Notes for An Introduction to Mathematical Cryptography

Mohamed-Amine Azzouz

2024-05-05

Contents

Lecture 1: An Introduction to Cryptography

Introduction

Note: these are a set of notes to help me summarize my casual reading of the textbook of the same name.
The book presents some mathematical background that is necessary to understand cryptography's algorithms, starting with algebra and basic number theory. I will only list those that I forgot for my own learning.
For the sake of notation, Bob encrypts a message for Alice to read. Over an insecure channel, Eve is there to try to decrypt it.

1.1 Simple substitution ciphers

- Idea: to encrypt a message, replace each letter with another. Someone intercepting the message would need to try 26! combinations using brute force.
- In particular, one could shift each letter by a fixed amount (modulo 26), giving us the formula:

$$\text{(Ciphertext Letter)} \equiv_{26} \text{(Plaintext Letter)} + \text{(Secret Key)}$$

1.3 Modular Arithmetic

More notation: $a \equiv_m b$ is the same as saying $a \equiv b \pmod{m}$.

1.3.2 The Fast Powering Algorithm

Proposition

Step 1. Compute the binary expansion of A as

$$A = A_0 + A_1 \cdot 2 + A_2 \cdot 2^2 + A_3 \cdot 2^3 + \ldots + A_r \cdot 2^r$$

with $A_0, \ldots, A_r \in \{0, 1\}$, where we may assume that $A_r = 1$.

Step 2. Compute the powers $g^{2i} \pmod N$ for $0 \le i \le r$ by successive squaring,

$$\begin{aligned} a_0 &\equiv g \pmod N \\ a_1 &\equiv a_0^2 \pmod N \\ a_2 &\equiv a_1^2 \equiv g^2 \pmod N \\ a_3 &\equiv a_2^2 \equiv g^4 \pmod N \\ &\vdots \\ a_r &\equiv a_{r-1}^2 \equiv g^{2^r} \pmod N. \end{aligned}$$

Each term is the square of the previous one, so this requires r multiplications.
Step 3. Compute $g^A \pmod N$ using the formula

$$\begin{aligned} g^A &= g^{A_0 + A_1 \cdot 2 + A_2 \cdot 2^2 + A_3 \cdot 2^3 + \ldots + A_r \cdot 2^r} \\ &= g^{A_0} \cdot (g^2)^{A_1} \cdot (g^2)^{A_2} \cdot (g^2)^{A_3} \cdot \ldots \cdot (g^2)^{A_r} \\ &\equiv a_{A_0}^0 \cdot a_{A_1}^1 \cdot a_{A_2}^2 \cdot a_{A_3}^3 \cdot \ldots \cdot a_{A_r}^r \pmod N. \end{aligned}$$

Note that the quantities $a_0, a_1, \ldots, a_r$ were computed in Step 2. Thus the product can be computed by looking up the values of the a_i's whose exponent A_i is 1 and then multiplying them together. This requires at most another r multiplications.

1.4 Prime Numbers, Unique Factorization, and Finite Fields

Proposition: Method to compute a^{-1} mod p

Use the Euclidean Algorithm to compute $u, v \in \mathbb{Z}$ such that:

$$au + pv = 1$$

Then, $a^{-1} \equiv_p u$.

1.5 Powers and Primitive Roots in Finite Fields

Proposition: Fermat's Little Theorem

Let p be a prime number and let a be any integer. Then,

$$a^{p-1} \equiv \begin{cases} 1 \pmod{p} & \text{if } p \nmid a, \\ 0 \pmod{p} & \text{if } p \mid a. \end{cases}$$

As a corollary, $a^{-1} \equiv_p a^{p-2}$ for prime p, $p \nmid a$.

1.7 Symmetric and Asymmetric Ciphers

0.6.1 1.7.1 Symmetric/Private Ciphers

For each key k, we get a pair of functions $e_k : \mathcal{M} \to \mathcal{C}$ and $d_k : \mathcal{C} \to \mathcal{M}$ satisfying the decryption property $d_k(e_k(m)) = m$ for all $m \in \mathcal{M}$.
In other words, for every key k, the function d_k is the inverse function of the function e_k.
In particular, this means that e_k must be one-to-one, since if $e_k(m) = e_k(m')$, then

$$m = d_k(e_k(m)) = d_k(e_k(m')) = m'.$$

It is safest for Alice and Bob to assume that Eve knows the encryption method that is being employed, i.e. she knows the functions e and d.

