Chapter 9

Mathematical Models in Public-Key Cryptology

{draft 5/26/99}

Joel Brawley
Shuhong Gao

Prerequisites: linear and modern algebra, elementary number theory

9.1 Theory and Models

Chapter 8 has described several of the classical models of cryptography in which the decryption key was the same as or easily derivable from the encryption key. This meant that the corresponding encryption and decryption algorithms were closely related in the sense that one could be easily deduced from the other. Such cryptographic systems are called *symmetric-key* or *conventional* systems, and their security relies exclusively on the secrecy of the keys. Other examples of private-key systems are the Data Encryption Standard (DES) [24] and IDEA [12], in which users of the system who share a secret key can communicate securely over an unsecure channel. In all of the private-key systems, two users who wish to correspond must have a common key *before* the communication starts, and in practice, establishing a common secret key can be expensive, difficult, and sometimes nearly impossible, especially in a large network where the users need not know each other.

In 1976, Diffie and Hellman [7] introduced a revolutionary new concept called *public-key cryptography* based on the simple observation that the encryption and decryption could be separated; i.e., they recognized that a knowledge of the encryption key (or equivalently, the encryption algorithm) need not imply a knowledge of the decryption key (or algorithm). In such a system, the encryption key can be made public, say in a public directory, while the decryption key can be kept secret. Anyone wishing to send a message to a person in the directory can simply look up the public encryption key for that person and use it to encrypt the message. Then, assuming the decryption key is known only to the intended receiver of the message, only that person can decrypt the message.

Of course in such a public-key system it must be computationally infeasible to deduce the decryption key (or the decryption algorithm) from the public key (or the public encryption algorithm), even when general information about the system and how it operates is known. This leads to the idea of one-way functions.

A function f is called a *one-way function* if for any x in the necessarily large domain of f, $f(x)$ can be efficiently computed but for virtually all y in the range of f, it is computationally infeasible to find any x such that $f(x) = y$.

Public-key cryptography requires a special set of one-way functions $\{E_k : k \in \mathcal{K}\}$ where $\mathcal{K}$, the so-called *key space*, is a large set of possible *keys*, and E_k is a map from a plaintext space $\mathcal{M}_k$ to a ciphertext space $\mathcal{C}_k$. The one-way nature of E_k implies that for virtually all ciphertexts $c = E_k(m)$ it is computationally infeasible to recover the plaintext m from a given k and c. However, since the legitimate recipient of the message must be able to recover m from c, more is required of these one-way functions. Specifically, each E_k must have an inverse D_k, and this inverse must be easily obtainable given some additional secret information d. The extra information d is called a *trap-door* of E_k and the functions E_k themselves are called *trap-door one-way functions*. It is also required that, with a knowledge of D_k, $m = D_k(c)$ be easy to compute for all c in the ciphertext space. Thus, a public-key cryptosystem consists of a family of trap-door one-way functions.

Before proceeding, we will make a few remarks on the length of messages. For a given key k, the function E_k usually acts only on plaintexts of fixed length whereas, in practice, a message can be of arbitrary length. However, the message can be cut into appropriate pieces, called blocks, so that E_k can act on each block. The whole message is then encrypted by encrypting each block individually. This operating mode is called the *Electronic Code Book* (ECB) mode. (Other operating modes include *Cipher Block Chaining* (CBC) mode, *Cipher Feedback* (CFB) mode, and *Output Feedback* (OFB) mode [24].) The point here is that the plaintext space (i.e., the domain of E_k) may be finite but a message of arbitrary length can be encrypted using E_k.

To summarize we give our first model of public-key cryptography.

DEFINITION 9.1 *A* **(deterministic) public-key cryptosystem** *consists of the following components:*

1. *A set $\mathcal{K}$ called the key space whose elements are called keys.*
2. *A rule by which each $k \in \mathcal{K}$ is associated with a trap-door one-way function E_k with domain $\mathcal{M}_k$ (the plaintext space) and range $\mathcal{C}_k$ (the ciphertext space).*
3. *A procedure for generating a random key $k \in \mathcal{K}$ together with a trap-door d for E_k and the inverse map $D_k : \mathcal{C}_k \longrightarrow \mathcal{M}_k$ such that*

$$D_k(E_k(m)) = m, \text{ for all } m \in \mathcal{M}_k.$$

The key space $\mathcal{K}$ is also called the *public-key space*, and the set of trap-doors d is called the *private-key space*. Relative to (3), it is also required that random keys $k \in \mathcal{K}$ and their corresponding trapdoors d be easy to generate.

In practice, the complete description of all the components (1)–(3) of a cryptosystem is public knowledge. A person (user) who wants to become a part of the communication network can proceed as follows:

- Use (3) to generate a random key $k \in \mathcal{K}$ and the corresponding trap-door d.
- Place the encryption function E_k (or equivalently the key k) in a public directory (say in the user's directory or home page), keeping d and the decryption function D_k secret.

Now suppose that Bob wants to send a message m to a user Alice. To do this, he simply looks up her public enciphering function E_{k_A} and computes $c = E_{k_A}(m)$ which he sends to Alice. On receiving c, Alice computes

$$D_{k_A}(c) = D_{k_A}(E_{k_A}(m)) = m,$$

thereby recovering the message. An eavesdropper might intercept c and can obtain E_{k_A} from public files, but cannot find m from c without knowledge of d_A (or equivalently D_{k_A}).

Actually, it is currently unknown as to whether one-way functions truly exist; indeed, a proof of the existence of such functions would settle the famous **P**=**NP** problem of computer science. However, there are a number of functions that are believed to be one-way. For example, it is assumed by experts that integer multiplication is a one-way function because it is very easy to multiply large integers, but it seems very hard to factor large numbers using the current knowledge and technology. This assumption is the basis for several public-key cryptosystems including the RSA and Rabin cryptosystems, discussed in Section 9.2.

In the above model of public-key cryptography, two identical plaintext messages m are always encrypted into the same ciphertext $c = E_k(m)$, and in the ECB mode, this feature can cause some leaking of information. For example, if the cryptosystem were used to encrypt personnel data and the salary fields were encrypted separately, then by simply looking at the ciphertexts one could identify people with the same salary. A natural question arises: Is it possible to design a public-key system in which such identical plaintexts are encrypted to different ciphertexts? Surprisingly, the answer is yes! The idea is to use random numbers (also referred to as redundant information or nonces).

