to the various interface specifications listed below.

8.4.1 Overview

OSNMA resides on the Galileo E1 OS (Open Service) signal; see Chapter 2. This signal consists of a pilot and data signal, the latter has a data rate of 120 bits per second, given by 250 symbols per second. The higher data rate, than, say, GPS, makes the signal more amenable to NMA methods. Additionally, the fact that authentication was being considered as Galileo was being developed meant that there was flexibility in the design (e.g., no concern for any legacy implementations).

That said, there were some limitations in the early system, which led to certain design choices. Specifically, authentication information needed to be generated on the ground and uploaded to some set of satellites. This authentication would apply to both the uploaded satellite but also to those that could not have the information uploaded. The resulting idea was to have cross

authentication of the constellation. This cross authentication is made possible using TESLA methods, namely a single TESLA chain spread through the constellation.

OSNMA uses a TESLA scheme similar to that described above. There are several trade-offs in the design as indicated in [6]:

- The strength of the needed digital signature to provide the authentication of the scheme must be traded against the size of the signature. As such, signature strengths of 112–128 bits are considered, but even these yield signature sizes on the order of 500 bits (see Section 5.5.4).
- TESLA improves the data inefficiencies of using digital signatures, but then one has to trade off how often the signature is sent, since that is what authenticates the TESLA chain.
- There are trade-offs between using a single TESLA chain versus one per satellite. The former allows for easier cross authentication, while the latter is more secure in that there is not a single point of failure.
- There are some slight computational trade-offs, in that verifying the TESLA chain requires hashing down to the TESLA root key. Such concerns may point to using slightly smaller hash functions.
- Security in the TESLA chain depends on key size and the size of the MACs. Again, smaller keys with sufficient strength, say 128 bits, suffice. Small MACs may be viable in that gathering multiple MACs raises the overall strength accordingly, since an adversary would need to guess all the MACs. Compare that situation with the much larger MAC sizes used to protect longer-lasting data.
- Finally, there are system trade-offs for the public key needed for the digital signatures. How often should it be changed? How can it be distributed in an authenticated manner?

The result of these considerations is the current OSNMA design.

8.4.2 Overall Architecture

The overall architecture follows the general framework of Figure 8.4. There are differences (e.g., in where the authentication information is generated). Another look at the authentication workflow is shown in Figure 8.7 (similar to figures in [10]). On the left side are the main authentication components in the signal-in-space: the TESLA Root key, the TESLA chain keys, the NAV messages, and the MACs, also known as tags. On the right side of the figure are the receiver functions:

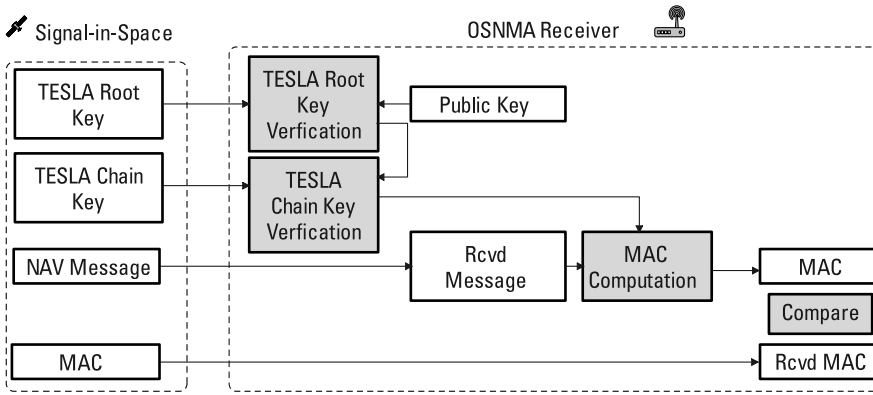

Figure 8.7 OSNMA functionality.

1. The TESLA root key is verified via a digital signature using the system public key;
2. The TESLA root key is used to verify the current TESLA chain key;
3. The TESLA chain key and the received NAV message(s) are used to compute the local MAC;
4. This computed MAC is compared with the received MAC. More details on the cryptographic functions are given in the next section.

Full details of OSNMA can be found in the following system documents:

Info Note. The *Galileo OSNMA Info Note* [10] provides an overview and use cases for OSNMA.

OSNMA SIS ICD. The document *Signal-in-Space Interface Control Document* [11] describes all the information in the signal for OSNMA. That information includes the message structure for the digital signatures, the TESLA keys, the MACs, and all associated messages.

Receiver Guidelines. This document [12] describes the functions that a receiver needs to perform for OSNMA. These functions include verification of the TESLA chain keys and the MACs.

In addition, see [13] for distribution of information over the internet such as the PKI certificates for OSNMA.

8.4.3 Cryptographic Methods

The operations shown in Figure 8.7 are achieved by a series of cryptographic methods. The paper [14] outlines the four main steps:

1. The public key needs to be authenticated any time it is updated. This goal is accomplished using a Merkle hash tree.
2. The resulting public key is used to verify the TESLA root key.
3. An authenticated TESLA root key can then be used to authenticate any TESLA chain key.
4. In particular, once the TESLA chain keys are authenticated, they can used to regenerate the MACs and compare those MACs to the received MACs. The result authenticates the messages.

We discuss each of these methods separately; all of them use the methods described in Part II.

8.4.3.1 Public Keys and a Merkle Tree

The Galileo system needs to accommodate distribution of the public keys for authentication of the TESLA chain via the TESLA root key. During the lifetime of the system, it is possible that more than one public key will be needed. As such, the system is set up to allow the user to verify a new key. These goals are accomplished by creating at the start of the system up to 16 possible public keys and using them as data to construct a Merkle hash tree. The result is shown in Figure 8.8, where the *Mi* represent the public keys along with any metadata.

As discussed in Section 4.6, the Merkle tree is constructed by taking successive hashes. That is, the hash for a parent is the hash of its two children concatenated together (e.g., the hash $X1,1$ is derived from the hashes $X0,2$ and $X0,3$). In that way, knowing the lower level nodes allows one to compute the values further up the tree. Using the properties of the hash function, any deviation from the original data (e.g., if a key were to be changed) would result in a mismatch in the top-level hash.

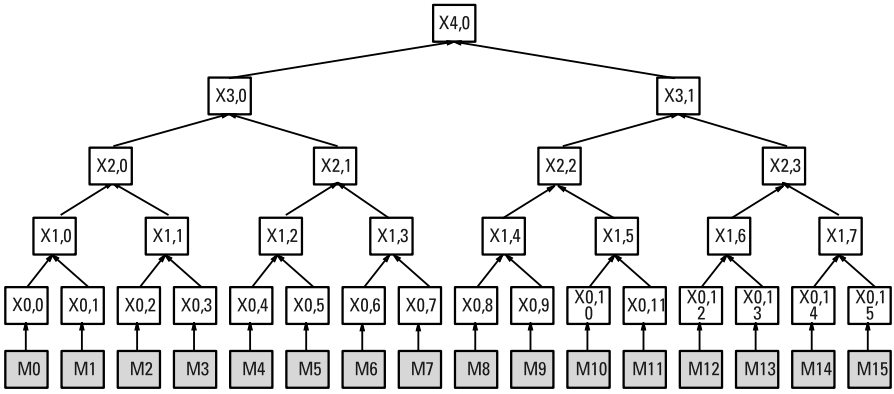

Figure 8.8 OSNMA Merkle Tree (Figure 18 in [11]).

For OSNMA, every receiver obtains the top-level hash at the factory. The public keys are sent over the air with the pertinent data. In addition, the values of the other hashes needed to reproduce the top-level hash are also transmitted. These values consist of the sibling hashes to the chain hash nodes above the current key. For example, above $M2$ is ($X0,2$), ($X1,1$), ($X2,0$), and ($X3,0$). The required sibling hashes would be ($X0,3$), ($X1,0$), ($X2,1$), and ($X3,1$). Using these hashes, which reveal no information, being hashes, the receiver can eventually compute the top-level hash $X4,0$ and compare it with its stored value. Thus, the receiver can verify the public key is correct. The current system uses SHA256 for the hash, although the ICD also allows for SHA3-256.

The benefit of this scheme is that the public key can be sent using the Galileo signal and yet still be authenticated. Since public keys usually do not need to be changed often, the limited size Merkle tree should be sufficient. There is minimal computational cost to the receiver since once it computes the required hashes, the public key is verified for that public key's lifetime, and presumably the receiver stores that state.

8.4.3.2 Signing TESLA Root Key

The public key is used to verify the digital signature of the TESLA root key. The signature is formed using ECDSA P-256 (and SHA256) or ECDSA P-521 (and SHA512). These resulting signature sizes are twice the size of the elliptic curve field. The inputs are the TESLA root key along with several metadata fields. Verification of the TESLA root key then follows using the standard ECDSA algorithms.

Receiver functionality as defined in [12] indicates that failure to validate the signature should result in the TESLA chain being discarded and the subsequent navigation data not authenticated. Otherwise, once the TESLA root key is authenticated, it can be used for the duration of the TESLA chain.

8.4.3.3 TESLA Chain

The TESLA chain is constructed as a series of hashes to produce successive keys; see Section 4.8 and the discussion earlier in this chapter. The hash function is again either SHA256 or SHA3-256. The hash function of a given key to produce the next key must be truncated. The SIS ICD [11] allows for key sizes between 96 bits and 256 bits in increments of 8 bits, although at the time of this writing the OSNMA operational configuration is fixed to 128-bit key lengths. The actual hash function uses as input the key, the system time, and a piece of unpredictable information that acts like a salt. The resulting chain ends with the key at index 0, which is the TESLA root key.

When a receiver obtains a TESLA chain key, it can perform the successive hashes down to the root key to verify that the thus obtained key matches the TESLA root key that it has authenticated. Note that if a receiver saves the

results of a such a calculation, it then only needs to compute hashes down to any previously verified key.

As we discussed earlier, there are trade-offs in terms of the length of the TESLA chain and the sizes of the keys. OSNMA allows for the flexibility to set these values during operation.

8.4.3.4 MACs

The truncated MACs, called *tags* in the Galileo specification, use a keyed hash along with the TESLA chain key. The MAC acts on a combination of information: the PRN of the satellite transmitting the data, the PRN of the satellite that is broadcasting the authentication information (these two quantities can differ), system time, a counter indicating where the tag falls, when the tag appears, and the navigation data. These fields are concatenated and possibly padded. The specific algorithm is HMAC using SHA256 or CMAC using AES.

Multiple tags can be sent in the OSNMA message. These tags can allow for cross authentication of the satellites or authentication of different data fields. There can be some flexibility in where these tags are placed, which is also indicated by the system using a fixed look-up table. This sequence information must also be verified via a MAC calculation along with the individual tags. Failure of any of the MACs indicates the resulting tags and messages should be discarded.

8.4.4 Performance

The article [15] shows several performance results related to OSNMA:

- For availability, the metric is to have at least four satellites in view in a 120-second window with sufficient authentication information available (i.e., if only certain satellites can transmit the information). In September 2022, this availability was over 99% worldwide.
- If sufficient tags are not received for all visible satellites, there could be an accuracy degradation with not being able to use everything in view. Studies show that this degradation was minimal, and presumably would become much less as OSNMA matures.
- Finally, TTFAF was measured in two cases: a stationary receiver with a clear view of the sky and a dynamic receiver in suburban conditions. Since a receiver must wait to receive the next TESLA chain key, which is at least 30 seconds, we expect that TTFAF will be greater than that value. The stationary receiver had TTFAF around 100 seconds, while the dynamic receiver was larger but still less than 200 seconds.

8.4.5 Future Plans

At the time of this writing, OSNMA is almost operational. Upgrades to the system are to move toward signal authentication methods, which we will detail in Chapter 10.

8.5 Chimera

The Chimera protocol on the GPS L1C signal was developed as part of the NTS-3 experiment satellite program. Chimera comprises methods for both protecting data and spreading codes, and as such it is fully defined in Chapter 10. This section describes the NMA portions of Chimera. The initial Chimera description is in [8] and the specifications in [16, 17].

8.5.1 NMA on Chimera

There are two flavors of NMA on Chimera applied to the L1C signal and CNAV2 messaging (see Chapter 2). CNAV2 messages are 18 seconds long and consist of three subframes:

Subframe 1: This subframe is 9 bits for the ITOW. It is protected by its own forward error correction.

Subframe 2: This subframe consists of 576 bits devoted to clock and ephemeris information.

Subframe 3: This subframe is variable date of 250 bits defined by specific page numbers.

Subframes 2 and 3 add a 24-bit CRC separately. In addition, LDPC coding is applied before the messages are sent. Since the cryptography is applied independent of the CRCs and before FEC, we do not discuss CRCs and FEC further.

The two flavors for NMA on NTS-3 are

Baseline Chimera: This NMA application uses only digital signatures to authenticate the CNAV2 data messages.

TESLA Chimera: This NMA application applies the TESLA protocol for the authentication.

A fundamental decision early in the development process was how many messages could be devoted to authentication. With only 250 bits per message in subframe 3 devoted to extra navigation information, any other enterprise data

may have demands on this limited resource. It was decided that a 20% allowance for authentication was possible and practical. Specifically, that means two subframe 3 messages per 10 messages (total). That tradeoff has ramifications for both NMA flavors, as we will see below, in terms of the sizes of the authentication information. It also indicates latency, since 10 messages, for example, represents 3 minutes of clock time.

8.5.2 Baseline Chimera

Baseline Chimera was the initial design; the specification is IS-AGT-100A [16]. The NMA portion uses a Chimera epoch comprising 10 consecutive messages. This collection of messages is authenticated with a single digital signature. The specific bits to be signed are indicated in Figure 8.9 (taken from Figures 20-3 and 204 in [16]). All messages have their subframe 1 and subframe 2 bits signed. The complete subframe 3 bits are also signed except for those messages that carry the digital signature. Note the CRCs are not signed, and the signing occurs before any forward error correction.

The bits in a Chimera epoch are concatenated together and some padding is applied.[3] The signature uses ECDSA for the field P-224 (see NIST SP 800-186, Section 3.2.1.2). The hash function is SHA-512 as defined in FIPS PUB 180-4. Using P-224 provides 112 bits of security, which was deemed sufficient for the NTS-3 experiments. Additionally, the resulting 448-bit signature fits into two 250-bit subframe 3 messages.

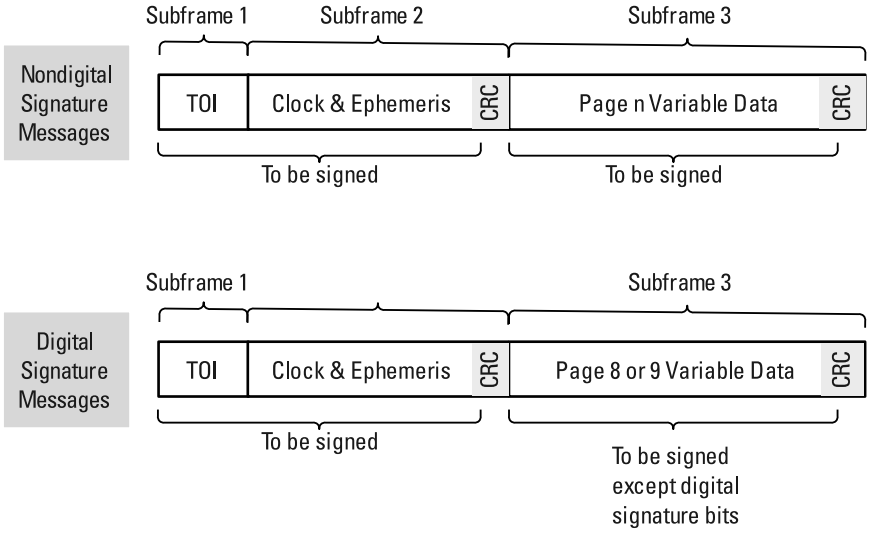

Figure 8.9 Baseline Chimera message bits to be signed [16].

The resulting signature is inserted as page 8 and page 9. Note these are the L1C page numbers used for NTS-3 and are not the official page numbers used by the GPS enterprise. Figure 8.10 shows the Chimera epoch and the locations of pages 8 and 9. Page 9 should always be the last message in the Chimera epoch. Page 8 can occur anywhere among the other 9 messages.

Actual performance will be measured as part of the NTS-3 experiments. Latency is predictable by the 3-minute Chimera epoch. AER will be a function of the SNR and the FEC on the L1C signal, which is more advanced than other GPS signals. However, note that the loss of any message in a Chimera epoch will invalidate the authentication for the whole epoch. Finally, TTFAF will need to be inferred, since only a single Chimera signal is likely to be available for use.

8.5.3 TESLA Chimera

TESLA Chimera was published after baseline Chimera. Its purpose is to swap out using digital signatures for NMA with using a TESLA scheme. Even though there is only a single Chimera signal planned, the TESLA Chimera protocol is designed for a constellation for signals. The specification is IS-AGT-101 [17].

The ability of TESLA to use MACs instead of digital signatures means that we should be able to lower the latency. Specifically, a TESLA Chimera epoch is defined to be half the length at 90 seconds or five CNAV2 messages. The last message in this epoch is devoted for authentication information. That arrangement still yields a 20% allotment, one message out of five. That authentication message is page 10, with the same warning that this is the page number on NTS-3 and will not necessarily hold for the GPS enterprise.

TESLA Chimera requires a TESLA chain. The specification is set up to allow a weeklong TESLA chain spread across up to 40 satellites. TESLA subchains are supported in generality (i.e., a receiver will be able to infer where

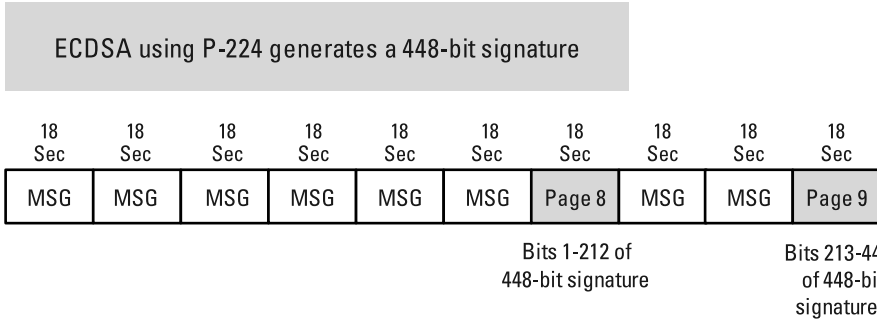

Figure 8.10 Baseline Chimera messages to be signed. Page 8 location is arbitrary.

a received key occurs in the overall chain by indexing). In particular, such indexing allows a receiver to infer when a new TESLA chain starts.

The bits to be protected by a MAC are shown in Figure 8.11 (taken from Figures 20-3 and 20-4 in [17]). All messages have their subframe 1 and subframe 2 bits protected. The complete subframe 3 bits are also protected except for the MACS in page 10. Note the CRCs are not protected, and the MACs are computed before any forward error correction.

One important difference with TESLA Chimera is that each message is protected by its own MAC. That means if a given message is received in error, only that message is not authenticated, except the loss of page 10 will invalidate the whole epoch. To accomplish this goal, the decision was made to use smaller MACs based on work from Stanford [8]. The size of the MAC is only 14 bits, which is very small from a standpoint of protecting data at rest, but was considered adequate for authenticating short-lived message in transit.

The MACs are computed by concatenating the bits as in Figure 8.11 and applying padding as needed.[3] The MAC is HMAC with SHA-512 (FIPS PUB 180-4). The lower 14 bits of the resulting HMAC are used for the MAC. The key used is the key for this TESLA epoch in the TESLA chain. The resulting MACs are inserted into subframe 3 of page 10 as shown in Figure 8.12.

As with any TESLA protocol, the TESLA chain needs to be authenticated. For TESLA Chimera, this is accomplished by sending a digital signature of one of the TESLA chain keys, which usually would be the TESLA root key but does not have to be. The resulting signature is inserted into two CNAV2 messages using pages 11 and 12 (with the same warning about NTS-3 page

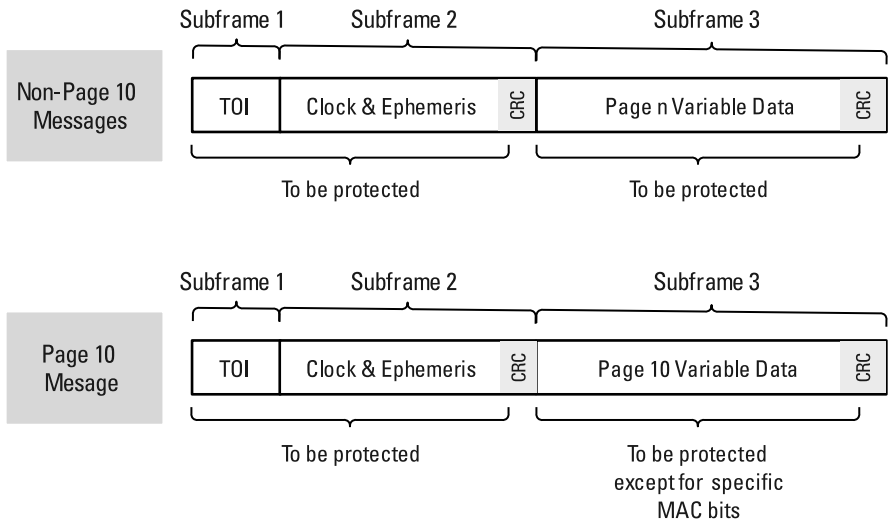

Figure 8.11 TESLA Chimera message bits to be protected by MACs.

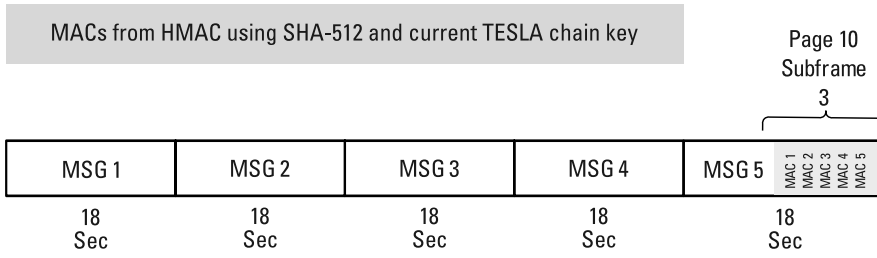

Figure 8.12 TESLA Chimera messages to be protected by MACs.

numbers). The insertion is the same as pages 8 and 9 for baseline Chimera. The resulting messages are added to the rest of the messaging. Strictly speaking, this increases the amount of messages devoted to authentication above 20%, but since the signature need not be sent that often, the effect should be minimal, especially when considering a TESLA chain across a whole constellation.

Finally, TESLA will improve performance. Latencies seemingly are the same, since a receiver needs to wait until the next TESLA epoch to receive the required TESLA chain key (i.e., that is how TESLA works). But note in a constellation, these page 10 messages could be staggered so that a receiver may be receiving a page 10 from some satellite sooner, and its key could potentially be used to derive the needed key. That observation should improve TTFAF. Finally, error performance should be similar, except note that loss of a single message does not prevent authentication of other messages due to the use of separate MACs.

8.5.4 An Experimental Design

NTS-3 is an experimental system, and thus Chimera and TESLA Chimera are experimental signals. That means the many of the design choices would be revisited if these designs were to become operational. Such specifics such as the epoch sizes, signature algorithms, and the MACs sizes would need to be updated.

8.6 Summary and the Future

NMA is perhaps the clearest example of where cryptography could and should be applied in satnav. The goals of integrity and authentication are essentially universal for data, and NMA are the methods to achieve that for navigation data. However, the nature of satnav—broadcast, low data rates, constrained devices—lead to very specific methods. Foremost is the use of the TESLA pro-

tocol, as we saw with OSNMA and Chimera; other GNSS systems are moving toward NMA methods.

TESLA exploits the advantages of shared-key methods via bit commitments, although asymmetric digital signatures are still needed. The reliance on public key cryptography is problematic due to emergence of quantum computing methods, which break RSA and ECDSA cryptography (see the appendix). As such, recent work examines what digital signatures could replace the use of ECDSA. Replace here means be able to fit into similar data message usage. The paper [18] explores the possible candidates; see the appendix for a longer discussion.

Finally, we stress again that these methods provide data authentication and not authentication of the signal timing. Methods to provide the latter protection are in the next chapter.

End Notes

1. This early work includes nonpublished results by the authors.
2. Consider protocols such as TLS or IPsec from Chapter 6. In these situations, each piece of data (packet, frame, etc.) is authenticated. But this authentication uses shared keys established by the protocols, and such session keys are not possible in broadcast satnav.
3. Cryptographic algorithms like hashes and signing are designed to work on arbitrary bit-length inputs by having any necessary padding built into the algorithm. However, most implementations of these algorithms assume that the inputs are a multiple of bytes instead of bits. As such, the specifications were written to add padding to make the inputs a multiple of 8 bits.

References

[1] *Vulnerability Assessment of the Transportation Infrastructure Relying on Global Positioning System*, Volpe Transportation Center and United States Department of Transportation, August 29, 2001, https://rosap.ntl.bts.gov/view/dot/8435.

[2] Scott, L., "Anti-Spoofing & Authenticated Signal Architectures for Civil Navigation Systems," in *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003),* September 12, 2003, pp. 1543–1552, http://www.ion.org/publications/abstract.cfm?jp=p& articleID=5339.

[3] Hein, G. W., et al., "Authenticating GNSS: Proofs against Spoofs, Part 1," *Inside GNSS,* July 2, 2007, https://insidegnss.com/authenticating-gnss-proofs-against-spoofs-part-1/.

[4] Hein, G. W., et al., "Authenticating GNSS Proofs against Spoofs, Part 2," *Inside GNSS,* September 1, 2007, https://insidegnss.com/authenticating-gnss-proofs-against-spoofs-part-2/.

[5] Wesson, K., M. Rothlisberger, and T. Humphreys, "Practical Cryptographic Civil GPS Signal Authentication," *Navigation,* Vol. 59, No. 3, September 2012, pp. 177–193, doi:10.1002/navi.14, https://onlinelibrary.wiley.com/doi/10.1002/navi.14.

[6] Fernandez-Hernandez, I., et al., "A Navigation Message Authentication Proposal for the Galileo Open Service: NMA Proposal for Galileo OS," *Navigation,* Vol. 63, No. 1, March 2016, pp. 85–102, doi:10.1002/navi.125, https://onlinelibrary.wiley.com/doi/10.1002/navi.125.

[7] Anderson, J. M., et al., "Chips-Message Robust Authentication (Chimera) for GPS Civilian Signals," in *Proceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017),* September 29, 2017, pp. 2388–2416, doi:10.33012/2017.15206, http://www.ion.org/publications/ abstract.cfm?jp=p&articleID=15206.

[8] Anderson, J., et al., "Authentication of Satellite-Based Augmentation Systems with Over-the-Air Rekeying Schemes," *Navigation,* Vol. 70, No. 3, 2023, doi:10.33012/navi.595.eprint, https://navi.ion.org/content/70/3/navi.595.full.pdf.

[9] Wullems, C., O. Pozzobon, and K. Kubik, "Signal Authentication and Integrity Schemes for Next Generation Global Navigation Satellite Systems," *European Navigation Conference (ENC-GNSS 2005),* 2005, https://eprints.qut.edu.au/38275/.

[10] European GNSS Agency, Galileo Open Service Navigation Message Authentication (OS-NMA)–Info Note., Publications Office of the European Union, 2021, doi:doi/10.2878/49446.

[11] European GNSS Agency, Galileo Open Service Navigation Message Authentication (OSNMA)–Signal-in-Space Interface Control Document (SIS ICD), Issue 1.0, December 2022, Publications Office of the European Union, 2022, doi: doi/10.2878/ 594840.

[12] European GNSS Agency, Galileo Open Service Navigation Message Authentication (OSNMA)–Receiver Guidelines, Issue 1.0, December 2022, Publications Office of the European Union, 2022, doi:doi/10.2878/256023.

[13] European GNSS Agency, Galileo Open Service Navigation Message Authentication (OSNMA)–Internet Data Distribution Interface Control Document (OSNMA IDD ICD), Issue 1.0, July 2023, Publications Office of the European Union, 2023, doi:doi/10.2878/325903.

[14] Fernandez Hernandez, I., et al., "Toward an Operational Navigation Message Authentication Service: Proposal and Justification of Additional OSNMA Protocol Features"in *2019 European Navigation Conference (ENC),* April 2019, pp. 1–6, doi:10.1109/EURONAV.2019.8714151, https://ieeexplore.ieee.org/abstract/document/8714151.

[15] Fernandez Hernandez, I., et al., "Galileo Authentication and High Accuracy: Getting to the Truth," *Inside GNSS*, February 13, 2023, https://insidegnss.com/galileo-authentication-and-high-accuracy-getting-to-the-truth/.

