Chapter 1

Overview of Cryptography

Contents in Brief

1.1 Introduction

Cryptography has a long and fascinating history. The most complete non-technical account of the subject is Kahn's *The Codebreakers*. This book traces cryptography from its initial and limited use by the Egyptians some 4000 years ago, to the twentieth century where it played a crucial role in the outcome of both world wars. Completed in 1963, Kahn's book covers those aspects of the history which were most significant (up to that time) to the development of the subject. The predominant practitioners of the art were those associated with the military, the diplomatic service and government in general. Cryptography was used as a tool to protect national secrets and strategies.

The proliferation of computers and communications systems in the 1960s brought with it a demand from the private sector for means to protect information in digital form and to provide security services. Beginning with the work of Feistel at IBM in the early 1970s and culminating in 1977 with the adoption as a U.S. Federal Information Processing Standard for encrypting unclassified information, DES, the Data Encryption Standard, is the most well-known cryptographic mechanism in history. It remains the standard means for securing electronic commerce for many financial institutions around the world.

The most striking development in the history of cryptography came in 1976 when Diffie and Hellman published *New Directions in Cryptography*. This paper introduced the revolutionary concept of public-key cryptography and also provided a new and ingenious method for key exchange, the security of which is based on the intractability of the discrete logarithm problem. Although the authors had no practical realization of a public-key encryption scheme at the time, the idea was clear and it generated extensive interest and activity in the cryptographic community. In 1978 Rivest, Shamir, and Adleman discovered the first practical public-key encryption and signature scheme, now referred to as RSA. The RSA scheme is based on another hard mathematical problem, the intractability of factoring large integers. This application of a hard mathematical problem to cryptography revitalized efforts to find more efficient methods to factor. The 1980s saw major advances in this area but none which rendered the RSA system insecure. Another class of powerful and practical public-key schemes was found by ElGamal in 1985. These are also based on the discrete logarithm problem.

One of the most significant contributions provided by public-key cryptography is the digital signature. In 1991 the first international standard for digital signatures (ISO/IEC 9796) was adopted. It is based on the RSA public-key scheme. In 1994 the U.S. Government adopted the Digital Signature Standard, a mechanism based on the ElGamal public-key scheme.

The search for new public-key schemes, improvements to existing cryptographic mechanisms, and proofs of security continues at a rapid pace. Various standards and infrastructures involving cryptography are being put in place. Security products are being developed to address the security needs of an information intensive society.

The purpose of this book is to give an up-to-date treatise of the principles, techniques, and algorithms of interest in cryptographic practice. Emphasis has been placed on those aspects which are most practical and applied. The reader will be made aware of the basic issues and pointed to specific related research in the literature where more indepth discussions can be found. Due to the volume of material which is covered, most results will be stated without proofs. This also serves the purpose of not obscuring the very applied nature of the subject. This book is intended for both implementers and researchers. It describes algorithms, systems, and their interactions.

Chapter 1 is a tutorial on the many and various aspects of cryptography. It does not attempt to convey all of the details and subtleties inherent to the subject. Its purpose is to introduce the basic issues and principles and to point the reader to appropriate chapters in the book for more comprehensive treatments. Specific techniques are avoided in this chapter.

1.2 Information security and cryptography

The concept of *information* will be taken to be an understood quantity. To introduce cryptography, an understanding of issues related to information security in general is necessary. Information security manifests itself in many ways according to the situation and requirement. Regardless of who is involved, to one degree or another, all parties to a transaction must have confidence that certain objectives associated with information security have been met. Some of these objectives are listed in Table 1.1.

Over the centuries, an elaborate set of protocols and mechanisms has been created to deal with information security issues when the information is conveyed by physical documents. Often the objectives of information security cannot solely be achieved through mathematical algorithms and protocols alone, but require procedural techniques and abidance of laws to achieve the desired result. For example, privacy of letters is provided by sealed envelopes delivered by an accepted mail service. The physical security of the envelope is, for practical necessity, limited and so laws are enacted which make it a criminal

privacy or confidentiality	keeping information secret from all but those who are authorized to see it.
data integrity	ensuring information has not been altered by unauthorized or unknown means.
entity authentication or identification	corroboration of the identity of an entity (e.g., a person, a computer terminal, a credit card, etc.).
message authentication	corroborating the source of information; also known as data origin authentication.
signature	a means to bind information to an entity.
authorization	conveyance, to another entity, of official sanction to do or be something.
validation	a means to provide timeliness of authorization to use or manipulate information or resources.
access control	restricting access to resources to privileged entities.
certification	endorsement of information by a trusted entity.
timestamping	recording the time of creation or existence of information.
witnessing	verifying the creation or existence of information by an entity other than the creator.
receipt	acknowledgement that information has been received.
confirmation	acknowledgement that services have been provided.
ownership	a means to provide an entity with the legal right to use or transfer a resource to others.
anonymity	concealing the identity of an entity involved in some process.
non-repudiation	preventing the denial of previous commitments or actions.
revocation	retraction of certification or authorization.

Table 1.1: *Some information security objectives.*

offense to open mail for which one is not authorized. It is sometimes the case that security is achieved not through the information itself but through the physical document recording it. For example, paper currency requires special inks and material to prevent counterfeiting.