To explain this idea more fully, suppose with each $k \in \mathcal{K}$ there is also associated a large set $\mathcal{R}_k$, called a *randomization set*, and a map

$$E_k : \mathcal{M}_k \times \mathcal{R}_k \longrightarrow \mathcal{C}_k.$$

In order to encrypt a plaintext $m \in \mathcal{M}_k$ with such a map, one picks a random number $r \in \mathcal{R}_k$ and computes the ciphertext

$$c = E_k(m, r).$$

Consequently, in the ECB mode of operation, a sequence of plaintext blocks $m_1, \ldots, m_t$ will be encrypted to a sequence $c_1, \ldots, c_t$, where $c_i = E_k(m_i, r_i)$ and each r_i is chosen independently and randomly from $\mathcal{R}_k$ for $i = 1, \ldots, t$. The set $\mathcal{R}_k$ is usually large, so the chance of picking the same r is small and the ciphertext values c_i will generally be different even when the corresponding plaintext blocks are identical.

Here again, just as in our first model of public-key cryptography, each E_k is required to be a one-way function with a trap door, but it need not be fully invertible since the recipient does not need to recover r. What is needed for decryption is that E_k be partially invertible; i.e., there needs to exist a function $D_k : \mathcal{C}_k \longrightarrow \mathcal{M}_k$ such that $D_k(E_k(m, r)) = m$, for all $m \in \mathcal{M}_k$ and all $r \in \mathcal{R}_k$. The function D_k is called a partial inverse of E_k since it recovers only part of the input to E_k. We call a one-way function E_k with this property a *partial-trap-door one-way function*, and we give our second model of public-key cryptography.

DEFINITION 9.2 *A* **probabilistic public-key cryptosystem** *consists of the following:*

1. *A set $\mathcal{K}$ called the key space whose elements are called keys.*

2. *A rule by which each $k \in \mathcal{K}$ is associated with a partial-trap-door one-way function E_k with domain $\mathcal{M}_k \times \mathcal{R}_k$ and range $\mathcal{C}_k$. (Here, $\mathcal{M}_k$ is called the plaintext space, $\mathcal{R}_k$ the randomization set, and C_k the ciphertext space.)*
3. *A procedure for generating a random key $k \in \mathcal{K}$ together with a partial-trap-door d for E_k and the map $D_k : \mathcal{C}_k \longrightarrow \mathcal{M}_k$ such that*

$$D_k(E_k(m, r)) = m, \text{ for all } m \in \mathcal{M}_k,\ r \in \mathcal{R}_k.$$

Again, the elements $k \in \mathcal{K}$ are called the public keys and the partial-trap-doors d the private keys. Obviously, the deterministic model is a special case of the probabilistic model in which $\mathcal{R}_k$ has only one element. In order for the probabilistic model to be useful and secure, the following properties are needed.

P1. Given a public key k it is easy to compute $E_k(m, r)$ for $m \in \mathcal{M}_k$ and $r \in \mathcal{R}_k$.

P2. Given a private key d, it is easy to compute $D_k(c)$ for $c \in \mathcal{C}_k$.

P3. Knowing k and $c \in \mathcal{C}_k$ it is infeasible to decide for any $m \in \mathcal{M}_k$ whether m can be encrypted to c under E_k. Thus, it is infeasible to determine D_k or d from the general information about the cryptosystem.

P4. It is easy to generate a random key $k \in \mathcal{K}$ and the corresponding private key d.

In the deterministic model it was required that it be infeasible to determine m from a knowledge of only E_k and c (as E_k is one-way). The corresponding requirement P3 for the probabilistic model is much stronger because if one can not even decide whether a plaintext m can be encrypted to a given c, then certainly one can not find a plaintext that can be encrypted to c. In this connection we note that in the deterministic model, it is trivial to decide if a plaintext $m \in \mathcal{M}_k$ can be encrypted to a given ciphertext $c \in \mathcal{C}_k$, as one can simply compute $E_k(m)$ and check whether or not it is c. Requirement P3 also implies that even when an adversary has a potentially matched pair (m, c) of plaintext and ciphertext, he or she can not even verify that there exists $r \in \mathcal{R}_k$ such that $E_k(m, r) = c$. Therefore, a probabilistic cryptosystem can provide a higher level of security than a deterministic one.

The first probabilistic public-key cryptosystem was given by McEliece in 1978 (see [24]). In this system, the trap-door is based on the fact that the encoding process for error-correcting codes can be easy whereas decoding can be hard. In 1985 ElGamal [8] proposed a probabilistic system whose trap-door is based on the fact that exponentiation in a finite field (to be described later) is easy, but the inverse process, the so-called discrete logarithm problem, can be hard. The ElGamal system is a modification of the Diffie-Hellman key exchange scheme [7], whose security is also based on the discrete logarithm problem. Both the ElGamal and the Diffie-Hellman systems will be discussed in Section 9.3.

We close our general discussion of cryptographic models with a few remarks concerning a further generalization of property P3 which required that it be hard to decide whether a given plaintext m can be encrypted to a given ciphertext c. Even under this condition, there can still be leaking of information. More specifically, it can happen that the probability that m is encrypted to a given c is significantly different for different values of m; consequently, for a given c it is possible to infer some partial probabilistic information about the plaintext space. To avoid such leaking, one can replace P3 by the following stronger condition.

P3′. Given k and $c = E_k(m, r) \in \mathcal{C}_k$, it is infeasible to infer any partial information about m.

While we shall not describe any systems with property P3′, we do give a few references. The first probabilistic cryptosystem proven to satisfy P3′ was given by Goldwasser and Micali [10], and later Blum and Goldwasser [4] gave a more efficient system.

In the discussions above we have tacitly assumed that the adversary was passive in that he or she could only eavesdrop on a communication. However, if the adversary were active and could inject or alter messages, then some systems (e.g., the Rabin and ElGamal systems, discussed later) are vulnerable to an *adaptive chosen ciphertext attack*, in which the adversary is assumed so powerful that he or she can obtain the decryptions of many ciphertexts of his or her own making, though *not* the target ciphertext. Recently, Cramer and Shoup [6] proposed a cryptosystem that is secure against such an attack and is believed to be practical as well.

