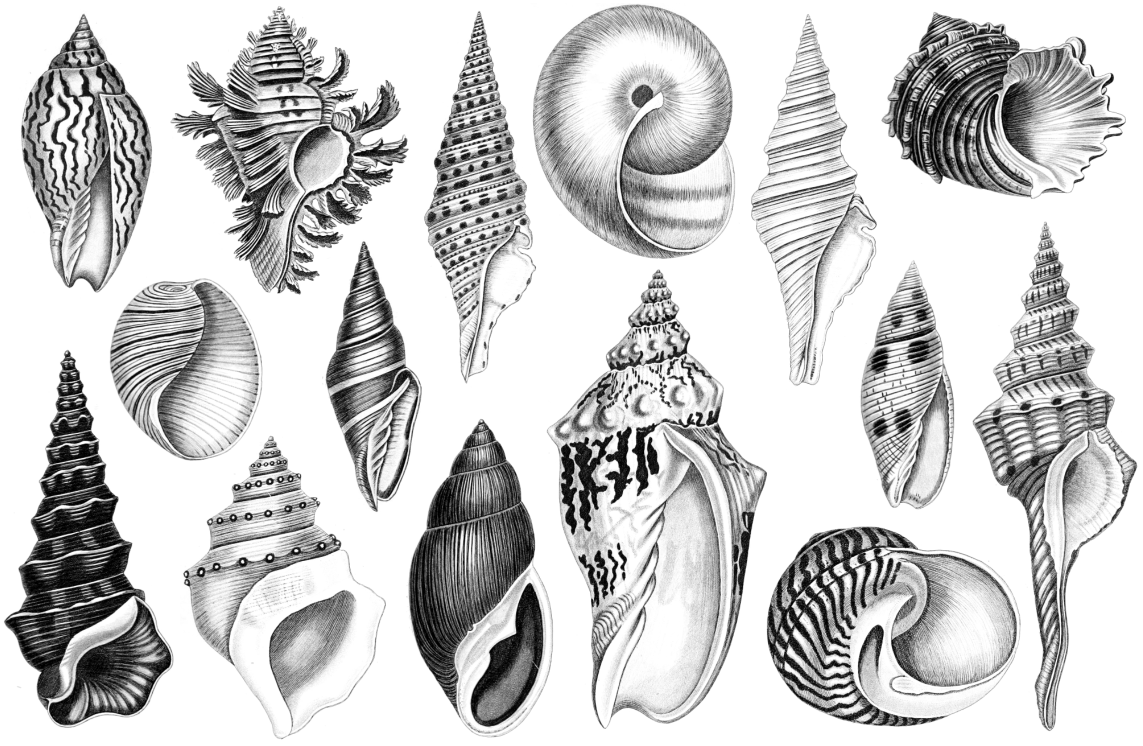

The Joy of Cryptography

Mike Rosulek ⟨rosulekm@eecs.oregonstate.edu⟩
School of Electrical Engineering & Computer Science
Oregon State University, Corvallis, Oregon, USA

Draft of January 22, 2018

Foreword

These are lecture notes for cs427 at Oregon State University, an introductory course in cryptography at the advanced undergraduate level. By reading and studying these notes, you should expect to learn how to:

- State and interpret the standard formal definitions for the most common cryptographic security properties (privacy and authentication).
- Formally prove security properties of sound cryptographic constructions, and break the security of unsound ones.
- Choose the appropriate cryptographic primitive for a task (block ciphers, hash functions, MACs, public-key encryption, etc.) while avoiding common pitfalls.

Along the way, you will also learn how the most common cryptographic constructions work.

Background

You will get the most out of these notes if you have a solid foundation in standard undergraduate computer science material:

- Discrete mathematics [required]: for modular arithmetic, discrete probabilities, simple combinatorics, and especially proof techniques.
- Algorithms & data structures [highly recommended]: for reasoning about computations at an abstract level.
- Theory of computation (automata, formal languages & computability) [recommended]: for even more experience formally reasoning about abstract processes of computation.