[16] Air Force Research Laboratory Space Vehicles Directorate, Satellite Navigation Technical Area, Interface Specification IS-AGT-100 Rev A: Chips Message Robust Authentication (Chimera) Enhancement for the L1C Signal: Space Segment/User Segment Interface, 2024.

[17] Air Force Research Laboratory Space Vehicles Directorate, Navigation Technology Satellite-3, Interface Specification IS-AGT-101: Timed Efficient Stream Loss-Tolerant Authentication (TESLA) Chips Message Robust Authentication (Chimera) Enhancement for the L1C Signal: Space Segment/User Segment Interface, 2024.

[18] Mendiola, M. A., et al., "Post-Quantum Authentication Schemes," in *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020),* September 25, 2020, pp. 3812–3825, doi:10.33012/2020.17747, http://www.ion.org/publications/abstract.cfm?jp=p&articleID= 17747.

9

Spreading Code Protection

The previous chapter showed the various methods used to protect the data layer in a satnav signal. This chapter has a similar focus on the lower layers of the signal, namely the carrier and modulation layers. Since in a satnav signal these layers manifest in the CDMA spreading code, we use spreading codes or, more generally, signal as the overall target for the protection. The overall goals are also authentication and integrity: a user wants to be sure they are tracking the legitimate signal and that it has not been altered, namely in terms of when it was transmitted. The actual cryptographic methods are different than for data, because the processing nature of the lower layers is different. Cryptography is used to create features in the signal, specifically changes to the spreading codes, which can be used to verify a true signal. Because one wishes for these features be exclusive to the transmitter, confidentiality is also a goal for the methods.

We first expand on the traits and goals of signal protection methods along with the constraints and performance measures. We present a brief history before listing the methods, concentrating first on the options for cryptography and then how the cryptography can be applied. Our main example is again an experimental signal on NTS-3, which uses markers inserted into the spreading code.

9.1 Protecting the Signal Lower Layers

Unlike NMA, which basically adapts standard data protection methods to navigation messages, protecting the satnav signal lower layers is not obvious. We desire essentially to authenticate the pseudoranges derived from the signal. The derivation comes via correlation from a signal that is very low power (buried in

the noise). Thus we need to put features that respect the pseudorange calculation while yielding authentication, similar to the way a digital signature adds a feature to data that still allows one to use the data.

9.1.1 Goals

A satnav signal in the lower layers has the following important traits:

- The purpose of the signal is to yield pseudorange via its timing. Thus the signal is very ephemeral: the timing changes continually, and it is usually not vital to know what the timing was far in the past, although note that some methods we will consider add latency that defines how far in the past.
- The signal needs to be known to the receiver so it can track it to obtain pseudorange. In other words, the receiver needs to be able to generate the replica spreading codes of the true signal. Any cryptographic features need to take this into account.
- Similarly, the signal is universally available. That means any features need to be usable by most likely everyone.

Given these traits, the security goals for the signal are foremost integrity and authentication. Integrity means that the signal can still be processed to yield the correct timing. Authentication means we can be sure that the signal came from the legitimate source. Using cryptographic features embedded in the signal to achieve these goals implies that we need confidentiality/exclusivity of the features.

There are implementation goals related to the security goals. For example, any scheme should be scalable to the large user base. That scalability includes distribution of any cryptographic material. There is also scaling with regard to the features themselves (i.e., changing the features as necessary to enhance the strength of the authentication).

9.1.2 Constraints

As with any security goals, their achievement should not have a large impact on performance. For example, if the features take away from the processing of the signal, for example, by reducing SNR, then that loss should be quantified and deemed acceptable. To expand on this notion, consider that:

- One impact would be in the correlation properties of the signal, that is, those properties assumed in a CDMA system (see Chapter 2). One

should consider both autocorrelation degradation and cross-correlation effects due to the added features.

- The features themselves need to be detected and processed. This need requires that sufficient power is available in these features to perform the authentication. That trait in turn means that power will likely be lost from the main signal to perform pseudorange calculation.
- We emphasize that for the features to be used to authenticate, they must first be discernible in the signal. If the features occur as only part of the signal, then that means more time, and thus latency, will be needed to extract the features.

Finally, consider implementation issues. Primary considerations are the space vehicles and user equipment, such as size, weight, (and) power, and cost (SWaP-C). There are concerns both with technology maturation and the timelines to implement any solutions. This latter point is even more severe when one considers that changes to the signal itself are likely very difficult to achieve in existing satellites; contrast that with some of the NMA methods where some solutions exist just in the ground segment. In the receiver, there are also considerations such as the resource utilization to do the cryptographic processing.

9.1.3 Measures

As with NMA, there are measures related to strength and latency. While there is the usual strength to be considered by the cryptographic methods utilized, strength is also related to the SNR needed to extract and process the features. That processing takes time to gather all the feature information, which is the main factor when considering latency.

There are the same measures of TTFAF and TBA as defined in Section 8.1. Note TBA can be considered for a signal from a single satellite or for time between full authenticated fixes. In addition, we introduce TTFA, which is the time it takes to gain the first authentication once the receiver is turned on. This time is measured after the signals are being tracked.

9.2 History

Using cryptography in the spreading codes is not as obvious for open signals as protecting data due to the traits mentioned in the previous section. As such, early literature in the use of cryptography is sparser than that for NMA.

Logan Scott's paper [1], which is seminal for NMA (Section 8.2), also suggests methods for spreading codes. Specifically, Level 2 deals with punctures

into the spreading code. The resulting scheme uses bit-commitment, although without using that term. These punctures are called *Spread Spectrum Security Codes (SSSC)***.** That paper also discusses Level 3, which uses private key methods. In parallel, the same ideas are put forth in a paper by Marcus Kuhn [2].

Since then, efforts to add cryptography to the spreading codes are focused on using private, shared keys. Naturally, these techniques are likewise not public. The recent breakthrough in public discussion came with the development of Chimera, detailed in Chapter 10 defined in IS-AGT100A [3] and IS-AGT-101 [4].

9.3 General Cryptographic Methods

Cryptography is used to generate features in the spreading codes of the satnav signal. The application of cryptography is very simple, namely using a cipher to generate ciphertext, which in turn generates the features. The variation in methods depends on where the key used by the cipher comes from and how it is managed. There are two methods:

- Using a shared secret between the transmitter and the receiver. This usage is a form of *transmission security (TRANSEC)*, analogous to how cryptographic methods used on data are a form of *information security*.
- Using secret information that only the transmitter knows. After transmission of the features, information is revealed to the receiver to process the features. This method is a form of bit commitment (Section 4.7).

Noticeably missing from these options is a direct use of asymmetric cryptography. The existence of such a method is an open question, which we elaborate on in the next chapter.

9.3.1 Cryptography and Spreading Codes

Consider Figure 9.1, which shows a generic outlook for using a cipher to generate the spreading code information. Any cipher could be used, either a block cipher or a stream cipher; for concreteness think AES. The encryption process uses plaintext that is known, say something that depends on time. The cipher mode is ECB (see Section 4.7), since there is almost always no concern about the ciphertext being repeated.

As shown in Figure 9.1, the ciphertext has two uses. It can either be used to generate the spreading codes in total; basically, the random nature of the ciphertext bits are the desired random-like spreading chips. The second use is to create features that are part of the spreading codes, for example, by puncturing

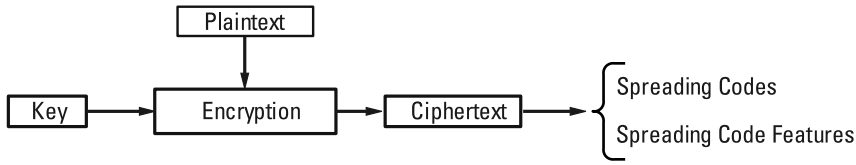

Figure 9.1 Using a cipher to protect spreading codes.

or marking certain portions of the spreading code. There are trade-offs with regard to performance, which we discuss in the next section.

Figure 9.1 shows the process for both the transmitter and the receiver. That is, a receiver does the same encryption operation to create replicas of the spreading code features. We stress that there is no decryption process.

The security goals follow from the nature of the cipher. As long as the key is kept private to only the intended parties, one can be assured of the exclusivity of the ciphertext, which will also yield integrity and authentication. The strength of this scheme depends on the size of the key. However, note that what is being protected is very ephemeral, which means the size could be smaller than what would be suggested for the use of protecting data.

The ciphertext gives exclusivity in that it cannot be generated without knowing the key. That means that the ciphertext should not be revealed before it is needed. Note confidentiality of the plaintext is not germane, because the plaintext is already known (i.e., it is specified in signal documentation). This trait of the scheme sets it apart from INFOSEC, where often ciphertext is available for anyone to see. We call this difference out explicitly.

> *The Need to Keep Ciphertext Exclusive*
> The ciphertext used to generate spreading code information needs to be kept secret before it is sent.

9.3.2 Shared Key Methods

Shared key or private access systems are not open, and some security mechanism is used to grant access. For instance, the user pays and gets access to the system, much like a subscription service. Access comes from being given the key used to generate the cryptographic features. Distributing the key requires some type of infrastructure to do so securely. Separate out-of-band methods could be used, using the protocols discussed in Chapter 7, because the keys should not be sent in the open. Nonnavigation examples of such schemes include satellite radio and satellite TV, which are two broadcast industries that use cryptographic mechanisms to control access to products. Both of these systems have a great

deal of data bandwidth, which is usually not the case for satnav systems. It is conceivable that a multitiered system could be used with multiple keys.[1]

Of course, a central issue to using shared keys is that the keys are shared by all users, since there is presumably just a single signal being transmitted. That trait puts a large trust burden on the users. In particular, the users' receivers must be able to protect the keys from being revealed. There are commercial solutions to protect such shared secrets. These systems can be difficult to get completely right.[2]

Related to this trust issue is the need to deal with inventory. That is, with some regularity, user equipment must be revalidated. The usual mechanism for this is key expiration, and getting a new key requires revalidation. But key expiration can be problematic since it affects the whole system, including all of the transmitters. Mechanisms like zero-trust (Section 6.5) may be relevant, which require that a person or device reestablishes its authorization to operate. This may be as simple as associating a device with an account and paying the bill. Usually, there is a feedback loop; once the bill is paid, the device is authorized, and some proof is given that it is the same device that was registered. Most services reserve the right to terminate your access if they deem you have not lived up to the agreement.

9.3.3 Bit Commitment Methods

Instead of sharing keys that must be kept private, the system could just release information after the signal has been transmitted. Since the transmitter has sent the information before it is revealed, this is a form of bit commitment (see Section 4.7). That release could be:

- The key used to generate the cryptographic features. This key would essentially only be used once, and the system would generate a new key each time before release.
- The specific cryptographic features. For example, the feature could be the whole spreading code or just smaller snippets depending on what was generated.

Releasing means using some separate, presumably much higher-speed communication channel. For our purposes in this chapter, we will assume whatever is needed of the communication channel and focus on the signal and the relationship of the features to the signal.

Using the out-of-band channel means that latencies can be an order of magnitude faster than say if the satnav signal itself was used. If the key is being

released, a larger data bandwidth means we do not have to reduce key sizes just to accommodate messages.

The information release must respect the timing in the signal. For instance, if a key is available on the communication channel at a given time, then that time is with respect to events in the signal (i.e., the signal must have already used the key). For instance, if the key is served on a website, then the website must be trusted to not release the key until the relevant time.

9.4 Applying Cryptography to Spreading Codes

The previous section showed how to generate the ciphertext needed to create the cryptographic features in a signal. This section describes those possible cryptographic features. We consider generating the complete spreading code sequence or smaller markers/punctures. We also describe how cryptography can be used to modify the chips themselves. Either shared-key or the bit commitment schemes just described can be used.

9.4.1 Ciphertext as Spreading Codes

Perhaps the most straightforward application of cryptography for spreading codes is to use the ciphertext as the whole spreading sequence. Consider Figure 9.2, where a series of plaintext blocks are encrypted into a series of ciphertext blocks. These blocks are combined into the spreading code. For example, they could just be concatenated to yield the spreading symbols.

This process begs the question: can such spreading codes be useful from a signal perspective? Recall CDMA (see Sections 2.2.1 and 6.4) as a technique to allow multiple users on the same channel. In this spread spectrum technique, the spreading codes are chosen to have particular properties, such as one code

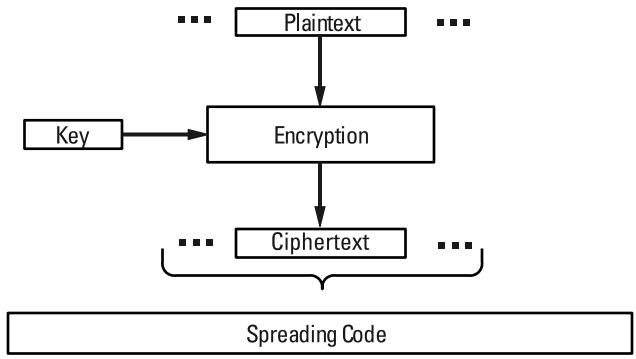

Figure 9.2 Generating complete spreading codes.

being orthogonal (uncorrelated) to another. The orthogonality is in the sense of a (discrete) dot product. Orthogonality means that different signals act basically as noise to the desired signal when acquiring and tracking.

Cryptography can guarantee this orthogonality provided the plaintext blocks are unique, say a counter tied to time and the specific transmitter. The reason is that a good cipher has the avalanche property (Chapter 3), which means changing even a single bit of the plaintext means the resulting ciphertext are essentially random to each other. That randomness is precisely what uncorrelated means. Strictly speaking, codes generated using cryptography are not exactly orthogonal, but the expected value of their correlation is zero.

The security goals are obtained through this method by being able to generate the sequence using the correct key and plaintext and using that to correlate correctly (i.e., track, the incoming signal). Of course, someone could have delayed the encrypted signal, but no one can generate the signal ahead of time without knowing the key.

9.4.2 Ciphertext as Markers

Suppose instead that the ciphertext is used to generate portions of the spreading code with the rest of the spreading code being the default for that signal. These portions are called *markers* or *punctures* because of their relationship with the overall spreading code. Examples include the SSSCs in Scott [1], the hidden markers in Kuhn [2], and the security punctures in Chimera [5, 6].

Consider Figure 9.3, where the usual cryptographic process is used to generate ciphertext. In this case, the ciphertext is split into to two uses. One

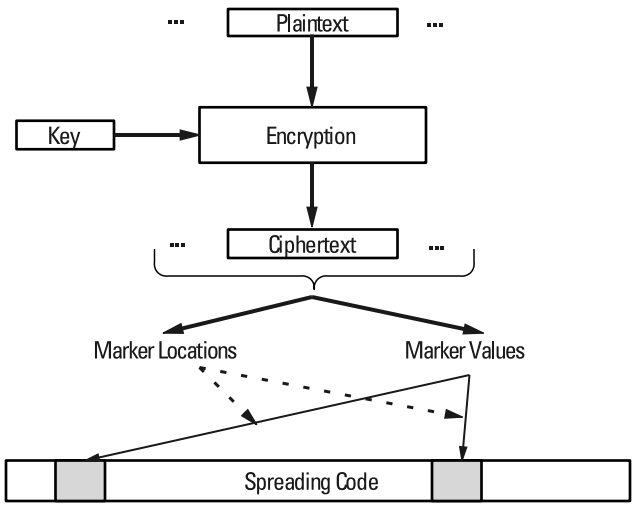

Figure 9.3 Generating spreading code markers.

set of ciphertext is used to specify the locations of the markers; the other set is used to specify the values of the markers. Only a portion of the spreading code is replaced by markers; that fraction is called the *duty factor* of this scheme.

Creating separate ciphertext can be done by adjusting the plaintext blocks. For example, a single bit denoting whether the ciphertext could be used for locations or values will suffice. See the next section for an example for NTS-3.

The security goals for using markers are different. Rather than just tracking the signal, we need a method that verifies the presence of the markers. Just correlating the full spreading code won't necessarily suffice depending on the SNR and duty factor. The marker replacement can be either literally replacing the spreading code chips with the ciphertext values, XOR-ing the ciphertext values into the spreading code chip values, or even just flipping the spreading code values based on the ciphertext. These options lead to different processing methods, for example, see [7]. As with any detection scheme, care will be needed to balance false alarm rates against detecting actual attacks on the signal (e.g., by falsifying the markers).

The need for having the marker values come from ciphertext is because they are then random and unpredictable. But why are the locations cryptographically determined? After all, if the locations are fixed, then the receiver knows where to collect the data for later processing. However, if the markers are placed in the same location, then a pulse jammer can be used to generate a denial-of-service attack by jamming only those locations. If the markers are placed at random locations, then the jammer must have continuous duty cycle *or* have enough gain to detect the puncture disrupting the underlying code in just a few chips.

Even assuming random locations, there are trade-offs with regard to the number of possible locations. A smaller number of potential locations simplifies the receiver design in terms of where they need to look for markers. But a larger number focuses a jammer to consider more locations.

In addition, one could decide that the marker positions are revealed at the time the marker values are revealed, or one could use the previous key and reveal the marker locations just in advance of them being used. That decision also impacts the jammer (i.e., do they have no knowledge of locations until after transmission or can they maybe try to process the signal ahead of time).

9.4.3 Ciphertext Modifying Spreading Chips

We can extend the marker idea to having the values and locations provide a more subtle change to the spreading code. We offer an example called *chip shape signaling*,[3] but there are many other possibilities. All that is needed is some feature in the signal that can be varied with the variation cryptographi-

cally controlled. As with the above spreading code and marker schemes, either shared-key or bit commitment schemes can be used.

We will embed a cryptographic message in the chips by using one of two chip-shapes. The user will process the chips, either in real time or after the fact depending on when the necessary information is available. Processing will be to see if they can detect the message. The message will be a pseudorandom sequence, bound to the channel and the time. The design is such that detection with a square chip will recover most of the energy in the chip, but decoding with the correct chip shape will increase the received signal power. We can also correlate with the wrong chip shape with a corresponding loss in signal power. We will call this the inverted signal. There are two concerns with the correlations. The first concern is with regard to power attacks. The second concern is doing the correlation across data bit boundaries.

The cryptographic message is generated using the same kind of cryptographic process in Figure 9.3. The shaped chips will correlate well with a rectangular pulse, allowing reception of the signal in a streaming fashion.

The team led by Sanjeev Gunawardena at AFIT [8, 9] have proposed manipulating signal generation details to act similarly to markers. We describe a scheme that we will call *chip state modulation (CSM)*, which is akin to watermarking.

Chips are generated with intentional ripples (see Figure 9.4). Either the + state or the – state are chosen based on cryptographically generated bits (indicated, respectively, as authentication features 1 and 2). The variations in the chip state are detectable once the key that generates them is known. This makes the chip-shape a steganographic feature.

To allow reception when the key is not known, the perturbation has to be small, as it will effectively rob power from the signal. Furthermore, to detect the variation, the data on the signal will need to be removed. If it takes N chips to detect a data bit reliably, and the difference between knowing the sense of a chip shape and not knowing is X%, then the number of chips that must be collected to achieve the same threshold for data bit detection $\frac{100}{X}N$. If we steal 10% of

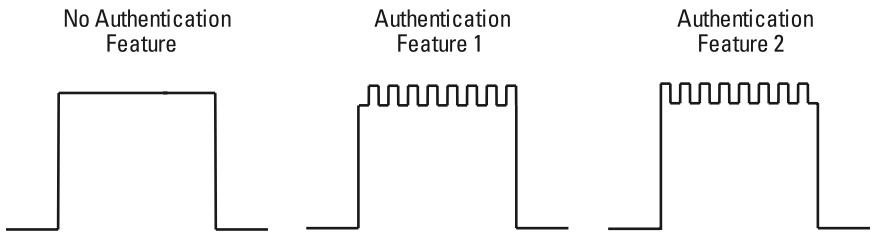

Figure 9.4 Two ripple chip states used to stegonagraphically hide the authentication data.

the power, that is, the duty factor is 10%, then we have to integrate over 10 bits to achieve the same performance we are getting when we integrate one bit. As an example, L_5 chips are at 10.23 Mcps, and the data rate is 100 symbols/second or 50 data bits per second. There are thus 102,300 chips in a symbol. At a duty factor of 10%, we need to integrate over 10 symbols, or 1,023,000 chips.

The advantages of using the chip shape idea are:

- User equipment: The user can decide when and how many chips are to be processed for authentication purposes.
- User equipment: The authentication marks are continuous. This trait makes the correlation process easier to perform and likely reduces the impact of timing mismatches.

The issues with this technique are:

- Spacecraft: It may be difficult to implement of the chip shape on the spacecraft, especially with Class-C amplifiers that are typically used. These are nonlinear and reasonably high in efficiency.
- Spacecraft: Direct digital synthesis would be required. As an example, the chip shapes in Figure 9.4 with 10 points per chip would likely require a 10x increase in circuit timing.
- User equipment: The details of the signal would require analog-to-digital converters with sufficient dynamic range to allow the detection of the feature.
- User equipment: The same internal timing issue might be a problem for user designs. We do note that UE designers often have more flexibility than spacecraft designers.

One other factor is that compared to punctures, there is less understanding of how this might perform both in design aspects and in operation. The reason is that there is decent literature on snapshot receivers [10, 11], with lots of implementation experience.

9.5 Chimera Markers

This section gives concrete examples of the use of markers. The focus is on the Chimera signal, which was originally designed for NTS-3. The markers are designed to work with either slow channel or fast channel Chimera. In addition, the same marker construction is used for either baseline Chimera or TESLA

Chimera. The only difference in the marker construction for any of these cases is where the cryptographic key comes from. Thus, we just consider the general marker creation procedure called Chimera. See Chapter 10 for relevant history of NTS-3 and Chimera. The interface specifications for Chimera are IS-AGT-100A [3] and IS-AGT-101 [4].

9.5.1 L1C Spreading Code Structure

Chimera is based on the GPS L1C signal. The authoritative source for GPS L1C is the interface specification IS-GPS-800 [12]. The spreading codes for the L1C signal comprise two components: the data component and the pilot component, each with their separate, near orthogonal spreading codes. Chimera focuses on the pilot component. While there is not data per se, the pilot component does have fixed overlay code that is length 1,800 symbols, which is 18 seconds at the 100 symbols per second rate. This overlay code is aligned with the 18-second messages on the data channel (see Section 8.5).

The pilot spreading code is a repeating length 10230 Weil-based code. The modulation, called TMBOC, is a time multiplexed scheme between two different BOC modulations. The main modulation is BOC(1,1), which alternates with a lower duty-cycle BOC(6,1). Specifically, every 33 spreading code chips consist of 29 BOC(1,1) chips and 4 BOC(6,1) chips; these are interwoven in a fixed pattern. The pertinent information is that the Chimera markers will only replace BOC(1,1) chips; the BOC(6,1) chips are left alone.

9.5.2 Chimera Marker Overview

The main idea is to replace some of the BOC(1,1) chips in the pilot signal with cryptographically generated locations and marker values. Figure 9.5 shows a nominal intended replacement; note the single square wave modulation, which is the chip modulation for BOC(1,1).

The marker procedure is hierarchical and is depicted in Figure 9.6. Each pilot symbol corresponds to a length-10230 spreading code which is a 10-ms duration. The spreading code is divided into 1-ms pieces called *sectors*. Each sector, which is 1,023 chips long, is divided into 31 *segments* of 33 chips in length. These 33 chips have the 29 BOC(1,1) chips and 4 BOC(6,1) in the prescribed pattern. The goal of the marker procedure is to replace sets of these 29 chips with the marker values.

The figure shows that some of the segments are of type S for slow channel or F for fast channel. Inserting markers for either channel is independent of the other. That is, each type uses separate calls to the cryptographic processing. In addition, the actual pattern of types varies by signal.

The duty factor can vary from using zero to seven of the 31 segments for each type. For example, using one segment means 29 chips are used for markers

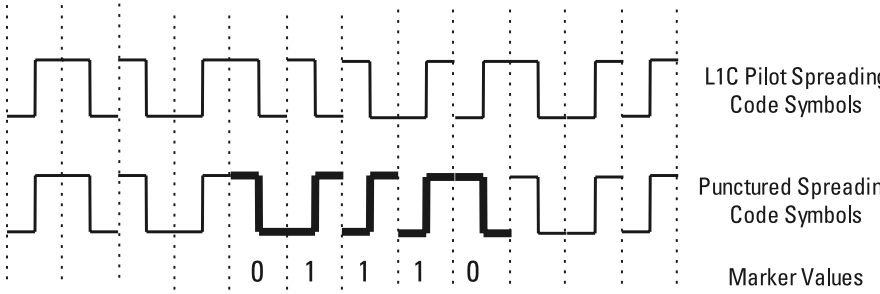

Figure 9.5 Marker insertion for Chimera (adapted from Figure 3-2 in [3]).

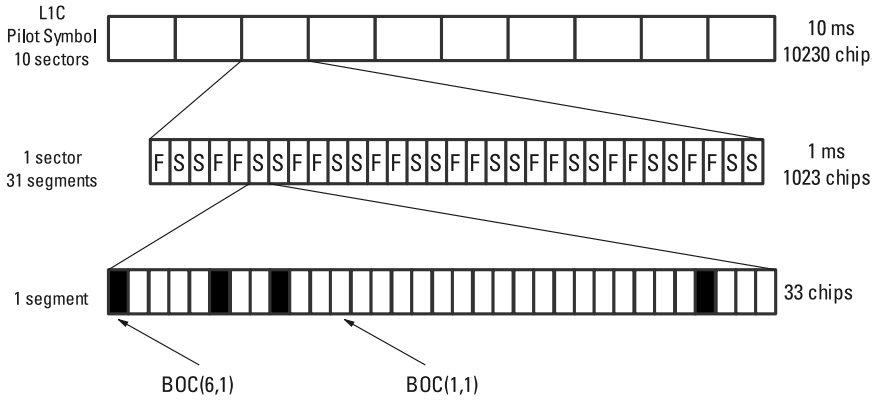

Figure 9.6 Hierarchy of inserting Chimera markers (adapted from Figure 20-8 in [3]).

in a 1,023-chip sector. That is about 2.83% of the pilot channel or 3.23% of just the BOC(1,1) chips. Using one marker per ms means 29,000 chips in a second, which gives a processing gain of around 44.6 dB, assuming that a receiver integrates for that long. The choice of duty factor shows the trade-off between allowing more energy for authentication is the price of taking away from the actual signal. We show some performance results below.

9.5.3 Cryptographic Processing

The cryptographic processing follows the scheme in Figure 9.3. AES-256 is used to generate two buffers of ciphertext. One buffer will be used to determine which segments are to be replaced with markers. The other buffer will be used to generate the 29 values for each marker segment. The buffers are created on a pilot symbol basis (i.e., for ten sectors).

The amount of ciphertext needed depends on the duty factor. For locations, there are either 16 or 15 possible segments depending on whether S or

F, respectively. That means 4 bits are needed to determine a choice randomly, so if d is the duty factor, the total number of bits needed for a pilot symbol for location is $40 \times d$ for each type. That number determines the number of 128-bit AES ciphertext blocks. A similar calculation applies for the marker values. Here we need 290 bits per pilot symbol (29 bits for each of 10 sectors), so the total number is $290 \times d$ for each type. Again, dividing by 128 determines the number of ciphertext blocks. We stress that the duty factors can be different for the slow and fast channels.

Generating these different ciphertext buffers requires that we have distinct plaintext. The plaintext block is shown in Figure 9.7 with these byte fields:

- SRC is the source (e.g., 0 for NTS-3);
- PRN is the PRN of the signal;
- Lnk is link (e.g., L1, L2, or L5);
- Z-count is 4 bytes and is defined in [13] and is set to the start of the current marker sector (it remains constant for the pilot symbol);
- Prd is the period is related to the Chimera period for fast channel Chimera;
- S/F denotes slow channel or fast channel;
- DF is the duty factor;
- FN is the function that determines if these are for marker locations or marker values;
- Ind is the index of the ciphertext block into the relevant ciphertext buffer;
- Cntr counts where this pilot symbol is in the Chimera epoch;
- Fill is "AF."

The use of the Z-count field means that the counter will not roll over very often; it counts in 1.5-second intervals, which is a bit more than two centuries.

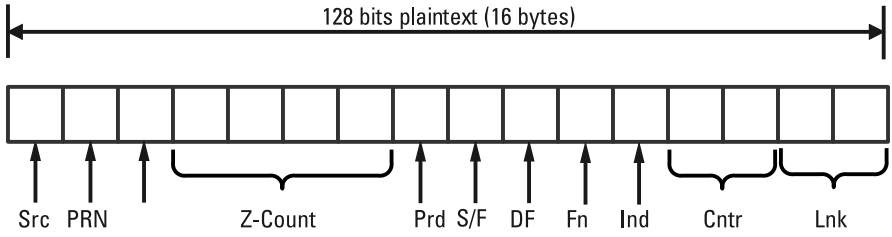

Figure 9.7 Plaintext for Chimera markers. (Adapted from Figure 20-11 in [3].)