9.2 Cryptosystems Based on Integer Factorization

Given two primes, say $p = 863$ and $q = 877$, it is an easy process to multiply them by hand to get the product $n = 756851$. However, it is not nearly so easy to determine by hand the factors p and q from only a knowledge of the product 756851. In a similar fashion, if p and q are large, say 1,000 digits each, then a computer can readily find the 2,000 digit product (since multiplying two k-digit numbers requires at most $O(k^2)$ operations), but even the fastest of today's computers cannot generally determine the factors from only the product. This leads us to consider two central problems in the history of mathematics, namely the problems of (a) determining whether a given integer is a prime, and (b) determining the factorization into primes of a given integer. These two problems have been attacked by some of the best mathematicians of all time, including the great C. F. Gauss (1777–1855) who wrote [9]:

> The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic. It has engaged the industry and wisdom of ancient and modern geometers to such an extent that it would be superfluous to discuss the problem at length ... Further, the dignity of the science itself seems to require solution of a problem so elegant and so celebrated.

Gauss wrote these words some 175 years before primality testing and the integer factorization problem were applied to modern day cryptography, so as important as they were in Gauss' day, they are even more important today.

It is clear from his words that Gauss realized that primality testing was a different problem from that of integer factorization, and since his time, significant progress has been made on both of these problems. For example, there is an efficient probabilistic algorithm called the Rabin-Miller test (see [3, 5]) which can recognize a composite number of say 1,000 digits without ever factoring that number, and the primality of a number can also be determined efficiently [1, 2]. There have also been developed over the years much improved factoring algorithms [13], but despite this progress, factoring a general composite number with as few as say 200 digits is still out of reach of the fastest computers using the best algorithms known today.

It is not our purpose here to delve into the theory of primality testing and integer factorization (for which we refer the reader to [13, 20] for recent developments). Instead, we simply wish to emphasize that it is easy to generate and multiply large prime numbers but it is not generally possible to factor the resulting answer in reasonable time; that is, integer multiplication appears to be a one-way function. This belief forms the basis for several public-key cryptosystems. We will discuss two of these after reviewing several ideas from modular arithmetic.

Let n be a positive integer and let $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ denote the ring of integers modulo n. It turns out that exponentiation in this ring is easy; i.e., for any $\alpha \in \mathbb{Z}_n$ and positive integer e, the computation of $\alpha^e \bmod n$ can be done efficiently. We demonstrate how this is done with an example.

Example 9.1

In order to compute α^{29}, first write 29 in binary form

$$29 = 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1.$$

It then follows that

$$\alpha^{29} = \alpha^{1 \cdot 2^4} \cdot \alpha^{1 \cdot 2^3} \cdot \alpha^{1 \cdot 2^2} \cdot \alpha^{0 \cdot 2^1} \cdot \alpha^1 = \left(\left(((\alpha^2 \cdot \alpha)^2 \cdot \alpha)^2\right)\alpha^0\right)^2 \cdot \alpha.$$

Thus, only four squarings and three multiplications are needed to compute $\alpha^{29} \bmod n$. (Here, it is important that the reductions modulo n be done at each squaring or multiplication to avoid large intermediate integers.)

The above idea can be generalized to show that α^e can be computed with $\log_2(e) - 1$ squarings and at most $\log_2(e) - 1$ multiplications, where $\log_2(e)$ is the length of e in its binary representation. So, for any $\alpha \in \mathbb{Z}_n$ and $e > 0$, $\alpha^e \bmod n$ can be computed efficiently.

Now consider the reverse of the operation of exponentiation modulo n. Assuming n is specified, two different problems arise: (a) given α and $y \in \mathbb{Z}_n$, find an integer x (if one exists) such that $\alpha^x \equiv y \bmod n$; and (b) given e and $y \in \mathbb{Z}_n$, find x (if one exists) such that $x^e \equiv y \bmod n$. These two problems are intrinsically different and each of them leads to a public-key cryptosystem.

Problem (a) is called *the discrete logarithm problem* modulo n and is believed hard for almost all n. (For certain values of n it is easy; e.g., when n has only small prime factors or when n is a prime but $n - 1$ has only small prime factors.) We will discuss this problem and its relation to the ElGamal system in the next section.

Problem (b) asks for the computation of an eth root of an integer y modulo n. This is easy when the complete factorization of n is known but believed hard otherwise. For cryptographic purposes, the most important case is when n is the product of two large (distinct) primes and this is the case we shall develop here. (Our discussion can be readily generalized to the situation where n is square-free.) The RSA system arises when we examine this situation with $\gcd(e, \phi(n)) = 1$ and the Rabin system comes from considering the case in which e divides $\phi(n)$. Here $\phi(n)$ is the familiar Euler ϕ-function which equals the number of integers in $\mathbb{Z}_n$ that are relatively prime to n. When $n = pq$ with p and q distinct primes, it is given by $\phi(n) = (p-1)(q-1)$.

Assume then that $n = pq$ and let e be an integer with $\gcd(e, \phi(n)) = 1$. Then there exists an integer d such that $ed \equiv 1 \bmod \phi(n)$. Using Fermat's little theorem, it is straightforward to show that

$$x^{ed} \equiv x \bmod n, \text{ for all } x \in \mathbb{Z}_n. \tag{9.1}$$

This means that for any $x \in Z_n$, $x^d \bmod n$ is an eth root of x modulo n; hence an eth root of x can be computed efficiently provided d is known. Thus an important question is whether d be computed efficiently.

The answer is yes if n can be factored but no otherwise. To explain this statement, first assume that the complete factorization of n into primes is known. Then $\phi(n)$ can be computed quickly. By applying the extended Euclidean algorithm [3] to e and $\phi(n)$, it is easy to find d such that $ed \equiv 1 \bmod \phi(n)$. Conversely, assume that a number d is known with $ed \equiv 1 \bmod \phi(n)$. Then $\Phi = ed - 1$ is a multiple of $\phi(n)$. By Exercise 9.3, n can be easily factored using Φ. Hence computing d from e and n is equivalent to factoring n. Therefore, the factors of n provide a trap-door for inverting the function $P_{e,n} : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ defined as

$$P_{e,n}(x) = x^e \bmod n,$$

where $x^e \bmod n$ denotes the smallest nonnegative integer congruent to x^e modulo n.

We can now describe the RSA cryptosystem, which bases its security on the belief that the class of functions $P_{e,n}$ are trap-door one-way functions.

