CONTENTS

Preface **xiii**
Message to students . . . xiv
Message to teachers . . . xvii
Acknowledgments . . . xix

UNCONDITIONAL CRYPTOGRAPHY

1 One-Time Pad and the Provable Security Mindset **3**
1.1 Defining the problem . . . 4
1.2 One-time pad . . . 6
1.3 How to formalize an attack scenario . . . 8
1.4 How to define and prove security of OTP . . . 12
1.5 How to interpret a security proof . . . 13
1.6 Limitations of security proofs . . . 15
1.7 What's so special about XOR? . . . 17

2 Rudiments of Provable Security **25**
2.1 Attack scenarios as libraries . . . 25
2.2 Interchangeable libraries . . . 28
2.3 How to distinguish two libraries . . . 33
2.4 How to prove that two libraries are interchangeable . . . 37
2.5 Abstract cryptographic primitives . . . 42
2.6 Modular cryptographic constructions . . . 45
2.7 ⋆ "Real-or-random" vs. "left-or-right" . . . 49

3 ⋆ Secret Sharing **61**
3.1 Defining secret sharing . . . 62

3.2 2-out-of-2 secret sharing 65
3.3 Polynomial interpolation 68
3.4 Multiplicative inverses modulo a prime 72
3.5 Shamir secret sharing 78

PSEUDORANDOMNESS

4 Modern Computational Cryptography 87
4.1 The concrete approach to provable security 88
4.2 The asymptotic approach to provable security 91
4.3 Indistinguishability 95
4.4 The bad-event technique 98
4.5 Birthday probabilities 105

5 Pseudorandom Generators 117
5.1 Defining pseudorandomness 118
5.2 Do PRGs exist? 121
5.3 Application: Pseudo-OTP 124
5.4 How to extend a PRG's stretch 127
5.5 How to attack pseudorandomness 130

6 Pseudorandom Functions 141
6.1 Motivating and defining PRFs 142
6.2 How to attack PRFs 144
6.3 An example proof involving PRFs 148
6.4 Building a PRG from a PRF: The CTR construction 153
6.5 ⋆ Building a PRF from a PRG: The GGM construction 158
6.6 PRFs for long inputs: The CBC-MAC construction 166

7 Pseudorandom Permutations 181
7.1 Defining PRPs 182
7.2 Comparing PRPs and PRFs, and the switching lemma 184
7.3 How to construct a PRP: The Feistel construction 186
7.4 PRPs in theory and practice 194
7.5 ⋆ Strong PRPs 198

SYMMETRIC-KEY ENCRYPTION

8 Chosen-Plaintext Attacks Against Encryption **205**
8.1 Definitions and leaking the plaintext length . . . 206
8.2 Deterministic encryption . . . 209
8.3 A randomized encryption scheme based on PRFs . . . 210
8.4 An example chosen-plaintext attack . . . 214
8.5 Block cipher modes . . . 215
8.6 Non-block-aligned plaintexts . . . 224
8.7 ⋆ Nonce-based encryption . . . 228

9 Chosen-Ciphertext Attacks Against Encryption **245**
9.1 Format-oracle attacks and malleability . . . 246
9.2 Defining security against chosen-ciphertext attacks . . . 251
9.3 Chosen-ciphertext attack strategies . . . 255
9.4 Message authentication codes . . . 261
9.5 Encrypt-then-MAC . . . 267
9.6 Authenticated encryption and associated data . . . 273

HASHING

10 Collision-Resistant Hash Functions **295**
10.1 Defining collision resistance . . . 296
10.2 Hash functions in the real world . . . 300
10.3 Composing collision-resistant functions . . . 303
10.4 The Merkle-Damgård construction . . . 306
10.5 PRFs from Merkle-Damgård: Length extension, NMAC, and HMAC 310

11 ⋆ Universal Hash Functions **325**
11.1 Defining universal hash functions . . . 326
11.2 Universal hash paradigm for PRFs . . . 330
11.3 CBC-MAC is a UHF . . . 333
11.4 Poly-UHF: An unconditionally secure UHF . . . 342

12 Random Oracles and Other Idealized Models **355**
12.1 Defining the random oracle model . . . 356
12.2 Using random oracles . . . 361
12.3 ⋆ The ideal permutation model . . . 367
12.4 ⋆ The ideal cipher model . . . 376

ASYMMETRIC-KEY CRYPTOGRAPHY

13 Key Exchange **391**
13.1 Cyclic groups . . . 391
13.2 Diffie-Hellman key exchange . . . 398
13.3 Key exchange in the abstract . . . 403
13.4 ⋆ Elliptic curves . . . 404