Disclaimer

You are reading an early draft of this book. Of course I make every effort to ensure the accuracy of the content, but the content has not yet benefitted from many critical readers. So, *caveat emptor!*

More than that, much interesting and hugely important material is not covered. Some more minor missing material is indicated as such and labeled "to-do", but conspicuously missing major sections are simply absent. Anyone with a pre-existing opinion about cryptography will likely be appalled at what is missing. In my defense, my university schedule is on the quarter system which necessitates that I generate 10 weeks of course material. Everything beyond that is produced via the much slower *labor-of-love* mechanism.

I welcome feedback from readers, educators, cryptographers — not only on errors and typos but also on the selection, organization, and presentation of the material.

Code-based games

The security definitions and proofs in these notes are presented in a style that is known to the outside world as *code-based games*. I've chosen this style because I think it offers significant pedagogical benefits:

- Every security definition can be expressed in the same style, as the indistinguishability of two games. In my terminology, the games are *libraries* with a common interface/API but different internal implementations. An adversary is any calling program on that interface. These libraries use a concrete pseudocode that reduces ambiguity about an adversary's capabilities. For instance, the adversary controls arguments to subroutines that it calls and sees only the return value. The adversary cannot see any variables that are privately scoped to the library.

- A consistent framework for definitions leads to a consistent process for *proving* and *breaking* security — the two fundamental activities in cryptography.

  In these notes, *breaking* a construction always corresponds to writing a program that expects a particular interface and behaves as differently as possible in the presence of two particular implementations of the interface.

  Proving security nearly always refers to showing a sequence of libraries (called *hybrids*), each of which is indistinguishable from the previous one. Each of these hybrids is written in concrete pseudocode. By identifying what security property we wish to prove, we identify what the endpoints of this sequence must be. The steps that connect adjacent hybrids are stated in terms of syntactic rewriting rules for pseudocode, including down-to-earth steps like factoring out and inlining subroutines, changing the value of unused variables, and so on.

- Cryptography is full of conditional statements of security: *"if A is a secure thingamajig, then B is a secure doohickey."* A conventional proof of such a statement would address the contrapositive: *"given an adversary that attacks the doohickey-security of B, I can construct an attack on the thingamajig-security of A."*

  In my experience, students struggle to find the right way to transform an abstract, hypothetical B-attacking adversary into a successful A-attacking adversary. By defining security in terms of games/libraries, we can avoid this abstract challenge, and indeed avoid the context switch into the contrapositive altogether. In these notes, the thingamajig-security of A gives the student a new *rewriting rule* that can be placed in his/her toolbox and used to bridge hybrids when proving the doohickey-security of B.

Code-based games were first proposed by Shoup[1] and later expanded by Bellare & Rogaway.[2] These notes adopt a simplified and unified style of games, since the goal is not to encompass every possible security definition but only the fundamental ones. The most significant difference in style is that the games in these notes have no explicit Initialize or Finalize step. As a result, all security definitions are expressed as *indistinguishability* of two games/libraries, even security definitions that are fundamentally about unforgeability. Yet, we can still reason about unforgeability properties within this framework. For instance, to say that no adversary can forge a MAC, it suffices to say that no adversary can distinguish a MAC-verification subroutine from a subroutine that always returns false. An index of security definitions has been provided at the end of the book.

One instance where the approach falls short, however, is in defining collision resistance. I have not been able to define it in this framework in a way that is both easy to use and easy to interpret (and perhaps I achieved neither in the end). See Chapter 12 for my best attempt.

Supplementary material

Security proofs in this book follow a standard pattern: We start from one "library" and perform a sequence of small, cumulative modifications. Each modification results in a separate hybrid library that is indistinguishable from the previous one.

I have prepared PDF slide decks to supplement the security proofs contained in the book. They are available from the course website. The slides allow the reader to step forward and backward through the proof's sequence of logical steps, seeing only the current hybrid library at any time (with changes highlighted and annotated).

Acknowledgements

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

About the cover

The cover design consists of assorted shell illustrations from *Bibliothèque conchyliologique*, published in 1846. The images are no longer under copyright, and were obtained from the Biodiversity Heritage Library (http://biodiversitylibrary.org/bibliography/11590). Like a properly deployed cryptographic primitive, a properly deployed shell is the most robust line of defense for a mollusk. To an uniformed observer, a shell is just a shell, and crypto is just crypto. However, there are a wide variety of cryptographic primitives, each of which provides protection against a different kind of attack. Just as for a seasoned *conchologist*, the joy is in appreciating the unique beauty of each form and understanding the subtle differences among them.