Definition: Kerckhoff's Principle

The security of a cryptosystem should depend only on the secrecy of the key, and not on the secrecy of the encryption algorithm itself.

If (K, M, C, e, d) is to be a successful cipher, it must have the following properties:

1. For any key $k \in K$ and plaintext $m \in M$, it must be easy to compute the ciphertext $e_k(m)$.
2. For any key $k \in K$ and ciphertext $c \in C$, it must be easy to compute the plaintext $d_k(c)$.
3. It must be hard to decrypt any set of ciphertexts without the key k.

Another desirable but hard property to get is:

4. Security against a known plaintext attack: if Eve is given any pair of plaintexts and their encryptions, she can't use that to find the key. (The simple substitution cipher does not follow that)

An even more secure property is:

5. Security against a chosen plaintext attack: if Eve chooses to have any pair of plaintexts and their encryptions available, she can't use that to find the key.

0.6.2 1.7.5 Random Bit Sequences and Symmetric Ciphers

We would like to construct a mapping $R : \mathcal{K} \times \mathbb{Z} \to \{0,1\}$ such that with a relatively small key $k \in \mathcal{K}$, one can encrypt arbitrarily large messages $j \in \mathbb{Z}$. This is the goal of a pseudo-random number generator, for R to be one it needs to fulfill:

1. For all $k \in \mathcal{K}$ and all $j \in \mathbb{Z}$, it is easy to compute $R(k, j)$.
2. Given an arbitrarily long sequence of integers $j_1, j_2, \ldots, j_n$ and given all of the values $R(k, j_1), R(k, j_2), \ldots, R(k, j_n)$, it is hard to determine k.
3. Given any list of integers $j_1, j_2, \ldots, j_n$ and given all of the values $R(k, j_1), R(k, j_2), \ldots, R(k, j_n)$, it is hard to guess the value of $R(k, j)$ with better than a 50% chance of success for any value of j not already in the list.

If we could find a function R with these three properties, then we could use it to turn an initial key k into a sequence of bits $R(k,1), R(k,2), R(k,3), R(k,4), \ldots$, which would be the key to a one-time pad. Indeed, the one-time pad, which consists in encrypting a message by XORing it with a one-time use key of the same length, is provably secure but hard to use because the keys need to be the same size as the message.
No function has been proven to be a proper pseudorandom number generator. Some good candidates that have been working so far are algorithms that use some kind of "mixup" operations, similarly to what operations in modular arithmetic can induce. This is the basis of encryption protocols like the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES), which are widely used today. Another method is to choose R such that its inverse is known to be very difficult to compute. However, this method is not as efficient for practical use.

0.6.3 1.7.6 Public/Asymmetric Ciphers Make a First Appearance

Public key cryptography is a way for Alice and Bob to communicate over an insecure channel without having secretly exchanged keys beforehand. If in private key cryptography, Alice and Bob first safely exchanged private keys before communicating, the revolutionary idea of public key cryptography is to use a public key, made by Alice, with which absolutely anyone can encrypt messages to and send to Alice. However, only Alice had the private key to decrypt the messages.
A real life analogue to this would be Alice installing a safe in a public space, with a slot (the public key) into which anyone can insert messages. Then, only Alice would have the key to the safe (the private key) to open it and read the messages.

Definition: Public key cryptography

An element $k = (k_{\text{priv}}, k_{\text{pub}})$ of the key space $\mathcal{K}$ is such that any public key has an

encryption $e_{k_{\text{pub}}} : \mathcal{M} \to \mathcal{C}$, and each private key has a decryption $d_{k_{\text{priv}}} : \mathcal{C} \to \mathcal{M}$. Those maps are such that for any $k \in \mathcal{K}$, $d_{k_{\text{priv}}}(e_{k_{\text{pub}}}(m)) = m$ for all $m \in \mathcal{M}$.

For an asymmetric cipher to be secure, it must be difficult for Eve to compute the decryption function $d_{k_{\text{priv}}}(c)$, even if she knows the public key k_{pub}.
In practice, public key algorithms are much slower than private key cryptography, so what usually happens is that an asymmetric cipher is used to share a key for a symmetric cipher, which is how data is actually sent.