14 Public-Key Encryption **419**
14.1 Security definitions for PKE . . . 419
14.2 One-time vs. many-time encryption . . . 422
14.3 El Gamal encryption . . . 426
14.4 The Fujisaki-Okamoto construction . . . 431
14.5 Hybrid encryption . . . 441
14.6 Key encapsulation . . . 444

15 RSA **457**
15.1 Arithmetic modulo a composite . . . 457
15.2 The RSA function and RSA assumption . . . 462
15.3 Building a KEM from RSA . . . 467
15.4 ⋆ Chinese remainder theorem . . . 474

16 Digital Signatures **489**
16.1 Security definitions for digital signatures . . . 490
16.2 Signatures from RSA . . . 492
16.3 Common signature idioms . . . 507

ADVANCED TOPICS

17 Encrypted Messaging and Ratcheting **523**
17.1 Forward secrecy and the symmetric ratchet . . . 524
17.2 Post-compromise secrecy and the asymmetric ratchet . . . 528
17.3 The double ratchet and the Signal protocol . . . 530
17.4 Group messaging and MLS . . . 535

18 Authenticated Key Exchange **545**
18.1 Defining the problem . . . 546
18.2 Authenticating with signatures . . . 550
18.3 Authenticating implicitly . . . 557
18.4 Authenticated key exchange in practice . . . 562

19 Zero-Knowledge Proofs **569**
19.1 The Schnorr identification protocol . . . 570
19.2 Sigma protocols . . . 575
19.3 More examples of sigma protocols . . . 578
19.4 Noninteractive proofs and signatures . . . 586

20 Post-Quantum Cryptography **601**
20.1 Capabilities of quantum computing . . . 602
20.2 The learning-with-errors problem . . . 604
20.3 Key exchange from LWE . . . 610

APPENDIX

A Math Review **625**
A.1 Strings . . . 625
A.2 Probability . . . 626
A.3 Modular arithmetic . . . 630
A.4 Exponents and logarithms . . . 635
A.5 Big-O . . . 635
A.6 Linear algebra . . . 636

B A Crash Course in Binary Finite Fields **641**

Bibliography **647**

Index **673**

PREFACE

The Joy of Cryptography is an undergraduate-level textbook that introduces readers to the fundamentals of provable security. With over four hundred exercises, it is suitable as the main reference for a first course on cryptography or as a secondary resource for a course on computer/network security.

PROVABLE SECURITY: Provable security is what demystifies cryptography and promotes it from an ad hoc game of cat-and-mouse to a principled discipline. It is how we answer questions like, *What does it mean to be secure? Can we* ***prove things*** *about security?*

This book teaches readers how to understand and especially *do* provable security: how to formally define security goals, how to prove security properties, and how to demonstrate that insecure things are insecure.

TOPICS: Cryptography involves many different flavors of algorithms, called *primitives,* each designed to provide a different flavor of security guarantee. This book provides a comprehensive treatment of the most important primitives that keep our digital lives safe: how they work, how they are different from each other, and why they are secure:

- Unconditionally secure cryptography: one-time pad and secret sharing.
- Fundamental primitives: pseudorandom generators, pseudorandom functions, and pseudorandom permutations.
- Symmetric-key encryption: chosen-plaintext and chosen-ciphertext attacks, authenticated encryption.

- Hashing: collision-resistant hash functions, universal hash functions, and random oracles.
- Asymmetric-key cryptography: key exchange, public-key encryption, and digital signatures.
- Advanced topics: encrypted messaging and ratcheting, authenticated key exchange, zero-knowledge proofs, and post-quantum cryptography.

OPEN ACCESS: An online, web-based version of the book is available for free at:

`joyofcryptography.com`.

This online version includes animated visualizations of all hybrid security proofs.

Typos and other errors can be reported at `github.com/rosulek/joc`.

MESSAGE TO STUDENTS

I'm honored to be your guide to my favorite corner of computer science. Cryptography has a well-deserved reputation as a beautiful, fascinating, exciting topic. However, it also tends to be regarded as difficult, even intimidating. I think it's important to understand what exactly makes the study of cryptography challenging, so you can be better prepared for what to expect.

WHAT MAKES CRYPTOGRAPHY DIFFICULT? Cryptography demands rigor, precision, and attention to detail; there are definitions, theorems, and proofs. Indeed, cryptography is a mathematical discipline.

However, I believe that what makes cryptography difficult to learn is not the depth of math involved. (A list of specific prerequisites appears later in this section.) Instead, I believe its difficulty stems from the following:

- In cryptography, we study algorithms at a high level of **abstraction.** Compare the following two kinds of questions:
  - *What can we say about combining AES and SHA3 (two standard cryptographic algorithms) in this particular way?*
  - *If A is an arbitrary algorithm with security property X, and B is an arbitrary algorithm with security property Y, then what can we say about combining A and B in this particular way?*

The first question asks about the security properties of one fully specified algorithm. The second question is more abstract; it is about an algorithm built from components A and B, which remain *unspecified.* In this book we generally deal in abstract questions like the second one.

