Chapter 1

Introduction

Cryptography is an interdisciplinary field of great practical importance. The subfield of public key cryptography has notable applications, such as digital signatures. The security of a public key cryptosystem depends on the difficulty of certain computational problems in mathematics. A deep understanding of the security and efficient implementation of public key cryptography requires significant background in algebra, number theory and geometry.

This book gives a rigorous presentation of most of the mathematics underlying public key cryptography. Our main focus is mathematics. We put mathematical precision and rigour ahead of generality, practical issues in real-world cryptography, or algorithmic optimality. It is infeasible to cover all the mathematics of public key cryptography in one book. Hence we primarily discuss the mathematics most relevant to cryptosystems that are currently in use, or that are expected to be used in the near future. More precisely, we focus on discrete logarithms (especially on elliptic curves), factoring based cryptography (e.g., RSA and Rabin), lattices and pairings. We cover many topics that have never had a detailed presentation in any textbook.

Due to lack of space, some topics are not covered in as much detail as others. For example, we do not give a complete presentation of algorithms for integer factorisation, primality testing, and discrete logarithms in finite fields, as there are several good references for these subjects. Some other topics that are not covered in the book include hardware implementation, side-channel attacks, lattice-based cryptography, cryptosystems based on coding theory, multivariate cryptosystems and cryptography in non-Abelian groups. In the future, quantum cryptography or post-quantum cryptography (see the book [50] by Bernstein, Buchmann and Dahmen) may be used in practice, but these topics are also not discussed in the book.

The reader is assumed to have at least a standard undergraduate background in groups, rings, fields and cryptography. Some experience with algorithms and complexity is also assumed. For a basic introduction to public key cryptography and the relevant mathematics the reader is recommended to consult Smart [572], Stinson [592] or Vaudenay [616].

An aim of the present book is to collect in one place all the necessary background and results for a deep understanding of public key cryptography. Ultimately, the text presents what I believe is the "core" mathematics required for current research in public key cryptography and it is what I would want my PhD students to know.

The remainder of this chapter states some fundamental definitions in public key cryptography and illustrates them using the RSA cryptosystem.

1.1 Public Key Cryptography

Two fundamental goals of cryptography are to provide privacy of communication between two entities and to provide authentication of one entity to another. Both goals can be achieved with symmetric cryptography. However, symmetric cryptography is not convenient in some applications for the following reasons. First, each pair of communicating entities needs to have a shared key. Second, these keys must be transmitted securely. Third, it is difficult to obtain signatures with non-repudiation (e.g., suitable for signing contracts).

In the mid 1970s, Merkle, Diffie and Hellman proposed the idea of **public key cryptography** (also sometimes called **asymmetric cryptography**). This idea was also proposed by Ellis at GCHQ, under the name "non-secret encryption". One of the earliest and most important public key cryptosystems is RSA, invented by Adleman, Rivest and Shamir in 1977 (essentially the same scheme was also invented by Cocks at GCHQ in 1973).

As noted above, a major application of public key cryptography is to provide authentication. An extremely important example of this in the real world is digital signatures for authenticating automatic software updates. The public key of the software developer is stored in the application or operating system and the software update is only performed if the digital signature on the update is verified for that public key (see Section 12.1 of Katz and Lindell [334] for more details). Signature schemes also provide message integrity, message authentication and non-repudiation (see Section 10.2 of Smart [572]). Other important applications of public key cryptography are key exchange and key transport for secure communication (e.g., in SSL or TLS).

1.2 The Textbook RSA Cryptosystem

We briefly describe the "textbook" RSA cryptosystem. The word "textbook" indicates that, although the RSA cryptosystem as presented below appears in many papers and books, this is definitely not how it should be used in the real world. In particular, public key encryption is most commonly used to transmit keys (the functionality is often called key transport or key encapsulation), rather than to encrypt data. Chapter 24 gives many more details about RSA including, in Section 24.7, a very brief discussion of padding schemes for use in real applications.