DEFINITION 9.3 RSA Cryptosystem

- *The public-key space $\mathcal{K}$ is the set of integer pairs (e, n) where n is a product of two large distinct primes, $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.*

- *For each $k = (e, n) \in \mathcal{K}$, the plaintext and ciphertext spaces are $\mathcal{M}_k = \mathcal{C}_k = \mathbb{Z}_n$.*

- *For each $k = (e, n) \in \mathcal{K}$, the encryption function is $E_k = P_{e,n}$.*

- *For each $k = (e, n) \in \mathcal{K}$, the corresponding private key is (d, n) where $ed \equiv 1 \bmod \phi(n)$, and the decryption function is $D_k = P_{d,n}$.*

We also need a rule for generating a random pair of public and private keys (e, n) and (d, n), but this is not difficult. We have mentioned that primality testing can be done efficiently and further there are many primes with a given number say of t digits (by the prime number theorem). This means that one can generate a t-digit prime as follows. Choose a random number of t digits and test it for primality. If it is not prime, repeat the procedure until a prime is obtained. In this way one can get a pair of primes p and q of any desired size. Then defining $n = pq$ and $N = (p-1)(q-1)$, one may choose a random integer $e \in \mathbb{Z}_N$ such that $\gcd(e, N) = 1$. For this e, it is a simple matter to compute d with $ed \equiv 1 \bmod N$. Then (e, n) is a public key and (d, n) is the corresponding private key.

We should point out that, even though computing d from e and n is equivalent to factoring n, it has not been proven that *inverting* $P_{e,n}$ is equivalent to computing d or factoring n, as there may exist some other method to compute eth roots modulo n without factoring n or computing d. This raises the following research question.

Open Problem. *Given a composite integer n and a positive integer e with $\gcd(e, \phi(n)) = 1$, prove or disprove that computing eth roots modulo n is equivalent to factoring n.*

We next consider the case in which e divides $\phi(n)$, used in the Rabin system. Here, the function $P_{e,n}$ is no longer a permutation on $\mathbb{Z}_n$ since for a given e and y, there can be several x that map to the same y under $P_{e,n}$. However, if e is small relative to n, say $e \leq (\log n)^c$ for some constant c, then it can be proved that finding the inverse images of y under $P_{e,n}$ is equivalent to factoring n. Thus, if e is (say) one of $2, 3, 5, 6, 7$, then $P_{e,n}$ is a candidate trap-door one-way function where the factors of n again provide the trap-door for inverting $P_{e,n}$. Such trap-door one-way functions can be used to build up public-key cryptosystems; indeed, the Rabin system uses only the function $P_{2,n}$ and is described as follows.

DEFINITION 9.4 Rabin cryptosystem

- *The public-key space is $\mathcal{K} = \{n : n = pq,$ where p and q are large distinct primes$\}$.*
- For each $n \in \mathcal{K}$, the plaintext space is a subset $\mathcal{M}_n \subset \mathbb{Z}_n$ such that $x_1^2 \not\equiv x_2^2 \bmod n$ for all different $x_1, x_2 \in \mathcal{M}_n$, and the ciphertext space $\mathcal{C}_n = \{x^2 \bmod n : x \in \mathcal{M}_n\}$.
- For each $n \in \mathcal{K}$, the encryption function is $E_n = P_{2,n}$, the private-key is the pair (p, q) such that $n = pq$, and the decryption function is described below.

To decrypt a ciphertext under the Rabin system, we need to describe how to compute square roots modulo n given the factors of n. The idea is to compute square roots modulo each prime factor of n separately and then use the Chinese remainder theorem to combine them to get a square root modulo n. Suppose that $c \in \mathbb{Z}_n$ has a square root modulo n. Then c also has square roots modulo p and q. Moreover, if r and s are some square roots of c modulo p and q, respectively, and if a and b are integers such that $ap + bq = 1$, then it is easy to check that

$$\pm aps \pm bqr$$

is a square root of c modulo n for any choice of $+$ and $-$ signs. This gives four square roots of c. It can be shown that c has exactly four square roots if and only if $\gcd(c, n) = 1$ and c has at least one square root modulo n. To get the correct plaintext one just needs to check which of the roots is in $\mathcal{M}_k$.

It remains to show how to compute square roots modulo a prime p. But this is also easy, especially when $p \equiv 3 \bmod 4$, for then $r = c^{(p+1)/4} \bmod p$ is a square root of c whenever c is a *quadratic residue* in $\mathbb{Z}_p$: that is, $c \equiv x^2 \bmod p$ for some $x \in \mathbb{Z}_p$. If p is congruent to 1 modulo 4, square roots modulo p can also be computed efficiently but will not be described here (see [3, 5] for details). Hence computing square roots modulo $n = pq$ is easy when p and q are known.

We conclude this section by showing that if one can compute square roots modulo n then one can actually factor n; that is, we show that computing square roots modulo n is equivalent to factoring n. This also provides a nice example illustrating the power of randomness in computing. Suppose that there is an algorithm σ which, when presented a quadratic residue $x \in \mathbb{Z}_n$, outputs a square root of x in $\mathbb{Z}_n$, denoted by $\sigma(x)$. (One may think of σ as an algorithm, a black box, or an oracle.) Then n can be factored by the following simple algorithm. First, pick a nonzero element $a \in \mathbb{Z}_n$ uniform randomly and compute $x = a^2 \bmod n$. Next, input x to σ and get $b = \sigma(x)$. Then compute $h = \gcd(a-b, n)$. It can be shown that for each run of the algorithm, the computed number h is a proper factor of n with probability at least $1/2$. If the algorithm is run t times, then the probability of getting a factor of n is at least $1 - (1/2)^t$. Thus for $t = 10$, the chance of finding a factor of n is over 99.9%! Therefore n can be factored quickly.

As we have indicated, breaking the Rabin system is equivalent to factoring integers. This is the first example of a public-key cryptosystem with provable security against a passive adversary who can only eavesdrop. Compared to the RSA system, the encryption in Rabin system is more efficient (only one square), and the decryption costs approximately the same as in RSA. However, the same proof above shows that the Rabin system is totally insecure against an active adversary who can mount a chosen ciphertext attack. Under this attack, an adversary chooses some (possibly many) ciphertexts and asks for the corresponding plaintexts, then deduces from them the secret key. The reader should be able to see why this attack works against the Rabin system. Because of this, the Rabin system is not used in practice.