Conceptually, the way information is recorded has not changed dramatically over time. Whereas information was typically stored and transmitted on paper, much of it now resides on magnetic media and is transmitted via telecommunications systems, some wireless. What has changed dramatically is the ability to copy and alter information. One can make thousands of identical copies of a piece of information stored electronically and each is indistinguishable from the original. With information on paper, this is much more difficult. What is needed then for a society where information is mostly stored and transmitted in electronic form is a means to ensure information security which is independent of the physical medium recording or conveying it and such that the objectives of information security rely solely on digital information itself.

One of the fundamental tools used in information security is the signature. It is a building block for many other services such as non-repudiation, data origin authentication, identification, and witnessing, to mention a few. Having learned the basics in writing, an individual is taught how to produce a handwritten signature for the purpose of identification. At contract age the signature evolves to take on a very integral part of the person's identity. This signature is intended to be unique to the individual and serve as a means to identify, authorize, and validate. With electronic information the concept of a signature needs to be redressed; it cannot simply be something unique to the signer and independent of the information signed. Electronic replication of it is so simple that appending a signature to a document not signed by the originator of the signature is almost a triviality.

Analogues of the "paper protocols" currently in use are required. Hopefully these new electronic based protocols are at least as good as those they replace. There is a unique opportunity for society to introduce new and more efficient ways of ensuring information security. Much can be learned from the evolution of the paper based system, mimicking those aspects which have served us well and removing the inefficiencies.

Achieving information security in an electronic society requires a vast array of technical and legal skills. There is, however, no guarantee that all of the information security objectives deemed necessary can be adequately met. The technical means is provided through cryptography.

1.1 Definition *Cryptography* is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication.

Cryptography is not the only means of providing information security, but rather one set of techniques.

Cryptographic goals

Of all the information security objectives listed in Table 1.1, the following four form a framework upon which the others will be derived: (1) privacy or confidentiality (§1.5, §1.8); (2) data integrity (§1.9); (3) authentication (§1.7); and (4) non-repudiation (§1.6).

1. *Confidentiality* is a service used to keep the content of information from all but those authorized to have it. *Secrecy* is a term synonymous with confidentiality and privacy. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible.
2. *Data integrity* is a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes such things as insertion, deletion, and substitution.
3. *Authentication* is a service related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. For these reasons this aspect of cryptography is usually subdivided into two major classes: *entity authentication* and *data origin authentication*. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed).
4. *Non-repudiation* is a service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary. For example, one entity may authorize the purchase of property by another entity and later deny such authorization was granted. A procedure involving a trusted third party is needed to resolve the dispute.

A fundamental goal of cryptography is to adequately address these four areas in both theory and practice. Cryptography is about the prevention and detection of cheating and other malicious activities.

This book describes a number of basic *cryptographic tools* (*primitives*) used to provide information security. Examples of primitives include encryption schemes (§1.5 and §1.8), hash functions (§1.9), and digital signature schemes (§1.6). Figure 1.1 provides a schematic listing of the primitives considered and how they relate. Many of these will be briefly introduced in this chapter, with detailed discussion left to later chapters. These primitives should

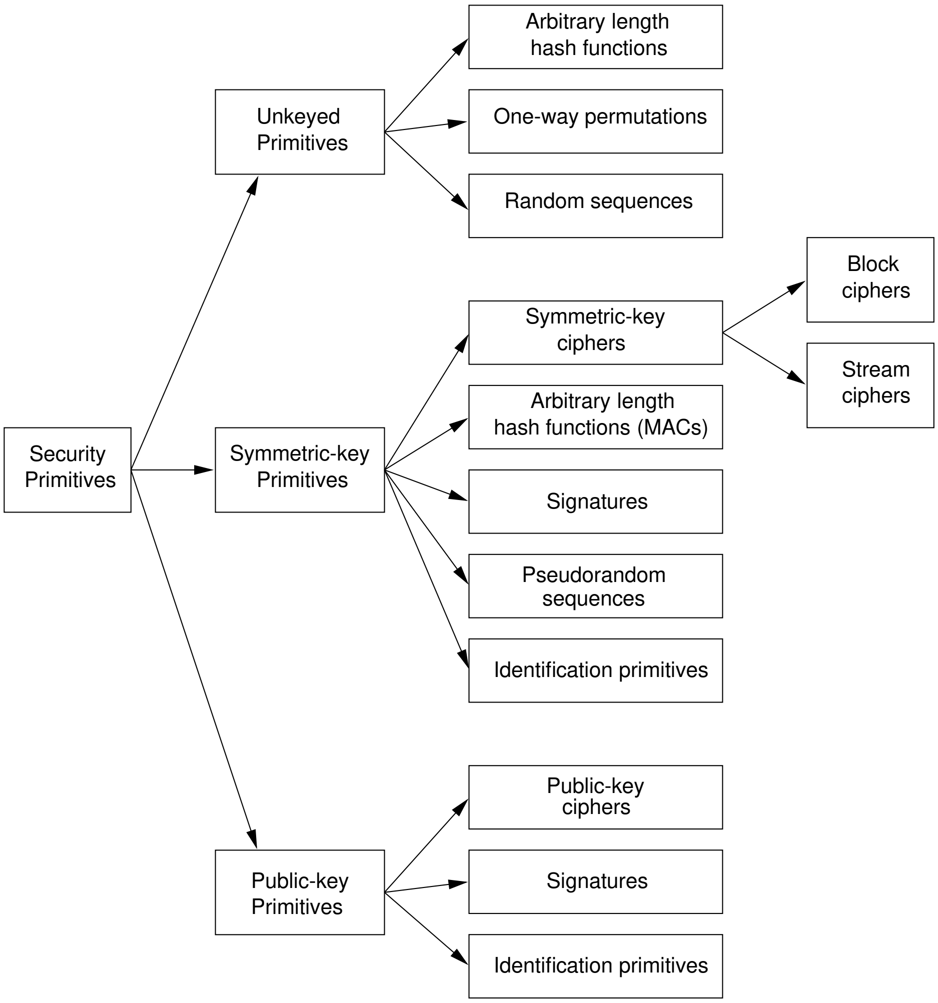

Figure 1.1: *A taxonomy of cryptographic primitives.*

be evaluated with respect to various criteria such as:

1. *level of security.* This is usually difficult to quantify. Often it is given in terms of the number of operations required (using the best methods currently known) to defeat the intended objective. Typically the level of security is defined by an upper bound on the amount of work necessary to defeat the objective. This is sometimes called the work factor (see §1.13.4).
2. *functionality.* Primitives will need to be combined to meet various information security objectives. Which primitives are most effective for a given objective will be determined by the basic properties of the primitives.
3. *methods of operation.* Primitives, when applied in various ways and with various inputs, will typically exhibit different characteristics; thus, one primitive could provide

very different functionality depending on its mode of operation or usage.

4. *performance.* This refers to the efficiency of a primitive in a particular mode of operation. (For example, an encryption algorithm may be rated by the number of bits per second which it can encrypt.)
5. *ease of implementation.* This refers to the difficulty of realizing the primitive in a practical instantiation. This might include the complexity of implementing the primitive in either a software or hardware environment.

The relative importance of various criteria is very much dependent on the application and resources available. For example, in an environment where computing power is limited one may have to trade off a very high level of security for better performance of the system as a whole.

Cryptography, over the ages, has been an art practised by many who have devised ad hoc techniques to meet some of the information security requirements. The last twenty years have been a period of transition as the discipline moved from an art to a science. There are now several international scientific conferences devoted exclusively to cryptography and also an international scientific organization, the International Association for Cryptologic Research (IACR), aimed at fostering research in the area.

This book is about cryptography: the theory, the practice, and the standards.

1.3 Background on functions

While this book is not a treatise on abstract mathematics, a familiarity with basic mathematical concepts will prove to be useful. One concept which is absolutely fundamental to cryptography is that of a *function* in the mathematical sense. A function is alternately referred to as a *mapping* or a *transformation*.

1.3.1 Functions (1-1, one-way, trapdoor one-way)

A *set* consists of distinct objects which are called *elements* of the set. For example, a set X might consist of the elements a, b, c, and this is denoted $X = \{a, b, c\}$.

1.2 Definition A *function* is defined by two sets X and Y and a *rule* f which assigns to each element in X precisely one element in Y. The set X is called the *domain* of the function and Y the *codomain*. If x is an element of X (usually written $x \in X$) the *image* of x is the element in Y which the rule f associates with x; the image y of x is denoted by $y = f(x)$. Standard notation for a function f from set X to set Y is $f\colon X \longrightarrow Y$. If $y \in Y$, then a *preimage* of y is an element $x \in X$ for which $f(x) = y$. The set of all elements in Y which have at least one preimage is called the *image* of f, denoted $\mathrm{Im}(f)$.

1.3 Example (*function*) Consider the sets $X = \{a, b, c\}$, $Y = \{1, 2, 3, 4\}$, and the rule f from X to Y defined as $f(a) = 2$, $f(b) = 4$, $f(c) = 1$. Figure 1.2 shows a schematic of the sets X, Y and the function f. The preimage of the element 2 is a. The image of f is $\{1, 2, 4\}$. □

Thinking of a function in terms of the schematic (sometimes called a *functional diagram*) given in Figure 1.2, each element in the domain X has precisely one arrowed line originating from it. Each element in the codomain Y can have any number of arrowed lines incident to it (including zero lines).

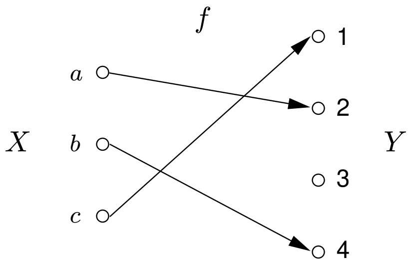

Figure 1.2: *A function f from a set X of three elements to a set Y of four elements.*

Often only the domain X and the rule f are given and the codomain is assumed to be the image of f. This point is illustrated with two examples.

1.4 Example (*function*) Take $X = \{1, 2, 3, \ldots, 10\}$ and let f be the rule that for each $x \in X$, $f(x) = r_x$, where r_x is the remainder when x^2 is divided by 11. Explicitly then

$$\begin{array}{lllll} f(1) = 1 & f(2) = 4 & f(3) = 9 & f(4) = 5 & f(5) = 3 \\ f(6) = 3 & f(7) = 5 & f(8) = 9 & f(9) = 4 & f(10) = 1. \end{array}$$

The image of f is the set $Y = \{1, 3, 4, 5, 9\}$. □

1.5 Example (*function*) Take $X = \{1, 2, 3, \ldots, 10^{50}\}$ and let f be the rule $f(x) = r_x$, where r_x is the remainder when x^2 is divided by $10^{50} + 1$ for all $x \in X$. Here it is not feasible to write down f explicitly as in Example 1.4, but nonetheless the function is completely specified by the domain and the mathematical description of the rule f. □

(i) 1-1 functions

1.6 Definition A function (or transformation) is $1 - 1$ *(one-to-one)* if each element in the codomain Y is the image of at most one element in the domain X.