Alice chooses two large primes p and q of similar size and computes $N = pq$. Alice also chooses $e \in \mathbb{N}$ coprime to $\varphi(N) = (p-1)(q-1)$ and computes $d \in \mathbb{N}$ such that

$$ed \equiv 1 \pmod{\varphi(N)}.$$

Alice's RSA **public key** is the pair of integers (N, e) and her **private key** is the integer d. To **encrypt** a message to Alice, Bob does the following:

1. Obtain an authentic copy of Alice's public key (N, e). This step may require trusted third parties and public key infrastructures, which are outside the scope of this book;

see Chapter 12 of Smart [572] or Chapter 12 of Stinson [592]. We suppress this issue in the book.

2. Encode the message as an integer $1 \leq \mathsf{m} < N$.

   Note that m does not necessarily lie in $(\mathbb{Z}/N\mathbb{Z})^*$. However, if $p, q \approx \sqrt{N}$ then the probability that $\gcd(\mathsf{m}, N) > 1$ is $(p+q-1)/(N-1) \approx 2/\sqrt{N}$. Hence, in practice one may assume that $\mathsf{m} \in (\mathbb{Z}/N\mathbb{Z})^*$.[1]

3. Compute and transmit the **ciphertext**

$$\mathsf{c} = \mathsf{m}^e \pmod{N}.$$

To **decrypt** the ciphertext, Alice computes $\mathsf{m} = \mathsf{c}^d \pmod{N}$ and decodes this to obtain the original message.

Exercise 1.2.1. Show that if $\gcd(\mathsf{m}, N) = 1$ then $(\mathsf{m}^e)^d \equiv \mathsf{m} \pmod{N}$. Show that if $\gcd(\mathsf{m}, N) \neq 1$ then $(\mathsf{m}^e)^d \equiv \mathsf{m} \pmod{N}$.

The RSA system can also be used as a digital signature algorithm. When sending a message m to Bob, Alice computes the signature $\mathsf{s} = \mathsf{m}^d \pmod{N}$. When Bob receives (m, s) he obtains an authentic copy of Alice's public key and then verifies that $\mathsf{m} \equiv \mathsf{s}^e \pmod{N}$. If the verification equation holds then Bob believes that the message m does come from Alice. The value m is not usually an actual message or document (which might be huge) but a short integer that is the output of some (non-injective) compression function (such as a hash function). We sometimes call m a **message digest**.

The idea is that exponentiation to the power e modulo N is a **one-way function**: a function that is easy to compute but such that it is hard to compute pre-images. Indeed, exponentiation modulo N is a **one-way permutation** on $(\mathbb{Z}/N\mathbb{Z})^*$ when e is co-prime to $\varphi(N)$. The private key d allows the permutation to be efficiently inverted and is known as a **trapdoor**. Therefore RSA is often described as a **trapdoor one-way permutation**.

A number of practical issues must be considered:

1. Can public keys be efficiently generated?
2. Is the cryptosystem efficient in the sense of computation time and ciphertext size?
3. How does Bob know that Alice's public key is authentic?
4. Is the scheme secure?
5. What does "security" mean anyway?

One aim of this book is to explore the above issues in depth. We will study RSA (and some other cryptosystems based on integer factorisation) as well as cryptosystems based on the discrete logarithm problem.

To indicate some of the potential problems with the "textbook" RSA cryptosystem as described above, we present a three simple attacks.

1. Suppose the RSA cryptosystem is being used for an online election to provide privacy of an individual's vote to everyone outside the electoral office.[2] Each voter encrypts

[1] If N is a product of two 150 digit primes (which is the minimum size for an RSA modulus) then the expected number of trials to find $1 \leq \mathsf{m} < N$ with $\gcd(\mathsf{m}, N) > 1$ is therefore $\approx 10^{150}$. Note that the age of the universe is believed to be less than 10^{18} seconds.

[2] Much more interesting electronic voting schemes have been invented. This unnatural example is chosen purely for pedagogical purposes.

their vote under the public key of the electoral office and then sends their vote by email. Voters don't want any other member of the public to know who they voted for.

Suppose the eavesdropper Eve is monitoring internet traffic from Alice's computer and makes a copy of the ciphertext corresponding to her vote. Since encryption is deterministic and there is only a short list of possible candidates, it is possible for Eve to compute each possible vote by encrypting each candidate's name under the public key. Hence, Eve can deduce who Alice voted for.