9.3 Cryptosystems Based on Discrete Logarithms

Let $\mathbf{F}_q$ be a finite field of q elements so that $q = p^n$ for some prime p and integer n. It is well known that the multiplicative group of nonzero elements of $\mathbf{F}_q$, denoted by $\mathbf{F}_q^*$, is a cyclic group of order $q-1$. Thus if α is a generator of this multiplicative group, then every nonzero element β in $\mathbf{F}_q$ is given by $\beta = \alpha^x$ for some integer x; in fact for each β there is a unique integer in the range $0 \leq x < q-1$ with this property. For a given x and α, the power α^x can be quickly computed by the square-and-multiply method as demonstrated in Example 9.1. The inverse problem, i.e., the problem of finding, for a given α and β, the x in the range $0 < x < q-1$ satisfying $\beta = \alpha^x$, is the discrete logarithm problem; it is believed to be hard for many fields. Thus, exponentiation in finite fields is a candidate for a one-way function.

Example 9.2

For the prime $p = 1999$, the ring $\mathbb{Z}_p$ is a finite field and the nonzero elements $\mathbb{Z}_p^*$ of $\mathbb{Z}_p$ form a group G under multiplication modulo p:

$$G = \mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}.$$

Furthermore, the element $\alpha = 3$ is a generator of G, and is also known as a *primitive element* modulo p:

$$G = \{1, \alpha, \alpha^2, \ldots, \alpha^{p-2}\} \bmod p.$$

It is easy to compute that

$$3^{789} \equiv 1452 \bmod p.$$

However, it is not nearly so easy to determine that $x = 789$, given only that x is in the range from 0 to 1997 and satisfies the equation

$$3^x \equiv 1452 \bmod 1999.$$

A more realistic challenge is to find an integer x such that

$$3^x \equiv 2 \bmod p, \text{ where } p = 142 \cdot (10^{301} + 531) + 1.$$

We know a solution exists but we don't know its value.

The above discussion can be generalized to any group G (whose operation is written multiplicatively). The *discrete logarithm problem* for G is to find, for given $\alpha, \beta \in G$, a nonnegative integer x (if it exists) such that $\beta = \alpha^x$. The smallest such integer x is called the discrete logarithm of β to the base α, and is written $x = \log_\alpha \beta$. In Example 9.2, $\log_3 1452 = 789$. Clearly, the discrete logarithm problem for a general group G is exactly the problem of inverting the exponentiation function $\exp : \mathbb{Z}_N \longrightarrow G$ defined by $\exp(x) = \alpha^x$ where N is the order of α.

The difficulty of this general discrete logarithm problem depends on the representation of the group. For example, consider G to be the cyclic group of order N. If G is represented as the additive group of $\mathbb{Z}_N$, then computing discrete logarithms in G is equivalent to solving the linear equation $ax \equiv b \bmod N$, where a, b are given integers; this can be easily done by using the extended Euclidean algorithm. If G is represented as a subgroup of the multiplicative group of a finite field as above or as a multiplicative group of elements from $\mathbb{Z}_m$ (where m may be composite or prime), then the problem can be "hard." For an elliptic curve group [14], the discrete logarithm problem seems to be harder. (As we have indicated, no one has been able to prove that these discrete logarithm problems are really hard, but they have been studied by number theorists for considerable time with only limited success.) For recent surveys and a more detailed study of the discrete logarithm problem, we refer the reader to [15, 18, 19]. We now describe two cryptosystems whose security is based on the assumption that the discrete logarithm problem is hard.

The Diffie-Hellman key exchange scheme is a protocol for establishing a common key between two users of a classical cryptosystem. As we mentioned earlier, for a large network of users of a conventional cryptosystem, the secure distribution of keys can be complicated and logistic. In 1976, Diffie and Hellman [7] gave the following simple and elegant solution for this problem.

DEFINITION 9.5 **Diffie-Hellman key exchange scheme.** *Given the public group G and an element $\alpha \in G$ of order N, two parties, say Bob and Alice, establish a common key using the following steps:*

- *Alice picks a random integer $a \in \mathbb{Z}_N$, computes $A = \alpha^a$, and sends it to Bob.*
- *Bob picks a random integer $b \in \mathbb{Z}_N$, computes $B = \alpha^b$, and sends it to Alice.*

- *Alice computes $B^a = \alpha^{ba}$, and Bob computes $A^b = \alpha^{ab}$. Their common key is $k = \alpha^{ab} = \alpha^{ba}$.*

An eavesdropper, who knows G and α from the public directory, after intercepting A and B, is then faced with the following problem.

DEFINITION 9.6 **Diffie-Hellman problem.** *Let G be a group and let $\alpha \in G$. Given $A = \alpha^a$ and $B = \alpha^b$, compute $k = \alpha^{ab}$.*

If one can solve the discrete logarithm problem, then it is clear that one can solve the Diffie-Hellman problem; hence the latter problem is no harder than the former. It is believed that the two problems are equivalent, and in fact this equivalence has been established for some special cases. In any event, the Diffie-Hellman key exchange scheme is secure provided the Diffie-Hellman problem is hard.

The Diffie-Hellman key exchange scheme is widely used (with some variants) in practice to generate "session keys," for example in secure internet transactions. This scheme itself is not a public-key cryptosystem; however, ElGamal [8] showed that it could easily be converted into one. Note that if Alice publishes $k_A = \alpha^a$ but keeps a secret, then anyone, say Charlie, can share a common key with Alice in the same way that Bob did; i.e., Charlie can pick a random integer c, and compute $r = \alpha^c$ and $k_A^c = \alpha^{ac}$. Before sending r to Alice, Charlie can encrypt any message m he wishes by simply computing the product $r_1 = m \cdot k_A^c$. Then he sends the pair (r, r_1) to Alice. Alice can compute $k_A^c = r^a$ from r and her private key a, and so decrypt m.

DEFINITION 9.7 **ElGamal cryptosystem.** *Given the public group G (written multiplicatively) and an element $\alpha \in G$ of order N, let $G_1 =<\alpha>$, the subgroup of G generated by α.*

- *The key space is $\mathcal{K} = G_1$.*
- *For each $k \in \mathcal{K}$, the plaintext and ciphertext spaces are*

$$\mathcal{M}_k = G, \quad \mathcal{C}_k = G_1 \times G = \{(\beta_1, \beta_2) : \beta_1 \in G_1,\, \beta_2 \in G\}.$$

The randomization set is $\mathcal{R}_k = \mathbb{Z}_N$.