Note that this means it will be constant for 150 pilot symbols. The import thing for the plaintext is that every plaintext block is then unique for each signal, function, and time. These choices of plaintext fields also indicate why there is no issue using ECB for the cipher mode.

9.5.4 Creating the Markers

The Chimera transmitter will create the markers for the period of the key, which changes for slow channel and fast channel (see Chapter 10). The process is sequential following these steps:

1. Start at some given pilot symbol.
2. Create location ciphertext buffer L and ciphertext value buffer V using the procedure in the previous section.
3 For each sector in the pilot symbol:
   - Sequentially take the next four bits from L to determine the next segment location. The specific algorithm is defined in [3]. The number of locations is determined by the duty factor. For concreteness, the indices of the chose segments are ordered from low to high.
   - For each chosen segment, take the next 29 bits from V and replace the BOC(1,1) chips with the values of these values.

The actual possible location depends on the PRN of the signal. Figure 9.6 shows a specific pattern for PRN 1 (modulo 32) (see Table 20-3 in [3]). Every pattern has 16 locations for slow channel and 15 locations for fast channel. While the 32 possible patterns are not perfectly random, the choices are enough to give sufficient variation between signals.

9.5.5 Performance

The performance scheme allows for pseudorange validation in a few seconds by collecting the marker information and performing a test (e.g., correlating against the predicted marker values). Since the pilot component is used, it is easy for a receiver to account for the pilot overlay code in the processing (i.e., there is no need to worry about unknown data).

The paper [14] has performance plots for the marker detection. For example, using a duty factor of 10%, capturing about 500 ms of the signal is sufficient to get near-perfect detection at a false alarm rate of 10^{-6}; 500 ms represents about 50 ms worth of marker material. These results assume a C/N0 of 25 dB-Hz.

9.6 Takeaways

Authenticating the timing in a satnav signal is paramount to assuring the position and timing solutions. One method to do that authentication is to have cryptographically generated features in the spreading code. These features could be processed and verified by a UE that can replicate the cryptographic generation. There are two different ways to do that cryptography: either using shared key ciphers or by using information only known to the transmitter but revealed after transmission. Additionally, other features could be modified in the signal that would be bound to time, such as the chip shape example.

There are fundamental trade-offs in these methods. Devoting a larger portion of the signal to the features (e.g., the markers) allows for easier verification. But that portion takes away from the actual satnav signal, which degrades navigation performance. The emerging system such as Chimera give a platform to measure these trade-offs.

End Notes

1. Multikey distribution systems can be implemented using key hierarchies. Examples include logical key hierarchy and one-way function trees [15].
2. The experience of DirectTV in its early days suggests that there is at least the possibility of substantial effort being directed into piracy. In at least one case, a provider has taken offensive cyber actions to enforce their obligations on users by destroying hacked equipment. See DirectTV and the Black Sunday Hack [16].
3. In the domain of electronic intelligence, the identification of radars by details in the waveform is known as specific emitter identification. This is analogous to using fingerprints for identification. Similar techniques have been proposed and demonstrated for signal quality measurements for GPS by a team led by Sanjeev Gunawardena at AFIT [8, 9].

References

[1] Scott, L., "Anti-Spoofing & Authenticated Signal Architectures for Civil Navigation Systems," Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003), September 12, 2003, pp. 1543–1552, http://www.ion.org/publications/abstract.cfm?jp=p& articleID=5339.

[2] Kuhn, M. G., "An Asymmetric Security Mechanism for Navigation Signals," in Information Hiding (J. Fridrich, ed.), Berlin: Springer, 2005, pp. 239–252, doi:10.1007/978-3-540-30114-1_17.

[3] Air Force Research Laboratory Space Vehicles Directorate, Satellite Navigation Technical Area, Interface Specification IS-AGT-100 Rev A: Chips Message Robust Authenication (Chimera) Enhancement for the L1C Signal: Space Segment/User Segment Interface, 2024.

[4] Air Force Research Laboratory Space Vehicles Directorate, Navigation Technology Satellite-3, Interface Specification IS-AGT-101: Timed Efficient Stream Loss-tolerant Authentication (TESLA) Chips Message Robust Authentication (Chimera) Enhancement for the L1C Signal: Space Segment/User Segment Interface, 2024.

[5] Anderson, R., *Security Engineering: A Guide to Building Dependable Distributed Systems,* Third Edition, Indianapolis, IN: John Wiley & Sons, 2020.

[6] AFRL, IS-AGT-100 CHIMERA Specification, April 17, 2019, https://www.gpsexpert.net/chimera-specification.

[7] O'Hanlon, B., et al., "SBAS Signal Authentication," Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022), Denver, Colorado, pp. 3369–3377.

[8] Echeverry, N. C., et al., "Signal Quality Monitoring through Analysis of Chip Shape Deformation for GNSS Signals," Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS + 2020), September 25, 2020, pp. 1454–1461, doi:10.33012/2020.17694, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=17694.

[9] Gunawardena, S., and F. van Graas, "High Fidelity Chip Shape Analysis of GNSS Signals Using a Wideband Software Receiver," Proceedings of the 25th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2012), September 21, 2012, pp. 874–883, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=10300.

[10] Petovello, M., "What Is Snapshot Positioning and What Advantages Does It Offer?" Inside GNSS, December 6, 2018, https://insidegnss.com/what-is-snapshot-positioning-and-what-advantages-does-it-offer/.

[11] van Diggelen, F., *A-GPS: Assisted GPS, GNSS, and SBAS,* Norwood, MA: Artech House, 2009, https://us.artechhouse.com/A-GPS-Assisted-GPS-GNSS-and-SBAS-P1729.aspx.

[12] GPS Program Office, NAVSTAR GPS Space Segment/Navigation User Segment ISGPS-800 Interface.

[13] GPS Program Office, NAVSTAR GPS Space Segment/Navigation User Segment Interfaces IS-GPS-200.

[14] Anderson, J. M., et al., "Chips-Message Robust Authentication (Chimera) for GPS Civilian Signals," Proceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017), September 29, 2017, pp. 2388–2416, doi: 10.33012/2017.15206, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=15206.

[15] Pande, A. S., and R. C. Thool, "Survey on Logical Key Hierarchy for Secure Group Communication," in 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), 2016, pp. 1131–1136, doi:10.1109/ICACDOT.2016.7877763.

[16] Poulsen, K., "DirecTV Attacks Hacked Smart Cards," The Register, January 25, 2001, https://www.theregister.com/2001/01/25/directv_attacks_hacked_smart_cards/.

10

Hybrid Protection of a Satnav Signal

The previous two chapters showed, respectively, how to use cryptography to protect navigation messages and navigation signals. Those discussions were separate, since each protection could be done independently of the other. However, that lack of coupling provides some vulnerabilities, which is a motivation for binding together data and signal protections. This chapter shows why and how such hybrid protections could be done.

We start by defining the issue and the general options for a solution along with the history of finding practical solutions. That leads to the first signal that gives such a solution: the Chimera signal, which was developed for broadcast on NTS-3. We present how all the features of Chimera described in the last two chapters yield the overall result. We finish discussing another emerging example from Galileo.

10.1 The Issue and Options

We first discuss the need to couple the data and signal protection together and then how to possibly achieve that. This problem has a lot of personal history with the authors.[1]

10.1.1 The Problem

Consider NMA and the process of transmitting authenticated messages. These messages are sent at a specific time, and indeed the message often has the exact time as part of the data. Thus, NMA binds the data loosely to time, but we say

loosely because the granularity is determined roughly by the data bandwidth. For example, at a rate of 100 symbols per second for L1C, each symbol lasts 10 ms, which ends up being 10,230 chips.

If a threat were to shift a message by a few chips, the data would be the same, but the relationship between the data and the carrier would be different. Tracking loops (see Section 2.2.2), track the CDMA and carrier and determine the data from an integrate and dump process, so the result of this shift is that there is less energy collected for the data. But less energy still may permit a receiver to demodulate the correct data and verify its authentication even in the shifted signal. Since each chip shift at 1 Mchip/sec chipping rate is about 300m in error, the effect on timing and positioning can be huge.

This type of attack was discussed by Psiaki and Humphreys [1]. The discussion is focused on the *security code estimation and replay (SCER)* attack. The idea is that an adversary can estimate a data bit value in real time and then broadcast a shift version at higher power with the same data bit. This paper is an example of the need to bind the timing in the signal (i.e., the spreading codes with the data).

10.1.2 The (Only?) Choices

We know of two ways to bind the data and signal. The first method is to use a shared secret for both TRANSEC and INFOSEC. The issue with that method is the need to share that secret with every user, and that may be problematic; this concern is why we did not mention that solution when describing NMA.

The other solution is to use bit-commitment for both the data and signal protections. The same information that is revealed after transmission could be used to authenticate the data, for example, like TESLA does, and the spreading codes. Of course, one of the prices paid for bit-commitment is latency.

10.1.3 A Holy Grail

In the course of our decades of research in using cryptography for navigation, the authors have given a lot of thought to the possibility of another method. We call this method the holy grail, because while it can be described in the abstract, we know of no way to achieve it.

The holy grail is to have a public key solution that can apply to the signal analogous to public key methods on the data. Specifically, we desire a method that does the following:

1. The transmitter creates a signal that anyone can track (e.g., some GNSS signal).

2. The transmitter embeds features that are created with a private key. These features obfuscate the signal in that it cannot be tracked as is.
3. The receiver receives a noisy version of this signal and cannot track it since it doesn't know features.
4. Instead, the receiver uses the corresponding public key function to produce a result that is a noisy version of the original signal. This noisy version can then be tracked.

The end result is a signal that only the transmitter can create, but anyone can receive. That property is not true for a shared key signal, since anyone knowing the signal can in principle transmit it.

We know of no way to do this, and one of us (JTG) thinks it is impossible and that the only solutions are the options just given, such as bit-commitment. The other of us (JJR) is still not convinced for two reasons. First, digital signatures do something similar to the above steps, just not at the signal level. Second, signal tracking is just correlation, and correlation is robust with regard to some operations, such as permutation. All that being said, we will leave this idea as an open problem for now.

10.2 Protecting Both the Signal and Data

The goals, constraints, and metrics for cryptographically binding the signal and data are mainly derived from protecting each individual part, which were defined in the previous two chapters.

10.2.1 Goals

Besides the traits in both the navigation signal and data, any binding of the two needs to be able to create features that are linked to both parts. That means unlike a layered approach where the layers are treated separately, in this case they must be coordinated. For example, information used to protect the data needs to also be related to protecting the spreading codes. Often this will be accomplished by having timing as an input to the feature creation.

The security goals are the combined security goals for both the data and signal as defined in the previous chapters. So, for example, we want authentication and integrity for both the data and signal. Fortunately, verification of these goals benefits from needing to satisfy the goals for data and signal. That is, if an adversary only affects say the signal, then even though the data may still verify, the receiver knows there is a problem because the signal fails. This trait is, of course, the main reason for doing the binding. Such joint verification is aided by using linked cryptographic keys.

Similar system goals also exist. The transmission will need to be coordinated with regard to the cryptographic feature generation, as, for example, with constellation-wide TESLA chains. Similar constellation-wide methods could apply for the spreading codes for a constellation. The receiver also has similar goals (i.e., the ability to capture and process the necessary information).

10.2.2 Constraints

The only main constraint manifested by the combined protections of both data and signal is that both the transmitter and receiver need to be able to coordinate the protections. In the transmitter, this means that the data-level processing needs to interact with the spreading code generation; that interaction may be different than usual satellite architectures. Similarly in the receiver, data processing, which occurs after tracking, would need to then help with verifying features in the spreading code. One example is looking in a saved signal buffer for markers, whose positions are derived from the data.

10.2.3 Measures

Measures for protection of each layer apply, and they induce overall metrics for the combined protections. For example, TTFAF overall must take into account TTFAF for the data and for the signal, and similarly for TBA. The effects on the normal navigation processing, such as tracking and data bandwidth, likewise are derived from the amalgamation of all the methods.

10.3 History

Except for some seminal thinking, it was not until the mid-2015s that efforts were made to create signals that had protections for both data and signal bound together. As with the previous two chapters, the seminal work is by Logan Scott in 2003 [2]. That paper discusses putting the spreading code markers, called SSSCs, into the spreading code such that the receiver can verify them after receiving information in the navigation message. That information is simultaneously protected with digital signatures.

In 2014, at the then Advanced GPS Technology branch of Air Force Research Laboratory/Space Vehicles Directorate (AFRL/RV), research was started to provide signal authentication. As is often the case, the task independently discovered the ideas in the paper by Scott.[2] More recently, Galileo is defining the Signal Authentication Service (SAS), formerly known as Assisted Commercial Authentication Service (ACAS), which also uses bit commitment and is discussed later in this chapter.

10.4 General Methods

Before describing the specific methods, we provide some general points about creating a hybrid method that protects both data and signal.

10.4.1 Hybrid Approaches Using Bit Commitment

Our focus is on bit-commitment schemes that bind the data and signal methods together. Consider Figure 10.1, which summarizes the general parts for the transmitter and receiver.

The transmitter performs these steps in Figure 10.1:

1. The transmitter obtains the necessary key material and parameters for the method. Key material could be the private key portion of an asymmetric cryptography scheme or a symmetric key that only it possesses at the moment, like a TESLA chain start key, or possibly both.
2. The transmitter uses the key to create authentication features in the data messages (e.g., digital signatures or the TESLA MACs). This procedure will necessitate bundling messages together for some well-defined scope.
3. Simultaneously, the transmitter creates spreading code authentication features. These features could be markers or the whole spreading code. This procedure is bound to the data procedure in time.
4. The transmitter broadcasts the overall signal and data messages.

The receiver steps in Figure 10.1 are:

1. The receiver must possess the right key material to authenticate the features. For data authentication, that means the public key portion of the asymmetric scheme. For other methods, such as TESLA or the

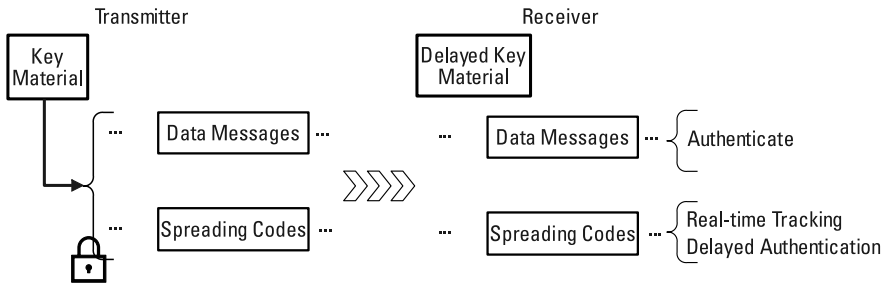

Figure 10.1 A general hybrid bit commitment scheme.

spreading code features, the receiver must wait to receive the material from a delayed source, either in-band or out-of-band.

2. The receiver gathers the requisite amount of data and data authentication features and verifies the data. This process could be just verifying a digital signature, or verifying the TESLA MACs after the TESLA chain has been authenticated separately.
3. A similar buffering must be done to authenticate the spreading code. For example, in the case of markers, the receiver will be tracking the underlying unmarked portions of the spreading code and then verify the marker values and positions once the requisite key material is created.
4. After these two steps, the receiver will be able to make a decision of authentication of the combined signal.

Figure 10.1 should not be construed to mean that the data is modulated on the spreading codes that are authenticated. All that is needed is that the spreading signal components are linked together. For example, the Chimera signal defined in the next section does this with the L1C signal's data and pilot components. The Galileo SAS signal uses two separate Galileo signals. All that is required is that the various signals are bound together in time.

As with any bit commitment method, the receiver must know time well enough to prevent any type of replay attack. That is, an adversary should not be able to themselves capture the delayed key material and use that to create a false signal that could be understood by a receiver with insufficient time. Fortunately, these timing requirements are on the order of tens of seconds or more and not onerous, especially considering that many devices are connected to a network with fairly good timing. Of course, any delay affects TTFAF and TBA.

Finally, we mention that the general setup in Figure 10.1 does not show the possible interaction between transmitters in the same constellation. For example, there can be information shared across the cryptographic chains among the space vehicle links, so that the appearance of features on one SV link will allow authentication of features on other SV links [3]. We referred to this possible cross authentication as *multichannel*. Such linking could also use a subconstellations architecture instead of the whole constellation.

10.5 NTS-3

Over nearly the past decade, AFRL has been sponsoring the development of *Navigation Technology Satellite – 3 (NTS-3)*, an experimental satellite that will explore new methods in satnav [4, 5]. The original NTS-1 and NTS-2 were launched in the 1970s to prove the concept of GPS. The U.S. Department of

the Air Force designated NTS-3 as one of its initial Vanguard programs. NTS-3 is currently planned to be launched in the near future, as of this writing.

One of the many experiments planned for NTS-3 is the initial broadcast of the Chimera modification to the L1C signal. Chimera is a hybrid system that cryptographically binds data and spreading code protections, which were discussed in Sections 8.5 and 9.5, respectively. See also [6, 7]. This section describes the combined signal modification.

Chimera comes in two separate methods: Baseline Chimera and TESLA Chimera. Baseline Chimera was designed first and uses digital signatures as the data authentication procedure. TESLA Chimera was designed later and uses TESLA for data authentication. The two interface specifications for these methods are publicly available: Baseline Chimera is defined in IS-AGT-100A [8] and TESLA Chimera/TESLA is defined in IS-AGT-101 [9]. Each method is described separately.

10.5.1 Baseline Chimera Overview

An overview of baseline Chimera is shown in Figure 10.2. The top portion shows the data channel on the L1C signal in terms of the data messages. The scope of data authentication is the *Chimera epoch* and is 3 minutes, which is 10 L1C CNAV2 messages. Two of the messages are devoted to transmitting the digital signature, as was discussed in Section 8.5.

There are two separate sets of markers for baseline Chimera, denoted as *slow channel Chimera* and *fast channel Chimera*. Figure 10.2 shows the two separate processes to generate these markers. The slow channel markers are

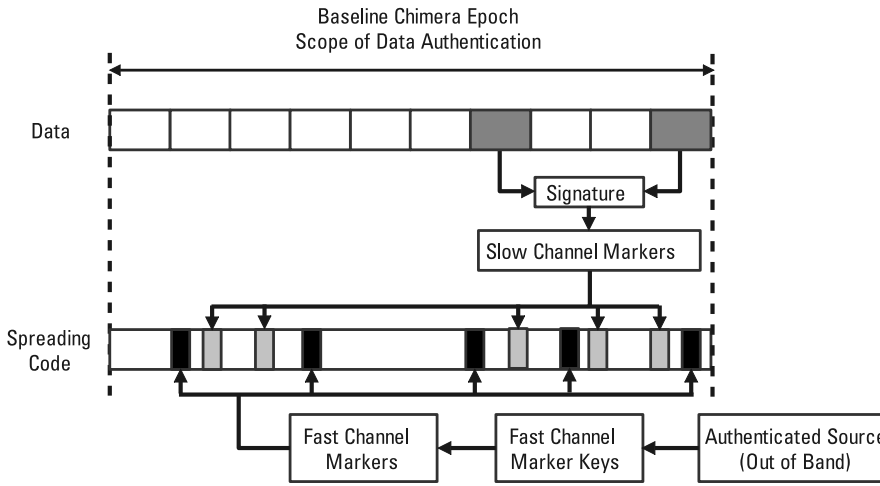

Figure 10.2 Baseline Chimera binding of data and signal (adapted from Figure 3.1 in [8]).

generated by deriving a key from the digital signature. That derivation implies that the slow channel also have the same scope of the Chimera epoch. The fast channel markers are generated using a separate key, which is obtained out-of-band. Their scope is defined by the period of that key, which is either 1.5 or 6 seconds. The marker generation includes generating random locations and values, as was discussed in Section 9.5.

10.5.1.1 Slow Channel

The philosophy of the slow channel Chimera is to have a signal with all authentication derived from the signal in space. This goal has the following implications for the data:

- We need to send the digital signatures as close as possible to the data messages that are being signed.
- We need to limit how much of the data messages, specifically the subframe 3 potions of the CNAV2 messages, are devoted to authentication. For example, 20% of the available data bits seems reasonable; available means the payload bits in the L1C subframe 3 pages.
- The digital signature has 112 bits of strength, which means a signature of size 448. That size signature fits into two CNAV2 messages.
- Thus the Chimera epoch for signing is 10 messages, or 180 seconds. See Figure 10.3, which shows the L1C data messages mapping to the Chimera epoch.

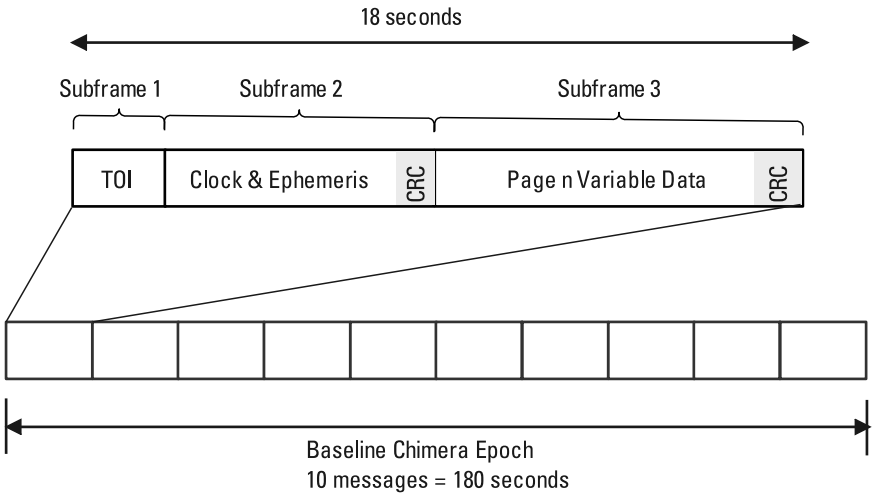

Figure 10.3 Baseline Chimera data (adapted from Figure 20.1 in [8]).

The digital signature is transmitted in two CNAV2 messages. Because the slow channel markers are generated using a key derived from the digital signature, at least one of the messages must be at the end of the epoch to ensure that the bit commitment aspects of the protocol are maintained. Figure 10.4 shows a nominal picture, with the location of page 8 and page 9 messages devoted to the digital signature (see Section 8.5).

The slow channel marker generation proceeds from first obtaining the slow channel marker key. This key is defined by taking the SHA-512 hash of the 448-bit digital signature and then truncating to 256 bits. This key is unique for each digital signature.

The specific marker generation was given in Section 9.5; see Figure 10.5. In particular, the slow channel markers can only be put in the S segments. The main point to stress is that the slow channel marker key persists for the whole Chimera epoch, namely 3 minutes. That means the same key is used for 18,000 pilot symbols.

10.5.1.2 Fast Channel

One detriment to slow channel Chimera is the long latency. Slow channel markers cannot be verified until the digital signature is received and verified. That requirement means receiving a complete Chimera epoch error-free, which could yield a latency of almost two Chimera epochs, or 6 minutes. If an error occurs, the signature will not verify and no slow channel marker key can be obtained.

That latency cannot be helped with regard to the data authentication, but for the spreading code markers, the latency can be decreased if the generating key can be changed more quickly. Reducing the period of the marker generation is the goal of fast channel Chimera. Fast channel Chimera is independent of the data channel, and it is useful if a receiver is getting any data it needs, like almanac and ephemera's, from a separate authenticated source.

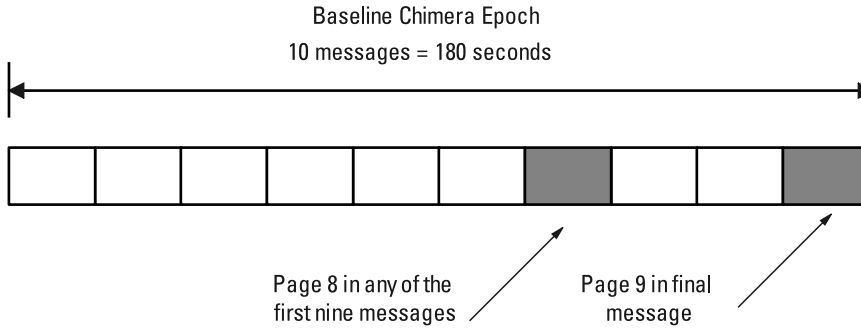

Figure 10.4 Baseline Chimera data messages (adapted from Figure 20.2 in [8]).

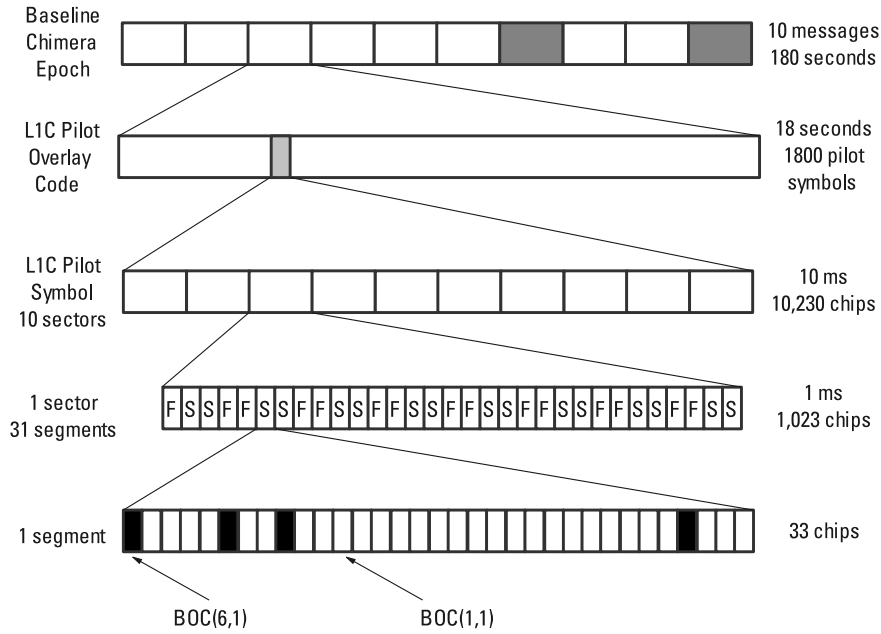

Figure 10.5 Another look at inserting Chimera markers (adapted from Figure 20-8 in [8]).

Fast channel Chimera has a long-lasting fast channel marker key generating key. This is the key that will be used to generate fast channel marker keys. This generating key is possessed only by trusted elements of the system (i.e., the satellites and trusted servers).

The fast channel marker key is derived by hashing the 32-bit GPS time with the generating key using SHA-512 and truncating the result to 256 bits. Note this function does not have a lot of entropy used to generating the keys, but if the generating key is changed at some longer time (say monthly), the entropy should be sufficient to generate the random markers. The duration of these keys has two possibilities: 1.5 seconds and 6 seconds. These *fast channel periods* are shown in Figure 10.6.

The marker generation is the same as in Figure 10.5, except that the F segments are used. Depending on the fast channel periods, 1,500 or 6,000 pilot symbols use the same marker key.

A receiver obtains the fast channel marker key from some out-of-band means. That requirement implies that a receiver must wait the period to receive the key, assuming the out-of-band source transmission is fast. The out-of-band communication channel is assumed to be authenticated, which is out of scope of the Chimera specification.

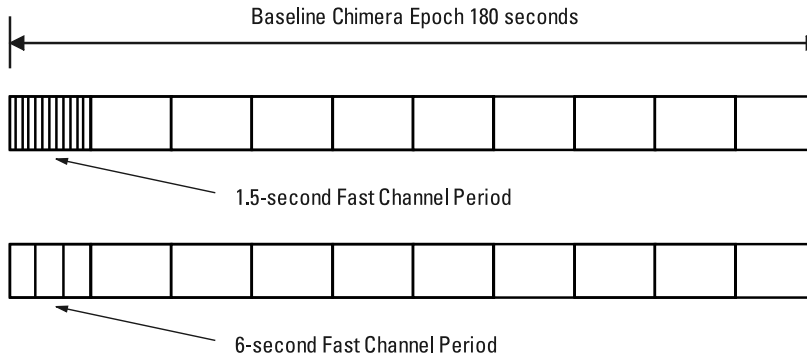

Figure 10.6 Baseline Chimera fast channel periods. (Adapted from Figure 20-9 in [8].)

10.5.2 TESLA Chimera Overview

As we saw in Chapter 8, TESLA is used for NMA because it is more data bandwidth efficient. After the development of baseline Chimera, the AFRL NTS-3 team began the development of TESLA Chimera [8]. TESLA Chimera replaces the slow channel procedures as indicated in Figure 10.7. Instead of digital signatures, information from the TESLA chain is used to generate the slow channel key. Digital signatures are still transmitted as described below. The important thing to note is that the fast channel portions of the TESLA Chimera signal are the same as baseline Chimera, in the sense either method uses fast channel keys for marker generation that are independent of the show channel methods.