Lecture 2: Discrete Logarithms and Diffie–Hellman

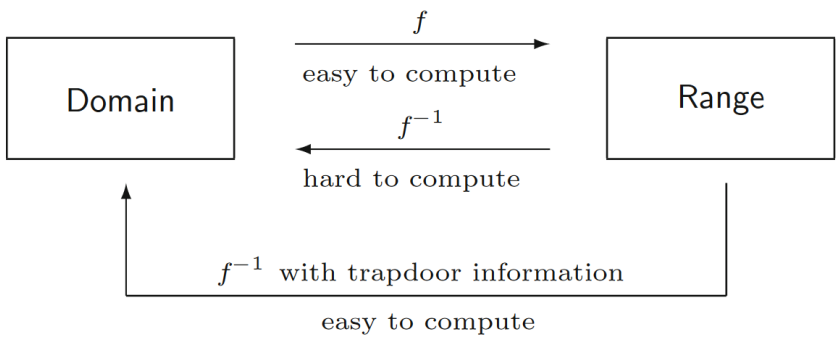

Figure 2.1: Illustration of a one-way trapdoor function

Idea: f is the encryption (sometimes using a public key), f^{-1} is the decryption that Eve tries to compute, with a trapdoor for those who have the key.

2.2 The Discrete Logarithm Problem

Definition: The Discrete Logarithm (DLP)

$g^x \equiv_p h$ always has a solution if p is prime, since in that case, $\mathbb{F}_p^* = \langle g \rangle$. When a solution exists, one can define the multivalued solution to be $x \equiv_{p-1} \log_g(h)$. The multivalue is from Fermat's Little Theorem.

Proposition

Notice that:

$$\log_p(ab) = \log_p(a) + \log_p(b).$$

Otherwise, the discrete logarithm is very hard to compute, since unlike the continuous logarithm, it has no monotone properties. In general, note that the problem can be stated in a general cyclic group.

2.3 Diffie–Hellman Key Exchange

- **Public parameter creation:**
  - A trusted party chooses and publishes a (large) prime p and an integer g having large prime order in F_p^*.

- **Private computations:**
  - Alice:
    - Choose a secret integer a.
    - Compute $A \equiv g^a \pmod{p}$.
  - Bob:
    - Choose a secret integer b.
    - Compute $B \equiv g^b \pmod{p}$.
- **Public exchange of values:**
  - Alice sends A to Bob.
  - Bob sends B to Alice.
- **Further private computations:**
  - Alice computes the number $B^a \pmod{p}$.
  - Bob computes the number $A^b \pmod{p}$.
- **Shared secret value:**
  - The shared secret value is $B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \pmod{p}$.

Definition: The Diffie-Hellman Problem

Let p be a prime number and g an integer. The Diffie–Hellman Problem (DHP) is the problem of computing the value of $g^{ab} \pmod{p}$ from the known values of $g^a \pmod{p}$ and $g^b \pmod{p}$.

2.4 The Elgamal Public Key Cryptosystem

One issue with Diffie-Hellman is that the secret key that is being shared is "random", in other words Alice and Bob have no true control over it. The Elgamal public key cryptostem allows Bob to send a crypted message to Alice, which she can recover.

Public parameter creation

- A trusted party chooses and publishes a large prime p and an element g modulo p of large (prime) order.

Key creation

- Alice:
  - Choose private key $1 \leq a \leq p-1$.

- Compute $A = g^a \pmod{p}$.
- Publish the public key A.

Encryption

- Bob:
  - Choose plaintext m.
  - Choose a random element k.
  - Use Alice's public key A to compute $c_1 = g^k \pmod{p}$ and $c_2 = m \cdot A^k \pmod{p}$.
  - Send ciphertext (c_1, c_2) to Alice.

Decryption

- Alice:
  - Compute $(c_1^a)^{-1} \cdot c_2 \pmod{p}$. This quantity is equal to m.

We know the Elgamal encryption is around as safe as the Diffie-Hellman problem.

Proposition

Fix a prime p and base g to use for Elgamal encryption. Suppose that Eve has access to an oracle that decrypts arbitrary Elgamal ciphertexts encrypted using arbitrary Elgamal public keys. Then she can use the oracle to solve the Diffie–Hellman problem. Conversely, if Eve can solve the Diffie–Hellman problem, then she can break the Elgamal PKC.

2.5 An Overview of the Theory of Groups

Definition

Let G be a group and let $a \in G$ be an element of the group. Suppose there exists a positive integer d with the property that $a^d = e$. The smallest such d is called the order of a. If there is no such d, then a is said to have infinite order.