- *For each $k \in \mathcal{K}$, the encryption function $E_k : \mathcal{M}_k \times \mathcal{R}_k \longrightarrow \mathcal{C}_k$ is given by $E_k(m, r) = (\alpha^r, k^r \cdot m)$.*

- *For each $k \in \mathcal{K}$, the corresponding private key is the integer $d \in Z_N$ such that $k = \alpha^d$, and the decryption function $D_k : \mathcal{C}_k \longrightarrow \mathcal{M}_k$ is given by $D_k(c_1, c_2) = c_2 \cdot (c_1^d)^{-1}$.*

It is easy to check that if $k = \alpha^d$, then $D_k(E_k(m, r)) = m$ for all $m \in M_k$ and $r \in \mathcal{R}_k$. To obtain a random key, one just chooses $d \in \mathbb{Z}_N$ at random and computes $k = \alpha^d$. In practice, one has to be extremely careful in choosing the group G so that the discrete logarithm problem is hard. Originally, Diffie and Hellman (1976) and ElGamal (1985) used the multiplicative group of $\mathbb{Z}_p$ for a large prime p. The above description of their systems is actually a natural generalization to an arbitrary group. The most studied groups for cryptographic purposes are multiplicative subgroups of $\mathbb{Z}_p$, $\mathbb{Z}_m$ (where m is a product of two large primes), $\mathbf{F}_{2^n}$, and elliptic curve groups over finite fields. While $\mathbb{Z}_p$ is currently the most popular choice, there is increasing interest in using elliptic curves over finite fields, particularly the fields $\mathbf{F}_{2^n}$ [14].

Note that breaking the ElGamal cryptosystem by a ciphertext-only attack is equivalent to solving the Diffie-Hellman problem. Thus the ElGamal cryptosystem is another example with provable security if the Diffie-Hellman problem is indeed hard. The major disadvantage of the system is the message expansion by a factor of two, but there are ways to improve it in both efficiency and security as we discuss next.

Observe that the multiplicative operation $k^r \cdot m$ in the encryption function E_k could be replaced by other operations. For example, if the elements in G are represented as binary strings of 0's and 1's, then we can let

$$E_k(m, r) = (\alpha^r, k^r \oplus m)$$

where m is any binary string and $\oplus$ is the bitwise "exclusive or" operation (XOR); e.g., $(1100) \oplus (0101) = 1001$. In this case, m does not have to be in G. If the elements in G are represented as binary strings of length ℓ, then the plaintext space $\mathcal{M}_k$ can be any subset of binary strings of length ℓ. Note that in the ElGamal cryptosystem, it is required that a plaintext be in the group. This is not trivial to achieve for some groups (e.g., elliptic curves), but the above approach solves this problem. Also, $\oplus$ is computationally cheaper than multiplication in a group.

However, the above alternative does not solve the problem of message expansion. One way around this is to use k^r as a key in a conventional cryptosystem, say DES or IDEA [12]. That is, define

$$E_k(m, r) = (\alpha^r, \widetilde{E}_{k^r}(m)),$$

where $\widetilde{E}$ is any encryption function in a conventional cryptosystem. Here m can be a message of arbitrary length and $\widetilde{E}$ operates on m, say, with cipher block chaining (CBC) mode. With this modification, the cryptosystem is sometimes called a *hybrid cryptosystem*, which combines the advantages of both public-key and conventional cryptosystems. Such cryptosystems are practical but do not have provable security (a typical phenomenon for conventional cryptosystems). To improve security, one can use a cryptographically strong pseudo-random bit generator to expand k^r to a much longer string and then XOR it with m.

It should be noted that the ElGamal cryptosystem is completely insecure against an adaptive chosen ciphertext attack, mentioned earlier. Indeed, given an encryption (c_1, c_2) of a message m, one can ask for the decryption of $(c_1, c_2 \cdot \alpha)$, which is $\alpha \cdot m$, so m can be deduced immediately.

9.4 Digital Signatures

Suppose that you wish to transmit an electronic file. A natural question is how one can put a piece of information at the end of the file that serves the same role as a handwritten signature on a document. It turns out that the digital signature is one of the main applications of public-key cryptography.

Handwritten signatures have the following main features:

- The signature is unique and unforgeable. It is proof that the signer deliberately signed the document. It convinces the recipient of the document and any third party, say a judge, that it has been signed by the claimed signer.

- The signature is not reusable. It is part of the document and can not be moved to another document. If the document is altered or the signature is moved to another document then the signature is no longer valid.

How do we realize a signature digitally? Since it is easy to copy, alter, or move a file on computers without leaving any trail, one needs to be very careful in designing a signature scheme. In keeping with the above properties of a handwritten signature, a digital signature should be a number that depends on some secret known only to the signer and on the content of the message being signed. It must also be verifiable: i.e., the recipient of the message (or any unbiased third party) should be able to distinguish between a forgery and a valid signature without requiring the signer to reveal any secret information (private key). Thus in a signature scheme, we need two algorithms: one used by the person signing the message and the other used by the recipient verifying the signature. In the following, we describe two methods based on the RSA and the ElGamal cryptosystems [8, 21]. Incidentally, the recently adopted Digital Signature Algorithm (DSA) in the US Digital Signature Standard (DSS) [17] is a variation of ElGamal signature scheme, and we will describe it as well.

To describe the RSA digital signature scheme, note that the encryption function $E_k = P_{e,n}$ and the decryption function $D_k = P_{d,n}$ in the RSA system are commutative: that is,

$$D_k(E_k(x)) = E_k(D_k(x)) \equiv x^{ed} \equiv x \bmod n, \text{ for all } x \in \mathbb{Z}_n.$$

Suppose that a user Alice has public key $k = (e, n)$ and private key (d, n) for the RSA cryptosystem. Then Alice can use her private key to encrypt a message (or a file) $m \in \mathbb{Z}_n$ and use the ciphertext $s = D_k(m) = m^d \bmod n$ as her signature for the message m. Anyone, seeing the message m and the signature s, can compute $m_1 = E_k(s)$ and accept the signature if and only if $m_1 = m$. This proves that Alice indeed signed the message m, since an adversary trying to forge a signature for Alice on a message m would have to solve the equation $s^e \equiv m \bmod n$ for $s \in \mathbb{Z}_n$ (which is presumably hard). So if Bob shows m and s to a judge and if $E_k(s) = m$, the judge should be convinced that no one but Alice could have signed the statement.