The specific TESLA data procedure is described in Section 8.5, including the TESLA chain derivation and the MAC creations. Here are some highlights:

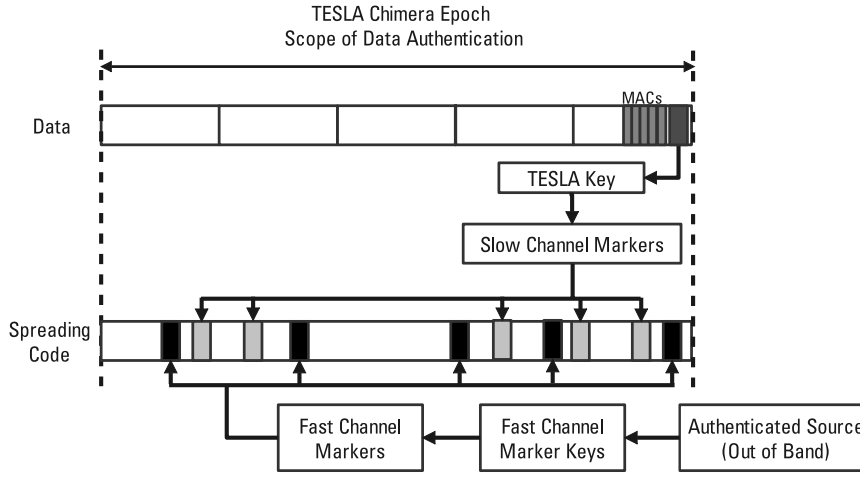

Figure 10.7 TESLA Chimera binding of data and signal. (Adapted from Figure 3.1 in [9].)

- There is one TESLA message in every five messages, so the *TESLA Chimera epoch* is 90 seconds. See Figure 10.8, which shows the L1C data messages mapping to the Chimera epoch.
- That still gives a 20% data usage, ignoring the occasional digital signature messages.
- The TESLA message in subframe 3 page 10 contains the MACs and the key used for the previous epoch.
- Each message in the TESLA Chimera epoch is authenticated using a 14-bit MAC.

The digital signature is transmitted in two CNAV2 messages similar to baseline Chimera, namely in subframe 3 page 11 and page 12. The signature

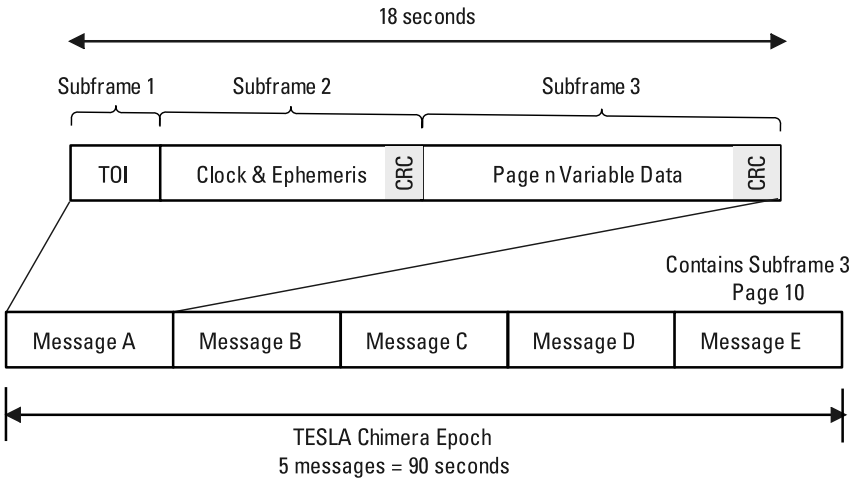

Figure 10.8 TESLA Chimera data. (Adapted from Figure 20.1 in [9].)

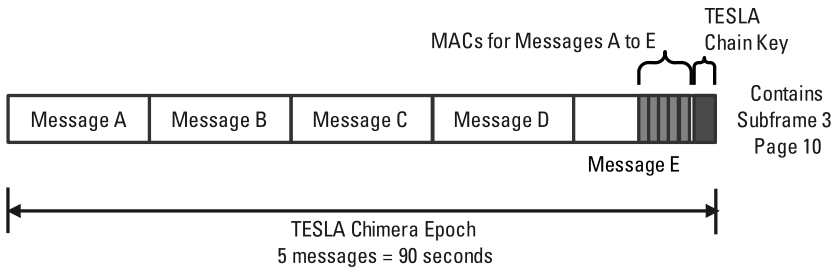

Figure 10.9 TESLA Chimera data messages. (Adapted from Figure 20.2 in [9].)

authenticates the TESLA chain, nominally by signing the TESLA chain root key, although the specification allows for any key to be signed. While the TESLA chain needs to be authenticated to have complete confidence in the MACs, the signature will verify the chain for its duration. A typical duration could be a week. Also note that the TESLA chain will likely be split across the constellations in practice. That trait means that verification of the chain will apply for every signal.

10.5.3 Processing

The nature of the Chimera signal requires some considerations for processing. In the transmitter, the baseline slow channel markers cannot be inserted into the signal until all the messages in the respective epoch (i.e., 3 minutes) are available and processed. After the messages are processed, then the transmitter can create the L1C signal and insert slow channel markers, as appropriate, using the key derived from the digital signature. Fast channel markers are also inserted at this time, but those do not add any appreciable latency because the fast channel periods are shorter.

TESLA Chimera is more efficient in that the marker key is derived from the TESLA chain, which the transmitter knows irrespective of the data. That availability of the marker key means that the transmitter can generate the TESLA Chimera messages one at a time and insert the markers as appropriate. This reduction in latency in the transmitter is another attraction for using TESLA Chimera.

For either Chimera, the receiver must buffer a sufficient portion of the received signal and wait until the respective epochs are done to receive the information for processing the slow channel markers. The fast channel markers can be verified more quickly assuming that the receive has access to the out-of-band fast channel marker keys. The authentication of the data can be done also once the epoch worth of data is accumulated.

10.5.4 Performance

The dependencies on the Chimera epochs means we can easily state some of the metrics:

- For TTFAF, a receiver needs to wait until receiving a complete epoch for the data authentication. Worse case is twice the epoch length, which is 6 minutes for baseline Chimera and 3 minutes for TESLA Chimera.
- Similarly, TBA is governed by the length of the epochs, or respectively, 3 minutes and 90 seconds.

- The error rates for doing the data authentication is based on the error rates for the L1C signal. See, for example, [10] and specifically [11] for Chimera.

Authenticating the signal has the same latency, but we also need to consider metrics for processing the markers. These metrics were discussed in Section 9.5. A receiver would likely process both slow and fast channel markers coherently.

Of course, the actual duty factors could differ between the channels. Appealing to [11], we note that the correlation loss due to the markers from Chimera is relatively small. For example, there is about a 0.9-dB loss for a 10% duty factor and only a 0.4-dB loss for 5% duty factor.

This brief discussion does not consider the processing complexity, power, technology required, and especially requirements on user clocks and their related complexity. Discussion on these topics will, hopefully, arise as the NTS-3 experiments are made public.

10.5.5 Multichannel Chimera

The main performance concern with a bit commitment such as Chimera is latency. For the slow channel chimera, this is substantial at 3 minutes. We look more closely at ways to mitigate this latency. The approach includes variations that are *not* consistent with IS-AGT-101 [9].

We first make some distinctions; an attestation of authenticity in Chimera, which we will call a *confirmation*, has two forms: *validation*, which occurs when a TESLA message is verified, and *authentication*, which occurs when a digital signature is verified.

Those definitions lead to these critical parameters:

- Time-Between-Validations (TBV);
- Time-Between-Authentication (TBA);
- Time-Between-Four-(Simultaneous)-Validations (TB4V);
- Time-Between-Four-(Simultaneous)-Authentications (TB4A).

The value four is chosen because the naive way to solve the PNT equations require four simultaneous measurements (Section 2.1). This idea gives rise to the interaction between the confirmation of the data and the confirmation of the pseudorange.

The main parameters influencing these statistics are the frequency of the validation and authentication messages. An additional variable is the *message phase*, that is, any offset (stagger) between SVs messages.

If all of the SVs are using the same message schedule, then (TBV, TB4V) and (TBA, TB4A) will be almost the same, respectively. Any difference will be any occasions where only three or fewer SVs are tracked without errors. If the message schedule is *phased*, then the time between confirmations can be significantly shorter.

This discussion is tied to the L1C TESLA Chimera signal. Of special note is that L1C messages have varying purposes that imply requirements on how often they should be broadcast. We use the term *promised* to indicate messages that have pledged frequencies of occurrence in the interface specification [12].

Figure 10.10 represents a TESLA 90-second epoch with a Digital Signal Epoch of 12 minutes. Two TESLA epochs are grouped together for a total of 3 minutes. The messages are denoted as P for promised and X for anything. The 3-minute groupings have two basic formats. The A grouping just contains two TESLA epochs, with T denoting the TESLA message. The B grouping contains a digital signature, denoted as D1 and D2. The broadcast pattern is A—A—A—B for a complete 12-minute cycle.

If the SV message timing is uniform (i.e., zero phase) then the $TBV \approx TB4V \approx 90$ seconds and $TBA \approx TB4A \approx 12$ minutes. If the SV message timing is phased, the performance changes markedly; see Figure 10.12.

The most natural thing is to shift the message phase into groups and either shift each group in phase together or shift the phase within each group. For instance, if you group SVs by orbital plane, there are six groups with four

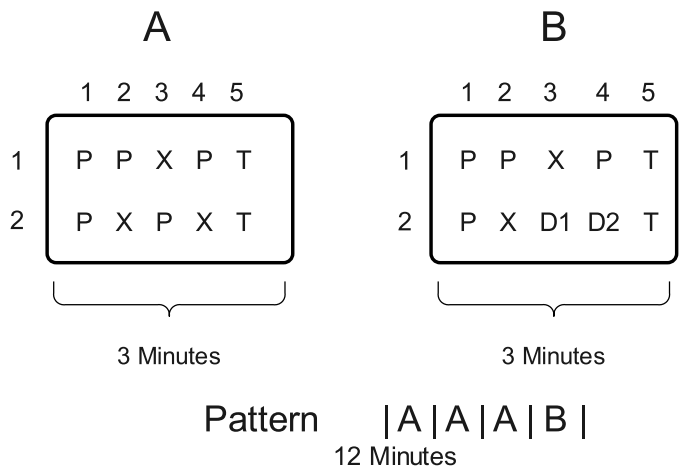

Figure 10.10 Example of a basic L1C message schedule.

SVs in a plane. One then could shift the messages for SVs in the same plane by 25% of the messages.

One could also shift the phase of all SVs in a plane and the shift would be one-sixth, and this is adjusted to be an even divisor of the message structure.

Recall that GPS is nearly a Walker constellation, with the SVs not evenly spaced in the orbital plane 2.2.1. Parameters for consideration are the number of groups for the message phase and whether the phasing is intragroup or by group**.** Other criteria can be used.[3]

The above discussion assumes that the TESLA chains on each SVs are independent. But the TESLA chains could be interwoven or coupled. This coupling has the effect of increasing the number of simultaneous confirmations. Figure 10.11 shows the between noncoupled and coupled signals for *M* SVs. The noncoupling situation uses the same TESLA chain for all SVs, while the coupling situation spreads out a single TESLA chain. The coupled situation is supported by [9].

Since using the same key is generally the definition of a cryptonet, we can think of the situation of each SV using its own key chain as each being on its

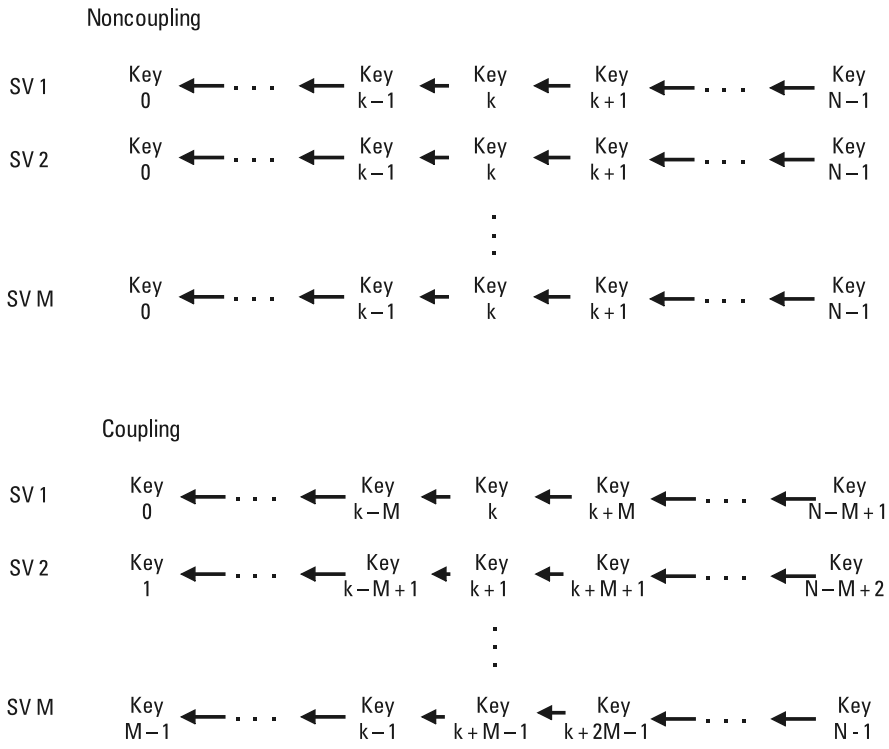

Figure 10.11 Noncoupling and coupling between TESLA streams.

own cryptonet. When two SVs use the same key chain, then they are on the same cryptonet. The limiting case is when the constellation is on one cryptonet. The effect of combining crypto-net is also known as *multichannel*.

If the messages phase is staggered, then the wait (TBV/TBA) can be *significantly* reduced (see Figure 10.12). The penalty is that the time to simultaneous validation/authentication (e.g., TB4V/TB4A) becomes *much* longer. The use of multichannel allows cross-validation of the pseudorange and recovers TB4V for the pseudorange but not the data. However, since the data is slowly changing, this is a significant recovery of performance [13].

There are practical matters to consider in the management of the cryptonets. One fact is that having the whole system on one cryptonet means that the disruption in the chain that occurs when a chain is exhausted and a new one

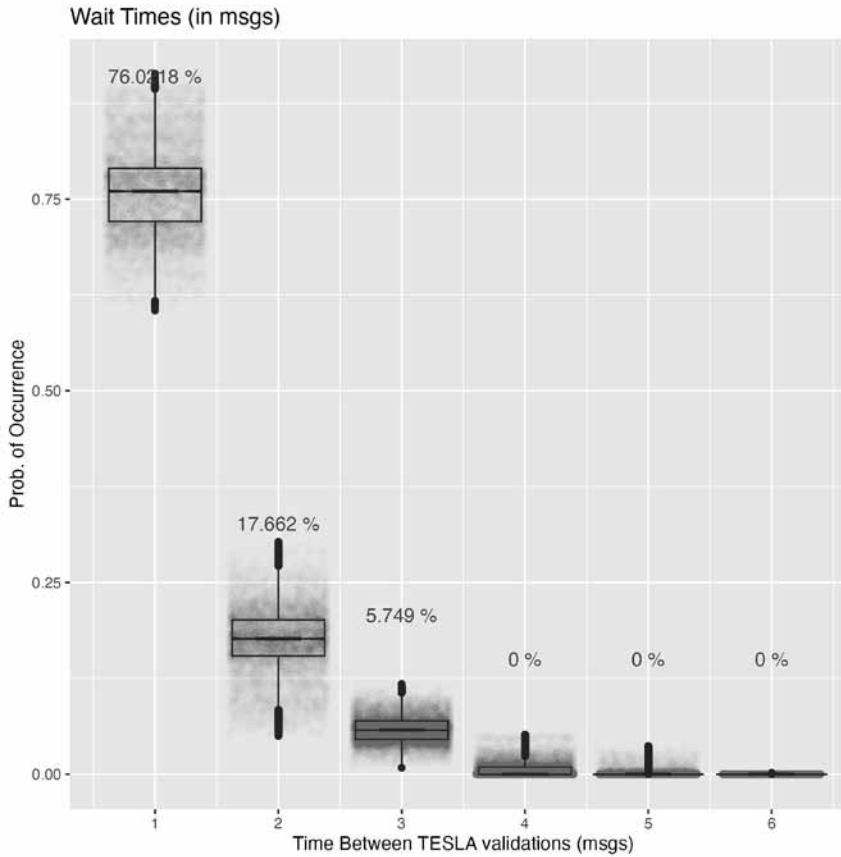

Figure 10.12 The effect of staggering the phase messages. Statistics based on one orbit (12 hrs) for 40,000 UE placed uniformly on the earth, with good viewing parameters (low C/N_0, all in view, etc.) Based on data in [13].

is started (i.e., cryptonet renewal) will be global. This situation could also occur in an unplanned event, such as if the private keys were somehow compromised. If the chains were decoupled, then possibly only some need to be immediately updated.

Besides the effects of cryptonet renewal, there are issues related to constellation management, initialization of new SVs, the retirement of SVs, and changes in PRN assignments. There are contingency effects, such as the temporary removal of an SV due to an anomaly. Additionally, constellation expansion and ease of operations should be considered.

How the constellation is configured must be conveyed to the user, and it is highly desirable that this be sufficiently compact that a new message is not required, so it needs to be tucked in either the TESLA message or the digital signature.

10.6 Galileo Signal Authentication Service

At the time of this writing, Galileo is creating the Signal Authentication Service (SAS), formerly known as the Assisted Commercial Authentication Service (ACAS), to provide an open architecture for signal authentication (see [14–16]). SAS uses a delayed release method to authenticate spreading codes on the encrypted E6-C signal. The delayed information comes from the OSNMA keys on E1; see Section 8.4.

The general scheme is shown in Figure 10.13. The transmitter side consists of three objects: the encrypted E6-C codes, the OSNMA messages and keys, and the E6-C spreading codes reencrypted again to produce the reencrypted codes (RECS). The receiver obtains all three of these objects to ultimately authenticate both data and the pseudorange timings. The RECS can be made available early, say up to a week ahead of time, since they are encrypted.

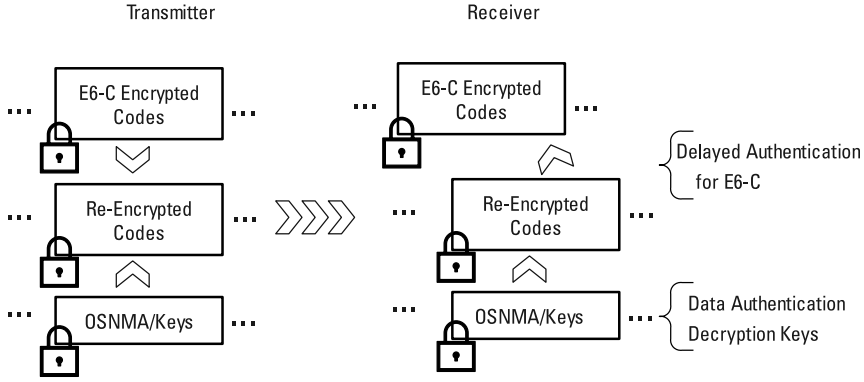

Figure 10.13 Overview of the SAS system.

The specific actions for the transmitter side of the system in Figure 10.13 are:

1. Create the E6-C spreading codes before broadcast to create the RECS;
2. The RECS are created by encrypting the E6-C encrypted spreading codes using OSNMA keys that will be revealed after the E6-C signal is transmitted;
3. Distribute the RECS through some out-of-band process to the receivers;
4. Broadcast E6-C and E1 (OSNMA) at their required time.

On the receiver end, the specific actions in Figure 10.13 are:

1. Obtain the RECS for its planned operation time;
2. Receive and buffer the portion of the E6-C signal that corresponds to the RECS;
3. Receive and process the OSNMA signal and obtain the keys to decrypt the RECS;
4. Correlate the decrypted RECS against the saved portion of the encrypted E6-C signal to verify the timing in the received signal.

We stress that the encrypted E6-C signals are never created in the receiver; that is, these spreading codes are still only available in real time to the holders of the private key material.

The security goals are obtained using all the objects in SAS. The OSNMA signal authenticates data and the keys used for RECS. Correlating the decrypted RECS against the saved E6-C signal authenticates timing. Note that this assumes that the Galileo components are synchronized in time, as SAS also provides authenticated E1-E6 group delays together with the RECS.

The paper in [14] introduces the SAS concept that is being implemented. Besides presenting a more detailed vision for the SAS system, the paper proposes computational solutions and some initial performance results based on simulation. The paper in [15] provides a good discussion of the underlying assumptions for SAS in order to look at performance under specific spoofing scenarios.

The main difference between SAS and Chimera is that SAS lets the receiver correlate against an encrypted signal by revealing portions of the encrypted spreading codes after the fact through the RECS. Chimera lets the receiver generate the encrypted spreading code features, though only at a later time. Both methods will allow for the requisite signal authentication. Note also that SAS is applied to signals that already exist.

10.7 Takeaways

This chapter explored combining two different cryptographic methods together to provide full authentication of a satnav signal. Specifically, the data authentication methods in the NMA described in Chapter 8 are bound with the spreading code methods explored in Chapter 9. In some way, these can be a layered approach, in that methods for each layer can be independent. For example, the specific cryptography used to authenticate the data bits is independent of the cryptography used to generate the markers in the spreading code.

Where independence fails is in the binding. Namely, the cryptographic key material used for both layers are linked together. For example, the public keys used in baseline Chimera determine the digital signature from which the slow channel marker keys are derived, and similarly for the TESLA chain keys and the slow channel marker keys for TESLA Chimera. Of course, out-of-band methods like the fast channel markers don't have this binding, but then they are separate from the data layer.

The Galileo SAS shows a different approach, in that the binding is done at the system level. A receiver needs to authenticate the data message, and then they can reach out to get the requisite spreading code information. That said, these methods all share the main bit commitment paradigm for protecting the hybrid signal: transmit a signal with data and spreading code authentication features, and then reveal information to the receiver to confirm those features.

End Notes

1. The authors have worked together, off and on, since before 2000. Many of our original thoughts are similar to those presented in this chapter.

2. More specifically, JJR and JTG with colleague Bob Lindell each separately came up with the core idea behind Chimera, which was then developed by the AFRL team. Subsequently, we learned of Scott's previous work a decade earlier. The term "Chimera" was coined by JJR.

3. An interesting related approach assessed the performance using the ability to discipline an inertial measurement unit (e.g., a commercial-grade aviation gyroscope) [17]. Their results show the advantage of using a message phase, and that nine groups performed well. The assignment was based on a reference grid, and the assignment to groups was done likely using a greedy algorithm.

References

[1] Psiaki, M. L., and T. E. Humphreys, "GNSS Spoofing and Detection," in *Proceedings of the IEEE,* Vol. 104, No. 6, 2016, pp. 1258–1270, doi:10.1109/JPROC.2016.2526658.

[2] Scott, L., "Anti-Spoofing & Authenticated Signal Architectures for Civil Navigation Systems," in *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003),* September 12, 2003, pp. 1543–1552, http://www.ion.org/publications/abstract.cfm?jp=p& articleID=5339.

[3] Gillis, J., J. J. Rushanan, and R. Lindell, "Chimera: A Framework for GNSS Signal Authentication," in *Joint Navigation Conference*, Dayton, OH: Institute of Navigation, June 6, 2016.

[4] Goldstein, D., AFRL PNT S&T Investments, November 12, 2015, Air Force Research Laboratory, https://web.stanford.edu/group/scpnt/pnt/PNT15/2015_Presentation_Files/I11-AFRLGPS_R&D.pdf.

[5] Slimak, K., Advanced GPS Technologies (AGT), Air Force Research Laboratory, May 1, 2015, https://www. gps.gov/multimedia/presentations/2015/04/partnership/slimak.pdf.

[6] Hinks, J., et al., "Signal and Data Authentication Experiments on NTS-3," in *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021),* September 24, 2021, pp. 3621–3641, doi:10.33012/2021.17964, http://www.ion.org/publications/abstract.cfm? jp=p&articleID=17964.

[7] Hinks, J., D. Chapman, and J. Anderson, "Navigation Technology Satellite–3: A Vanguard for Space-Based Position, Navigation, and Timing," in *Proceedings of the 2021 International Technical Meeting of The Institute of Navigation,* January 28, 2021, pp. 491–496, doi:10.33012/2021.17844, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=17844.

[8] Air Force Research Laboratory Space Vehicles Directorate, Satellite Navigation Technical Area, Interface Specification IS-AGT-100 Rev A: Chips Message Robust Authentication (Chimera) Enhancement for the L1C Signal: Space Segment/User Segment Interface, 2024.

[9] Air Force Research Laboratory Space Vehicles Directorate, Navigation Technology Satellite-3, Interface Specification IS-AGT-101: Timed Efficient Stream Loss-Tolerant Authentication (TESLA) Chips Message Robust Authentication (Chimera) Enhancement for the L1C Signal: Space Segment/User Segment Interface, 2024.

[10] Ortega Espluga, L., et al., "Data Decoding Analysis of Next Generation GNSS Signals," in *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, Miami, Florida, September 2019, pp. 1090–1105.

[11] Anderson, J. M., et al., "Chips-Message Robust Authentication (Chimera) for GPS Civilian Signals," in Pr*oceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017),* September 29, 2017, pp. 2388–2416, doi:10.33012/2017.15206, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=15206.

[12] GPS Program Office, *NAVSTAR GPS Space Segment/Navigation User Segment IS-GPS-800 Interface*.

[13] Gillis, J. T., and R. Allen, "TESLA Chimera Discrete Event Simulation for GPS Authentication," in *Proceedings of the International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2024),* September 20, 2024.

[14] Fernandez-Hernandez, I., et al., "Semiassisted Signal Authentication for Galileo: Proof of Concept and Results," *IEEE Transactions on Aerospace and Electronic Systems* Vol. 59, No. 4, 2023, pp. 4393–4404, doi:10.1109/TAES.2023.3243587.

[15] Winkel, J., I. Fernandez-Hernandez, and C. O'Driscoll, "Implementation Considerations for ACAS and Simulation Results," *IEEE Transactions on Aerospace and Electronic Systems,* 2023, arXiv: 2307.12398 [eess.SP], https://arxiv.org/abs/2307.12398.

[16] Commission Implementing Decision (EU) 2024/1882, https://eurlex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L_202401882.

[17] Esswein, M. C., and M. L. Psiaki, "Optimal Authentication Staggering Groups for Chimera, *Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022),* September 23, 2022, pp. 3378–3392, doi:10.33012/2022.18440, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=18440.

11

Other Things and Going Forward

This final chapter offers some miscellaneous topics and summarizes our final thoughts. We begin with other places where cryptography applies to navigation, one important example being location authentication or the secure location problem. We then discuss some more esoteric cryptographic methods and whether they may be applicable. We follow with a brief look at future trends in navigation, (i.e., where the cryptographic methods we discussed could be going). Finally, we end with what we believe are the most important lessons from this book.

11.1 Other Navigation Examples

This book shows the important and prevalent methods where cryptography is used to attain security goals in satnav. This section visits some less common methods and problems that did not fit into earlier portions of the book. The emphasis, of course, is on cryptography *and* navigation.

11.1.1 Snapshot Receivers

All of the methods used in this book to protect the satnav signal ultimately had the receiver do the final processing. For example, the receiver verifies the data using digital signatures or TESLA MACs and verifies the signals by confirming markers. The idea behind a snapshot receiver is to have the target receiver let some other receiver do the work. This concept has been around for a while [1].

The idea is shown in Figure 11.1. The target receiver captures information about all of the signals in space, perhaps after some signal processing, like

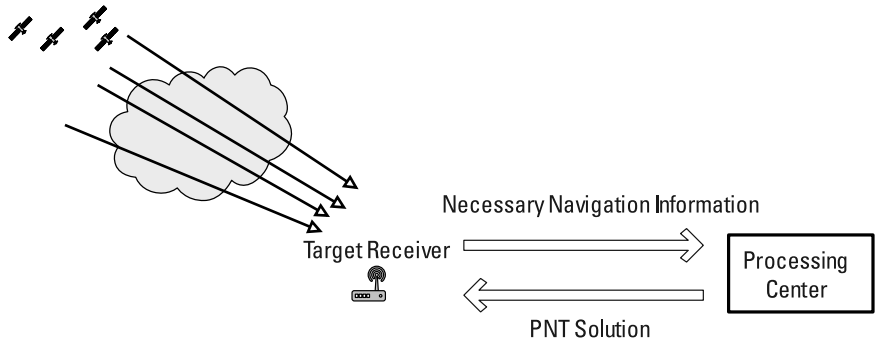

Figure 11.1 Snapshot receiver.

reducing to baseband. The target receiver then sends all of that information to some processing center, which could be another more capable receiver, or the cloud, and so on. After reception, the processing center computes the PNT solution from the data as if they were the actual receiver, and sends the solution back to target receiver.