Proposition

Let G be a finite group. Then every element of G has finite order. Further, if $a \in G$ has order d and if $a^k = e$, then $d \mid k$.

Proposition: Lagrange's Theorem

Let G be a finite group and let $a \in G$. Then the order of a divides the order (or size) of G. More precisely, let $n = |G|$ be the order of G and let d be the order of a, i.e., a^d is the smallest positive power of a that is equal to e. Then $a^n = e$ and $d \mid n$.

2.6 How Hard Is the Discrete Logarithm Problem?

Definition

Definition (Order Notation) Let $f(x)$ and $g(x)$ be functions of x taking values that are positive. We say that "f is big-O of g" and write $f(x) = \mathcal{O}(g(x))$ if there are positive constants c and C such that $f(x) \leq cg(x)$ for all $x \geq C$. In particular, we write $f(x) = \mathcal{O}(1)$ if $f(x)$ is bounded for all $x \geq C$. The next proposition gives a method that can sometimes be used to prove that $f(x) = \mathcal{O}(g(x))$.

Proposition

If the limit

$$\lim_{x \to \infty} \frac{f(x)}{g(x)}$$

exists (and is finite), then $f(x) = \mathcal{O}(g(x))$.

2.7 A Collision Algorithm for the DLP

Proposition: Trivial bound for DLP

Let G be a group and let $g \in G$ be an element of order N. (Recall that this means that $g^N = e$ and that no smaller positive power of g is equal to the identity element e.) Then the discrete logarithm problem

$$g^x = h$$

can be solved in $\mathcal{O}(N)$ steps and $\mathcal{O}(1)$ storage, where each step consists of multiplication by g.

Proposition: Shanks's Babystep–Giantstep Algorithm

Let G be a group and let $g \in G$ be an element of order $N \geq 2$. The following algorithm solves the discrete logarithm problem $g^x = h$ in $\mathcal{O}(\sqrt{N} \cdot \log N)$ steps using $\mathcal{O}(\sqrt{N})$ storage.

1. Let $n = 1 + \lfloor \sqrt{N} \rfloor$, so in particular, $n > \sqrt{N}$.
2. Create two lists,
   - List 1: $e, g, g^2, g^3, \ldots, g^n$,
   - List 2: $h, h \cdot g^{-n}, h \cdot g^{-2n}, h \cdot g^{-3n}, \ldots, h \cdot g^{-n^2}$.
3. Find a match between the two lists, say $g^i = h \cdot g^{-jn}$.
4. Then $x = i + jn$ is a solution to $g^x = h$.

2.8 The Chinese Remainder Theorem

We want to solve a system of congruences modulo integers that are coprime to each other.

Proposition: Chinese Remainder Theorem

Let $m_1, m_2, \ldots, m_k$ be a collection of pairwise relatively prime integers. This means that $\gcd(m_i, m_j) = 1$ for all $i \neq j$. Let $a_1, a_2, \ldots, a_k$ be arbitrary integers. Then the system of simultaneous congruences

$$\begin{aligned} x &\equiv a_1 \pmod{m_1}, \\ x &\equiv a_2 \pmod{m_2}, \\ &\vdots \\ x &\equiv a_k \pmod{m_k} \end{aligned} \tag{2.7}$$

has a solution $x = c$. Further, if $x = c$ and $x = c'$ are both solutions, then $c \equiv c' \pmod{m_1 m_2 \cdots m_k}$.

To solve system: rewrite one of the congruences as a proper equality, then substitute it in another congruence and invert to find the solution modulo the product of the moduli. Keep going with more and more congruences to have reduced to one solution.
In many cases, finding the modulo sqaure root is easy:

Proposition

Let p be a prime satisfying $p \equiv 3 \pmod 4$. Let a be an integer such that the congruence $x^2 \equiv a \pmod p$ has a solution, i.e., such that a has a square root modulo p. Then

$$b \equiv \pm a^{\frac{p+1}{4}} \pmod{p}$$

is a solution.

In the case where p is not prime, it's hard to check whether the number in question even has a square root. However, in the case the prime factorization is known, one can solve the system made of p's prime factors, and then use the Chinese Remainder Theorem, meaning the main barrier to solving the square root problem is prime factorizaiton. Note the problem is especially easy if the prime factorization is of many small primes.