There is one catch though — and this occurs when all or a significant fraction of the elements in Z_n represent valid messages. In this case, one could easily forge a signature as follows. Pick $s \in \mathbb{Z}_n$ at random and compute $m = E_k(s)$ where k is Alice's public key. Then with high probability, m is a valid message and in this case, since $E_k(s) = m$ holds, s is a valid signature of Alice for m. To avoid this possibility in practice, one adds some redundant information to the message. Namely, we require the message to have some additional structure (e.g., it should be in some standard format). Thus, a random element in $\mathbb{Z}_n$ will be a valid message with only vanishing probability. This comment also applies to the DSA and the ElGamal signature schemes discussed below.

The ElGamal cryptosystem cannot, as it stands, be used to generate signatures, but it can be modified to suit signature purposes. In this case, the signature scheme is probabilistic in that there are many possible valid signatures for every message and the verification algorithm accepts any of the valid signatures as authentic. Suppose that p is a large prime for which computing discrete logarithms in $\mathbb{Z}_p$ is infeasible, and $\alpha \in \mathbb{Z}_p$ is a primitive element. Also suppose that Alice chooses a random integer $a \in \mathbb{Z}_{p-1}$ as her private key and $\beta = \alpha^a \bmod p$ as her public key. To sign a message $m \in \mathbb{Z}_{p-1}$, Alice can

- Pick a random $k \in \mathbb{Z}_{p-1}$, with $\gcd(k, p-1) = 1$.
- Compute γ and δ where

$$\gamma = \alpha^k \bmod p, \quad \delta = (m - a\gamma)k^{-1} \bmod (p-1).$$

- Sign the message m with (γ, δ).

Since $\beta^\gamma \gamma^\delta \equiv \alpha^{a\gamma + k\delta} \equiv \alpha^m \bmod p$, anyone can verify Alice's signature:

- Get Alice's public key β (α and p are public).
- Compute $e_1 = \beta^\gamma \gamma^\delta \bmod p$ and $e_2 = \alpha^m \bmod p$.
- Accept the signature as valid only if $e_1 = e_2$.

The US Digital Signature Standard (DSS) was adopted on December 1, 1994. In DSS, a digital signature algorithm (DSA) is proposed and it is a variation of the ElGamal signature scheme. We describe DSA briefly as follows. Choose primes p and q with $q|(p-1)$ and

$$2^{159} < q < 2^{160}, \;\; 2^{L-1} < p < 2^L.$$

That is, q has 160 bits and p has L bits where $512 \le L \le 1024$ and L is a multiple of 64. Suppose $\alpha \in \mathbb{Z}_p$ has order q: i.e., $\alpha \not\equiv 1 \bmod p$ but $\alpha^q \equiv 1 \bmod p$. A user, say Alice, has a random nonzero integer $a \in \mathbb{Z}_q$ as her private key and $\beta = \alpha^a \bmod p$ as her public key. To sign a message $m \in \mathbb{Z}_q$, Alice can

- Pick a random nonzero $k \in \mathbb{Z}_q$.
- Compute $\gamma = (\alpha^k \bmod p) \bmod q, \;\; \delta = (m + a\gamma)k^{-1} \bmod q$.
- Sign the message m with (γ, δ).

To verify Alice's signature (γ, δ) for the message m, the receiver can

- Get Alice's public key β.
- Compute

$$e_1 = m\delta^{-1} \bmod q, \; e_2 = \gamma\delta^{-1} \bmod q, \; \gamma_1 = (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q.$$

- Accept the signature as valid only if $\gamma_1 = \gamma$.

To see why this works, note that

$$k\delta \equiv m + a\gamma \bmod q.$$

Thus

$$\alpha^{k\delta} \equiv \alpha^{m+a\gamma} \bmod p \equiv \alpha^m \beta^\gamma \bmod p.$$

Since q is prime and $\delta \not\equiv 0 \bmod q$, we see that the map $x \mapsto x^\delta$ in the multiplicative group generated by α is a permutation. Thus

$$\alpha^k \equiv \alpha^{m\delta^{-1}} \beta^{\gamma\delta^{-1}} \bmod p.$$

Reducing both sides modulo q (after they are reduced modulo p), we have $\gamma = \gamma_1$.

Note that in the above signature scheme, the size of a signature equals (in RSA) or doubles (in both ElGamal and DSA) the size of the message being signed. This can be awkward in practice, especially if the message being signed is long. One way to overcome this problem is to first hash the message to a string of fixed size and then sign the hashed value of the message. Together with DSS, the National Institute of Standards and Technology (NIST) has also published a Secure Hash Standard (SHS) [17]. In SHS, a Secure Hash Algorithm (SHA) is proposed, which maps a message of arbitrary length to a binary string of length 160. We will not describe the details here, and the interested reader is referred to the references.

Public-key cryptography has many other applications, including identification, authentication, authorization, data integrity, and smart cards. In fact, NIST proposed in February 1997 a standard for entity authentication using public-key cryptography; the reader can consult the website `<http://csrc.ncsl.nist.gov/fips/>`. For further study we recommend the books [16, 22, 23, 24], and for the early history of cryptography, we suggest [11]. Computational number theory is nicely covered in [3, 5, 20]. Additional information on cryptographic methods, algorithms, and protocols can be found at `<http://www.ssh.fi/tech/crypto/>`.

9.5 Exercises and Projects

1. Develop an algorithm for computing $\alpha^e \bmod n$ using the square-and-multiply method indicated in Example 9.1. Implement your algorithm and test it for $n = 12345$; $\alpha = 123$; and $e = 0, 111, 12344, 54321$.

2. Let $n = 863 \cdot 877 = 756851$ and let $e = 5$. Given that 863 and 877 are primes, find $\phi(n)$ and compute d such that $ed \equiv 1 \bmod \phi(n)$.

3. The following problems relate to the discussion of the RSA system.

a. Given $n = pq = 591037$ and $\phi(n) = 589500$, determine p and q by first determining a quadratic equation that p satisfies and then solving it using the quadratic formula.

b. Let M be a given multiple of $\phi(n)$. Write $M = 2^e m$ where m is odd. Prove that for random $a, b \in \mathbb{Z}_n$, the probability that $\gcd(n, a^{m2^i} - b^{m2^i})$ is a proper factor of n for some $0 \leq i \leq e$ is at least $1/2$. As a consequence, show that n can be factored efficiently when a multiple of $\phi(n)$ is given.