This setup implies just normal navigation processing, but the same setup can be used for cryptographic processing. For example, perhaps the processing center contains the requisite cryptographic material and the target receiver does not. That remote processing would allow the target receiver to get the benefits of the cryptography without possessing the information. References about this concept include [2, 3].

Such a framework could be used with all the cryptographic methods for data and signal protections. The issue is the latency inherent in the process and the out-of-band communication channel required to send the information. This information can be large if, for example, raw navigation signal information is transmitted.

11.1.1.1 Location Authentication

The *location authentication* problem, also known as *secure location,* is for a target receiver to prove their location to someone else. There are possible variations to this problem:

- The target receiver knows where it is and wants to prove that information to someone;
- Someone else queries the target receiver to verify their position after possibly computing the position.

An example of an application for location authentication is to limit access to some information except at certain locations. One could consider this

as another authentication factor: along with something one has, is, and knows, there could also be the factor of where someone is.

A fun example is location-based augmented reality games. Released in 2016, Pokémon Go is a game that encourages the collection of Pokémon, which are made-up creatures, that can be virtually found at specific locations. The idea is that you check your phone looking for a Pokémon and get to see the location base advertisement, which is the business model behind the game. It didn't take long for location position spoofers to proliferate, so that the collection of rare Pokémon could be accomplished without leaving your home. Most of these spoofings seem to operate by falsifying the GPS coordinates supplied by the phone to the Pokémon game. Thus location authentication is failing due to issues in the receiver itself.

Another example is location service based on an internet IP address, which has been around for a while. The techniques are based on discovering peculiarities in internet routing and the traceroute tool, which was initially developed in 1987. An early paper [4] describes an architecture and correctly points out the difficulties of using GPS for location proof by itself and proposes to use a differential GPS system.

Such techniques provide a template for how one might produce a location-verification service:

- Create and broadcast a signal with secure features below the noise floor;
- The target receiver samples and buffers the received RF;
- The receiver signs this buffer and submits it to some processing center for verification;
- The processing center verifies the origin of the information and processes the signal to see if it is consistent with the stated location.

There are nuances, such as how much interaction is needed in this protocol. This example uses snapshot receivers that we just described.

A similar recent example has been suggested by Scott using the Chimera signal [5] (i.e., by leveraging authentication information already built into the signal). Such schemes essentially use a snapshot receiver infrastructure not to obtain position but to send both position and corroborating evidence. This scheme removes the need for remote processing to compute the position; it only needs to verify it.

11.1.2 Using Location in Cryptography

Another concept is for a receiver to use its location as input to cryptographic functions. Early examples are given in [6]: geoencryption. Related work is based

on that of Dorothy Denning in her paper in 1996 [4]. The core idea is to tie decryption to a location and time. An example problem would be theater film distribution. If a digital copy of a film was encrypted so that it could only be decrypted in a finite time interval (time-locked) and location, then you could use the internet to distribute the file without concern about piracy. However, there is the problem of lack of entropy. The granularity of locations on the surface area is about 5.1×10^{14} meters, so that every square centimeter is indexed in just under 64 bits. A year is a bit over half a million minutes, which is indexed by 20 bits. Coarser granularity would mean less entropy into the plaintext.

To encrypt data, one would compute the desired location/time of the target, rounded to an appropriate value so that the result is predictable from the receiver, and then compute a keyed hash of that value using some preplaced key. From this keyed hash, the key for geolocking could be derived, which then could be used to encrypt the data.

Decryption reverses these steps and requires that the user submit the correct position into the key to get the correct geolocking key. In particular, there is an implicit chain of trust from the GPS signal-in-space through the creation of the geolocking key. This trust comes from antispoof methods applied to the signals and integrity of the hash computation. The latter implies a need for trust in the user equipment. Furthermore, as with all things GPS, if someone records the signal and plays it back as if it were at the correct location, the protocol would be broken. Any methods one proposes only authenticate that the signal is correct; these methods cannot verify the signal path was correct without some outside measurement.

Notice that without much work, this type of protocol can be modified to give location authentication. Assuming a trusted receiver, it can simply pass the geolocking key as a witness of a specific location. We could modify the UE to output a digitally signed version of PNT, and it would be a testament to the UE having been at that location at that time.

11.2 But What About ...?

Cryptography is a very broad field, and while we have tried to match that breadth in our discussion, there are necessarily some topics that may have niche applications for navigation. An additional topic is the impact of quantum technologies on cryptography, which is discussed in the appendix.

11.2.1 Homomorphic Encryption

The idea behind *homomorphic encryption* is to create a cryptographic system where operations can be performed on ciphertext without decrypting the oper-

ands. The term "homomorphism" in mathematics means a function that preserves operations. More formally as an example, if $f(x, y)$ is some function on two variables, we want the following property of the encryption scheme:

$$f\left(E_K\left[P_1\right], E_K\left(P_2\right)\right) = E_K\left[f\left(P_1, P_2\right)\right]$$

where the P_i are plaintexts. A usage example would be something like aggregation of private data. The data could be gathered and totaled, say, without decrypting. Later, the processed answer could be decrypted without ever having revealed the inputs.

There are many schemes that have been proposed, many of which only achieve partial homomorphic encryption. That is, only certain functions can be done. One simple example is RSA encryption, where $f(x, y) = x \cdot y$ is achieved by fact that

$$\left(M_1, M_2\right)^e = M_1^e M_2^e \left(\text{mod } n\right)$$

Recent schemes have been proposed that can do full homomorphic encryption, such as operations one would find useful in signal processing, but they are often very inefficient [7].

Navigation applications of homomorphic encryption are related to using positioning as inputs to cryptographic processes, such as discussed in the last section. Homomorphic encryption would allow performing those calculations while keeping the position private.

11.2.2 Blockchain

We talked briefly about blockchain in Section 4.6. Recall the usefulness to capture a secure ledger that records many transactions. As such, blockchain is useful in providing auditing and logging information, an important desire for e-currency.

A recent application to navigation is GEODNET [8]. The idea is to create a decentralized system that provides assistance for higher accuracy for satnav like GPS. The blockchain concepts are used to incentivize the various entities in the system to provide service.

11.2.3 Physical Layer Key Exchange

Many of the cryptographic methods have focused on the physical layer of the satnav signal, such as in Chapter 9. It is worth mentioning that the physical layer can also be used to provide cryptographic solutions. One example of this is

the work of the past decades on *physical layer key exchange*. The idea is for Alice and Bob to agree on a secret key by leveraging a random physical channel. The nuance is that anyone can also observe the channel, so any protocol needs to account for possible eavesdroppers.

The concept of communicating in the face of eavesdroppers dates back to Wyner's wiretap model [9]. In this example, it can be shown under certain channel noise model assumptions that Alice can communicate to Bob with guarantees on bounding how much an eavesdropper can learn. Note that cryptographic hashing can reduce any such eavesdropper information to essentially nothing. This idea can be extended to various physical media. Here the idea is that Alice and Bob perform some kind of protocol exchange, like sending simultaneous pulses through a disturbed medium, measure some random feature, and use that to extract a key. One example uses an optical channel through a turbulent atmosphere [10]. See [11] for a a more theoretical look at these protocols.

Can such protocols have a usage in navigation? Perhaps a as supplement to other protocols (e.g., to provide ad hoc shared keying during snapshot or other processing).

11.2.4 Implementation Security

Our last topic is not cryptography per se but the enabler of cryptography, namely how to achieve security in the implementation of cryptographic methods. Implementation security depends on the platform and which part of the platform we are concerned with. The security goals are related to the function: confidentiality of processing and stored material, integrity of the underlying processes, and protection against denial of service.

At the higher levels, implementation security falls under the realm of cybersecurity and INFOSEC. Examples include protecting operating systems and networks. We touched on some aspects in the discussion on protocols in Chapter 6. Lower-level concerns fall under the realm of *embedded security*: extending the security goals to the actual hardware implementations. Embedded security has many challenges due to the hardware devices often being very resource constrained, having little connection to user, being part of a larger system, and being special purpose. This is a very broad topic with lots of threats and countermeasures (see, e.g., [12–14]).

Implementation security and embedded security apply to satnav, of course, and indeed we hinted at the need anytime we talk about trusted processing or receivers. Creating good embedded security solutions, such as the TPM, can be used a a foundation for the overall security in a platform. Moreover, such solutions can become standardized and certified.

11.3 Future Trends

Even though some satnav is ingrained in the daily lives of the world (i.e., the use in billions of cell phones) satnav continues to evolve. We mention some of the trends at the time of this writing. But satnav is not the only way to do navigation, and so we briefly mention other approaches and how cryptography may have an impact.

11.3.1 Future Satnav

We offer some observations about some of the various satnav systems and where they may be going.

- For GPS, perhaps the obvious item is the launch of NTS-3 as this book is being published and the host of cryptographic and authentication experiments. These experiments are led by the use of Chimera; the Chimera experiments will allow for real-time testing of the proposed signal from space. Such testing hopefully paves the way for future implementation.
- Galileo has already invested in cryptographic methods through its OS-NMA signal. The use of SAS is emerging, and signal authentication methods embedded in the signal are forthcoming.
- We expect similar methods to emerge in all GNSS systems in the years ahead.

These satnav systems are government owned. In recent years there has been a rise in privately owned services, mainly due to the reduced cost of LEO solutions.

11.3.2 Other Navigation Methods

There has been an ever-increasing push for complementary PNT, such methods as independent clocks, inertial measurement units (IMUs), celestial navigation, magnetic navigation, visual-aided navigation, and so on. All these methods offer very good features, some of which help with security. For example, an independent clock cannot be spoofed itself, and any threats end up being threats to the platform. Even so, there is much work emerging on the nonsecurity integrity aspects. For example, see [15] for a discussion of visual-aided navigation. These methods are truly complementary, in the sense they do not, as of now, have the ubiquity of satnav.

Does cryptography matter in complementary PNT? That depends on the required security goals and the protections. If visual-aided navigation depends on tools such as good maps then that information needs to be protected. Fortunately, those protections probably can leverage standard protocols, as discussed in Chapter 6.

11.4 Our Final Thoughts

Our central goal for this book was to present cryptography in the context of satnav for those interested in either or both of those topics. A related goal was to offer all the lessons that we have accumulated from decades of working on these topics. Here then are the most important principles as we see them.

Satnav Security Is an Element of Security
Satnav provides positioning and timing, and both of these are very important to the operational goals of perhaps every system. Thus, securing position and timing is needed for security overall.

Satnav Security Has Unique Features
The security goals of positioning and timing differ from the usual security goals. Even when concepts such as authentication and integrity are similar, the ephemeral nature of the position solution suggests different methods.

TRANSEC Is Not INFOSEC
Security goals and protections for the signal in space are not the same as the goals and methods to protect information.

"Cryptography est omnis divisa in partes tres"
There are three main areas of cryptographic methods:

- Symmetric ciphers, where shared keys are used to primarily do encryption. An example is AES.
- Hashing, which reduces data to a small fingerprint, sometimes also using a key. Examples are SHA2 family and HMAC.
- Asymmetric or public key cryptography, which uses key pairs to accomplish encryption, signing, and key exchange. Examples are RSA, ECDSA, and Diffie-Hellman.

Protocols Are Harder than Algorithms
When people think of cryptography, they often focus on the algorithms and keys such as AES, RSA, and ECDSA. But cryptographic protocols perform the actual work to achieve security goals by leveraging algorithms in multiple steps.

Protecting Navigation Data Is Almost Straightforward
While methods to authenticate data, say using digital signatures, are standard, using those methods in a satnav signal present challenges of data bandwidth and latency.

Protecting Satnav Signal Timing Is Not as Straightforward
Securing the timing derived from spreading codes can be difficult in the face of needing methods usable by all users.

Cryptography Is Just One Piece of the Puzzle
This book necessarily focuses on cryptographic methods, but we stress that such methods are only part of the overall enterprise solution. Systems concerns, concept of operations (CONOPs), cost/benefits, and so on must all be taken into account.

References

[1] Petovello, M., "What Is Snapshot Positioning and What Advantages Does It Offer?" *Inside GNSS,* December 6, 2018, https://insidegnss.com/what-is-snapshot-positioningand-what-advantages-does-it-offer/.

[2] Lo, S., et al., "Signal Authentication: A Secure Civil GNSS for Today," *Inside GNSS,* 2009, https://insidegnss.com/wp-content/uploads/2018/01/sepoct09-Lo.pdf.

[3] Boughton, R. S., et al., "Extracting Location from RF Samples," Private Communication, 2002.

[4] Denning, D. E., and P. F. MacDoran, "Location-Based Authentication: Grounding Cyberspace for Better Security," *Computer Fraud & Security,* February 1996, pp. 12–16, doi:10.1016/S1361-3723(97)82613-9, https://linkinghub.elsevier.com/retrieve/pii/S1361372397826139.

[5] Divis, D. A., "New Chimera Signal Enhancement Could Spoof-Proof GPS Receivers," *Inside GNSS,* June 3, 2019, https://insidegnss.com/new-chimera-signal-enhancement-could-spoof-proof-gps-receivers/.

[6] Scott, L., et al., "A Location Based Encryption Technique and Some of Its Applications," in *Proceedings of the 2003 National Technical Meeting of the Institute of Navigation,* 2003, pp. 734–740.

[7] Gorantala, S., R. Springer, and B. Gipson, "Unlocking the Potential of Fully Homomorphic Encryption," *Communications of the ACM,* Vol. 66, No. 5, 2023, pp. 72–81, doi:10.1145/3572832, https://doi.org/10.1145/3572832.

[8] GEODNET Foundation, 2024, https://geodnet.com/.

[9] Wyner, A. D., "The Wire-Tap Channel," *The Bell System Technical Journal,* Vol. 54, No. 8, 1975, pp. 1355–1387, doi:10.1002/j.1538-7305.1975.tb02040.x.

[10] Drake, M. D., et al., "Optical Key Distribution System Using Atmospheric Turbulence as the Randomness Generating Function: Classical Optical Protocol for Information Assurance, *Optical Engineering,* Vol. 52, No. 5, 2013, p. 055008, doi:10.1117/1.OE.52.5.055008, https://doi.org/10.1117/1.OE.52.5.055008.

[11] Bloch, M., and J. Barros, *Physical-Layer Security: From Information Theory to Security Engineering,* Cambridge, UK: Cambridge University Press, 2011.

[12] Hou, X., and J. Breier, *Cryptography and Embedded Systems Security,* Cham, Switzerland: Springer Nature Switzerland, 2024, https://books.google.com/books?id=veKv0AEACAAJ.

[13] Bhunia, S., and M. Tehranipoor, *Hardware Security: A Hands-on Learning Approach,* San Francisco, CA: Morgan Kaufmann Publishers, 2018.

[14] van Woudenberg, J., and C. O'Flynn, *The Hardware Hacking Handbook: Breaking Embedded Security with Hardware Attacks,* San Francisco, CA: No Starch Press, Inc., 2022.

[15] Zhu, C., M. Meurer, and C. Günther, "Integrity of Visual Navigation—Developments, Challenges, and Prospects," *Navigation,* Vol. 69, No. 2, 2022.

Appendix: The Influence of Quantum

This discussion of cryptographic strength depends on the capabilities of an adversary. For example, the time to brute force a shared key is taken to be the strength of symmetric ciphers like AES. Such measures depend on a standard, classical computing model. But what happens if that model changes? This appendix discusses the fundamental change in the computing model induced by quantum technologies.

At a simple level, classical computing is concerned with *bits* and the rules for digital logic, which Claude Shannon identified with Boolean algebra in his master's thesis [1]. Starting in the 1970s physicists started to look at the quantum mechanical aspects of computing, including the observation that something now called *qubits* could hold more information than classical bits, with some serious restrictions. Further *quantum information theory* was not a simple extension of Shannon's information theory.

In 1982, Richard Feynman [2] suggested the first problem that could be solved by a quantum computer and not evidently by a classical computer: the simulation of quantum phenomena. By the mid-1990s, the first two algorithms applicable to problems outside the field of physics were articulated. The first, in 1994, was Shor's algorithm for integer factorization; it is superpolynomially faster than the best-known classical factorization method. The second algorithm, Grover's algorithm in 1996, provides quadratic speed-up for searching an unstructured list. These two algorithms are the best-known quantum computing algorithms and their ideas are at the root of many other of the suggested algorithms for quantum computing (see [3] or *Quantum Zoo* [4]).

Grover's algorithm is a threat to symmetric cryptography, although there is an easy fix. Shor's algorithm is an existential threat to common asymmetric cryptographic algorithms (Chapter 5) based on factorization and, generally, any hidden subgroup problems like elliptic curve cryptography. The impact of Shor's algorithm leads to the need for post-quantum methods, and we present some methods that are germane to satnav applications along with the motivating mathematics based on NP-complete problems. A full suite of post-quantum methods have been recommended by NIST [5], such as the lattice-based methods: CRYSTALS-Kyber, a key encapsulation method, takes the place of Diffe-Hellman key exchange; CRYSTALS-Dilithium and Falcon for digital signing. There is also a stateless hash method SPHINCS+. These join AES256 as NIST's initial post-quantum suite. An issue is that post-quantum cryptographic methods, and in particular, hash-based and lattice-based methods require significantly more bits for a given level of security than elliptic curve methods. This increase is an issue for bandlimited methods, such as in-band digital signing of navigation data. As such, we discuss a method for digital signing based on multivariate cryptography that has the potential to resolve this issue.

As a final note, it is worth considering a warning from Richard Feynman in *The Character of Physical Law* (Lecture 6) [6]:

> On the other hand, I think I can safely say that no one understands quantum mechanics. So do not take this lecture too seriously, ... I am going to tell you what nature behaves like. If you will simply admit that maybe she does behave like this, you will find her a delightful, entrancing thing. Do not keep saying to yourself, if you can possibly avoid it, "but how can it be like that?" because you will get "down the drain", into a blind alley from which nobody has yet escaped. Nobody knows how it can be like that.

A.1 The Components of Quantum Computing

A classical bit can be represented as a switch or any other two-state value. Mathematically, this representation is as an element in $Z_2 = \{0, 1\}$. The value of the bit is deterministic and equal to one of these values. It can be copied and manipulated by gates (i.e., electronic circuits, but the result is always deterministic).[1]

In contrast, the most basic unit of computing in a quantum computer is the *qubit*. The state of a qubit is more subtle than a bit: it is a quantum mechanical phenomenon with five special properties:

- *Superposition:* The ability to exist in any combination of the two pure states simultaneously in different proportions. That is, the state is *both* a mix of the pure states, informally a 0 state and a 1 state. The limitation

is that at some point, *the measurement*, the qubit will collapse into one of the two basis states, randomly, with likelihood proportional to weights in the superimposed state.

- *Unitary operations:* Computational actions on a qubit. These actions function like gates for classical bits. Unitary operations are invertible, so actions on qubits are reversible. For example, Toffoli gates makes classical logical operations reversible so that quantum computers include classical computations [7].
- *Measurement:* The *collapse*, to the output value of the qubit. Measurement is the exceptional action that is not reversible (non-unitary).
- *Entanglement:* The ability of multiple qubits to couple with each other so that the measurement of one determines the measurement of the other.
- *No Cloning:* It is impossible to copy or clone the state of a qubit in general. This restriction has profound implications for the computing model used in constructing a quantum computer [8, 9]. Structures such as cache and memory buses are no longer convenient. A qubit can exchange state with another qubit, but this is not duplication, so not *copying*.

A.1.1 States and Superposition

The typical quantum physics notation for vectors is "kets" $|x\rangle$ for a column vector, and "bra" $\langle y|$ for a row vector. The contents of a ket (or bra) is something that specifies the state involved. If a number is specified, then it is usually the index of the basis vector under consideration. Under this convention, $|0\rangle$ is the vector $\begin{bmatrix}1\\0\end{bmatrix}$ and $|1\rangle$ is the vector $\begin{bmatrix}0\\1\end{bmatrix}$. These are the two fundamental states. Similarly, $\langle 0|$ is [1, 0], and $\langle 1|$ is [0, 1]. This is called the Dirac notation, and many variations seem to be in use, hence our definition of the contents as "something that specifies the state." If you haven't noticed, the usual Euclidean inner product, in physics called the *bracket*, is a bra next to a ket.

As a simple example, let our qubit be light (photons) [10]: the basis vectors will be horizontal polarization ($|h\rangle$) and vertical polarization ($|v\rangle$). Light exists in a combination or superposition of polarizations:

$$\ell = \alpha|h\rangle + \beta|v\rangle \tag{A.1}$$

where α and β are complex numbers such that

$$Norm(\ell) = \overline{\alpha}\alpha + \overline{\beta}\beta = |\alpha|^2 + |\beta|^2 = 1 \tag{A.2}$$

The coefficients α and β are the *amplitudes*, and $\overline{\alpha}\alpha$ is the probability that the light will be polarized horizontally when measured and similarly for $\overline{\beta}\beta$ when polarized vertically.

The state of a qubit is then the set of the possibilities, here, the polarization, each with its respective amplitude. The magnitude squared of the amplitude is the probability of the event. The act of converting the amplitudes to a value is is a measurement. The value that occurs is probabilistic, with proportions given by the magnitude squared of the amplitudes. As a consequence of this framework, the multiplication of the state by any complex number of magnitude one is irrelevant. Thus, states differing by $e^{i\theta}$ are all the same, and in particular, ± 1 states are indistinguishable. For instance, it does not matter if the polarization vector is along the positive x-axis or the negative x-axis when horizontally polarized.

Since quantum phenomena has multiple realizations, not just polarized light,[2] there are lots of representations of qubits, and photonics (light) is not the most common currently. Hencefort, we take the "computational" basis states as $|0\rangle$ and $|1\rangle$ and the magnitude squared of the coefficient of the basis states will be the probability that that basis state will occur when measured.

A.1.2 Matrices: Hermitian, Unitary, and Measurement

We treat quantum mechanics from a matrix view, which is consistent with the quantum computing literature and the Matrix Mechanics view of quantum mechanics, developed by W. Heisenberg and M. Born.[3] Our discussion here follows [11], especially chapters 4 and 5.

A *Hermitian matrix* is a matrix for which $H = H^{\dagger}$ (i.e., H equals its adjoint), which is its conjugate transpose. Such matrices have real eigenvectors even if the entries are complex. Furthermore, the eigenvectors of a Hermitian matrix are orthogonal and thus induce a decomposition of the space. If the matrix has real entries, the term *symmetric* is used.

A simple example of a Hermitian matrix is the *Hadamard* matrix:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{A.3}$$

When it acts on the usual basis, we get

$$|+\rangle = H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle \tag{A.4}$$

and

$$|-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \tag{A.5}$$

The resulting basis is called the Hadamard basis. If H is used to act on this basis, it returns to the original basis [11]. Note that if one were to measure say $|+\rangle$, we would obtain either $|0\rangle$ or $|1\rangle$ equally likely, because the square of each coefficient is one half.

This example motivates the *measurement postulate*: Any observation (measurement) of a quantum mechanical system is specified by a Hermitian matrix, O. If the state is $|\Psi\rangle$ before the measurement, then

- The possible outcomes of measuring Ψ with O are $\left\{ v_j = \frac{1}{\left|P_j \Psi\right|} P_j \Psi \right\}$;
- The probability of measuring ν_j is $|P_j|\Psi\rangle|^2$.

Here P_j is the projection matrix onto the jth eigenvector. The important thing is that Hermitian matrices yield the measurements. To emphasize: the measurement of $|\Psi\rangle = a|0\rangle + b|1\rangle$ by O is $|0\rangle$ with probability $|a|^2$ and $|1\rangle$ with probability $|b|^2$ and the state after the measurement is $|0\rangle$ or $|1\rangle$, respectively. Note $|1\rangle$ and $-|1\rangle$ are the same state.

Quantum computing needs gates; that is, a way to transform a set of qubits to another set of qubits. Gates are accomplished using unitary operators defined by unitary matrices. A *unitary matrix* is one in which the conjugate transpose is the inverse, $U^{-1} = U^{\dagger}$. Note that H is also unitary and thus is a gate—the *Hadamard* gate. It transforms a state into the superposition of two states, as shown above.

Unitary matrices give quantum operations because the evolution of a quantum mechanical system is specified by a unitary operation.[4] Unitary matrices have orthogonal eigenvectors and the eigenvalues lie on the unit circle.

A.1.3 Entanglement

Because the governing equations for quantum effects involve probability, it is unsurprising that qubits can be made dependent on each other. What perhaps is surprising is the consequence of such coupling.

An example of coupling is a two-photon state represented on the basis vectors for horizontal and vertical polarizations:

$$|h\rangle|h\rangle, |h\rangle|v\rangle, |v\rangle|h\rangle, |v\rangle|v\rangle \tag{A.6}$$

or in shorthand:

$$|hh\rangle, |hv\rangle, |vh\rangle, |vv\rangle \tag{A.7}$$

In our usual basis, we write this as

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle \tag{A.8}$$

which is the basis space for the tensor product: $|hv\rangle = |h\rangle \otimes |v\rangle$, or $|01\rangle = |0\rangle \otimes |1\rangle$.

Not every vector in the span of the tensor product is a tensor product. In mathematics, elements that can be represented as a tensor product are the decomposable elements. In this context, states in the span that can be decomposed into products are called unentangled or separable and those that cannot are called entangled. As an example, take two qubits, *A* and *B*, that satisfy:

$$A : \alpha_A |0\rangle_A + b_A |1\rangle_A \tag{A.9}$$

$$B : \alpha_B |0\rangle_B + b_B |1\rangle_B \tag{A.10}$$

Then

$$X = A \otimes B \tag{A.11}$$

$$= \alpha_A \alpha_B |0\rangle_A \otimes |0\rangle_B + \alpha_A b_B |0\rangle_A \otimes |1\rangle_B + b_A \alpha_B |1\rangle_A \otimes |0\rangle_B + b_A b_B |1\rangle_A \otimes |1\rangle_B \tag{A.12}$$

$$= \alpha_A \alpha_B |00\rangle + \alpha_A b_B |01\rangle + b_A \alpha_B |10\rangle + b_A b_B |11\rangle \tag{A.13}$$

And, of course, $\overline{a_A} a_A + \overline{b_A} b_A = 1$ and $\overline{a_B} a_B + \overline{b_B} b_b = 1$, because each set of probabilities must add to 1.

By construction, *X* is an unentangled state. The intuition is that if a state is separable, then it can be decomposed into two states that are uncorrelated. If the state is entangled, it cannot be so decomposed, and the states are correlated, and in fact are correlated *in any basis*. This gives rise to the "spooky action at a distance" that Einstein famously objected to [12].[5]

It is easy to construct an entangled state; in fact

$$\Phi^+ = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \tag{A.14}$$

is such a state—it is known as the first Bell State. To see this, assume it was unentangled, then we would have the system of equations:

$$1/\sqrt{2} = \alpha_A \alpha_B \tag{A.15}$$

$$0 = \alpha_A b_B \tag{A.16}$$

$$0 = b_A \alpha_B \tag{A.17}$$

$$1/\sqrt{2} = b_A b_B \tag{A.18}$$

It's not too hard to see that the inner equations require α_A, b_B (respectively, b_A or α_B) or both to be zero, which is not consistent with the top and bottom equations.

To explain how to construct the $|\Phi^+\rangle$, we need two gates (unitary operators): the "Not" and subsequently the Controlled Not or CNot gates. The Not operation acts to send $|0\rangle \rightarrow |1\rangle$ and $|0\rangle \rightarrow |1\rangle$ and is given by:

$$Not = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{A.19}$$

It is worth looking at what the Not operation does on a mixed state, $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ then

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{A.20}$$

The Not gate also transforms the Hadamard basis: $M|+\rangle = |+\rangle$ and $M|-\rangle = -|-\rangle$. It is tempting to call the Not gate "Swap," but that is not what the literature does.

A more interesting gate on two qubits is the "controlled-not" gate, *CNot*.

$$CNot = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.21}$$

The name arises in the following way:

$$|00\rangle \xrightarrow{CNot} |00\rangle; \quad |01\rangle \xrightarrow{CNot} |01\rangle \tag{A.22}$$

$$|10\rangle \xrightarrow{CNot} |11\rangle; \quad |11\rangle \xrightarrow{CNot} |10\rangle \tag{A.23}$$

So, in the standard basis ($|0\rangle$, $|1\rangle$), the first qubit ("control") determines the second qubit ("target") and the operation looks like the classical xor of the control and target. Furthermore, CNot as a transformation in the Hadamard basis produces

$$|++\rangle \xrightarrow{CNot} |++\rangle; \quad |+-\rangle \xrightarrow{CNot} |--\rangle \tag{A.24}$$

$$|-+\rangle \xrightarrow{CNot} |-+\rangle; \quad |--\rangle \xrightarrow{CNot} |+-\rangle \tag{A.25}$$

and you could argue that the role of control and target are switched from the standard basis.

If we start with two qubits in the $|0\rangle$ state and apply the Hadamard-gate (the H-Gate) to the first qubit to put it into the $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ state, and then use this as the control gate to the second qubit, something rather amazing happens: the qubits are entangled, and the first Bell state is created. We show this schematically in Figure A.1.

