
Undergraduate Topics in Computer Science

Series Editor

Ian Mackie, University of Sussex, Brighton, UK

Advisory Editors

Samson Abramsky , Department of Computer Science, University of Oxford, Oxford, UK

Chris Hankin , Department of Computing, Imperial College London, London, UK

Mike Hinchey , Lero – The Irish Software Research Centre, University of Limerick, Limerick, Ireland

Dexter C. Kozen, Department of Computer Science, Cornell University, Ithaca, NY, USA

Andrew Pitts , Department of Computer Science and Technology, University of Cambridge, Cambridge, UK

Hanne Riis Nielson , Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark

Steven S. Skiena, Department of Computer Science, Stony Brook University, Stony Brook, NY, USA

Iain Stewart , Department of Computer Science, Durham University, Durham, UK

‘Undergraduate Topics in Computer Science’ (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems, many of which include fully worked solutions.

The UTiCS concept relies on high-quality, concise books in softback format, and generally a maximum of 275–300 pages. For undergraduate textbooks that are likely to be longer, more expository, Springer continues to offer the highly regarded Texts in Computer Science series, to which we refer potential authors.

More information about this series at <http://www.springer.com/series/7592>

Duncan Buell

Fundamentals of Cryptography

Introducing Mathematical
and Algorithmic Foundations

Duncan Buell
Department of Computer Science
and Engineering
University of South Carolina
Columbia, SC, USA

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-030-73491-6 ISBN 978-3-030-73492-3 (eBook)
<https://doi.org/10.1007/978-3-030-73492-3>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2021

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

I taught in a computer science department. Our cryptography class, which I just finished teaching for the fourth time, is for upper-level undergraduates (juniors and seniors) and also for graduate credit. This is, I think, similar to what is available in a large number of universities in the United States. Our course is also cross-listed with the mathematics department, and we usually have a combination of math and computer science students taking the course.

The computer science students know how to program, but they are often not so good at the math. In contrast, the math students can usually do the math but don't have the same experience in programming. I have normally accepted programming assignments in any of Python, C++, or Java. I have pointed out that Python provides multiprecision integer arithmetic as a default, and have pointed the students using C++ to the `gmp` package and the students using Java to the `BigNum` package. I have also encouraged students to work in groups on the programming assignments, allowing groups to combine both the mathematical understanding and the ability to code up the algorithms and heuristics presented.

I have assumed some minimal background in asymptotic analysis of algorithms. I think I have kept the big- $\mathcal{O}(\ast)$ to a minimum, and I think virtually all the needed discrete math (modular arithmetic, groups, rings, and such) actually appears in this text.

I think most students in the United States see at least the rudiments of matrix reduction in high school; I have tried to make the linear algebra limited to what most students might actually see in high school; and I have not used any linear algebra beyond the mechanical process of matrix reduction.

Although there is a certain background in groups and rings, I have tried to make that as straightforward as possible. One advantage of this material is that it isn't pure theory; one can actually see concrete examples of the theorems and, I would hope, understand that the theory is mostly just a way of discussing the material and not the end in itself.

It is, in fact, precisely because this material does and should appear in both departments that I decided to write this book. There are several good crypto books that are suitable for a course that is just mathematics. There is at least one that is suitable for a lower-level course targeting a more general audience.

Cryptography, as done in this century, is heavily mathematical, but it also has roots in what is computationally feasible. What I would hope this text does, that I think other texts do not do, is balance the theorems of mathematics against the feasibility of computation. This is the essence of my including the material of Chap. 8 in the text. My background is mathematics. My experience (except for a first year in a math department after my doctorate) is in computer science departments, with 15 years at a research lab doing high performance computing research in support of the National Security Agency. My “take” on cryptography, including my time with NSA, is that it is something one actually “does”, not a mathematical game one proves theorems about. There is deep math; there are some theorems that must be proved; and there is a need to recognize the brilliant work done by those who focus on theory. But at the level of an undergraduate course, I think the emphasis should be first on knowing and understanding the algorithms and how to implement them, and also to be aware that the algorithms must be implemented carefully to avoid the “easy” ways to break the cryptography.

Exercises

Most of the exercises in the later chapters are programming problems. This is largely due to the difficulty in posing realistic problems that can be done entirely by hand. As for computational resources: Many of the exercises, and many of the necessary exercises, require some computation. There are many tools that can be used. The (ancient, but certainly feasible) Unix tool `bc` will do the multiprecision arithmetic necessary for many of the computations and can be scripted using something like

```
bc < myscript.txt
```

to redirect from standard input. Some students will prefer Matlab® or Mathematica or Maple. Others may find SageMath a better tool. The key for some of the simpler problems is that students use the software as a sophisticated calculator but do the algorithms themselves rather than just calling functions built by someone else. The key for the genuine programming problems is that students learn the details of the algorithms, at least for small examples that would not require multiprecision arithmetic packages. Following the classic advice to “make it right before you make it better”, if students can code single precision examples that work, then most of the work to code multiprecision examples will only be the transition from single to multiple precision, and the tracing of the single precision example can be used as ground truth for whether the multiple precision version has bugs.