1.7 Definition A function (or transformation) is *onto* if each element in the codomain Y is the image of at least one element in the domain. Equivalently, a function $f\colon X \longrightarrow Y$ is onto if $\mathrm{Im}(f) = Y$.

1.8 Definition If a function $f\colon X \longrightarrow Y$ is $1-1$ and $\mathrm{Im}(f) = Y$, then f is called a *bijection*.

1.9 Fact If $f\colon X \longrightarrow Y$ is $1 - 1$ then $f\colon X \longrightarrow \mathrm{Im}(f)$ is a bijection. In particular, if $f\colon X \longrightarrow Y$ is $1 - 1$, and X and Y are finite sets of the same size, then f is a bijection.

In terms of the schematic representation, if f is a bijection, then each element in Y has exactly one arrowed line incident with it. The functions described in Examples 1.3 and 1.4 are not bijections. In Example 1.3 the element 3 is not the image of any element in the domain. In Example 1.4 each element in the codomain has two preimages.

1.10 Definition If f is a bijection from X to Y then it is a simple matter to define a bijection g from Y to X as follows: for each $y \in Y$ define $g(y) = x$ where $x \in X$ and $f(x) = y$. This function g obtained from f is called the *inverse function* of f and is denoted by $g = f^{-1}$.

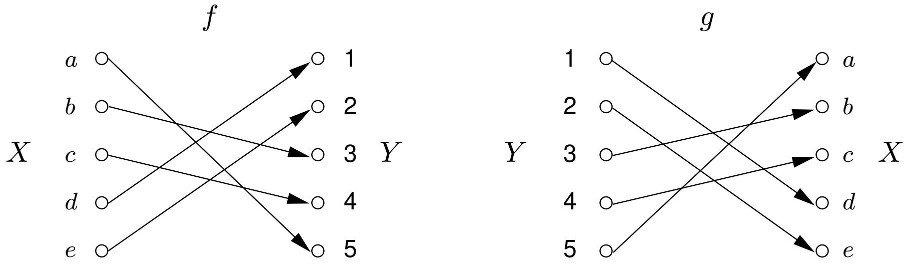

Figure 1.3: *A bijection f and its inverse $g = f^{-1}$.*

1.11 Example (*inverse function*) Let $X = \{a, b, c, d, e\}$, and $Y = \{1, 2, 3, 4, 5\}$, and consider the rule f given by the arrowed edges in Figure 1.3. f is a bijection and its inverse g is formed simply by reversing the arrows on the edges. The domain of g is Y and the codomain is X. □

Note that if f is a bijection, then so is f^{-1}. In cryptography bijections are used as the tool for encrypting messages and the inverse transformations are used to decrypt. This will be made clearer in §1.4 when some basic terminology is introduced. Notice that if the transformations were not bijections then it would not be possible to always decrypt to a unique message.

(ii) One-way functions

There are certain types of functions which play significant roles in cryptography. At the expense of rigor, an intuitive definition of a one-way function is given.

1.12 Definition A function f from a set X to a set Y is called a *one-way function* if $f(x)$ is "easy" to compute for all $x \in X$ but for "essentially all" elements $y \in \mathrm{Im}(f)$ it is "computationally infeasible" to find any $x \in X$ such that $f(x) = y$.

1.13 Note (*clarification of terms in Definition 1.12*)

(i) A rigorous definition of the terms "easy" and "computationally infeasible" is necessary but would detract from the simple idea that is being conveyed. For the purpose of this chapter, the intuitive meaning will suffice.

(ii) The phrase "for essentially all elements in Y" refers to the fact that there are a few values $y \in Y$ for which it is easy to find an $x \in X$ such that $y = f(x)$. For example, one may compute $y = f(x)$ for a small number of x values and then for these, the inverse is known by table look-up. An alternate way to describe this property of a one-way function is the following: for a random $y \in \mathrm{Im}(f)$ it is computationally infeasible to find any $x \in X$ such that $f(x) = y$.

The concept of a one-way function is illustrated through the following examples.

1.14 Example (*one-way function*) Take $X = \{1, 2, 3, \dots, 16\}$ and define $f(x) = r_x$ for all $x \in X$ where r_x is the remainder when 3^x is divided by 17. Explicitly,

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$f(x)$	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1

Given a number between 1 and 16, it is relatively easy to find the image of it under f. However, given a number such as 7, without having the table in front of you, it is harder to find

x given that $f(x) = 7$. Of course, if the number you are given is 3 then it is clear that $x = 1$ is what you need; but for most of the elements in the codomain it is not that easy. □

One must keep in mind that this is an example which uses very small numbers; the important point here is that there is a difference in the amount of work to compute $f(x)$ and the amount of work to find x given $f(x)$. Even for very large numbers, $f(x)$ can be computed efficiently using the repeated square-and-multiply algorithm (Algorithm 2.143), whereas the process of finding x from $f(x)$ is much harder.