If either qubit is measured in any basis, the other qubit will be the same. Moreover, the probability that a qubit is measured in the standard basis is 50% for either basis function being the outcome. Close examination shows that the qubit is measured in the Hadamard basis; the outcome is also 50% for each basis vector.[6]

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \tag{A.26}$$

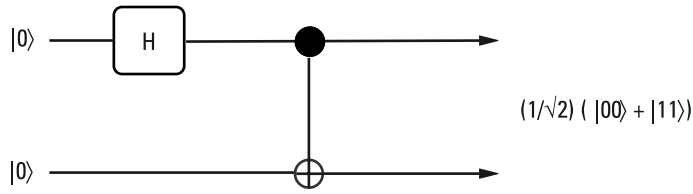

Figure A.1 H-Gate followed by Controlled-Not-Gate.

If we were to simulate these actions on a classical computer, we can see the effect of superposition and entanglement. If we entangle two of our qubits, i, j, we would remove $\{\alpha_k, \beta_k\}_{\{k = i, j\}}$ from the simulation and replace it with four variables $\{\alpha_k, \beta_k, \gamma_k, \delta_k\}_{\{k=i,j\}}$ and we must keep track of all four values, {00, 01, 10, 11} and what each transformation would do to these values. Classically, we would need to act on four-dimensional vectors, ({0, 0, 0, 0}, etc.). If n qubits are involved, then the simulation would track 2^n values, which shows how the ability to simulate a quantum computer on a classical computer is limited.

Even so, it is important to note that the output of a quantum computer is a classical value: a single measurement.

Quantum Computing Output
The measured value at the output of a quantum computation is determined probabilistically, and repeated trials will yield values in proportion to the probability of their occurrence.

The strategy in quantum computing, then, is to arrange a calculation that is likely to yield the desired result with a high-enough probability.

A.1.4 No Cloning

The no-cloning theorem places restrictions on the copying of qubits, and this prevents fan-out architectures that are common in conventional digital circuits. Thus, many basic advantages of the von Neuman architecture, and its descendants, like the Harvard architecture, become expensive or, in the case of caching (the full state) is impossible. The situation is nuanced, as a CNOT gate copies the control qubit if it is in the $|0\rangle$ or $|1\rangle$ state in the standard basis, but it will not copy the Hadamard basis [13].

A.2 Quantum Computing Algorithms

The no-cloning theorem, the requirement for reversible computing using unitary operations on the state, and Hermitian measurement calls for a quite different model of computing than the one we are familiar with. This section describes the important quantum computing algorithms: Grover's and Shor's, after giving a simpler example by Deutsch.

In this book, we do not discuss everything that would be needed to realize quantum computing. In particular, much effort is needed to implement quantum error correction [14].

A.2.1 Deutsch's Algorithm

The first example of a problem that could be solved by a quantum computer "better" than a classical computer appeared in 1985, and it has become known as *Deutsch's problem* [15]. It is simple, though perhaps contrived. It was updated to more or less the form presented here, and extended to a multibit version that showed *exponential advantage* of quantum computing over classical computing in 1996 [16]. That generalization is the *Deutsch-Jozsa problem*; however, it is equally contrived.

The problem introduces an important concept in the analysis of cryptographic algorithms: *oracles*. As an example, given a plaintext, PT, and a ciphertext, CT, an oracle would return true for a key that encrypted PT into CT and false for any other key. A *classical oracle* is a function that returns a Boolean value or Boolean vector. Classical oracles represent functions that we also want to implement on a quantum computer. Toward that goal, a *quantum oracle* is a classical oracle that has been extended to be reversible. This extension can be done by implementing the classical oracle in reversible gates and adding an ancillary state XOR'd with the oracle to make it reversible. Thus, given a classical oracle, *f*, the unitary equivalent is

$$U_f \left|xy\right\rangle = U_f \left|x\right\rangle \left|y\right\rangle \tag{A.27}$$

$$= \left|x\right\rangle \left|f(x) \oplus y\right\rangle \tag{A.28}$$

$$= \left|x, f(x) \oplus y\right\rangle \tag{A.29}$$

That is, applying the oracle for f to bits x and y yields the bits x and $f(x) \oplus y$, where $\oplus$ is XOR. Note that U_f is self-inverse, and it can be shown that U_f is unitary, and so is a valid quantum gate.

Very often, this gets used in a construction called a *phase oracle*, which is an oracle represented as

$$\left|x\right\rangle \xrightarrow{U_f} (-1)^{f(x)} \left|x\right\rangle \tag{A.30}$$

Here, the $f(x) = 0$ value of the oracle returns the state ($|x\rangle$), and the $f(x) = 1$ value of the oracle returns the negated state $-|x\rangle$. We will use the phase oracle construction in the discussion of Grover's algorithm (see Section A.2.2).

Deutsch's problem is discussed in most books on quantum computing, we most closely followed [7]; [17, 18] has more discussion on physical realizations of quantum computers and [9] is a fairly succinct book, with a nice explanation

of the Dirac notation in their Appendix A. Those seriously interested should consider [8].

To set up Deutsch's problem, consider that there are four functions on a single Boolean variable (i.e., $f: X \in \mathbb{Z}_2 \rightarrow \mathbb{Z}_2$). The two constant functions are $f_0(X) = 0$ and $f_1(X) = 1$; and the two "balanced" functions: $f_2(X) = X$ and $f_3(X) = \neg X$ (where ¬ complements a Boolean value, e.g., ¬1 = 0).

Deutsch's Problem
Given an *unknown* oracle, determine if it is constant or balanced.

Classically, this determination takes two evaluations of the oracle; we will demonstrate that it is possible to instead use *one* evaluation of the oracle in a quantum computing setting.

Applying the oracle U_f to an initial state $|a\rangle|b\rangle$ yields $|a\rangle|f(a) \oplus b\rangle$. Using the Hadamard basis, $U_f|-\rangle|+\rangle$ yields $|-\rangle|U_f+\rangle$ and we can expand $|U_f+\rangle$ as

$$\left|U_f +\right\rangle = \frac{1}{\sqrt{2}}\left(U_f\left|0\right\rangle + U_f\left|1\right\rangle\right) \tag{A.31}$$

$$= \frac{1}{\sqrt{2}}\left((-1)^{f(0)}\left|0\right\rangle + (-1)^{f(1)}\left|1\right\rangle\right) \tag{A.32}$$

If $f = f_0$ then the output is $|+\rangle$, if $f = f_1$ then the output is $-|+\rangle$, so for constant functions the output is $\pm|+\rangle$. If $f = f_2$ the the output is $|-\rangle$ and for $f = f_3$ the output is $-|-\rangle$, so for balanced functions the answer is $\pm|-\rangle$. Since global phase does not matter, the ± are irrelevant, and we have a discriminant for balanced versus constant in one function evaluation.

There is one issue to be dealt with—we don't measure in the $|+\rangle$, $|-\rangle$ (Hadamard) basis.

However, the H gate is self-inverse and $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$, so we can use the Hadamard gate to get to the computational basis and our result. This completes the circuit for Deutsch's problem. We show a schematic in Figure A.2.

Notice that the problem solved is pretty restricted; we don't know *any* values of the function, just the specific answer to the posed question. In particular, we are given the oracle without knowing it; hence the use of the term "oracle." Questions that were not contrived had to wait for Peter Shor [19, 20] (factorization) and Lou Grover [21] (search), discussed next.

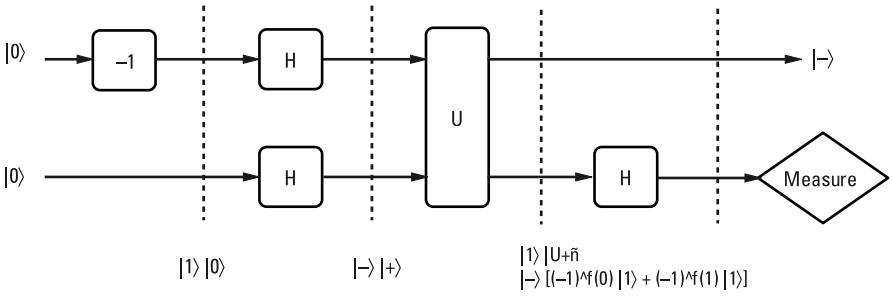

Figure A.2 Quantum program to solve Deutsch's problem.

The trick of making a non-reversible oracle reversible is the basis for converting classical computing, non-reversible operations to being realizable on a quantum computer. The cost is adding additional qubits to allow reversion of the operation. Since this is a polynomial operation, it turns out that quantum computers have a mechanism to perform classical computations with, at most, polynomial inefficiency (see Section A.3.3).

A.2.2 Grover's Algorithm

In 1996, Lou Grover announced the second important quantum computing algorithm, a "fast" search algorithm for searching an ordered list [21]. The quotes on "fast" are because the speed-up is *not* exponential, it is quadratic. At this time, there is a decent argument that the quadratic speed-up is not sufficient to provide a real advantage for quantum computing [14].

Using a classical computer, searching an unsorted list of $N=2^n$ elements to find a particular element requires $N/2$ trials to have a 50% probability of finding that element. Grover's algorithm can find the distinguished element in $\sqrt{N}$ trials with high probability. As an example, trying to brute force a symmetric cryptographic n-bit key with a known plaintext-cipher text pair requires searching $N=2^n$ possible keys, and so a classical brute force attack is accomplished in 2^{n-1} trials (50% of the time). The equivalent work under Grover's algorithm is $\sqrt{N}=\sqrt{2n}=2^{n/2}$.

The fix for this threat from quantum computing is pretty easy: design symmetric algorithms with twice the key strength. For instance, AES supports 192- and 256-bit key sizes, which, if one were to apply Grover's algorithm, would have a nominal strength of 96 and 128 bits, respectively. AES also is defined for 128-bit keys, and if Grover's algorithm could be efficiently implemented, then the effective strength would be 64 bits, which would *not* be secure. Clearly, the symmetric design community has been taking this threat into account since the time of the NIST AES competition [22]. It is known that Grover's algorithm is optimal for the unordered search problem [23].

AES and Quantum Computing
To be clear, AES256 is secure even in the event of quantum computing.

A.2.2.1 Grover's Algorithm Overview

Grover's algorithm uses two ideas. The first is that of a phase oracle, which was introduced in Section A.2.1, and the second is the *amplification trick*. We use a cryptographic context, but the algorithm is general and has many potential applications. For the key search problem, given a plaintext-ciphertext pair (PT-CT), we seek the key k^* so that $CT = E_{k^*}[PT]$.

A.2.2.2 The Oracle

Let $I(x)$: $x \in \rightarrow \{0, 1\}$ be the indicator function for the solution

$$I(k) = \begin{cases} 1 & if \quad CT = E_k[PT] \\ 0 & Otherwise \end{cases} \tag{A.33}$$

This function needs to be implemented in the quantum computer as a reversible function. Form a unitary operator U_k as a phase oracle (we suppress the ancillary $|-\rangle$ as it is unchanged by the oracle) by:

$$y = U_k|x\rangle = \begin{cases} -|x\rangle & if \quad CT = E_k[PT] \\ |x\rangle & Otherwise \end{cases} \tag{A.34}$$

$$= (-1)^{I(k)}|k\rangle \tag{A.35}$$

This operator is diagonal, and the diagonals are of magnitude one; it is therefore unitary and realizable on a quantum computer [7, 9].

We begin with n states and put them into the first Bell state: $|\Phi\rangle i = H|0\rangle i$ and then entangle these. The result is a superposition of all 2^n states equally likely. The oracle is applied at this point, and the result is a phase change of –1 if the state corresponds to a correct key (note that there could be more than one possible keys that work). If a measurement were to occur at this point, the states would be indistinguishable, as the phase change is not detectable.

A.2.2.3 Amplification about the Mean

Here is where a truly inspired thought came to Lou Grover: *amplification about the mean* or sometimes *inversion about the mean*. After the application of the phase oracle, the situation is that almost all the states are the same, and at least

one is different, namely one that corresponds to a key that sends the plaintext to ciphertext.

For any data, the difference between single-measurement and the mean is the deviation:

$$d_i = x_i - \bar{x} = x_i - \frac{1}{N}\sum_{j=1}^{N} x_i \tag{A.36}$$

And so the deviation vector, $X - \overline{X}$, can be written:

$$T = X - \bar{X} = \left(I - \frac{1}{N}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \vdots & \cdots & 1 \\ 1 & 1 & \cdots & 1 \end{bmatrix} \right) X \tag{A.37}$$

$$= \left(I - \frac{1}{N}\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \right) X \tag{A.38}$$

The matrix T is not unitary, so it cannot be realized as a quantum operation.

However, there is a related idea: rotate the state about a plane that is orthogonal to the uniform vector $(1/\sqrt{N}\,[1, 1, \cdots, 1]^T$ and this operator, D, is *unitary*. Now, the actual space we are dealing with is complex, and the maximum sameness state is superposition; that is $|+\rangle^{\otimes n}$. This operator is an example of a *Householder transformation.*[7]

The operator $\langle \mathrm{M}| = 1/N\,[1, 1, \cdots, 1]$ acting on a ket yields the average and hence the "mean" part of the name "inversion about the mean." The amplification is that the same states are slightly above the mean, and the different state, which is the desired key, is below the mean so that it is changed, while the same states have relatively less change.

A.2.2.4 The Algorithm

Thus, the solution to convert the difference in phase induced by the phase oracle is to rotate about the uniform state. The inversion acts as an amplification because it converts the phase difference into a scale difference. The whole algorithm is:

1. Start with n qubits in the Hadamard $|{+}{+}\cdots{+}\rangle$ state, call these the state, $|s\rangle$. This is the superposition of all possible n-bit keys.
2. Add in one more state, $|-\rangle$, as an ancillary bit for the inversion about the mean.
3. Apply the phase oracle, U_k to the state. This operation returns a superposition of whether or not each key value is valid.
4. Loop to apply inversion about the mean D. Each iteration slightly increases the magnitude of the correct answer.
5. Iterate for a total of $\pi/4\ \sqrt{2^n}$ times.
6. Measure the state: with a high probability this will be the correct state.
7. Repeat as necessary to validate the situation.

The probability of choosing the correct state can be shown to be much greater than 1/2 so at most a small number of applications is required. That number of applications does not affect the order of magnitude of the complexity.

A.2.2.5 Grover Summary

Grover's approach is known to be asymptotically optimal [23]. This result was known in 1999 before the competition that defined AES ended.

Caution: Not infrequently, claims will be made that Grover's algorithm can be used to solve a variety of problems beyond search. For instance, [18] makes claims about the applicability of Grover's algorithm to difficult problems like the traveling salesman problem. It is true that Grover can reduce the exponential time by a square root, but the result is still exponential. Most interpretations of "solving" a problem mean polynomial-time solutions, not exponential-time solutions. Grover's algorithm cannot turn an exponentially difficult problem into a polynomially difficult problem.

A.2.3 Shor's Algorithm and Impacts

In 1994 Peter Shor announced a stunning result at a conference in Santa Fe, New Mexico, which was later published in the *SIAM* journal in 1997; it is now known as Shor's algorithm [9, 11, 19, 20]. The idea capitalized on an old, well-known result in number theory, Euler's theorem (see Section 5.3) and the quantum version of the Fourier transform. The algorithm yields a very significant speed-up on factoring larger numbers, and as such it was the first practical application that showed the potential of quantum computing breaking cryptography, namely RSA. Shor extended the applicability of the approach to the hidden subgroup problem, thus including an attack on elliptic curve-based cryptography.

Unlike the situation in symmetric cryptography, where the mitigation is to increase the key size, the response to Shor's algorithm in cryptography can only be to select a new set of problems to base the algorithms on; that is, to not use the affected algorithms. Hence, the NIST announcement that they intend to standardize new key exchange protocols and digital signing method, based on NP-complete problems (see the next section).

The current classical attack on RSA, the general number field sieve [24], requires $O(e^{\sqrt[3]{N}})$ time to factor N. Shor's method factors in polynomial time by using a hybrid approach of classical and quantum computing, which represents a super-polynomial speed-up over the best-known classical factoring algorithm. Shor's algorithm is complicated, and we will provide a rough sketch, but the interested reader should consult any reference on quantum computing, for example [8, 9, 17].

A.2.3.1 A Classical Analog of Shor's Algorithm

Recall from Section 5.3 that given an integer N, then $m^{\varphi(N)} = 1 \pmod N$ for all $m \in \mathbb{Z}_N^*$, where $\varphi(N)$ is the Euler totient function. In general, the order of m mod N is the smallest k with $m^k = 1 \bmod N$. Observe that if $m^k = 1 \bmod N$ and k is even, then $\left(m^{\frac{k}{2}}+1\right)\left(m^{\frac{k}{2}}-1\right) = 0 \bmod N$ and so one of $\left(m^{\frac{k}{2}} \pm 1\right)$ shares a factor of N. This observation defines an (inefficient) factorization method. Let N be the number we wish to factor:

1. Choose a number, $m < N$.
2. Check to see if m shares a factor of N. If it does, reduce N accordingly, and continue.
3. Compute the order of m; that is, find the smallest k such that $m^k = 1 \bmod N$.
4. If k is odd, start over with a new m.
5. If k is even, then check to see which of $\left(m^{\frac{k}{2}} \pm 1\right) \bmod N$ shares a factor. Reduce N accordingly, and continue.

This is terribly inefficient, and the bottleneck is finding an index k such that $m^k = 1 \bmod N$. Finding such a k is the same as looking at the function $f(k) = m^k$ and finding values so that $f(j) = f(k)$. This period finding is equivalent to a birthday problem (see Section 4.1.3). The complexity of a hit is about $O(\sqrt{N})$, which is the efficiency of trying all the numbers less than $\sqrt{N} = 2^{\frac{n}{2}}$. This algorithm is less efficient than the number sieve.

Shor's insight is that finding the period of a discrete function, given a list of values (its graph), can be done with the inverse Fourier transform and superposition can give the all of the values at once in a quantum computing setting. The complication is that you don't get to read the spectrum off because any measurement is associated with collapsing state. And so, as in the case with Grover's algorithm, and just about every quantum computer calculation, we need to design the measurement so that the desired, or at least a useful, value is produced with a high probability.

A.2.3.2 The Quantum Fourier Transform

The classical Fourier transform of a finite sequence indexed $i = 0, .., N-1$ is obtained by multiplication of the sequence as a column vector by the $N \times N$ matrix $\Omega = [\omega^{\ell,m}]$, where

$$\Omega_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & & \vdots & \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix} \tag{A.39}$$

This matrix is unitary and, therefore, suitable for implementation on a quantum computer. The values $\omega^{(j)(k)}$ are subject to the usual periodic reduction $\omega^{jk} = \omega^{jk \bmod N}$, which we have not indicated to emphasize the unitary nature of the transformation. Notice that multiplication by Ω_{2n} can be made efficient by an observation of John Tukey [25]. The speedup of the fast Fourier transform (FFT) is $O(N \log N)$, compared to $O(N^2)$ for the naive matrix version. The trick is recursive matrix decomposition of (A.41) into two copies of Ω_n and diagonal multiplication followed by even-odd permutations.

The actual implementation of the *Quantum Fourier transform (QFT)* is even more efficient than the FFT, taking $O(\log N)$ operations [8, 9].

$$QFT\left(|X\rangle\right) = QFT\left(\sum_{j=0}^{N-1} \alpha_j |j\rangle\right) \tag{A.40}$$

$$= \sum_{j=0}^{N-1} \alpha_j QFT\left(|j\rangle\right) \tag{A.41}$$

Where

$$QFT\left(\left|j\right\rangle\right) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} \left|k\right\rangle \tag{A.42}$$

$$= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i \frac{jk}{N}} \left|k\right\rangle \tag{A.43}$$

When $N=2^v$ then $\frac{jk}{N} = k\frac{j}{2^v}$. If we $j < 2^v$ then $j = x_{v-1}2^{v-1} + x_{v-2}2^{v-2} + \ldots x_0$ and $\frac{j}{2^v} = 0.x_{v-1}x_{v-2}\ldots x_0$ is a binary number in [0, 1]. After much more tedious algebra, where one expands k as a binary representation and realize that the integer part, l, of anything in the exponent is $e^{i2\pi l} = 1$, the expression reduces to:

$$QFT\left(\left|j\right\rangle\right) = \frac{1}{\sqrt{2}}\left|0\right\rangle + e^{2\pi i 0.x_0} \left|1\right\rangle \otimes$$

$$\frac{1}{\sqrt{2}}\left|0\right\rangle + e^{2\pi i 0.x_1 x_0} \left|1\right\rangle \otimes$$

$$\frac{1}{\sqrt{2}}\left|0\right\rangle + e^{2\pi i 0.x_2 x_1 x} \left|1\right\rangle \otimes \ldots$$

$$\frac{1}{\sqrt{2}}\left|0\right\rangle + e^{2\pi i 0.x_v - 1 x_v - 2 \ldots x_0} \left|1\right\rangle \tag{A.44}$$

Unlike its classical counterpart, the output of the QFT *is not* the full set of measurements. Instead, the measurement is the probabilistic collapse of the state. The FFT operates $O(v2^v)$ operations, and the QFT operates $O(v)$ operations. But the output of the QFT only gives a single measurement, not all of the Fourier coefficients.

In order to get a good estimate of the period, m ancillary bits are used and the measurement is arranged so that the first m bits of the period are available.

To actually determine the period from the measurement, a classical computing algorithm, the ironically named quantum period finding algorithm, is used [7, 26].

In order to improve the accuracy of the quantum period finding estimate, the size of the QFT is increased substantially, and in total, $O(\nu^2 + 3\nu)$ qubits are required, which makes the algorithm $O(\nu^2)$. A number of attempts may be required to factor a number; the probability of success for each trial is greater than one-sixth. This includes getting a number of odd order, the failure of the QFT to yield a proper value, and so on. Thus, 10 trials yield better than a 99% probability of success.

A.3 Post-Quantum Cryptography

The biggest driver for change in cryptography in the early twenty-first century is the possibility of quantum computing. As we saw in the last section, two algorithms, Grover's and Shor's, pose the possibility of a real challenge to established methods. Fortunately, both of these challenges are being answered long before the threat has been realized. The threat to Grover's algorithm was quickly met with NIST's choice of AES and 256-bit key sizes. The challenge posed to asymmetric algorithms by Shor's algorithm has been met with the introduction of NIST's lattice-based methods recently (e.g., Kyber and Dilithium), although some additional niche issues remain, especially for bandlimited systems like satnav in the area of digital signatures [31]. There is promise in the constrained digital signature area, and at this time, multivariate-based signature schemes like unbalanced oil and vinegar are of interest. This section delves into the need for post-quantum cryptography, the theory that drives choosing post-quantum algorithms, and brief descriptions of some of those algorithms.

A.3.1 Symmetric Algorithms

The main threat posed by quantum computing to symmetric algorithms is searching with Grover's algorithm. It has been known for some time that Grover's is asymptotically optimal [23]. For the time being, it seems that quadratic speed-up afforded by Grover's algorithm will not be sufficient to allow actual attacks [14]. Nonetheless, it seems prudent to assume that it *could* occur and to double the key size. Thus, the use of AES256 seems like a simple standardized answer.

The one rub on this will be in constrained situations. AES256 uses 14 rounds and AES128 uses 10 rounds; that difference could be significant in energy density in *highly* energy-constrained systems. If the information has a short useful life, then it might be worth considering AES192. The key needs to be in use, and that includes the life of the data being protected, for much less time than it would take to break the key.

A.3.2 Asymmetric Cryptography

The threat to existing asymmetric algorithms is based on Shor's algorithm, which can factor integers and find hidden subgroups, the core feature of RSA and elliptic curve cryptography. While the threat from Shor's algorithm does not appear to be imminent, it is increasingly likely that it will be realized.

That potential threat of Shor's algorithm has led to the choosing of replacement public-key algorithms. NIST has selected two lattice-based algorithms [27], Dilithium [28] and Falcon [29], and one hash-based algorithm, Sphinc+ [30] as replacement signature schemes for RSA and elliptic curve signature (e.g., ECDSA). All of these algorithms have much longer signatures than are appropriate for signing in-band satnav messages [31]. We do not discuss hash-based approaches in this book; in general the signatures are larger for hash based signatures.

An additional round of NIST digital signature competition [32] has opened with one of the candidates, including several based on multivariate cryptography, that has the potential to have short signatures [33] at the expense of (very) large private signing and public verification keys.

A.3.3 Complexity Theory

The central question in this section is: What can a quantum computer do better than a classical computer? The branch of computer science devoted to such questions is complexity theory; see [8, 34, 35]. We distinguish between classical computers, based on Boolean logic, the ones we are all familiar with, and a quantum computer, which operates using different instructions using the quantum gates as defined in Section A.1. The goal of this section is to introduce language that allows us to discuss how quantum computers differ from classical computers in terms of the difficulty of the problems that can be solved.

We first establish that a quantum computer can compute everything a classical computer can. A classical computer has a small set of logical operations that define its capabilities. A *universal set of gates* are operations that any Boolean function can be constructed from [7–9]. Most representations of universal gates in a classical computer are not reversible: you cannot reconstruct the inputs from the output. For instance, the NOT and AND functions are universal for a classical computer [7]. There is a classical construction, the *Toffoli gate* that makes a classical Boolean function reversible, and thus is realizable in a quantum computer.

The Toffoli gate is a CNOT gate with two control bits (CCNOT). That is, if the two control bits are both set to 1, then the third value is inverted. Otherwise, there is no change. The Toffoli gate reversibly implements the NOT and AND functions, and thus implements any classical algorithm.

This fact means that a quantum computer is at least as capable as a classical computer. That is not to say that the efficiencies are comparable; limitations like those induced by the no-cloning theorem force radically different architectures on quantum computers. On the other hand, other features, such as entanglement, mean that quantum computers have the possibility of significantly more capability than a classical computer in some contexts, as we saw previously with Grover's and Shor's algorithms.

Next, we need to define how hard a problem is. A problem is said to be in *class P* if solving the problem takes polynomial resources. That is, the time for the solution grows polynomially with the size of the input; there are also suitable constraints on the other resources so that one cannot trade, say, exponential memory for faster time, and so on. The strict definition involves a deterministic *Turing machine* [7, 8, 34, 36]. A problem is in *class NP* if verifying a given solution to the problem is in class *P*; that is, it takes polynomial time to check the solution. Since if we can compute an answer, we can check it, class *P* is contained in class *NP* ($P \subseteq NP$, see Figure A.3). The nomenclature is because this definition is equivalent to the problem being solved by a nondeterministic Turning machine in polynomial time [34].

For the development of asymmetric algorithms, say for digital signatures, we need a problem that has the following characteristics:

1. The verification problem is easy (class *P*);
2. The forging problem is difficult (not class *P*);
3. The signing problem is easy (class *P*).

Items 1 and 2 imply the problem is class NP, and in order to get Item 3, there is usually a trapdoor where the signer picks parameters that make the signing easy, and by keeping them secret makes the forging problem difficult.

For example, the problem of determining the factors of a large number is sufficiently hard for a classical computer, although it is not known if factoring is

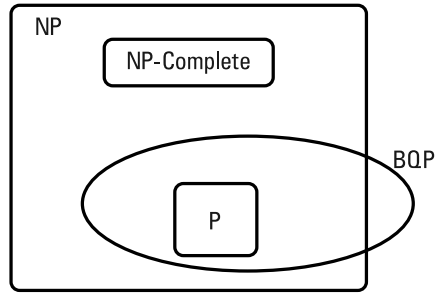

Figure A.3 Complexity class containment, assuming that $P \neq NP$, it is not known if $= BQP \subset NP$.

in class P the fastest known algorithm is in class super polynomial (*SP*) – growing faster than any polynomial, but less than exponentially. The number field sieve for factoring N grows like $e^{\sqrt[3]{N}}$ (see Section A.2.3). The RSA algorithm is based on the difficulty of factoring (see Section 5.4), but it is not sufficiently hard for a quantum computer (Shor's algorithm A.2.3). Likewise, elliptic curve based signatures, like ECDSA, are not adequate to withstand attack by a quantum computer (generalized Shor's algorithm).

One of the daunting things about complexity theory is the problem of class containment: when a complexity class D is contained in class C, $D \subseteq C$, we would like to say that C and D are different. But proving that the containment is strict, that is, that the classes are not equal seems to be very hard in many interesting cases. The above case of $P \subseteq NP$ is an example: it is not actually known if $P \neq NP$. In fact, this question is one of the Clay Institute's. "Millennium Problems" and *the* outstanding problem in theoretical computer science [37]. This question is usually phrased "Is $P = NP$?"

The final piece we need for classical computers is the notion of *complete*. We say that a problem, d, in a class C is C-complete if every problem $e \in C$ can be efficiently reduced to d. For instance, if d is NP-Complete, then any algorithm that solves d can be used to solve any problem in NP with a polynomial reduction as the measure of efficiency. That is, if you find a *polynomial algorithm* for an NP-complete problem, then $P = NP$. NP-complete problems are the usual intuitive examples of exponentially hard problems.