- Concrete examples can help you understand *what* an algorithm does, but not *why* it's secure. That's because security is a **global property** about the behavior of a system on *all* inputs; it's not something you can see through concrete examples. Besides that, our typical security goal is that when an algorithm is working as expected, its outputs *"look like random junk."* What good does it do to show examples of random junk, other than to instill a false sense of security?
- Security means reasoning about **what is *not* possible**—what an adversary *can't* do. It usually takes more care and effort to prove such negative statements.
- Cryptography involves multiple participants, with fundamentally **different perspectives** of the same system. It's crucial to understand which perspective is being invoked at any given time, because what is true from one may be false from another. You will need to learn how to inhabit and reconcile these multiple perspectives.

MATHEMATICAL PREREQUISITES: The book is meant to be accessible to a third- or fourth-year undergraduate student in a typical computer science (CS) degree program.

You should be comfortable with material that is standard in a typical first course on discrete math, including:

- **basic discrete objects:** functions (injective/surjective), sets and set operations (union, intersection, Cartesian product), tuples, strings, concatenation;
- **simple combinatorics:** counting objects as 2^n, $\binom{n}{2}$, etc;
- **discrete probability:** union bound, principles of multiplication and addition, conditional probabilities;
- **logic:** boolean operators, implications, contrapositive, quantifiers;
- **induction and recursion;**
- **rules of exponents and logarithms;**
- **binary numbers;**

- **basic modular arithmetic:** addition, subtraction, multiplication, idempotence of modular reduction: $(a + b) \% n = ((a \% n) + (b \% n)) \% n$;
- **basic linear algebra:** matrices and vectors, matrix multiplication, transpose.

Some of these concepts are briefly reviewed in appendix A, but the book is not ideal as a first exposure. Mathematical concepts not listed above (e.g., multiplicative modular inverses, totient function, Chinese remainder theorem, finite fields) are introduced as needed, and prior exposure is not assumed.

COMPUTER SCIENCE PREREQUISITES: This book uses source code to unambiguously define not only cryptographic algorithms but also their security goals. You must be able to correctly interpret the meaning of that source code. It is especially important to understand when two different programs "do the same thing."

Source code in this book is given only in a high-level pseudocode, so it's not important to have experience with any particular programming language. What's more important is a solid conceptual understanding of:

- **flow control:** conditional and looping constructs;
- **variables:** types and scope, especially the idea of scopes that are inaccessible from certain blocks of code;
- **information dependency:** When does one value/variable influence another?
- **simple data structures:** sets, arrays, associative arrays (dictionaries), strings;
- **subroutines:** calling principles, factoring out lines of code into a subroutine, inlining subroutine calls, recursion.

These elementary concepts are typically covered in an introductory course sequence in programming.

As I mentioned above, cryptography also involves a high degree of **abstraction.** It is therefore helpful, though not mandatory, to have prior exposure thinking about computational processes in this way. In undergraduate CS curricula, this usually happens in a theory of computation (automata) course.

EXERCISES: Each chapter contains a selection of exercises. Exercises marked with H have hints, which are located at the end of the chapter. Those marked with ⋆ are, in my opinion, significantly above average in difficulty or scope.

MESSAGE TO TEACHERS

SELECTION OF MATERIAL: Two philosophies have guided my selection of material for this book:

- Emphasize **provable security** above all else. Provable security is not just "learning about cryptography using definitions, theorems, and proofs," but defining and proving things specifically about *security properties*—that is, an algorithm's behavior in the presence of a particular adversarial attack. There are plenty of mathematical facts that involve cryptography but are not about security properties (for example, this factoring/primality algorithm is correct, elliptic curve addition satisfies the axioms of a group, or there is a unique polynomial interpolating a set of points). The book includes some material of this flavor, but only selectively.
- Within the realm of provable security, focus on the constructions and techniques that inform cryptography in the real world.

If you and I haven't agreed on what to include in a cryptography textbook, it can probably be understood through the lens of these two guiding principles.

ORGANIZATION: The book is designed so that material builds cumulatively on previous concepts. However, there are a few exceptions:

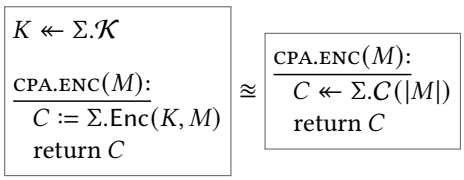
- Chapters and sections marked ★ could be skipped in the interest of time. Subsequent material does not build heavily on them.
- Chapters on advanced topics (17–20) do not build on each other, so they can be read in any order.