2.9 The Pohlig–Hellman Algorithm

The main thing to remember from the Chinese Remainder Theorem is that a congruence problem modulo m made of relatively prime factors is the same as solving the problem with those factors and then applying the Chinese Remainder Theorem to group the congruences together.
We know that the DLP consists in solving $g^x \equiv_p h$. Since this implies that $x \equiv_{p-1} \log_g(h)$ and $p-1$ is composite, one can try to reduce the DLP by decomposing $p-1$ into prime factors, and then use the Chinese Remainder Theorem to the solutions of those easier problems. This means p must not be chosen such that $p-1$ has many small prime factors, else the DLP can easily be solved. This is all illustrated below:

Proposition: Pohlig–Hellman Algorithm

Let G be a group, and suppose that we have an algorithm to solve the discrete logarithm problem in G for any element whose order is a power of a prime. To be concrete, if $g \in G$ has order q^e, suppose that we can solve $g^x = h$ in $\mathcal{O}(S_{q^e})$ steps. (Note that $S_{qe} = q^{e/2}$)
Now let $g \in G$ be an element of order N, and suppose that N factors into a product of prime powers as

$$N = q_1^{e_1} \cdot q_2^{e_2} \cdot \ldots \cdot q_t^{e_t}.$$

Then the discrete logarithm problem $gx = h$ can be solved in

$$\mathcal{O}\left(\sum_{i=1}^{t} S_{{q_i}^{e_i}} + \log N\right)$$

steps using the following procedure:

1. For each $1 \leq i \leq t$, let $g_i = g^{N/{q_i}^{e_i}}$ and $h_i = h^{N/{q_i}^{e_i}}$. Notice that g_i has prime power order ${q_i}^{e_i}$, so use the given algorithm to solve the discrete logarithm problem $g^{y_i} = h_i$. Let $y = y_i$ be a solution to this problem.

2. Use the Chinese remainder theorem to solve the system

$$x \equiv_{q_1^{e_1}} y_1,\ x \equiv_{q_2^{e_2}} y_2,\ \ldots,\ x \equiv_{q_t^{e_t}} y_t.$$

One can also reduce the problem when m is the power of a prime into a subproblem with that prime. The idea is that if g has order q^e, then $g^{q^{e-1}}$ has order q:

Proposition

Let G be a group. Suppose that q is a prime, and suppose that we know an algorithm that takes S_q steps to solve the discrete logarithm problem $g^x = h$ in G whenever g has order q. Now let $g \in G$ be an element of order q^e with $e \geq 1$. Then we can solve the discrete logarithm problem $g^x = h$ in $\mathcal{O}(eS_q) = \mathcal{O}(e\sqrt{q})$ steps.

The algorithm to actually reduce such a DLP is the folllowing:

Proposition

- $h = g^x$ means that $h^{q^{e-1}} = (g^x)^{q^{e-1}}$, which is $g^{x_0 q^{e-1}} \cdot (g^{q^e})^{x_1 + x_2 q + \ldots + x_{e-1} q^{e-2}} = (g^{q^{e-1}})^{x_0}$. We solve for x_0 with the simpler DLP.
- Then recursively, knowing $x_0, x_1, \ldots, x_{i-1}$, we get x_i by solving $(g^{q^{e-1}})^{x_i} = (h \cdot g^{-x_0 - x_1 q - \ldots - x_{i-1} q^{i-1}})^{q^{e-i-1}}$ with the simpler DLP.
- Finally, we get $x = x_0 + x_1 q + x_2 q^2 + \ldots + x_{e-1} q^{e-1}$.

2.10 Rings, Quotient Rings, Polynomial Rings, and Finite Fields

This is not needed until Chapter 6 and 7. It's just about how the concepts of congruence, divisibility, and prime numbers (irreducibility) encountered before can be generalized to rings, polynomial rings, and finite fields.

Proposition: Random stuff about finite fields

(a) For every $d \geq 1$, there exists an irreducible polynomial $m \in \mathbb{F}_p[x]$ of degree d.

(b) For every $d \geq 1$, there exists a finite field with p^d elements.

(c) Finite fields of the same size are isomorphic.