For quantum computers, we need a different class of problems: bounded error quantum polynomial (BQP). This complexity class contains problems that can be solved on a quantum computer in polynomial time with probability $\geq 1/3$. Note that given that bound, we can execute the solution polynomial times to drive the error probability to a very low value. Informally, BQP is the set of problems that can be easily solved on a quantum computer:

- It is known that $P \subset BQP$;
- It is not known if $BQP \subset NP$;
- If $P = NP$, then most problems are easy, and the way we look at hard cryptographic problems is wrong.

The relationships are illustrated in Figure A.3.

Assuming that $P \neq NP$, then NP-complete problems are good places to look for hard problems that can be used in cryptography. Another point worth considering is how hard problems are used in cryptography. The conjecture $P \neq NP$ is necessary, but to be practical, we need a *trapdoor function*. This is a function f that is easy to compute, $f \in P$, whose inverse is classically hard $f^{-1} \notin P$.

This statement gets amended to be: We seek a trapdoor function f that is easy to compute, $f \in P$, whose inverse is hard, $f^{-1} \notin$ BQP.

It is worth saying that an additional problem can emerge if we design a trapdoor so that we *think* solving the trapdoor is equivalent to solving an *NP*-complete problem: it may not be. The issue is often what we think is equivalent is *not actually equivalent.* It happens regularly in the initial design of asymmetric algorithms that there is another method to solve the trapdoor, which, while not solving the *NP*-complete problem, solves our trapdoor. In other words, trade-offs are often made to make the algorithms tractable, and these modifications may compromise the algorithm.

A.3.4 Code and Lattice-Based Cryptography

A *lattice* Λ is a discrete set of points in $\mathbb{R}^n$ closed under addition and multiplication by integers. It is formed from the basis: $\{b_1, b_2, \ldots b_n\}$ of n-dimensional vectors, that is, $b_i \in \mathbb{R}^n$ as column vectors. We assume that the vectors are independent. We then have $\Lambda = \{x : \mathrm{x} = \sum_{i=1}^{n} \alpha_i b_i$ for $\alpha_i \in \mathbb{Z}$.

Lattices have a natural problem associated with them, called the *shortest vector problem*. Given a lattice, the problem is to find the shortest vector in the lattice. This problem is believed to be hard in some cases [38]. The problem is illustrated in Figure A.4. The problem can be extended naturally to finding the closest lattice element to a given vector. In general, the theory of lattices is deep, and even in $\mathbb{R}^2$ there are connections to, for instance, elliptic functions [39, 40].

A.3.4.1 Code-Based Encryption

Lattices also occur in discrete spaces, such as GF(q^n)= $\mathbb{F}_q^n$, where they are usually called subspaces. Subspaces given by a set of basis vectors, not necessarily spanning the whole space, is a *linear code*. It is known that the problem of decoding a general linear code; that is, finding the closest codeword to a given vector, in $\mathbb{F}_q^n$ is NP-complete [41]. In 1978, Robert McEliece, working at the Jet Propulsion Laboratory, developed an asymmetric scheme based on the general intractability of decoding [42].

The trapdoor for this problem is to start with a tractable code, such as a Goppa (n, k) code [43], which can correct t-bits of error. Let the generator matrix for the code be $G \in \mathbb{F}_q^{k\times n}$. Create two random matrices, nonsingular $S \in \mathbb{F}_q^{k\times k}$ and and $P \in \mathbb{F}_q^{k\times n}$, a permutation matrix, and set $\hat{G} = SGP$. The private key is (G, S^{-1}, P^{-1}). The public key is the pair ($\hat{G}$, t).

To encrypt a message m, the vector $E(m)= c = m\hat{G} + e$ is computed; here, as is traditional in communication theory, m is a row vector. The row e represents t errors. This message can be decrypted because the original code has the desired error correction property. But it can only be decrypted by the holder of the private key.

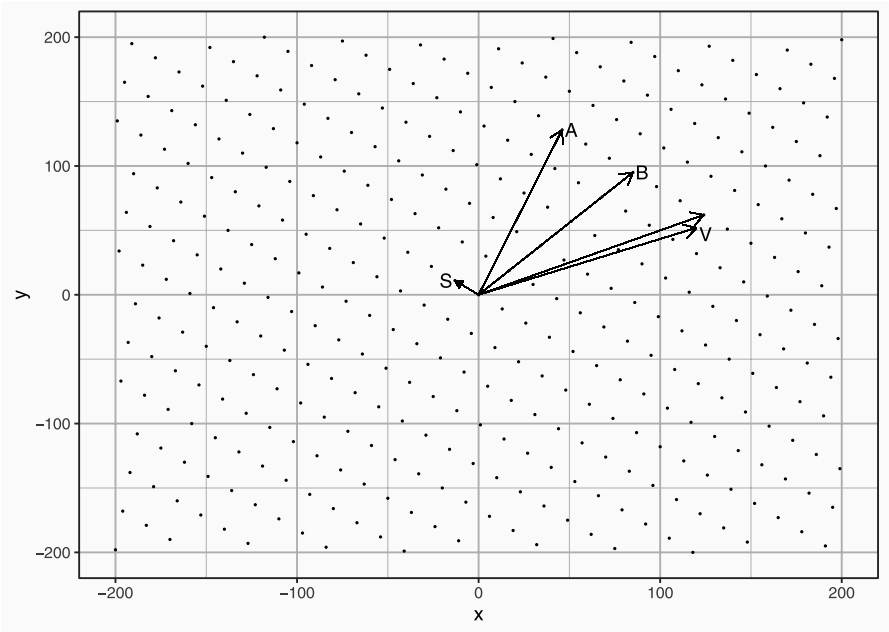

Figure A.4 A lattice in $\mathbb{R}^2$. The shortest vector (S) is [13, −11], and the basis vectors are A = [46, 118] and B = [85, 95]. An arbitrary vector V = [120, 52] is noted next to the closest vector on the lattice, which is at [124, 62].

Unfortunately, the sizes for the (public and private keys, signatures) in these schemes are very large [44].

A.3.4.2 Lattice Cryptography

We now turn to a general lattice Λ. In the cryptographic setting, the underlying space is based on finite fields, albeit large ones. The dimensions of the vector spaces are also large. This vastly complicates many problems, like finding the shortest vector or the closest lattice point to a vector not on the lattice.

In analogy with general error correction in algebraic coding theory, the problem of finding the shortest vector in a lattice is known to be difficult. If one asks for an approximate shortest vector, one can get within $2^{N/2}$ using the LLL algorithm [45] and do a bit better, although still exponential, using an algorithm due to C. P. Schnorr [46]. These bounds are as good as one can do [47]. These estimates are used to calculate the security parameters in developing lattice algorithms.

The essence is that the problem, $Ax = b$ is easy in $\mathbb{R}^n$ and when the underlying space is $\mathbb{Z}_p^n$. These can be solved, given A and b for x, via Gaussian elimination. The related problem $Ax + e = b$, where e is an error vector, easy in

$\mathbb{R}^n$, but hard in $\mathbb{Z}_p^n$. This becomes a lattice problem, because the solution set will be periodic.

Define $\Lambda^{\perp}(A)= \{z \in \mathbb{Z}_p^n | Az = 0\}$, then adding any vector in $\Lambda^{\perp}(A)$ to a solution of $Ax + e = b$.

The problem of finding $\{x, b\}$ is called the *learning with errors* problem. Of course, there are details, such as the probability distribution that the errors are drawn from, among others. One fact that makes this problem interesting is the result by Regev [48]: the complexity of the average-case problem is, within a polynomial factor, the same as the worst-case.

A.3.4.3 Dilithium

The NIST proposed standard for post-quantum digital signatures includes Dilithium [49]. The algorithm is in a finite ring, $R_q = (Z)_q[X^{256} + 1]$ where $q = 2^{23} - 2^{13} + 1$, which is prime, and just over 8 million.

To generate a key, choose a A a $k \times l$ matrix, in Dilithium 4×4, with entries in R_q. That size is still 1024×1024. Choose two secret vectors, s_1 and s_2 and compute $t = As_1 + s_2$, where all operations are in R_q. The values in s_1 and s_2 are restricted to small, less than ν. The public key is the pair (A, t). The secret key is (s_1, s_2). The signing operation is a bit more complex to explain.

The issue is to avoid leakage, and this is done by masking the computation in two ways. The first way is to mask the signature by transmitting the high-order bits of the computation that involves a masking vector y. The second way is to make a test on the high- and low-order bits of the computation, and if the two criteria are not met, then a new y is chosen. This latter method is *rejection sampling*.

The masking vector is chosen to have coefficients less than γ_1, and this is chosen strategically. It is large enough that the eventual signature does not reveal the secret key (i.e., the signing algorithm is zero-knowledge) yet small enough that the signature is not easily forged [49]. Then $w = Ay$ is decomposed into w_h high order bits of w and w_l, the low order bits of w. A c is generated from w_l and a hash of the message. The challenge c is selected with 60 coefficients ± 1, and the rest zero. And the signature candidate is $z = y + cs_1$, but using this would leak information about s_1. So, if cs_1 is too large, then a new y is chosen and the process repeats. If this test is passed, then the low-order bits of $Ax - ct$ are checked, and if they are too large, a new y is chosen. The various parameters are chosen so that the expected number of repetitions is between 4 and 7 [49]. The signature is then the pair (z, c); additionally, the size parameters ν and γ_1 are part of the algorithm's infrastructure.

The verifier computes $w' = Az - ct$ and accepts if all the coefficients of z are small involving γ_1 and if c is the hash of the message and w_1 are the high-order bits of w'.

Dilithium uses SHAKE-128, NIST's SHA-3 hash, to generate the entries of A using a seed ρ [28, 49], and an additional trick that reduces the public key by more than a factor of two, adding 150 bytes to the signature. The resulting public key is 1.3 KB, and the signatures are 2.4K bytes. If we compare this value to the size of A, which is $4 \times 4 \times 256 \times 3 \approx 12$ KB, the savings *are* substantial. Note that the representation of the matrix A is as a 4×4 element of R_q, but when written over Z^{256} that becomes a 1024×1024 matrix, and this matrix is the one to think of in terms of the lattice view.

Falcon [50], which stands for fast Fourier lattice-based compact signatures over NTRU is another lattice-based signature method based on the shortest integer solution (SIS) problem [51]. This approach yields a (minimum) signature of 666 bytes.

However, these numbers are far greater than could be used for in-band signing of navigation messages [31].

A.3.5 Multivariate Cryptography

The above lattice-inspired methods suffer from needing large digital signatures. For satnav, we need relatively small signatures to fit in data bandwidth-constrained messaging. One possible set of methods is based on the multivariate problem; unfortunately, this set is promising but, as of the writing of this book, not mature. Several proposed methods have been broken.

The problem of solving a system of multivariate quadratic (MQ) equations is known to be NP-complete [52]. An example multivariate quadratic polynomial in three variables is

$$= aX^2 + bY^2 + dXY + eXZ + fYZ + gX + hY + iZ + j \qquad \text{(A.45)}$$

where the indeterminates are X, Y, X, and the lowercase roman letters (a, b, ... j) are the coefficients from some field. The zeros of p would be a set of values for which $p(X, Y, Z) = 0$.

One version of this problem that has attracted much attention is the zeros of multivariate polynomial equations over a finite field. Even for the case of quadratic polynomials, this problem is NP-complete [52].

A.3.5.1 Unbalanced Oil and Vinegar

Recall the standard idea in asymmetric algorithms is to use a hard NP problem that can be verified but not solved. The signer can sign because they have access to a trapdoor.

The evaluation of a system of multivariate quadratic polynomials is easy, but solving the same system is NP-complete, in general. We thus could use a system of multivariate polynomials to verify a signature. But we need trapdoor

for the signer. The answer is to hide a linear problem inside the system of multivariate polynomials. The trapdoor is how the linear problem is embedded into the quadratic system.

The signer publishes a system of, hard-to-solve, multivariate polynomials *S*; this is the public key. When the signer wants to sign a document, *M*, they will supply a set of values $\overline{X}$, for which

$$0 = S(\overline{X}) - \text{vec}(\text{hash}(M)) \tag{A.46}$$

Here vec(hash(*M*)) takes the hash into a vector of the same dimension as the system (e.g., turns it into a byte or word vector).

The verifier can check this easily; it is just an evaluation of a polynomial system. Given S and a document, the verifier computes vec(hash(*M*)) and verifies (A.51). If the equation holds, then the document is taken as authentic.

The question then is how to form a signature? We need a trapdoor that is parametric, so it can be reused, and it has to be at least as hard to solve for the trapdoor as it is to solve the original problem. This last requirement is why various attempts to create the trapdoor have failed.

In this structure, problems that can be solved are linear ones; after all, we can use Gaussian elimination to solve a linear set of equations. The trapdoor is then how one disguises a multivariate linear problem as a multivariate quadratic problem.

In the case of unbalanced oil and vinegar (UOV), a set of quadratic equations is used with a special property. There will be two variable sets, vinegar, which appears without restrictions in the equations, and oil variables that don't appear in any expression with other oil variables. So, the vinegar variables mix, and the oil variables do not. The result is that the equations are quadratic in the vinegar variables and bi-linear in the oil variables. The effect of this construction is that if one does a partial evaluation of the system *P*, on the vinegar variable, the resulting system, $P|_v$ is a linear system in the oil variables. So, we can solve the system.

We will need to change variables from the set [*v*, *o*] to a set, $[x_1, ..., x_{o+v}]$ where the new equations are quadratic in all variables. The new system becomes S, the verification polynomials. The signer will need to keep this transformation secret, and the original set of equations is also secret. These form the signer's key.

The public key is the transformed equations, S, and the signature is $[\overline{x}_1, ..., \overline{x}_{o+v}]$ where $S(\overline{x}_1, ..., \overline{x}_{o+v})$ – vec(hash(*M*)) So, the forger is confronted with a multivariate quadratic system.

The obvious thing to attack is to find the underlying linear system rather than solve the MQ problem. And as several submissions to the first NIST post-quantum competition on digital signatures have unfortunately shown, it

is possible to make a mistake here. There were two algorithms, GeMSS and Rainbow, that made it to the third round and were broken there. The attacks have so far left the UOV untouched.

Interestingly, the first proposal, in 1988, along these lines, called C^* or Matsumoto-Imai, was broken in 1995, and a counterproposal was made (oil and vinegar (OV)) in 1997, and that was broken in 1998, too. The third attempt in 1998 by Kipnis et al. [53] has stood the test of time, which is UOV.

The unbalanced comes as the algorithm has a different number of vinegar variables (v) from the number of oil variables. Thus, the variables are unbalanced. They were equal ($o = v$), in the first proposal (OV). It seems that secure values have $v \approx 2o$ or v $\approx 3o$.

Multivariate methods are attractive for satnav because the signatures are relatively small. That convenience comes at the price of very large keys, both private and public. There have been several attempts to modify UOV, to make the key sizes smaller, and so on, but most of them have been broken. Will something like UOV work for satnav? We think so.

A.4 Takeaways: Quantum and Post-Quantum

Quantum computing has the potential to be revolutionary in its ability to solve *some* problems much faster than classical computers. It is out of the scope of this book to touch on noncryptographic uses, such as in optimization and machine learning. Its impact to cryptography is given by Grover's and Shor's algorithms. Grover's impact to symmetric ciphers can be mitigated by using large enough keys, such as, AES256. Shor's impact to asymmetric algorithms is more serious, since Shor's algorithm breaks in polynomial time the widely used methods of RSA and elliptic curve cryptography.

The solution for Shor's impact is found in post-quantum methods. Lattice-based methods (and hash-based, which were not described here) are currently being standardized. Multivariate methods are more applicable to the needs of satnav, but as yet are not mature.

The question that always arises is when will quantum computing become a reality? The technical challenges are severe and the field is filled with both optimism and pessimism. Still, because of the long lead time to update public standards (e.g., for the internet), the emphasis to be prepared with post-quantum algorithms is both ongoing and justified.

End Notes

1. There are classical computing systems that stretch the two-state model, such as *fuzzy systems* [54] and random algorithms [55]. However, implementations of these are deter-

ministic in the sense that there is a single defined—although possibly varying value. This situation is not the case with qubits.

2. One can construct a qubit out of a fermion or a boson; light is a boson. For more information on physical qubits, we suggest [18]; and for more programming introduction we recommend [17].

3. An alternative view, developed by E. Schrödinger is based in functional analysis, and is called the Schrödinger picture. These two views were reconciled by realizing that they were equivalent representations. The models are both representations of *Hilbert spaces* and are related in the same fashion that $L_2[-\pi, \pi]$, the set of square integral functions on $[-\pi, \pi]$ and l_2, the set of square summable sequences are the same (map functions to their Fourier coefficients, etc.).

4. The requirement that quantum computer operations be given by unitary operations comes from quantum mechanics [8, 18, 56]. Very briefly, the evolution of a quantum system is given as a solution to Shrödeinger's equation:

$$i\hbar \frac{\partial}{\partial t}\left|\Psi\right\rangle = \hat{H}\left|\Psi\right\rangle \Psi(0,x)$$

Where $\hat{H}$ is a Hermitian operator, which is the *Hamiltonian* for the system (usually a generalization of energy). The formal solution to this equation is:

$$\left|\Psi(t,x)\right\rangle\rangle = e^{\frac{i}{\hbar}\hat{H}(t)}\left|\Psi(0,x)\right\rangle$$

which is a generalization of the variations of constants formula [57, 58]. The exponential of a Hermitian operator is a unitary operator [56, 58].

5. The quote seems a translation from a letter to Max Born in the 1930s. It is a pithy objection and consistent with the cited paper. The 2022 Nobel Prize in physics was awarded to a trio of physicist "for experiments with entangled photons, establishing the violation of Bell inequalities and pioneering quantum information science" [59]; that is, establishing that entanglement is a real phenomena. The first demonstration was in the mid-1970s.

6. The state $|\Phi^{+}\rangle$ is one of four maximally entangled Bell state states which are

$$\Phi^{+} = \frac{1}{\sqrt{2}}\left(\left|00\right\rangle + \left|11\right\rangle\right)$$
$$\Phi^{-} = \frac{1}{\sqrt{2}}\left(\left|00\right\rangle + \left|11\right\rangle\right)$$
$$\Psi^{+} = \frac{1}{\sqrt{2}}\left(\left|01\right\rangle + \left|10\right\rangle\right)$$
$$\Psi^{-} = \frac{1}{\sqrt{2}}\left(\left|01\right\rangle + \left|10\right\rangle\right)$$

7. A *Householder transformation* is a reflection about a hyper-plane orthogonal to a given unit vector v:

$$R_v = I - 2vv^\dagger$$

They are used in a variety of signal-processing algorithms.

References

[1] Shannon, C. E., "A Symbolic Analysis of Relay and Switching Circuits," Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1940, https://dspace.mit.edu/handle/1721.1/11173.

[2] Feynman, R. P., "Simulating Physics with Computers," *International Journal of Theoretical Physics,* Vol. 21, No. 6, June 1982, pp. 467–488, doi:10.1007/BF02650179.

[3] Montanaro, A., "Quantum Algorithms: An Overview," *npj Quantum Information 2.1,* Jan. 12, 2016, p. 15023, doi:10.1038/npjqi.2015.23, http://arxiv.org/abs/1511.04206.

[4] Jordan, S., *Quantum Algorithm Zoo,* June 26, 2022, https://quantumalgorithmzoo. org/.

[5] "NIST Announces First Four Quantum-Resistant Cryptographic Algorithms," *NIST,* July 5, 2022, https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms.

[6] Feynman, R., *The Character of Physical Law,* MIT Press, 1996.

[7] Wong, T. G., *Introduction to Classical and Quantum Computing*, Omaha, NE: Rooted Grove, 2022.

[8] Nielsen, M. A., and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition,* Cambridge, UK: Cambridge University Press, 2011.

[9] Mermin, N. D., *Quantum Computer Science: An Introduction,* Cambridge, UK: Cambridge University Press, 2007.

[10] Romero, J., and G. Milburn, *Photonic Quantum Computing,* April 4, 2024, doi:10.48550/arXiv.2404.03367, arXiv:2404.03367[quant-ph].

[11] Rieffel, E., and W. Polak, *Quantum Computing: A Gentle Introduction,* Cambridge, MA: The MIT Press, 2011.

[12] Einstein, A., B. Podolsky, and N. Rosen, "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?" *Physical Review,* Vol. 47, No. 10, May 1935, pp. 777–780, doi:10.1103/PhysRev.47.777, https://link.aps.org/doi/10.1103/PhysRev.47.777.

[13] Scherer, W., *Mathematics of Quantum Computing: An Introduction,* Cham, Switzerland: Springer, 2019.

[14] Babbush, R., et al., "Focus Beyond Quadratic Speedups for Error-Corrected Quantum Advantage," *PRX Quantum,* Vol. 2, No. 1, March 2021, p. 010103, doi:10.1103/PRXQuantum.2.010103, https://link.aps.org/doi/10.1103/PRXQuantum.2.010103.

[15] Deutsch, D., "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer," in *Proceedings of the Royal Society of London A: Mathematical and Physical Sciences,* Vol. 400, No. 1818, July 1985, pp. 97–117, doi:10.1098/rspa.1985.0070, https://royalsocietypublishing.org/doi/10.1098/rspa.1985.0070.

[16] Deutsch, D., and R. Jozsa, "Rapid Solution of Problems by Quantum Computation," in *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences,* Vol. 439, No. 1907, 1992, pp. 553–558, doi:10.1098/rspa.1992.0167, https://royalsocietypublishing.org/doi/10.1098/rspa.1992.0167.

[17] Wong, H. Y., *Introduction to Quantum Computing: From a Layperson to a Programmer in 30 Steps,* Second Edition, Cham, Switzerland: Springer, 2023.

[18] LaPierre, R., *Introduction to Quantum Computing,* Cham Switzerland: Springer, 2021.

[19] Shor, P. W., "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science,* 1994, pp. 124–134, doi:10.1109/SFCS.1994.365700.

[20] Shor, P. W., "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing,* Vol. 26, No. 5, October 1997, pp. 1484–1509, doi:10.1137/S0097539795293172, arXiv:quant-ph/9508027, http://arxiv.org/abs/quant-ph/9508027.

[21] Grover, L. K., *A Fast Quantum Mechanical Algorithm for Database Search,* arXiv.org, May 29, 1996, doi:10.1145/237814.237866, https://arxiv.org/abs/quantph/9605043v3.

[22] Information Technology Laboratory Computer Security Division, *AES Development-Cryptographic Standards and Guidelines—CSRC,* CSRC–NIST, December 29, 2016, https://csrc.nist.gov/Projects/Cryptographic-Standards-andGuidelines/Archived-Crypto-Projects/AES-Development.

[23] Zalka, C., "Grover's Quantum Searching Algorithm Is Optimal," *Physical Review A,* Vol. 60, No. 4, October 1, 1999, pp. 2746–2751, doi:10.1103/PhysRevA.60.2746, arXiv:quant-ph/9711070, http://arxiv.org/abs/ quant-ph/9711070.

[24] Case, M. A., "A Beginner's Guide to the General Number Field Sieve," Oregon State University, ECE575 Data Security and Cryptography Project, 2003, https://www.semanticscholar.org/paper/A-Beginner-%E2%80%99-s-Guide-To-The-General-Number-Field-Case/d314adec0df02a073e0c77c050defa2232f535d4.

[25] Cooley, J. W., and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation,* Vol. 19, No. 90, 1965, pp. 297–301.

[26] Lipton, R. J., and K. W. Regan, *Introduction to Quantum Algorithms via Linear Algebra,* Second Edition, Cambridge, MA: The MIT Press, 2021.

[27] Information Technology Laboratory NIST, *Selected Algorithms 2022–Post-Quantum Cryptography—CSRC,* CSRC–NIST, January 3, 2017, https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[28] Bai, S., et al., "Crystals-Dilithium," 2021.

[29] Fouque, P.- A., et al., "Falcon: Fast-Fourier Lattice-Based Compact Signatures over NTRU," Submission to the NIST's Post-Quantum Cryptography Standardization Process, 36, No. 5, 2018, pp. 1–75.

[30] Schwabe, P., *SPHINCS+*, https://sphincs.org/.

[31] Mendiola, M. A., et al., "Post-Quantum Authentication Schemes," in *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020),* September 25, 2020, pp. 3812–3825, doi:10.33012/2020.17747, http://www.ion.org/publications/abstract.cfm?jp=p&articleID=17747.

[32] Information Technology Laboratory NIST, *Post-Quantum Cryptography: Digital Signature Schemes—CSRC,* CSRC–NIST, August 29, 2022, https://csrc.nist.gov/projects/pqc-dig-sig.

[33] Information Technology Laboratory NIST, *Round 1 Additional Signatures-Post-Quantum Cryptography: Digital Signature Schemes—CSRC,* CSRC–NIST, August 29, 2022, https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[34] Sipser, M., *Introduction to the Theory of Computation,* International Thomson Publishing, 1996.

[35] Mao, W., *Modern Cryptography: Theory and Practice,* Upper Saddle River, NJ: Prentice Hall, Hewlett-Packard Professional Books, 2004.

[36] Molina, A., and J. Watrous, "Revisiting the Simulation of Quantum Turing Machines by Quantum Circuits," in *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences,* Vol. 475, No. 2226, June 2019, p. 20180767, doi:10.1098/rspa.2018.0767, arXiv:1808.01701 [quant-ph].

[37] *P vs NP*, Clay Mathematics Institute, July 27, 2024, https://www.claymath.org/millennium/p-vs-np/.

[38] Bernstein, D. J., J. Buchmann, and E. Dahmen (eds.), *Post-Quantum Cryptography,* Berlin: Springer, 2008.

[39] Silverman, J. H., "An Introduction to the Theory of Elliptic Curves," Brown University and NTRU Cryptosystems, Inc., Summer School on Computational Number Theory and Applications to Cryptography, University of Wyoming, June 19–July 7, 2006.

[40] Silverman, J. H., *The Arithmetic of Elliptic Curves,* Vol. 106, New York: Springer, 2009.

[41] Berlekamp, E., R. McEliece, and H. van Tilborg, "On the Inherent Intractability of Certain Coding Problems (Corresp.)," *IEEE Transactions on Information Theory,* Vol. 24, No. 3, May 1978, pp. 384–386, doi:10.1109/TIT.1978.1055873, https://ieeexplore.ieee.org/document/1055873.

[42] McEliece, R. J., "A Public-Key Cryptosystem Based on Algebraic Coding Theory," *Deep Space Network Progress Report,* Vol. 44, January 1, 1978, pp. 114–116, https://ui.adsabs.harvard.edu/abs/1978DSNPR.44.114M.

[43] Huffman, W. C., and V. Pless, *Fundamentals of Error-Correcting Codes*, Cambridge, UK: Cambridge University Press, 2003, doi:10.1017/CBO9780511807077, https://www.cambridge.org/core/product/identifier/9780511807077/type/book.

[44] Bernstein, D. J., "Grover vs. McEliece," in *Post-Quantum Cryptography* (N. Sendrier, ed.), Vol. 6061, Berlin: Springer Berlin Heidelberg, 2010, pp. 73–80, doi:10.1007/978-3-642-12929-2_6, http://link.springer. com/10.1007/978-3-642-12929-2_6.

[45] Nguyen, P. Q., and B. Vallee (eds.), *The LLL Algorithm: Survey and Applications,* Information Security and Cryptography, Berlin: Springer Berlin Heidelberg, 2010, doi:10.1007/978-3-642-02295-1, https://link.springer.com/10.1007/978-3-642-02295-1.

[46] Goldreich, O., and S. Goldwasser, "On the Limits of Nonapproximability of Lattice Problems," *Journal of Computer and System Sciences,* Vol. 60, No. 3, June , 2000, pp. 540–563, doi:10.1006/jcss.1999.1686.url:https://www.sciencedirect.com/science/article/pii/S0022000099916860.

[47] Ajtai, M., "Optimal Lower Bounds for the Korkine-Zolotareff Parameters of a Lattice and for Schnorr's Algorithm for the Shortest Vector Problem," *Theory of Computing,* Vol. 4, No. 1, 2008, pp. 21–51, doi:10.4086/toc.2008.v004a002, https://theoryofcomputing.org/articles/v004a002.

[48] Regev, O., "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography," Vol. 56, No. 6, Article 34 , May 5, 2009, doi:10.48550/arXiv.2401.03703, arXiv:2401.03703[quant-ph].

[49] Ducas, L., et al., "Crystals–Dilithium: Digital Signatures from Module Lattices," in *IACR Transactions on Cryptographic Hardware and Embedded Systems,* Vol. 2018, No. 1, pp. 238–268, https://eprint.iacr .org/2017/633.

[50] Falcon, https://falcon-sign.info/.