NOVEL APPROACH TO PROVABLE SECURITY: The main difference between this book and competitors is its overall approach to provable security. Below is a summary of my most *opinionated* choices.

All security definitions use a unified **game-based style**, where an interaction between adversary and challenger is specified precisely using pseudocode. I call these games *libraries*. The adversary plays the role of a calling program who can access the game/library through its designated interface of subroutines.

All security definitions are **distinguishing games**. Rather than defining a single game, in which the adversary tries to guess a secret bit b, I explicitly write both the $b = 0$ and $b = 1$ branches as **two separate games**; the adversary's goal is

to distinguish between them. For example, an encryption scheme Σ has IND$-CPA security if the following two games are indistinguishable (definition 8.1.1):



In the left library, K is privately scoped to the library and initialized at the beginning of time. Thus, every call to the CPA.ENC subroutine uses the same value of K. In the right library, $\Sigma.\mathcal{C}(\ell)$ denotes the set of possible ciphertexts that result from encrypting length-ℓ plaintexts.

Even authenticity properties, as in digital signatures and authenticated encryption, are formalized as distinguishing games. For example, a digital signature scheme is secure if its verification oracle is indistinguishable from an oracle that "always" says *no* (see definition 16.1.2 for the more precise formulation).

All security proofs use a **game-hopping approach**, whose focus is not on explicitly writing a reduction algorithm but on making well-defined modifications to the pseudocode of a game/library. *This unified approach to proofs is the book's "killer feature."* Each "hop" in the proof is a small change to the pseudocode of a game, along with a justification that the change has only a negligible effect on the adversary. Many changes/hops have self-evident justification (e.g., removing an unreachable statement). Some consecutive hybrids are indistinguishable because of a cryptographic assumption (e.g., F is a secure PRF). In these cases, we do not write an explicit reduction algorithm but instead do the following. The assumption will be defined via two games, G_0 and G_1, which we assume are indistinguishable. We rewrite one hybrid game in our proof, so that G_0 appears as a separate, self-contained module (being sure that "factoring out" G_0 has no effect on the game's behavior). Now, with a literal instance of G_0 appearing as a subgame, we may replace it with G_1, and then inline the result to obtain a monolithic game again. Thus, we make a series of three indistinguishable changes to a game (factor out G_0, swap with G_1, inline G_1), each of which we can justify has negligible or no effect on the adversary. An explicit reduction algorithm is not the center of attention. Rather, it exists implicitly as the part of the game that remains unchanged when G_1 replaces G_0. An example is worth a thousand words, so I simply recommend examining the proof of claim 8.3.2 (CPA security of the $R \| (F(K, R) \oplus M)$ encryption scheme) as a good representative example.

I make **real-or-random** indistinguishability, not left-or-right indistinguishability, the default approach for defining security of encryption. This choice reinforces

a theme of the book, that we can define security by requiring outputs to "look like random junk." Left-or-right security for encryption is an outlier, and students in my experience find it confusingly artificial: Why would an encryption algorithm require us to supply *two* plaintexts? The left-or-right style is still presented (sections 2.7, 8.1.2, and 9.2.1) as an alternative, and the two styles are contrasted in the text and in exercises. I define CCA security not by forbidding the adversary from decrypting challenge ciphertexts but by "patching" the decryption oracle to give the expected plaintext in these cases (definition 9.2.1). In general, I try to acknowledge that security definitions are choices, not sacred proclamations, so there are often several valid ways to approach a security definition.

MACs are not presented as a separate primitive; rather, message authentication is discussed as a property already enjoyed by PRFs. Thus, we do not discuss nonce-based or randomized MACs.

RESOURCES: If you are interested in creating your own materials that follow this book's formatting style, you can find some tools available at:

`github.com/rosulek/joc`.

ACKNOWLEDGMENTS

I am grateful for thoughtful technical feedback on this edition of *The Joy of Cryptography* from Martin Albrecht, Joël Alwen, Yevgeniy Dodis, Nadim Kobeissi, Hugo Krawczyk, Chris Peikert, and Douglas Stebila. Thank you to Jake Johanson and Zane Othman-Gomez for extensively beta-testing an early draft of this book and providing an unfiltered student perspective. Sincere, special thanks are due to the outstanding teachers and mentors that shaped my professional journey: Terry Balk taught me math (I wish we had had a chance to chat about cryptography!), Manoj Prabhakaran taught me cryptography, and Michael Loui taught me to embrace scholarly writing.

Some financial support for writing this book has been kindly provided by the National Science Foundation (awards #1149647, #1617197), the Oregon State University Open Textbook Initiative, and Google.

There would be no *Joy of Cryptography* without the support and encouragement of the amazing Laura Rosulek. *ILY!*