Acknowledgements

It would be inappropriate for me not to thank my most recent cryptography class for which the first version of this was their textbook, and the previous classes for which earlier versions were written. I received great feedback from the most recent class, some corrections, and some suggestions for what should be changed and what had to be changed in order to become more readable. Everyone who teaches will complain about some of the students, but I have been fortunate to teach and work with a number of superstars; I am grateful to have had such students.

Columbia, SC, USA
February 2021

Duncan Buell

Contents

1	Introduction	1
1.1	History	1
1.2	Introduction	3
1.3	Why Is Cryptography Used?	5
1.4	Modes of Encryption	6
1.5	Modes of Attack	7
1.6	How Many Seconds in a Year?	7
1.7	Kerckhoffs' Principle	9
1.8	Exercises	9
	References	10
2	Simple Ciphers	11
2.1	Substitution Ciphers	12
2.1.1	Caesar Ciphers	12
2.1.2	Random Substitutions	12
2.1.3	Vigenère as an Example of Polyalphabetic Substitutions	13
2.2	Language Characteristics and Patterns	14
2.2.1	Letter Frequency	14
2.2.2	Word Boundaries	15
2.2.3	Cribbing	16
2.2.4	Entropy	16
2.3	Transposition Ciphers	19
2.3.1	Columnar Transpositions	19
2.3.2	Double Transposition	20
2.4	Playfair	20
2.5	ADFGX	21
2.6	Cryptanalysis	22
2.6.1	Breaking a Substitution Cipher	22
2.6.2	Breaking a Transposition Cipher	23
2.7	The Vernam One-Time Pad	23

2.8	Exercises	24
2.8.1	Cipher Text for Substitution Cipher Problems (3) and (4)	25
	References	26
3	Divisibility, Congruences, and Modular Arithmetic	27
3.1	Divisibility	27
3.2	The Euclidean Algorithm	29
3.2.1	The Naive Euclidean Algorithm	30
3.2.2	The Extended Euclidean Algorithm	31
3.2.3	The Binary Euclidean Algorithm	32
3.2.4	The Subtract-Three-Times Euclidean Algorithm	33
3.2.5	GCDs of Large Integers	34
3.3	Primes	35
3.4	Congruences	36
3.5	The Euler Totient	43
3.6	Fermat's Little Theorem	43
3.7	Exponentiation	44
3.8	Matrix Reduction	45
3.9	Exercises	46
	References	47
4	Groups, Rings, Fields	49
4.1	Groups	49
4.2	Rings	53
4.3	Fields	54
4.4	Examples and Expansions	54
4.4.1	Arithmetic Modulo Prime Numbers	54
4.4.2	Arithmetic Modulo Composite Numbers	57
4.4.3	Finite Fields of Characteristic 2	60
4.5	Exercises	60
	References	61
5	Square Roots and Quadratic Symbols	63
5.1	Square Roots	63
5.1.1	Examples	65
5.2	Characters on Groups	67
5.3	Legendre Symbols	67
5.4	Quadratic Reciprocity	68
5.5	Jacobi Symbols	68
5.6	Extended Law of Quadratic Reciprocity	69
5.7	Exercises	70
	Reference	71

6	Finite Fields of Characteristic 2	73
6.1	Polynomials with Coefficients mod 2	73
6.1.1	An Example	73
6.2	Linear Feedback Shift Registers	75
6.3	The General Theory	79
6.4	Normal Bases	80
6.5	Exercises	85
	References	85
7	Elliptic Curves	87
7.1	Basics	87
7.1.1	Straight Lines and Intersections	88
7.1.2	Tangent Lines	90
7.1.3	Formulas	90
7.1.4	The Mordell-Weil Group	91
7.2	Observation	93
7.3	Projective Coordinates and Jacobian Coordinates	94
7.4	An Example of a Curve with Many Points	94
7.5	Curves Modulo a Prime p	96
7.6	Hasse's Theorem	96
7.7	Exercises	97
	Reference	98
8	Mathematics, Computing, and Arithmetic	99
8.1	Mersenne Primes	99
8.1.1	Introduction	100
8.1.2	Theory	100
8.1.3	Implementation	103
8.1.4	Summary: Feasibility	104
8.1.5	Fermat Numbers	104
8.1.6	The Arithmetic Trick Is Important	105
8.2	Multiprecise Arithmetic and the Fast Fourier Transform	105
8.2.1	Multiprecise Arithmetic	105
8.2.2	Background of the FFT	106
8.2.3	Polynomial Multiplication	106
8.2.4	Complex Numbers as Needed for Fourier Transforms	107
8.2.5	The Fourier Transform	108
8.2.6	The Cooley–Tukey Fast Fourier Transform	109
8.2.7	An Example	110
8.2.8	The FFT Butterfly	116
8.3	Montgomery Multiplication	117
8.3.1	The Computational Advantage	120
8.4	Arithmetic in General	120