[51] Gentry, C., C. Peikert, and V. Vaikuntanathan, "Trapdoors for Hard Lattices and New Cryptographic Constructions," in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, 2008, pp. 197–206, https://eprint.iacr.org/2007/432.

[52] Bellini, E., et al., *An Estimator for the Hardness of the MQ Problem*, *International Conference on Cryptology in Africa,* Cham, Switzerland: Springer Nature, 2022, pp. 323–347, https://eprint.iacr.org/2022/708.

[53] Kipnis, A., J. Patarin, and L. Goubin, "Unbalanced Oil and Vinegar Signature Schemes," in *Advances in Cryptology—EUROCRYPT '99* (J. Stern, ed.), Vol. 1592, Lecture Notes in Computer Science, Berlin: Springer, 1999, pp. 206–222, doi:10.1007/3-540-48910-X_15, http://link.springer.com/10.1007/3-540-48910-X_15.

[54] Zadeh, L. A., and R. A. Aliev, *Fuzzy Logic Theory and Applications,* World Scientific, 2018, doi:10.1142/10936.

[55] Mitzenmacher, M., and E. Upfal, *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*, Second Edition, Cambridge, UK: Cambridge University Press, 2017.

[56] Sharkey, K. L., A. Chance, and A. Khan, *Quantum Chemistry and Computing for the Curious: Illustrated with Python and Qiskit® Code,* Birmingham, UK: Packt Publishing, 2022.

[57] Rehmann, U., *Variation of Constants–Encyclopedia of Mathematics,* https://encyclopediaofmath.org/wiki/Variation_of_constants.

[58] Balakrishnan, A. V., *Applied Functional Analysis,* Second Edition, Springer Science Group, 2020.

[59] The Nobel Prize in Physics 2022, https://www.nobelprize.org/prizes/physics/2022/summary/.

About the Authors

Joseph J. Rushanan joined the MITRE Corporation in 1986 and worked on a wide variety of problems in his first years, including published results in binary sequences, parallel processing, and wireless and secure communication. The latter work includes a joint patent on using atmospheric turbulence to generate random keys. He is currently a principal mathematician.

In 1998, he began working on GPS-related projects. His contributions include being part of the M-code signal design team and the L1C signal design team. For the latter, he invented the spreading codes for the L1C signal, the so-called Weil Codes and was codesigner for the TMBOC modulation on the L1C signal. He currently leads the planning for the cryptographic and authentication experiments for the Navigation Technology Satellite - 3 (NTS-3) program, which includes maintaining the two associated Chimera specifications. Besides cryptography, he has been doing research in quantifying PNT assurance. He received the 2019 Captain P.V.H. Weems Award from the Institute of Navigation for "sustained contributions to the design of GPS, including M-Code, the L1C signal, and the promotion of assurance concepts for all GPS users."

Dr. Rushanan has been an adjunct lecturer at Northeastern University's Khoury College of Computer Sciences since 2012, where he teaches classes in applied cryptography in the cybersecurity graduate program. He has previously taught theoretical computer science classes at Northeastern University and was an adjunct mathematics professor at (then) Bentley College.

He received a BS/MS in mathematics from the Ohio State University in 1982 and a PhD in mathematics from the California Institute of Technology in 1986.

James T. Gillis joined Hughes Aircraft Co.'s Space and Communications Group in 1979, writing operating systems for real-time networking on supercomputers. His next assignment was in the high-bay writing test support programs for thermal-vacuum testing of spacecraft.

In 1983, he joined the Controls Analysis Department at The Aerospace Corporation, where he first encountered GPS, working on the Kalman Filter. In 1994, he was part of the Delta-Clipper team, which won the Space Frontier Foundation's Vision to Reality award. He led several independent assurance teams for NASA programs, including NSCAT and Cassini.

In 1998, he joined the GPS program office and worked on the GPS Selective Availability Anti-Spoofing Module integration team. He was cochair of the M-Code security design team. In 2004, he received a program recognition award as part of a team supporting Operation Iraqi Freedom. In 2007, he led a team that won an Aerospace Corp. President's Award for resolving issues with the GPS IIF crypto ASIC.

In 2008, he moved to what is now the Data Science and Artificial Intelligence department at the Aerospace Corp as a senior project leader, where he now works. He has been involved with space receiver design and applications and analyzing GPS space service volume. His interests include cryptography, authenticated space-based navigation, quantum computing, physics-informed machine learning, and the Navigation Technology-3 experiments.

He received a BS in system science and mathematics from Washington University in St. Louis in 1979 and an MS in system science from the University of California Los Angeles (UCLA) in 1983. In 1986, he began an Aerospace Corp. Fellowship and was awarded a PhD in electrical engineering from UCLA in 1988. In 1991, he was a visiting research associate at the University of Maryland's Institute for Systems Research.

Index

Absorbing, 83
Add-rotate-XOR ciphers, 63
Add round key, 62
Advanced Encryption Standard (AES)
  about, 59–60
  cipher, choosing, 60
  execution time, 62
  key size and rounds, 60–62
  performance, 62–63
  structure, 60–62
AES-GCM, 92
Ascon, 93
Assisted Commercial Authentication Service (ACAS), 256, 270
Assurance. *See* PNT assurance
Asymmetric cryptography. *See* Public key cryptography
Atomic, consistent, isolated, and durable (ACID), 184
ATT&ACK, 187
Attacks
  on ciphers, 51–52
  DH, 130–31
  DSA, 124–25
  RSA, 121, 300
Audience, this book, 4–5
Authenticated ciphers, 93
Authenticated encryption with associated data (AEAD), 90–91, 93
Authentication
  ciphers and, 89–94
  digital signatures, 113
  identity and, 7–10
  link layer, 162
  location, 276–77
  multifactor, 148
Authentication error rate (AER), 211
Authentication Header (AH), 163
Authentication server (AS), 157
Automatic Dependent Surveillance-Broadcast (ADS-B), 198
Availability
  about, 6, 7
  cellular enterprise perspective, 168
  PNT assurance and, 9, 161
  satnav signals, 194
  as security goal, 162, 163
  sensors and, 184
Avalanche property, 54–55, 82

Baseline Chimera
  about, 227, 228
  binding of data and signal, 259
  data, 260
  fast channel, 261
  message bits, 228
  messages to be signed, 229
  overview, 259–63
  performance, 229
  slow channel, 260–61
  See also Chimera
Beidou, 17, 28
Binary carrier offset (BOC), 30
Binary phase shift keying (BPSK), 30

Bit commitment
about, 78
cryptographic hashes, 98–99
hybrid protection, 257–58
spreading codes, 240–41
TESLA and, 219
Blinding, 132
Blockchains, 97–98, 279
Block ciphers, 47
"Blue Books," 177–78
Bounded error quantum polynomial (BQP), 306–7
Breaking ciphers, 45
Brute force, 52
Byte substitution, 60

Caesar cipher, 44–45
CBC-MAC, 90–91
CCNOT gate, 304
CCSDS Packet Transfer Protocol, 182
Celestial navigation, 281
Certificate authority (CA), 134
Certificate revocation list (CRL), 135
Certificates, 135–36
ChaCha20, 63–64
Challenge-response, 148–49
Chimera
about, 186, 227
baseline, 227, 228–29, 259–63
fast channel, 261–63
NMA on, 227–28
SAS versus, 271
slow channel, 260–61
TESLA, 227, 229–31, 263–66
Chimera markers
about, 245–46
creating, 249
cryptographic processing, 247–49
L1C spreading code structure, 246
overview, 246–47
performance, 249
plaintext for, 248
Chip offset, 34
Chips, 244–45
Chip shape signaling, 243–45
Chip state modulation (CSM), 244
Chosen plaintext, 51
Cipher block chaining (CBC)
about, 65, 69
choice of, 70
decryption, 69, 70
encryption, 69
pros and cons, 69–70
See also CBC-MAC
Cipher modes
about, 65
cipher block chaining (CBC), 65, 69–70
counter mode (CTR), 65, 67–69, 70
electronic codebook (ECB), 65, 66–67, 70
overview, 65–66
which to use, 70
Ciphers
about, 43
add-rotate-XOR, 63
AES, 59–63
attacks on, 51–52
authentication and, 89–93
avalanche property, 54–55
block, 47
breaking, 45
Caesar, 44–45
ChaCha20, 63–64
classic, 44
confidentiality and, 50–53
desired properties of, 48–55
Feistel construction, 57–58
general principles for creating, 55–59
information theory and, 49–50
lightweight, 63–64
overview of, 46–48
performance, 59
polyalphabetic, 46
primary goal of, 50–51
pseudorandom generation and, 54
Simon and Speck, 64
stream, 47–48, 59
substitution, 45–46
substitution-permutation construction, 56–57
takeaways, 72
transposition, 46
Ciphertext
about, 46, 51
duty factor and, 247
as markers, 242–43
as spreading codes, 241–42

Classical oracle, 294
Class NP, 305
Clock bias, 18
Cloning, no, 287, 293
Code-based encryption, 307–8
Code division multiple access (CDMA)
about, 29
codes, 29–30, 31
layer, 30
physical layer and, 160–61
signal example, 33
Collisions, 80
Common Vulnerabilities and Exposures (CVE), 187
Communication networks, 159
Competition for Authenticated Encryption Security, Applicability, and Robustness (CAESAR), 93
Completeness, 150
Complexity theory, 304–7
Confidentiality
about, 6
ciphers and, 50–53
exclusivity and, 6–7
link layer, 162
PNT assurance and, 9
Confidentiality, integrity, and availability (CIA), 6
Confusion, 55
Constellations, 27
Construction equipment security and registration (CESAR), 194
Continuity, 7, 9
Control network, 24
Control segment
about, 15, 24
information, 24–25
location of components, 23
in sample enterprise, 187–88, 190–91
in satnav infrastructure, 183
security goals for, 25
Correlation process, 31–36
Counter mode (CTR), 65, 67–69, 70
Cryptoanalysis, 51
Cryptocurrency, 98
Cryptographic hashes
about, 78
birthday match, 80–82
bit commitment application, 98–99
chains, trees, and blockchain application, 96–98
collisions, 80
construction, 82–87
digital fingerprints application, 94–95
message authentication codes (MACs), 87–89
preimage resistance, 79
properties, 79
TESLA application, 99–102
Cryptographic identification, 148–50
Cryptographic protocols
about, 143–44
ensuring trust and, 6
goal of, 144
identity methods, 147–52
key management, 152–58
need for trust and, 145
network stack, 158–67
principles, 144–47
risk and, 146
trust in, 147
Cryptography
about, 4, 10
asymmetric, 10, 109, 304
lattice, 308–9
location in, 277–78
multivariate, 310–12
post-quantum, 303–12
public key (asymmetric), 10, 109–40
satnav enterprise and, 177–202
spreading codes and, 238–39, 241–45
symmetric, 10, 43–72
taxonomy, 11
CRYSTALS-Dilithium, 286
CRYSTALS-Kyber, 286

Database, 22–23, 24
Data Encryption Standard (DES), 55
Datagram TLS (DTLS), 166
Davies-Meyer compression function construction, 84
Decryption
about, 46–47
location and, 278
RSA, 119
See also Encryption
Deutsch-Joza problem, 294
Deutsch's algorithm, 294–96

Deutsch's problem, 294–96
Dictionary attack, 95
Diffie-Hellman key exchange
about, 109, 128–29
elliptic curve methods, 131
implications and attacks, 130–31
procedure, 129
protocol, 129–30
sizes, 131
strengths, 131
Diffusion, 55, 115
Digital fingerprints application, 94–95
Digital signature algorithm (DSA)
about, 109, 122
definition, 123–24
elliptic curve methods, 125–27
implications and attacks, 124–25
justification, 124
performance, 128
signature generation, 123
signature verification, 124
strengths, 127–28
Dilithium, 309–10
Dilution of precision (DOP), 27
Distance function, 17
Double signatures, 132–33
Duty factor, 243, 246–47

Electronic attack (EA), 194
Electronic codebook (ECB), 65, 66–67, 70
Elliptic curve DSA (ECDSA), 127–28, 134, 140, 150–51, 225
Elliptic curve methods, 125–27, 131
Encapsulating Security Payload (ESP), 163–64
Encryption
about, 46
code-based, 307–8
homomorphic, 278–79
location and, 278
public key cryptography, 111–12
RSA, 118
Engineering, control segment, 24
Enigma machine, 45
Entanglement
about, 287, 289–90
H-Gate, 292
Not gate, 291–92
"spooky action at a distance" and, 290
state construction, 290–91
See also Quantum computing
Ephemeris service, 189, 192
Euclid's algorithm, 116
Euler's totient theorem, 116, 117
European Geostationary Navigation Overlay Service (EGNOS), 17
Exclusive-ors (XORs), 47–48
Exclusivity, 6–7

Fast channel Chimera, 261–63
Fast Fourier Transform (FFT), 301, 302
Feistel construction, 57–58
Fermat's little theorem, 117
Feynman, Richard, 285, 286
Filtering, 22
Frequency division multiple access (FDMA), 29, 160–61
Frequency hopping, 161
Future trends, 281–82

Galileo OSNMA
about, 209, 221
cryptographic methods, 223–26
functionality, 223
future plans, 227
MACs, 226
overall architecture, 222–23
overview, 221–22
performance, 226
public keys and Merkle Tree, 224–25
signing TESLA root key, 225
TESLA chain, 225–26
TESLA scheme, 222
Galois counter mode (GCM), 90–91, 92–93
Generator of a prime, 116–17
GEODNET, 279
Global Navigation Satellite System (GNSS)
about, 3, 17
confidentiality, 184
in EA prevention, 194
signal broadcasting, 31–32
Global Positioning System (GPS), 3, 23, 28
GLONASS, 17, 28, 29
Ground antenna, 188, 191
Ground segment
about, 15
control segment, 15, 24–25, 183
mission segment, 15, 22–24, 183–84
security architecture in, 184

Grover's algorithm
about, 285–86, 296
algorithm, 298–99
amplification about the mean, 297–98
oracle, 297
overview, 297
summary, 299
See also Quantum computing algorithms

Hadamard matrix, 288–89
Hash-based message authentication codes (HMACs), 89
Hash chains, 96
Hashes, 77–79, 94
Hash function construction
about, 82–84
Davies-Meyer construction, 84
Merkle-Damgård construction, 82–83, 84
SHA-2 family and, 84–86
SHA-3 and, 83, 84, 86–87
sponge construction, 83–84, 86–87
Hash functions
about, 77
avalanche property, 82
birthday match, 80–82
bit commitment application, 98–99
chains, trees, and blockchain application, 96–98
collisions, 80
digital fingerprints application, 94–95
goals, 78–82
illustrated, 78
message authentication codes (MACs), 87–89
performance, 82
preimage resistance, 79
TESLA application, 99–102
Hermitian matrix, 288
Homomorphic encryption, 278–79
Householder transformation, 298
Hybrid protection
bit commitment, 257–58
constraints, 256
general methods, 257–58
goals, 255–56
history, 256
issue and options, 253–55
measures, 256
NTS-3, 258
See also Satnav signals
Hypertext Transfer Protocol (HTTP), 167
Hypertext Transfer Protocol Secure (HTTPS), 167

Identity methods
about, 147
cryptographic identification, 148–50
general traits, 147–48
satnav and, 179
zero-knowledge proofs, 150–52
Implementation security, 280
Inclination, 26
Independent clocks, 281
Inertial measurement units (IMUs), 281
Information theory, 49–50
INFOSEC, 183, 186, 280
Integrity
about, 7
digital signatures, 113
link layer, 162
PNT, 8–9
PNT assurance and, 9
Internet Key Ex-change (IKE), 164
Internet of Things (IoTs), 168–69
Internet Protocol (IP), 163
Internet Protocol security (IPsec), 63, 164
Internet Security Association and Key Management Protocol (ISAKMP), 164

Jamming, 197–98

Keccak, 86–87
Keplerian parameters, 25
Kerberos system, 157, 158
Kerckhoff's principle, 52–53
Key distribution center (KDC), 156
Key exchange, 114–15
Key management
about, 152
comparisons, 158
elements of, 152–53
public key methods, 153–55
in sample enterprise, 188–89, 192–93
satnav and, 180–81
symmetric methods, 156–58
See also Cryptographic protocols

Key mixing, 56
Known plaintext, 51

Lattice cryptography, 308–9
Lattices, 307
Least-squares estimation, 184, 185
Lightweight ciphers, 63–64
Link layer (layer 2), 161–63
Location
  authentication, 276–77
  Chimera markers and, 249
  in cryptography, 277–78
  GPS components, 23
Logical Link Control sublayer, 161
Longitude of ascending node, 26

Magnetic navigation, 281
Map, this book, 11
Markers, spreading code, 242–43
Masking vector, 309
Meaconing, 198–99
Measurement, 287, 288–89
Measures, 211
Media Access Control sublayer, 161
Medium earth orbit (MEO) constellation, 27
Merkle-Damgård construction, 82–83, 84
Merkle trees, 96, 97, 224–25
Message authentication codes (MACs)
  about, 87–88
  ciphers in creating, 89
  general view, 88
  hash-based (HMACs), 89
  for NMA (system view), 217
  OSNMA, 226
  signatures and, 114
  TESLA, 216–19
Message digest, 79, 82, 94, 132–33
Misalignment compensation, 35–36
Mission segment
  about, 15, 22
  functionality, 22–23
  location of components, 23
  predictive model, 23–24
  in sample enterprise, 187, 190
  in satnav infrastructure, 183–85
  security goals for, 24
Mix columns, 62
Modulo, 115–16

Monitor stations, 188, 191
Multichannel Chimera, 266–70
Multifactor authentication, 148
Multivariate cryptography, 310–12

National Institute of Standards and Technology (NIST), 5, 181–82
Navigation message authentication (NMA)
  about, 186, 209, 231
  on Chimera, 227–31
  constraints, 210–11
  digital signatures, 213–15
  general methods for, 212–21
  goals, 210
  history, 211–12
  navigation data protection and, 209–11
  OSNMA, 209, 221–27
  out-of-band, 219–20
  proposal, 202
  recent trends, 212
  summary, 220–21
  summary and the future, 231–32
  with TESLA, 215–19
Navigation signals, 15, 16–19
Navigation Technology Satellite (NTS-3)
  about, 231, 235, 245, 258–59
  Baseline Chimera, 259–63
  multichannel Chimera, 266–70
  performance, 265–66
  processing, 265
  TESLA Chimera, 263–65
Needham-Schroeder protocol, 155, 156
Network layer (layer 3), 163–64
Network stack
  about, 159–60
  link layer (layer 2), 161–63
  network layer (layer 3), 163–64
  observations, 167
  OSI network layer model, 159, 160
  physical layer (layer 1), 160–61
  transport layer (layer 4), 164–66
  upper layers, 167
  See also Cryptographic protocols
No-cloning theorem, 287, 293, 305
Noncryptographic hashes, 78–79, 94
Noncryptographic identification methods, 148
Nonrepudiation, 8, 10, 114, 137
Number theory, 115–18

One time pad, 53
Online Certificate Status Protocol (OCSP), 135
Open Service NMA. *See* Galileo OSNMA
Orbital plane, 26
Orbits, 25–27
Organization, this book, 11–12
OSI network layer model, 159, 160
Out-of-band (OOB) method, 18–19
Out-of-band methods, for NMA, 219–20

Performance
  baseline Chimera, 229
  Chimera markers, 249
  DSA, 128
  Galileo OSNMA, 226
  NTS-3, 265–66
  RSA, 121–22
  TESLA Chimera, 231, 265–66
Personal area networks (PANs), 161, 168
Physical analogs, 110
Physical layer (layer 1), 160–61
Physical layer key exchange, 279–80
PNT assurance
  about, 3, 8–9
  authentication and identity and, 9–10
  availability and, 9
  confidentiality and, 9
  high-level, 6
  integrity and, 9
  nonrepudiation and, 10
Polyalphabetic cipher, 46
Position, navigation, and timing (PNT), 3, 8, 29
Postquantum algorithm, 189
Post-quantum cryptography
  about, 303
  asymmetric cryptography, 304
  code and lattice-based cryptography, 307–10
  complexity theory, 304–7
  multivariate cryptography, 310–12
  symmetric algorithms, 303
  See also Quantum computing
Precision Pointing Security Module (PPS-SM), 200–202
Preimage resistance, 79
Pretty Good Privacy (PGP), 137, 139
Prime, generator of a, 116–17
Privacy-Enhanced Mail (PEM), 137
Private key, 111
Pseudorandom generation, 54
Pseudorandomness, 72
Public key cryptography
  about, 10, 109–10
  algorithms, 109–10
  DH key exchange protocol, 128–31
  DSA, 122–28
  encryption, 111–12
  goals, 111–15
  history, 111
  key exchange, 114–15
  math foundations, 115–18
  motivation, 110
  physical analogs, 110
  PKI, 133–36
  RSA, 118–22
  secure email application, 136–39
  signatures and, 112–14, 131–33
  takeaways, 140
  uses, 109
Public key infrastructure (PKI)
  about, 110, 133–34
  certificate authority (CA), 134, 135
  certificates, 135–36
  elements, 134
  hierarchy, 135
  main points, 136
  operations, 134–35
  registration authority (RA), 134
  structure, 134
Public key Needham-Schroeder, 155
Public keys, 111, 181, 224–25
Public key verification, 154

Quantum computing
  about, 285–86
  components of, 286–93
  entanglement, 287, 289–93
  measurement, 287, 288–89
  no cloning, 287, 293
  output, 293
  superposition, 286–88
  unitary operations, 287
Quantum computing algorithms
  about, 293
  Deutsch's algorithm, 294–96
  Grover's algorithm, 296–99
  Shor's algorithm, 299–303
Quantum Fourier transform (QFT), 301–3

Quantum information theory, 285
Quasi-Zenith Satellite System (QZSS), 3
Qubits, 285, 288, 291–92

Radio Frequency Identification (RFID) standards, 64
Rainbow tables, 95
Ranging signal
about, 27
composition of, 29
layers, 27–29
properties, 31
security goals for, 31
Reencrypted codes (RECS), 270–71
Regional Navigation Satellite System (RNSS), 3
Registration authority (RA), 134
Rejection sampling, 309
Risk and risk management, 71, 144–46, 181–82
Rivest-Shamir-Adleman (RSA)
about, 109, 118
attacks, 121, 300
decryption, 119
definition, 118–19
encryption, 118
implications, 120
justification, 119
parameters, 118
performance, 121–22
practicalities, 120
signatures, 120
strengths, 121
strength versus size, 121

SAASM Card Integration Program (SCIP), 200
Sample enterprise
common elements, 189
components, 187–89
control segment, 187, 190–91
ephemeris service, 189, 192
ground antenna, 188, 191
key management, 192–93
mission segment, 190
monitor stations, 188, 191
space segment, 191–92
user equipment (UE), 189, 193–94
See also Satnav
Satnav
about, 15–16
approaches, 195
control segment, 15, 24–25, 183, 187–88
cryptography and, 177–202
distribution of shared keys in, 180
future, 281
ground segment, 15, 22–25
identity and, 179
infrastructure, 181–87
key management and, 180–81
mission segment, 15, 22–24, 183–84, 187
navigation signals and, 16–19
orbits and, 25–27
overview, 15
protocols and, 178–81
public keys in, 181
sample enterprise, 187–94
security characteristics, 282
segments, 15, 19–25
signal data, 36
signals, 27–31
space and security, 177–78
space segment, 15, 19–20, 182–83
standalone regional systems, 17
threats, 186–87
trust and, 178–79
user segment, 15, 20–22, 185–86
Satnav signals
about, 194–95
carrier and modulation layers, 196
data layer, 196–97
hybrid protection, 253–72
layers and security goals, 195–97
protection history, 200–202
taxonomy of threats, 197–99
Scheduling, control segment, 24
Schnorr identification scheme, 151
Secure email application
about, 136
general framework, 137–39
goals, 137
illustrated, 138
nuances, 139
Secure/Multipurpose Internet Mail Extensions (S/MIME), 137
Secure Real-time Transport Protocol (SRTP), 166

Secure Sockets Layer (SSL), 165–66
Secure verification, digital fingerprints, 94–95
Sensor network, 22
SHA-2 family
about, 84
execution time, 86
Merkle-Damgård construction, 85
SHA-512 and, 85
tailoring security and performance, 85–86
SHA-3, 83, 84, 86–87
Shannon, Claude, 46, 49, 55–56, 285
Shared key methods, 239–40
Shift rows, 60
Shor's algorithm
about, 285, 299–300
classical analog, 300–301
potential threat of, 304
quantum Fourier transform (QFT), 301–3
See also Quantum computing algorithms
Signal Authentication Service (SAS)
about, 256, 258, 270
Chimera versus, 271
overview, 270
security goals, 271
transmitter specifications, 271
Signal data, 36
Signatures
about, 112–13
blind, 132
double, 132–33
DSA, 123–24
goals of, 113
MACs and, 114
NMA, 213–15
public key cryptography and, 112–14, 131–33
RSA, 120
trusted authority (TA) and, 131–32
use illustration, 113
Simon and Speck, 64
Simple Mail Transfer Protocol (SMTP), 136–37
Size, weight, (and) power, and cost (SWaP-C), 237
Slow channel Chimera, 260–61
Snapshot receivers
about, 275–76
illustrated, 276
location authentication, 276–77
Soundness, 151
Space Attack Research and Tactic Analysis (SPARTA), 187
Space-based augmentation system (SBAS), 17, 186
Space segment
about, 15, 19
design, 182
orbital components, 182
payload components, 19–20
sample enterprise, 191–92
security goals for, 20
SVs, 182–83, 188
Space vehicles (SVs), 25, 182–83, 188
Specific emitter identification (SEI), 103
SPHINCS+, 286
Sponge construction, 83–84, 86–87
Spoofing, 199
Spreading codes
about, 235
bit commitment methods, 240–41
Chimera markers, 245–49
ciphertext as, 241–42
complete, spreading codes, 243–45
constraints, 236–37
cryptography and, 238–39, 241–45
goals, 236
history, 237–38
L1C structure, 246
measures, 237
protecting signal lower layers, 235–37
shared key methods, 239–40
Spread spectrum security codes (SSSCs), 201
Squeezing, 83
Steganography, 70–71
Stream ciphers, 47–48, 59
Substitution ciphers, 45–46
Substitution-permutation construction, 56–57
Superposition, 286–88
Symmetric algorithms, 303
Symmetric cryptography, 10, 43
Symmetric key Needham-Schroeder, 156
Synchronization, time, 18
Synthetic initialization vector, 92

TESLA Chimera
about, 227, 229, 263
binding of data and signal, 263
data, 264
data messages, 264
data procedure, 263–64
as experimental design, 231
message bits protected by MACs, 230–31
overview, 263–66
performance, 231, 265–66
processing, 265
TESLA chain, 229–30
See also Chimera
Threats
jamming, 197–98
meaconing, 198–99
satnav infrastructure, 186–87
spoofing, 199
taxonomy of, 197–99
Ticket-granting service (TGS), 157–58
Tickets, 158
Time between authentication (TBA), 211, 266, 267
Timed Efficient Stream Loss-Tolerant Authentication (TESLA)
about, 78, 99
authentication via digital signatures, 101–2
as bit commitment scheme, 219
chains, 216, 225–26, 229–30, 256, 268
creating and verifying MACs (for NMA), 216
cryptographic components, 99
data stream, 100–101
framework for using, 215–17
Galileo OSNMA, 222
hash chain of keys, 100, 101
message stream and MACs, 101
NMA and, 215–19
overview, 100
parts, 100
properties, 102
root key, 101, 225
shared key MACs, 212
start key, 101
streams, noncoupling/coupling between, 268
subchains, 218
Time-division-multiple-access (TDMA), 160–61
Time stamps, 131–32, 154
Time synchronization, 18
Time to first authenticated fix (TTFAF), 211, 215, 226, 256
Toffoli gate, 30
TRANSEC, 189, 194, 238
Transport Control Protocol (TCP), 165, 166
Transport layer (layer 4), 164–67
Transposition, 46
Trapdoor function, 306–7
Trapdoor one-way function, 115
Trust
in cryptographic protocol, 147
ensuring, 145–46
need for, 145
satnav and, 178–79
Trusted authority (TA), 131–32
Trusted Platform Module (TPM), 179
Turing machine, 305

Unbalanced oil and vinegar (UOV), 310–12
User Datagram Protocol (UDP), 165, 166
User equipment (UE), 20–21, 189, 193–94
User segment
about, 15, 20
elements of, 21–22
functional layout, 21
in satnav infrastructure, 185–86
security goals for, 22

Virtual private networks (VPNs), 164
Visual-aid navigation, 281

Walker constellation, 28
Watermarking, 71, 95
Watermarks, 95
Web of trust protocol, 139
Wide Area Augmentation System (WAAS), 17
Wi-Fi Protected Access 3 (WPA3), 162

X.509 standard certificate structure, 135–36

Zero-knowledge proofs, 150–52
Zero trust, 169–70, 192