1.15 Example (*one-way function*) A *prime number* is a positive integer greater than 1 whose only positive integer divisors are 1 and itself. Select primes $p = 48611$, $q = 53993$, form $n = pq = 2624653723$, and let $X = \{1, 2, 3, \ldots, n-1\}$. Define a function f on X by $f(x) = r_x$ for each $x \in X$, where r_x is the remainder when x^3 is divided by n. For instance, $f(2489991) = 1981394214$ since $2489991^3 = 5881949859 \cdot n + 1981394214$. Computing $f(x)$ is a relatively simple thing to do, but to reverse the procedure is much more difficult; that is, given a remainder to find the value x which was originally cubed (raised to the third power). This procedure is referred to as the computation of a modular cube root with modulus n. If the factors of n are unknown and large, this is a difficult problem; however, if the factors p and q of n are known then there is an efficient algorithm for computing modular cube roots. (See §8.2.2(i) for details.) □

Example 1.15 leads one to consider another type of function which will prove to be fundamental in later developments.

(iii) Trapdoor one-way functions

1.16 Definition A *trapdoor one-way function* is a one-way function $f\colon X \longrightarrow Y$ with the additional property that given some extra information (called the *trapdoor information*) it becomes feasible to find for any given $y \in \mathrm{Im}(f)$, an $x \in X$ such that $f(x) = y$.

Example 1.15 illustrates the concept of a trapdoor one-way function. With the additional information of the factors of $n = 2624653723$ (namely, $p = 48611$ and $q = 53993$, each of which is five decimal digits long) it becomes much easier to invert the function. The factors of 2624653723 are large enough that finding them by hand computation would be difficult. Of course, any reasonable computer program could find the factors relatively quickly. If, on the other hand, one selects p and q to be very large distinct prime numbers (each having about 100 decimal digits) then, by today's standards, it is a difficult problem, even with the most powerful computers, to deduce p and q simply from n. This is the well-known *integer factorization problem* (see §3.2) and a source of many trapdoor one-way functions.

It remains to be rigorously established whether there actually are any (true) one-way functions. That is to say, no one has yet definitively proved the existence of such functions under reasonable (and rigorous) definitions of "easy" and "computationally infeasible". Since the existence of one-way functions is still unknown, the existence of trapdoor one-way functions is also unknown. However, there are a number of good candidates for one-way and trapdoor one-way functions. Many of these are discussed in this book, with emphasis given to those which are practical.

One-way and trapdoor one-way functions are the basis for public-key cryptography (discussed in §1.8). The importance of these concepts will become clearer when their application to cryptographic techniques is considered. It will be worthwhile to keep the abstract concepts of this section in mind as concrete methods are presented.

1.3.2 Permutations

Permutations are functions which are often used in various cryptographic constructs.

1.17 Definition Let $\mathcal{S}$ be a finite set of elements. A *permutation* p on $\mathcal{S}$ is a bijection (Definition 1.8) from $\mathcal{S}$ to itself (i.e., $p\colon \mathcal{S} \longrightarrow \mathcal{S}$).

1.18 Example (*permutation*) Let $\mathcal{S} = \{1,2,3,4,5\}$. A permutation $p\colon \mathcal{S} \longrightarrow \mathcal{S}$ is defined as follows:

$$p(1) = 3,\ p(2) = 5,\ p(3) = 4,\ p(4) = 2,\ p(5) = 1.$$

A permutation can be described in various ways. It can be displayed as above or as an array:

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 2 & 1 \end{pmatrix}, \tag{1.1}$$

where the top row in the array is the domain and the bottom row is the image under the mapping p. Of course, other representations are possible. □

Since permutations are bijections, they have inverses. If a permutation is written as an array (see 1.1), its inverse is easily found by interchanging the rows in the array and reordering the elements in the new top row if desired (the bottom row would have to be reordered correspondingly). The inverse of p in Example 1.18 is $p^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 1 & 3 & 2 \end{pmatrix}$.

1.19 Example (*permutation*) Let X be the set of integers $\{0,1,2,\dots,pq-1\}$ where p and q are distinct *large* primes (for example, p and q are each about 100 decimal digits long), and suppose that neither $p-1$ nor $q-1$ is divisible by 3. Then the function $p(x) = r_x$, where r_x is the remainder when x^3 is divided by pq, can be shown to be a permutation. Determining the inverse permutation is computationally infeasible by today's standards unless p and q are known (cf. Example 1.15). □

1.3.3 Involutions

Another type of function which will be referred to in §1.5.3 is an involution. Involutions have the property that they are their own inverses.

1.20 Definition Let $\mathcal{S}$ be a finite set and let f be a bijection from $\mathcal{S}$ to $\mathcal{S}$ (i.e., $f\colon \mathcal{S} \longrightarrow \mathcal{S}$). The function f is called an *involution* if $f = f^{-1}$. An equivalent way of stating this is $f(f(x)) = x$ for all $x \in \mathcal{S}$.

1.21 Example (*involution*) Figure 1.4 is an example of an involution. In the diagram of an involution, note that if j is the image of i then i is the image of j. □

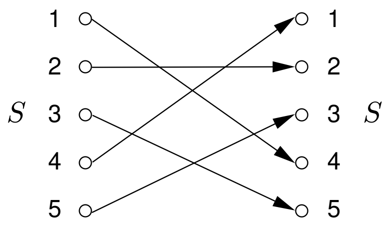

Figure 1.4: *An involution on a set S of 5 elements.*

1.4 Basic terminology and concepts

The scientific study of any discipline must be built upon rigorous definitions arising from fundamental concepts. What follows is a list of terms and basic concepts used throughout this book. Where appropriate, rigor has been sacrificed (here in Chapter 1) for the sake of clarity.