8.5	Exercises	120
	References	121
9	Modern Symmetric Ciphers—DES and AES	123
9.1	History	123
9.1.1	Criticism and Controversy	124
9.2	The Advanced Encryption Standard	125
9.3	The AES Algorithm	127
9.3.1	Polynomial Preliminaries: The Galois Field $GF(2^8)$	127
9.3.2	Byte Organization	128
9.4	The Structure of AES	129
9.4.1	The Outer Structure of the Rounds	129
9.4.2	General Code Details	129
9.4.3	KeyExpansion	130
9.4.4	SubBytes	133
9.4.5	ShiftRows	136
9.4.6	MixColumns	137
9.4.7	AddRoundKey	140
9.5	Implementation Issues	141
9.5.1	Software Implementations	142
9.5.2	Hardware Implementations	144
9.6	Security	145
9.7	Exercises	146
	References	147
10	Asymmetric Ciphers—RSA and Others	149
10.1	History	149
10.2	RSA Public-Key Encryption	150
10.2.1	The Basic RSA Algorithm	150
10.3	Implementation	151
10.3.1	An Example	152
10.4	How Hard Is It to Break RSA?	153
10.5	Other Groups	153
10.6	Exercises	155
	References	155
11	How to Factor a Number	157
11.1	Pollard rho	158
11.2	Pollard $p - 1$	160
11.2.1	The General Metaphysics of $p - 1$	161
11.2.2	Step Two of $p - 1$	161

11.3	CFRAC	162
11.3.1	Continued Fractions	162
11.3.2	The CFRAC Algorithm	165
11.3.3	Example	167
11.3.4	Computation	168
11.4	Factoring with Elliptic Curves	169
11.5	Exercises	170
	References	170
12	How to Factor More Effectively	173
12.1	Shortcomings of CFRAC	173
12.2	The Quadratic Sieve	173
12.2.1	The Algorithm	174
12.2.2	The Crucial Reasons for Success and Improvement over CFRAC	174
12.3	Once More Unto the Breach	175
12.4	The Multiple Polynomial Quadratic Sieve	176
12.4.1	Yet One More Advantage	177
12.5	The Number Field Sieve	177
12.6	Exercises	178
	References	178
13	Cycles, Randomness, Discrete Logarithms, and Key Exchange	179
13.1	Introduction	179
13.2	The Discrete Logarithm Problem	180
13.3	Difficult Discrete Log Problems	181
13.4	Cycles	182
13.5	Cocks-Ellis-Williamson/Diffie-Hellman Key Exchange	182
13.5.1	The Key Exchange Algorithm	183
13.6	The Index Calculus	183
13.6.1	Our Example	184
13.6.2	Smooth Relations	184
13.6.3	Matrix Reduction	185
13.6.4	Individual Logarithms	187
13.6.5	Asymptotics	187
13.7	Key Exchange with Elliptic Curves	187
13.8	Key Exchange in Other Groups	188
13.9	How Hard Is the Discrete Logarithm Problem?	189
13.10	Exercises	189
	References	190

14	Elliptic Curve Cryptography	191
14.1	Introduction	191
14.1.1	Jacobian Coordinates	192
14.2	Elliptic Curve Discrete Logarithms	193
14.3	Elliptic Curve Cryptography	193
14.4	The Cost of Elliptic Curve Operations	194
14.4.1	Doubling a Point	194
14.4.2	Left-to-Right “Exponentiation”	195
14.5	The NIST Recommendations	196
14.6	Attacks on Elliptic Curves	198
14.6.1	Pohlig-Hellman Attacks	198
14.6.2	Pollard Rho Attacks	198
14.6.3	Pollard Rho for Curves	200
14.6.4	Pollard Rho in Parallel	201
14.7	A Comparison of Complexities	202
14.8	Exercises	202
	References	202
15	Lattice-Based Cryptography and NTRU	205
15.1	Quantum Computing	205
15.2	Lattices: An Introduction	207
15.3	Hard Lattice Problems	208
15.4	NTRU	209
15.5	The NTRU Cryptosystem	210
15.5.1	Parameters	210
15.5.2	Creating Keys	211
15.5.3	Encrypting a Message	211
15.5.4	Decrypting a Message	211
15.5.5	Why This Works	213
15.5.6	Preventing Errors in Decryption	213
15.6	Lattice Attacks on NTRU	214
15.7	The Mathematics of the Lattice Reduction Attack	216
15.7.1	Other Attacks on NTRU	217
15.7.2	Lattice Reduction	217
15.8	NTRU Parameter Choices	218
15.9	Exercises	220
	References	220
16	Homomorphic Encryption	223
16.1	Introduction	223
16.2	Somewhat Homomorphic Encryption	224
16.3	Fully Homomorphic Encryption	224
16.4	Ideal Lattices	224
16.5	Learning with Errors	227

16.6	Security, and Homomorphic Evaluation of Functions	228
16.7	An Apologetic Summary	228
	References	228
Appendix A: An Actual World War I Cipher		231
Appendix B: AES Code		253
Index		275