2. To speed up encryption it is tempting to use small encryption exponents, such as $e = 3$ (assuming that $N = pq$ where $p \equiv q \equiv 2 \pmod 3$). Now suppose Bob is only sending a very small message $0 < \mathsf{m} < N^{1/3}$ to Alice; this is quite likely, since public key cryptography is most often used to securely transmit symmetric keys. Then $\mathsf{c} = \mathsf{m}^3$ in $\mathbb{N}$, i.e., no modular reduction has taken place. An adversary can therefore compute the message m from the ciphertext c by taking cube roots in $\mathbb{N}$ (using numerical analysis techniques).

3. A good encryption scheme should allow an adversary to learn absolutely nothing about a message from the ciphertext. But with the RSA cryptosystem one can compute the Jacobi symbol $(\frac{\mathsf{m}}{N})$ of the message by computing $(\frac{\mathsf{c}}{N})$ (this can be computed efficiently without knowing the factorisation of N; see Section 2.4). The details are Exercise 24.1.11.

The above three attacks may be serious attacks for some applications, but not for others. However, a cryptosystem designer often has little control over the applications in which their system is to be used. Hence it is preferable to have systems that are not vulnerable to attacks of the above form. In Section 24.7 we will explain how to secure RSA against these sorts of attacks, by making the encryption process randomised and by using padding schemes that encode short messages as sufficiently large integers and that destroy algebraic relationships between messages.

1.3 Formal Definition of Public Key Cryptography

To study public key cryptography using mathematical techniques it is necessary to give a precise definition of an encryption scheme. The following definition uses terminology about algorithms that is recalled in Section 2.1. Note that the problem of obtaining an authentic copy of the public key is not covered by this definition; the public key is an input to the encryption function.

Definition 1.3.1. Let $\kappa \in \mathbb{N}$ be a **security parameter** (note that κ is not necessarily the same as the "key length"; see Example 1.3.2). An **encryption scheme** is defined by the following spaces (all depending on the security parameter κ) and algorithms.

M_κ	the space of all possible messages;
PK_κ	the space of all possible public keys;
SK_κ	the space of all possible private keys;
C_κ	the space of all possible ciphertexts;
KeyGen	a randomised algorithm that takes the security parameter κ, runs in expected polynomial-time (i.e., $O(\kappa^c)$ bit operations for some constant $c \in \mathbb{N}$) and outputs a public key $\mathsf{pk} \in \mathsf{PK}_\kappa$ and a private key $\mathsf{sk} \in \mathsf{SK}_\kappa$;
Encrypt	a randomised algorithm that takes as input $\mathsf{m} \in \mathsf{M}_\kappa$ and pk, runs in expected polynomial-time (i.e., $O(\kappa^c)$ bit operations for some constant

$c \in \mathbb{N}$) and outputs a ciphertext $\mathsf{c} \in \mathsf{C}_\kappa$;

Decrypt an algorithm (not usually randomised) that takes $\mathsf{c} \in \mathsf{C}_\kappa$ and sk, runs in polynomial-time and outputs either $\mathsf{m} \in \mathsf{M}_\kappa$ or the invalid ciphertext symbol $\perp$.

It is required that

$$\text{Decrypt}(\text{Encrypt}(\mathsf{m}, \mathsf{pk}), \mathsf{sk}) = \mathsf{m}$$

if $(\mathsf{pk}, \mathsf{sk})$ is a matching key pair. Typically we require that the fastest known attack on the system requires at least 2^κ bit operations.

Example 1.3.2. We sketch how to write "textbook" RSA encryption in the format of Definition 1.3.1. The KeyGen algorithm takes input κ and outputs a modulus N that is a product of two randomly chosen primes of a certain length, as well as an encryption exponent e.

Giving a precise recipe for the bit-length of the primes as a function of the security parameter is non-trivial for RSA. The complexity of the best factoring algorithms implies that we need $2^\kappa \approx L_N(1/3, c)$ for some constant c (see Chapter 15 for this notation and an explantion of factoring algorithms). This implies that $\log(N) = O(\kappa^3)$ and so the bit-length of the public key is bounded by a polynomial in κ. A typical benchmark is that if $\kappa = 128$ (i.e., so that there is no known attack on the system performing fewer than 2^{128} bit operations) then N is a product of two 1536-bit primes.