[1]Victor Shoup: *Sequences of Games: A Tool for Taming Complexity in Security Proofs.* ia.cr/2004/332

[2]Mihir Bellare & Philip Rogaway: *Code-Based Game-Playing Proofs and the Security of Triple Encryption.* ia.cr/2004/331

Copyright

Contents

0 Review of Concepts & Notation **1**

0.1 Modular Arithmetic 1
0.2 Strings 2
0.3 Functions 2
0.4 Probability 2
0.5 Asymptotics (Big-O) 4

1 One-Time Pad **5**

1.1 One-Time Pad Definition 5
1.2 Properties of One-Time Pad 6

2 The Basics of Provable Security **12**

2.1 Library-Based Security Reasoning 12
2.2 A General-Purpose Security Definition for Encryption 17
2.3 How to Prove Security with the Hybrid Technique 21
2.4 How to Demonstrate Insecurity with Attacks 24

3 Secret Sharing **30**

3.1 Definitions 30
3.2 Insecure Aproach for Secret Sharing 32
3.3 A Simple 2-out-of-2 Scheme 33
3.4 Polynomial Interpolation 36
3.5 Shamir Secret Sharing 38
★ 3.6 Visual Secret Sharing 42

4 Basing Cryptography on Limits of Computation **47**

4.1 Polynomial-Time Computation 48
4.2 Negligible Probabilities 49
4.3 Indistinguishability 51
4.4 Birthday Probabilities & Sampling With[out] Replacement 52

5 Pseudorandom Generators **60**

5.1 Definition 60
5.2 Application: Shorter Keys in One-Time-Secret Encryption 63
★ 5.3 Taking the Contrapositive Point-of-View 66
5.4 Extending the Stretch of a PRG 67

6 Pseudorandom Functions **73**

6.1 Definition 73
6.2 Attacking Insecure PRFs 75
★ 6.3 A Theoretical Construction of a PRF from a PRG 76

7 Pseudorandom Permutations **83**

7.1 Definitions 83
7.2 Switching Lemma 84

7.3 Feistel Ciphers . . . 85
7.4 Strong Pseudorandom Permutations . . . 89

8 Security against Chosen Plaintext Attacks 92
8.1 Implications of CPA Security . . . 92
8.2 Pseudorandom Ciphertexts . . . 94
8.3 CPA-Secure Encryption from PRFs . . . 97

9 Block Cipher Modes of Operation 103
9.1 Common Modes . . . 103
9.2 CPA Security for Variable-Length Plaintexts . . . 106
9.3 Security of OFB Mode . . . 108
9.4 Padding & Ciphertext Stealing . . . 111

10 Chosen Ciphertext Attacks 118
10.1 Padding Oracle Attacks . . . 118
10.2 What Went Wrong? . . . 121
10.3 Defining CCA Security . . . 123
10.4 CCA Insecurity of Block Cipher Modes . . . 125
10.5 A Simple CCA-Secure Scheme . . . 126

11 Message Authentication Codes 136
11.1 Definition . . . 136
★ 11.2 A PRF is a MAC . . . 140
11.3 CBC-MAC . . . 145
11.4 Encrypt-Then-MAC . . . 146

12 Hash Functions 153
12.1 Defining Security . . . 153
12.2 Hash-Then-MAC . . . 156
12.3 Merkle-Damgård Construction . . . 159
12.4 Length-Extension Attacks . . . 161

13 The RSA Function 166
13.1 Modular Arithmetic & Number Theory . . . 166
13.2 The RSA Function . . . 169
13.3 Chinese Remainder Theorem . . . 170
13.4 The Hardness of Factoring N . . . 172
13.5 Malleability of RSA, and Applications . . . 175

14 Diffie-Hellman Key Agreement 181
14.1 Cyclic Groups . . . 181
14.2 Diffie-Hellman Key Agreement . . . 182
14.3 Decisional Diffie-Hellman Problem . . . 183

15 Public-Key Encryption 187
15.1 Security Definitions . . . 187

15.2 One-Time Security Implies Many-Time Security 188
15.3 ElGamal Encryption 191
15.4 Hybrid Encryption 194

Index of Security Definitions **198**