Encryption domains and codomains

- $\mathcal{A}$ denotes a finite set called the *alphabet of definition*. For example, $\mathcal{A} = \{0, 1\}$, the *binary alphabet*, is a frequently used alphabet of definition. Note that any alphabet can be encoded in terms of the binary alphabet. For example, since there are 32 binary strings of length five, each letter of the English alphabet can be assigned a unique binary string of length five.
- $\mathcal{M}$ denotes a set called the *message space*. $\mathcal{M}$ consists of strings of symbols from an alphabet of definition. An element of $\mathcal{M}$ is called a *plaintext message* or simply a *plaintext*. For example, $\mathcal{M}$ may consist of binary strings, English text, computer code, etc.
- $\mathcal{C}$ denotes a set called the *ciphertext space*. $\mathcal{C}$ consists of strings of symbols from an alphabet of definition, which may differ from the alphabet of definition for $\mathcal{M}$. An element of $\mathcal{C}$ is called a *ciphertext*.

Encryption and decryption transformations

- $\mathcal{K}$ denotes a set called the *key space*. An element of $\mathcal{K}$ is called a *key*.
- Each element $e \in \mathcal{K}$ uniquely determines a bijection from $\mathcal{M}$ to $\mathcal{C}$, denoted by E_e. E_e is called an *encryption function* or an *encryption transformation*. Note that E_e must be a bijection if the process is to be reversed and a unique plaintext message recovered for each distinct ciphertext.[1]
- For each $d \in \mathcal{K}$, D_d denotes a bijection from $\mathcal{C}$ to $\mathcal{M}$ (i.e., $D_d \colon \mathcal{C} \longrightarrow \mathcal{M}$). D_d is called a *decryption function* or *decryption transformation*.
- The process of applying the transformation E_e to a message $m \in \mathcal{M}$ is usually referred to as *encrypting* m or the *encryption* of m.
- The process of applying the transformation D_d to a ciphertext c is usually referred to as *decrypting* c or the *decryption* of c.

[1]More generality is obtained if E_e is simply defined as a $1-1$ transformation from $\mathcal{M}$ to $\mathcal{C}$. That is to say, E_e is a bijection from $\mathcal{M}$ to $\mathrm{Im}(E_e)$ where $\mathrm{Im}(E_e)$ is a subset of $\mathcal{C}$.

- An *encryption scheme* consists of a set $\{E_e : e \in \mathcal{K}\}$ of encryption transformations and a corresponding set $\{D_d : d \in \mathcal{K}\}$ of decryption transformations with the property that for each $e \in \mathcal{K}$ there is a unique key $d \in \mathcal{K}$ such that $D_d = E_e^{-1}$; that is, $D_d(E_e(m)) = m$ for all $m \in \mathcal{M}$. An encryption scheme is sometimes referred to as a *cipher*.
- The keys e and d in the preceding definition are referred to as a *key pair* and sometimes denoted by (e, d). Note that e and d could be the same.
- To *construct* an encryption scheme requires one to select a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, a key space $\mathcal{K}$, a set of encryption transformations $\{E_e : e \in \mathcal{K}\}$, and a corresponding set of decryption transformations $\{D_d : d \in \mathcal{K}\}$.

Achieving confidentiality

An encryption scheme may be used as follows for the purpose of achieving confidentiality. Two parties Alice and Bob first secretly choose or secretly exchange a key pair (e, d). At a subsequent point in time, if Alice wishes to send a message $m \in \mathcal{M}$ to Bob, she computes $c = E_e(m)$ and transmits this to Bob. Upon receiving c, Bob computes $D_d(c) = m$ and hence recovers the original message m.

The question arises as to why keys are necessary. (Why not just choose one encryption function and its corresponding decryption function?) Having transformations which are very similar but characterized by keys means that if some particular encryption/decryption transformation is revealed then one does not have to redesign the entire scheme but simply change the key. It is sound cryptographic practice to change the key (encryption/decryption transformation) frequently. As a physical analogue, consider an ordinary resettable combination lock. The structure of the lock is available to anyone who wishes to purchase one but the combination is chosen and set by the owner. If the owner suspects that the combination has been revealed he can easily reset it without replacing the physical mechanism.

1.22 Example (*encryption scheme*) Let $\mathcal{M} = \{m_1, m_2, m_3\}$ and $\mathcal{C} = \{c_1, c_2, c_3\}$. There are precisely $3! = 6$ bijections from $\mathcal{M}$ to $\mathcal{C}$. The key space $\mathcal{K} = \{1, 2, 3, 4, 5, 6\}$ has six elements in it, each specifying one of the transformations. Figure 1.5 illustrates the six encryption functions which are denoted by E_i, $1 \le i \le 6$. Alice and Bob agree on a trans-

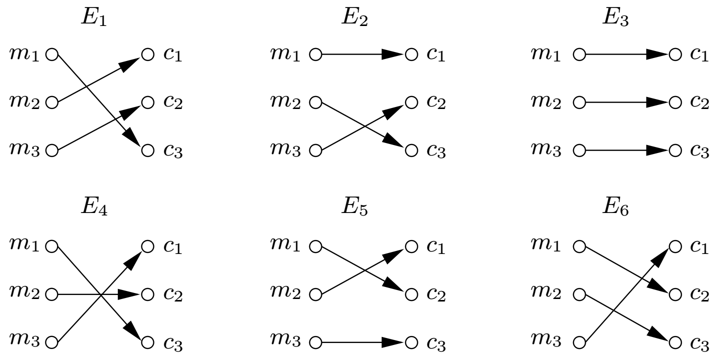

Figure 1.5: *Schematic of a simple encryption scheme.*

formation, say E_1. To encrypt the message m_1, Alice computes $E_1(m_1) = c_3$ and sends c_3 to Bob. Bob decrypts c_3 by reversing the arrows on the diagram for E_1 and observing that c_3 points to m_1.