As we will discuss in Chapter 12, one can generate primes in expected polynomial-time and hence KeyGen is a randomised algorithm with expected polynomial-time complexity.

The message space M_κ depends on the randomised padding scheme being used. The ciphertext space C_κ in this case is $(\mathbb{Z}/N\mathbb{Z})^*$, which does not agree with Definition 1.3.1 as it does not depend only on κ. Instead one usually takes C_κ to be the set of $\lceil \log_2(N) \rceil$-bit strings.

The Encrypt and Decrypt algorithms are straightforward (though the details depend on the padding scheme). The correctness condition is easily checked.

1.3.1 Security of Encryption

We now give precise definitions for the security of public key encryption. An **adversary** is a randomised polynomial-time algorithm that interacts with the cryptosystem in some way. It is necessary to define the **attack model**, which specifies the way the adversary can interact with the cryptosystem. It is also necessary to define the **attack goal** of the adversary. For further details of these issues see Sections 10.2 and 10.6 of Katz and Lindell [334], Section 1.13 of Menezes, van Oorschot and Vanstone [418], or Section 15.1 of Smart [572].

We first list the **attack goals for public key encryption**. The most severe one is the **total break**, where the adversary computes a private key. There are three other commonly studied attacks, and they are usually formulated as **security properties** (the security property is the failure of an adversary to achieve its attack goal).

The word **oracle** is used below. This is just a fancy name for a magic box that takes some input and then outputs the correct answer in constant time. Precise definitions are given in Section 2.1.3.

- **One way encryption (OWE):** Given a challenge ciphertext c the adversary cannot compute the corresponding message m.
- **Semantic security:** An adversary learns no information at all about a message from its ciphertext, apart from possibly the length of the message.

This concept is made precise as follows: Assume all messages in M_κ have the same length. A **semantic security adversary** is a randomised polynomial-time algorithm A that first chooses a function $f : \mathsf{M}_\kappa \to \{0,1\}$ such that the probability, over uniformly chosen $\mathsf{m} \in \mathsf{M}_\kappa$, that $f(\mathsf{m}) = 1$ is $1/2$. The adversary A then takes as input a challenge $(\mathsf{c}, \mathsf{pk})$, where c is the encryption of a random message $\mathsf{m} \in \mathsf{M}_\kappa$, and outputs a bit b. The adversary is **successful** if $b = f(\mathsf{m})$.

Note that the standard definition of semantic security allows messages $\mathsf{m} \in \mathsf{M}_\kappa$ to be drawn according to any probability distribution. We have simplified to the case of the uniform distribution on M_κ.

- **Indistinguishability (IND):** An adversary cannot distinguish the encryption of any two messages m_0 and m_1, chosen by the adversary, of the same length.

  This concept is made precise by defining an **indistinguishability adversary** to be a randomised polynomial-time algorithm A that plays the following game with a challenger: First the challenger generates a public key and gives it to A. Then (this is the "first phase" of the attack) A performs some computations (and possibly queries to oracles) and outputs two equal length messages m_0 and m_1. The challenger computes the **challenge ciphertext** c (which is an encryption of m_b where $b \in \{0,1\}$ is randomly chosen) and gives it to A. In the "second phase" the adversary A performs more calculations (and possibly oracle queries) and outputs a bit b'. The adversary is **successful** if $b = b'$.

For a fixed value κ one can consider the probability that an adversary is successful over all public keys pk output by KeyGen, and (except when studying a total break adversary) all challenge ciphertexts c output by Encrypt, and over all random choices made by the adversary. The adversary breaks the security property if the success probability of the adversary is noticeable as a function of κ (see Definition 2.1.10 for the terms noticeable and negligible). The cryptosystem achieves the security property if every polynomial-time adversary has negligible success probability as a function of κ. An adversary that works with probability 1 is called a **perfect adversary**.

We now list the three main attack models for public key cryptography.