Lecture 3: Integer Factorization and RSA

XXXXXXXXXXDDDD

3.1 Euler's Formula and Roots Modulo pq

XXX

3.2 The RSA Public Key Cryptosystem

XXX

3.3 Implementation and Security Issues

XXX

3.4 Primality Testing

XXX

3.5 Pollard's $p-1$ Factorization Algorithm

XXX

3.6 Factorization via Difference of Squares

XXX

3.7 Smooth Numbers and Sieves

XXX

3.8 The Index Calculus and Discrete Logarithms

XXX

3.9 Quadratic Residues and Quadratic Reciprocity

XXX

3.10 Probabilistic Encryption

XXX

Lecture 4: Digital Signatures

:3 XXXXXXXXXXDDDD

4.1 What Is a Digital Signature?

XXX

4.2 RSA Digital Signatures

XXX

4.3 Elgamal Digital Signatures and DSA

XXX

Lecture 5: Combinatorics, Probability, and Information Theory

:3 XXXXXXXXXXDDDD

5.1 Basic Principles of Counting

XXX

5.2 The Vigenere Cipher

XXX

5.3 Probability Theory

XXX

5.4 Collision Algorithms and Meet-in-the-Middle Attacks

XXX

5.5 Pollard's ρ Method

XXX

5.6 Information Theory

XXX

5.7 Complexity Theory and $\mathcal{P}$ Versus $\mathcal{NP}$

XXX

Lecture 6: Elliptic Curves and Cryptography

:3 X

6.1 Elliptic Curves

XXX

6.2 Elliptic Curves over Finite Fields

XXX

6.3 The Elliptic Curve Discrete Logarithm Problem

XXX

6.4 Elliptic Curve Cryptography

XXX

6.5 The Evolution of Public Key Cryptography

XXX

6.6 Lenstra's Elliptic Curve Factorization Algorithm

XXX

6.7 Elliptic Curves over $\mathbb{F}_2$ and over $\mathbb{F}_{2^k}$

XXX

6.8 Bilinear Pairings on Elliptic Curves

XXX

6.9 The Weil Pairing over Fields of Prime Power Order

XXX

6.10 Applications of the Weil Pairing

XXX

Lecture 7: Lattices and Cryptography

:3 X

7.1 A Congruential Public Key Cryptosystem

XXX

7.2 Subset-Sum Problems and Knapsack Cryptosystems

XXX

7.3 A Brief Review of Vector Spaces

XXX

7.4 Lattices: Basic Definitions and Properties

XXX

7.5 Short Vectors in Lattices

XXX

7.6 Babai's Algorithm

XXX

7.7 Cryptosystems Based on Hard Lattice Problems

XXX

7.8 The GGH Public Key Cryptosystem

XXX

7.9 Convolution Polynomial Rings

XXX

7.10 The NTRU Public Key Cryptosystem

XXX

7.11 NTRUEncrypt as a Lattice Cryptosystem

XXX

7.12 Lattice-Based Digital Signature Schemes

XXX

7.13 Lattice Reduction Algorithms

XXX

7.14 Applications of LLL to Cryptanalysis

XXX

Lecture 8: Additional Topics in Cryptography

:3 X

8.1 Hash Functions

XXX

8.2 Random Numbers and Pseudorandom Number

XXX

8.3 Zero-Knowledge Proofs

XXX

8.4 Secret Sharing Schemes

XXX

8.5 Identification Schemes

XXX

8.6 Padding Schemes and the Random Oracle Model

XXX

8.7 Building Protocols from Cryptographic Primitives

XXX

8.8 Blind Digital Signatures, Digital Cash, and Bitcoin

XXX

8.9 Homomorphic Encryption

XXX

8.10 Hyperelliptic Curve Cryptography

XXX

8.11 Quantum Computing

XXX

8.12 Modern Symmetric Cryptosystems: DES and AES

XXX

Bibliography

[1] J.H. Silverman, Jill Pipher, Jeffrey Hoffstein, *An Introduction to Mathematical Cryptography*, Springer New York, NY, Undergraduate Texts in Mathematics, 2008, DOI: https://doi.org/10.1007/978-0-387-77993-5, Copyright: Springer Science+Business Media, LLC, part of Springer Nature 2008, Softcover ISBN: 978-1-4419-2674-6 (Published: 01 December 2010), eBook ISBN: 978-0-387-77994-2 (Published: 15 December 2008), Edition: 1, Pages: XVI, 524, Illustrations: 29 b/w illustrations, Topics: Number Theory, Coding and Information Theory, Data Structures and Information Theory, Cryptology, Information and Communication, Circuits, Order, Lattices, Ordered Algebraic Structures, ISSN: 0172-6056, E-ISSN: 2197-5604