When $\mathcal{M}$ is a small set, the functional diagram is a simple visual means to describe the mapping. In cryptography, the set $\mathcal{M}$ is typically of astronomical proportions and, as such, the visual description is infeasible. What is required, in these cases, is some other simple means to describe the encryption and decryption transformations, such as mathematical algorithms. □

Figure 1.6 provides a simple model of a two-party communication using encryption.

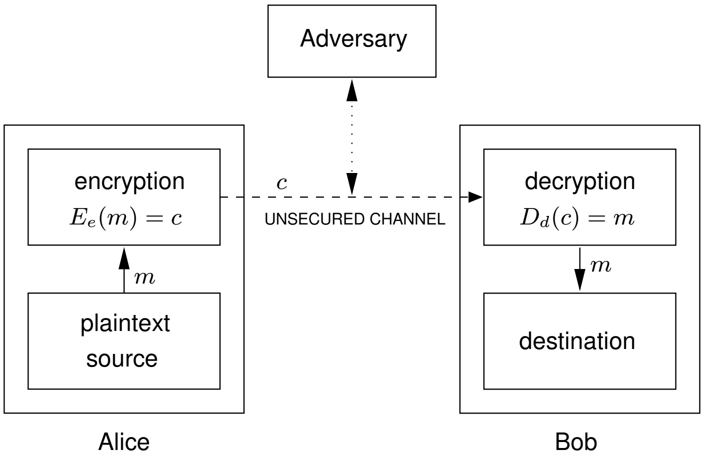

Figure 1.6: *Schematic of a two-party communication using encryption.*

Communication participants

Referring to Figure 1.6, the following terminology is defined.

- An *entity* or *party* is someone or something which sends, receives, or manipulates information. Alice and Bob are entities in Example 1.22. An entity may be a person, a computer terminal, etc.
- A *sender* is an entity in a two-party communication which is the legitimate transmitter of information. In Figure 1.6, the sender is Alice.
- A *receiver* is an entity in a two-party communication which is the intended recipient of information. In Figure 1.6, the receiver is Bob.
- An *adversary* is an entity in a two-party communication which is neither the sender nor receiver, and which tries to defeat the information security service being provided between the sender and receiver. Various other names are synonymous with adversary such as enemy, attacker, opponent, tapper, eavesdropper, intruder, and interloper. An adversary will often attempt to play the role of either the legitimate sender or the legitimate receiver.

Channels

- A *channel* is a means of conveying information from one entity to another.
- A *physically secure channel* or *secure channel* is one which is not physically accessible to the adversary.
- An *unsecured channel* is one from which parties other than those for which the information is intended can reorder, delete, insert, or read.
- A *secured channel* is one from which an adversary does not have the ability to reorder, delete, insert, or read.

One should note the subtle difference between a physically secure channel and a secured channel – a secured channel may be secured by physical or cryptographic techniques, the latter being the topic of this book. Certain channels are assumed to be physically secure. These include trusted couriers, personal contact between communicating parties, and a dedicated communication link, to name a few.

Security

A fundamental premise in cryptography is that the sets $\mathcal{M}, \mathcal{C}, \mathcal{K}, \{E_e : e \in \mathcal{K}\}, \{D_d : d \in \mathcal{K}\}$ are public knowledge. When two parties wish to communicate securely using an encryption scheme, the only thing that they keep secret is the particular key pair (e, d) which they are using, and which they must select. One can gain additional security by keeping the class of encryption and decryption transformations secret but one should not base the security of the entire scheme on this approach. History has shown that maintaining the secrecy of the transformations is very difficult indeed.

1.23 Definition An encryption scheme is said to be *breakable* if a third party, without prior knowledge of the key pair (e, d), can systematically recover plaintext from corresponding ciphertext within some appropriate time frame.

An appropriate time frame will be a function of the useful lifespan of the data being protected. For example, an instruction to buy a certain stock may only need to be kept secret for a few minutes whereas state secrets may need to remain confidential indefinitely.

An encryption scheme can be broken by trying all possible keys to see which one the communicating parties are using (assuming that the class of encryption functions is public knowledge). This is called an *exhaustive search* of the key space. It follows then that the number of keys (i.e., the size of the key space) should be large enough to make this approach computationally infeasible. It is the objective of a designer of an encryption scheme that this be the best approach to break the system.

Frequently cited in the literature are *Kerckhoffs' desiderata*, a set of requirements for cipher systems. They are given here essentially as Kerckhoffs originally stated them:

1. the system should be, if not theoretically unbreakable, unbreakable in practice;
2. compromise of the system details should not inconvenience the correspondents;
3. the key should be rememberable without notes and easily changed;
4. the cryptogram should be transmissible by telegraph;
5. the encryption apparatus should be portable and operable by a single person; and
6. the system should be easy, requiring neither the knowledge of a long list of rules nor mental strain.