- **Passive attack/chosen plaintext attack (CPA):** The adversary is given the public key.
- **Lunchtime attack (CCA1):**[3] The adversary has the public key and can also ask for decryptions of ciphertexts of its choosing during the first stage of the attack (i.e., before the challenge ciphertext is received).
- **Adaptive chosen-ciphertext attack (CCA):** (Also denoted CCA2.) The adversary has the public key and is given access to a decryption oracle O that will provide decryptions of any ciphertext of its choosing, with the restriction that O outputs $\perp$ in the second phase of the attack if the challenge ciphertext is submitted to O.

One can consider an adversary against any of the above security properties in any of the above attack models. For example, the strongest security notion is indistinguishability under an adaptive chosen ciphertext attack. A cryptosystem that achieves this security level is said to have **IND-CCA security**. It has become standard in theoretical cryptography to insist that all cryptosystems have IND-CCA security. This is not because

[3]The name comes from an adversary who breaks into someone's office during their lunch break, interacts with their private key in some way, and then later in the day tries to decrypt a ciphertext.

CCA attacks occur frequently in the real world, but because a scheme that has IND-CCA security should also be secure against any real-world attacker.[4]

Exercise 1.3.3. Show that the "textbook" RSA cryptosystem does not have IND-CPA security.

Exercise 1.3.4. Show that the "textbook" RSA cryptosystem does not have OWE-CCA security.

Exercise 1.3.5. Prove that if a cryptosystem has IND security under some attack model then it has semantic security under the same attack model.

1.3.2 Security of Signatures

Definition 1.3.6. A **signature scheme** is defined, analogously to encryption, by message, signature and key spaces depending on a security parameter κ. There is a KeyGen algorithm and algorithms:

Sign	A randomised algorithm that runs in polynomial-time (i.e., $O(\kappa^c)$ bit operations for some constant $c \in \mathbb{N}$), takes as input a message m and a private key sk, and outputs a signature s.
Verify	An algorithm (usually deterministic) that runs in polynomial-time, takes as input a message m, a signature s and a public key pk, and outputs "valid" or "invalid".

We require that $\text{Verify}(\mathsf{m}, \text{Sign}(\mathsf{m}, \mathsf{sk}), \mathsf{pk}) =$ "valid". Typically, we require that all known algorithms to break the signature scheme require at least 2^κ bit operations.

The main **attack goals for signatures** are the following (for more discussion see Goldwasser, Micali and Rivest [258], Section 12.2 of Katz and Lindell [334], Section 15.4 of Smart [572], or Section 7.2 of Stinson [592]):

- **Total break:** An adversary can obtain the private key for the given public key.
- **Selective forgery:** (Also called **target message forgery**.) An adversary can generate a valid signature for the given public key on any message.
- **Existential forgery:** An adversary can generate a pair (m, s) where m is a message and s is a signature for the given public key on that message.

  The acronym **UF** stands for the security property "unforgeable". In other words, a signature scheme has UF security if every polynomial-time existential forgery algorithm succeeds with only negligible probability. Be warned that some authors use UF to denote "universal forgery", which is another name for selective forgery.

As with encryption there are various attack models.

- **Passive attack:** The adversary is given the public key only. This is also called a "public key only" attack.
- **Known message attack:** The adversary is given various sample message-signature pairs for the public key.

[4] Of course, there are attacks that lie outside the attack model we are considering, such as side-channel attacks or attacks by dishonest system administrators.

- **Adaptive chosen-message attack (CMA):** The adversary is given a signing oracle that generates signatures for the public key on messages of their choosing.

  In this case, **signature forgery** usually means producing a valid signature s for the public key pk on a message m such that m was not already queried to the signing oracle for key pk. Another notion, which we do not consider further in this book, is **strong forgery**; namely to output a valid signature s on m for public key pk such that s is not equal to any of the outputs of the signing oracle on m.

As with encryption, one says the signature scheme has the stated security property under the stated attack model if there is no polynomial-time algorithm A that solves the problem with noticeable success probability under the appropriate game. The standard notion of security for digital signatures is **UF-CMA** security.

Exercise 1.3.7. Give a precise definition for UF-CMA security.

Exercise 1.3.8. Do "textbook" RSA signatures have selective forgery security under a passive attack?

Exercise 1.3.9. Show that there is a passive existential forgery attack on "textbook" RSA signatures.

Exercise 1.3.10. Show that, under a chosen-message attack, one can selective forge "textbook" RSA signatures.