4. You are an RSA cryptosystem user with public key $n = 756851$ and $e = 5$. Suppose that a number in $\mathbb{Z}_n$ is always written as a 6 digit number (padding zeros in front if necessary). Then the numbers in $\mathbb{Z}_n$ represent a triple of letters under the correspondence $00 \leftrightarrow A$, $01 \leftrightarrow B$, …, $25 \leftrightarrow Z$; e.g., $1719 = 001719 \leftrightarrow ART$. Your private key is the number d computed in Exercise 9.2. Decrypt the following ciphertext:

375365	752560	389138	193982	283519	350016	92892	86995
604644	125895	706746	323635	574615	226430	533566	419464

5. Suppose that p is a prime congruent to 3 modulo 4 and c is a quadratic residue modulo p. Prove that $x = c^{(p+1)/4} \bmod p$ is a square root of c modulo p.

6. Suppose Bob is a Rabin cryptosystem user with public key $n = 5609$ and private key $(p, q) = (71, 79)$. With each number in $\mathbb{Z}_n$ representing two letters as in Exercise 9.4, decrypt the following ciphertext:

924	642	2299	223	5374	121	2217	4474	719	839	5060
1474	3607	3763	2015	3586	3204	5060	10	2017	169	5101
446	4837	288	2217	4474	719	839	5060	1474	3988	

7. Investigate how to choose a convenient plaintext space for Rabin's cryptosystem. That is, for $n = pq$, find a subset M_n of $\mathbb{Z}_n$ such that (a) $x_1^2 \not\equiv x_2^2 \bmod n$ for all different $x_1, x_2 \in M_n$; (b) for any $x \in \mathbb{Z}_n$, it is easy to decide whether $x \in M_n$; and (c) M_n should be large, say of size $O(n)$.

8. Decrypt the following ElGamal ciphertexts. The parameters are $p = 3119$, $\alpha = 7$, $\beta = 1492$, and $d = 799$. Each number in $\mathbb{Z}_p$ represents two letters as in Exercise 9.4.

(1139, 1035)	(79, 1438)	(1489, 2725)	(2928, 87)	(691, 888)
(3010, 1012)	(1316, 1734)	(1790, 1385)	(2775, 1267)	(1807, 2319)
(2910, 2668)	(142, 238)	(123, 1994)	(916, 2055)	(3053, 2491)
(810, 247)	(1674, 2521)	(617, 1798)	(2705, 144)	(776, 650)
(1440, 311)	(1620, 713)	(938, 572)	(2209, 968)	(1037, 45)

9. Let $p = 877$ and $\alpha = 2$. Alice uses the ElGamal signature scheme, and her public key is $\beta = 253$.

a. Verify that $(137, 217)$ is a valid signature of Alice for the message $m = 710$.

b. Suppose your secret key is $a = 133$. Sign the message $m = 606$.

10. Suppose that Bob uses the DSA with $q = 103$, $p = 10 \cdot q + 1 = 1031$, $\alpha = 14$, $a = 75$, and $\beta = 742$. Determine Bob's signature on the message $x = 1001$ using the random value $k = 49$, and verify the resulting signature.

References

[1] L. M. Adleman and M.-D. A. Huang, *Primality Testing and Abelian Varieties over Finite Fields*, Lecture Notes in Mathematics, Volume 1512, Springer-Verlag, Berlin, 1992.

[2] A. O. L. Atkin and F. Morain, "Elliptic curves and primality proving", *Math. Comp.* **61**, 29–68 (1993).

[3] E. Bach and J. Shallit, *Algorithmic Number Theory, Volume I: Efficient Algorithms*, MIT Press, Cambridge, MA, 1996.

[4] M. Blum and S. Goldwasser, "An efficient probabilistic public-key cryptosystem which hides all partial information", *Advances in Cryptology — CRYPTO 84*, Lecture Notes in Computer Science, Volume 196, Springer-Verlag, Berlin, 1985, 289–299.

[5] H. Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics 138, Springer-Verlag, Berlin, 1993.

[6] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack", *Advances in Cryptology — CRYPTO 98*, Lecture Notes in Computer Science, Volume 1462, Springer-Verlag, Berlin, 1998, 13–25.

[7] W. Diffie and M. E. Hellman, "New directions in cryptography", *IEEE Trans. Info. Theory* **22**, 644–654 (1976).

[8] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Trans. Info. Theory* **31**, 469–472 (1985).

[9] C. F. Gauss, *Disquisitiones Arithmeticae*, Braunschweig, 1801. English Edition, Springer-Verlag, New York, 1986.

[10] S. Goldwasser and S. Micali, "Probabilistic encryption", *Journal of Computer and System Science* **28**, 270–299 (1984).

[11] D. Kahn, *The Codebreakers: The Story of Secret Writing*, Macmillan, New York, 1968.

[12] X. Lai, "On the design and security of block ciphers", *ETH Series in Information Processing*, Volume 1, Hartung-Gorre Verlag, Konstanz, Switzerland, 1992.

[13] A. K. Lenstra and H. W. Lenstra, Jr., *The Development of the Number Field Sieve*, Lecture Notes in Mathematics, Volume 1554, Springer-Verlag, Berlin, 1993.

[14] A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer, Boston, 1993.

[15] A. J. Menezes, I. F. Blake, X. Gao, R. C. Mullin, S. A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*, Kluwer, Boston, 1993.

[16] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1996.

[17] National Institute for Standards and Technology, *Secure Hash Standard* (SHS), FIPS 180-1 (1995); *Digital Signature Standard* (DSS), FIPS 186-1 (1998).

[18] A. M. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance", *Advances in Cryptology — EUROCRYPT 84*, Lecture Notes in Computer Science, Volume 209, Springer-Verlag, Berlin, 1985, 224–314.

[19] A. M. Odlyzko, "Discrete logarithms and smooth polynomials", in *Finite Fields: Theory, Applications, and Algorithms*, G. L. Mullen and P. J.-S. Shiue (eds.), Contemporary Mathematics, Volume 168, American Mathematical Society, Providence, RI, 1994, 269–278.

[20] C. Pomerance, ed., *Cryptography and Computational Number Theory*, *Proc. Symp. Appl. Math.*, Volume 42, American Mathematical Society, Providence, RI, 1990.

[21] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM* **21**, 120–126 (1978).

[22] B. Schneier, *Applied Cryptography*, 2nd ed., Wiley, New York, 1995.

[23] G. J. Simmons, ed., *Contemporary Cryptography: The Science of Information Integrity*, IEEE Press, New York, 1992.

[24] D. R. Stinson, *Cryptography: Theory and Practice*, CRC Press, Boca Raton, FL, 1995.