This list of requirements was articulated in 1883 and, for the most part, remains useful today. Point 2 allows that the class of encryption transformations being used be publicly known and that the security of the system should reside only in the key chosen.

Information security in general

So far the terminology has been restricted to encryption and decryption with the goal of privacy in mind. Information security is much broader, encompassing such things as authentication and data integrity. A few more general definitions, pertinent to discussions later in the book, are given next.

- An *information security service* is a method to provide some specific aspect of security. For example, integrity of transmitted data is a security objective, and a method to ensure this aspect is an information security service.

- *Breaking* an information security service (which often involves more than simply encryption) implies defeating the objective of the intended service.
- A *passive adversary* is an adversary who is capable only of reading information from an unsecured channel.
- An *active adversary* is an adversary who may also transmit, alter, or delete information on an unsecured channel.

Cryptology

- *Cryptanalysis* is the study of mathematical techniques for attempting to defeat cryptographic techniques, and, more generally, information security services.
- A *cryptanalyst* is someone who engages in cryptanalysis.
- *Cryptology* is the study of cryptography (Definition 1.1) and cryptanalysis.
- A *cryptosystem* is a general term referring to a set of cryptographic primitives used to provide information security services. Most often the term is used in conjunction with primitives providing confidentiality, i.e., encryption.

Cryptographic techniques are typically divided into two generic types: *symmetric-key* and *public-key*. Encryption methods of these types will be discussed separately in §1.5 and §1.8. Other definitions and terminology will be introduced as required.

1.5 Symmetric-key encryption

§1.5 considers symmetric-key encryption. Public-key encryption is the topic of §1.8.

1.5.1 Overview of block ciphers and stream ciphers

1.24 Definition Consider an encryption scheme consisting of the sets of encryption and decryption transformations $\{E_e\colon e \in \mathcal{K}\}$ and $\{D_d\colon d \in \mathcal{K}\}$, respectively, where $\mathcal{K}$ is the key space. The encryption scheme is said to be *symmetric-key* if for each associated encryption/decryption key pair (e, d), it is computationally "easy" to determine d knowing only e, and to determine e from d.

Since $e = d$ in most practical symmetric-key encryption schemes, the term symmetric-key becomes appropriate. Other terms used in the literature are *single-key*, *one-key*, *private-key*,[2] and *conventional* encryption. Example 1.25 illustrates the idea of symmetric-key encryption.

1.25 Example (*symmetric-key encryption*) Let $\mathcal{A} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \dots, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$ be the English alphabet. Let $\mathcal{M}$ and $\mathcal{C}$ be the set of all strings of length five over $\mathcal{A}$. The key e is chosen to be a permutation on $\mathcal{A}$. To encrypt, an English message is broken up into groups each having five letters (with appropriate padding if the length of the message is not a multiple of five) and a permutation e is applied to each letter one at a time. To decrypt, the inverse permutation $d = e^{-1}$ is applied to each letter of the ciphertext. For instance, suppose that the key e is chosen to be the permutation which maps each letter to the one which is three positions to its right, as shown below

$$e = \begin{pmatrix} \mathrm{A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z} \\ \mathrm{D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z\ A\ B\ C} \end{pmatrix}$$

[2] Private key is a term also used in quite a different context (see §1.8). The term will be reserved for the latter usage in this book.

A message

$$m = \text{THISC IPHER ISCER TAINL YNOTS ECURE}$$

is encrypted to

$$c = E_e(m) = \text{WKLVF LSKHU LVFHU WDLQO BQRWV HFXUH}. \qquad \square$$

A two-party communication using symmetric-key encryption can be described by the block diagram of Figure 1.7, which is Figure 1.6 with the addition of the secure (both con-

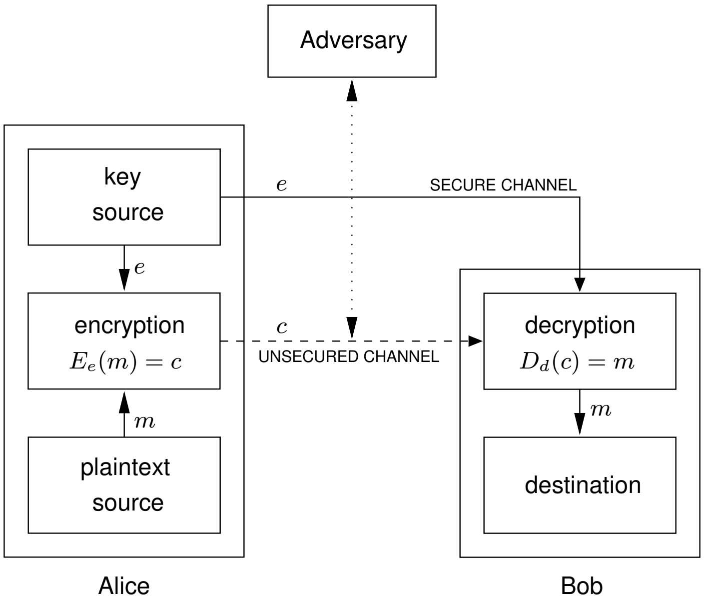

Figure 1.7: *Two-party communication using encryption, with a secure channel for key exchange. The decryption key d can be efficiently computed from the encryption key e.*

fidential and authentic) channel. One of the major issues with symmetric-key systems is to find an efficient method to agree upon and exchange keys securely. This problem is referred to as the *key distribution problem* (see Chapters 12 and 13).

It is assumed that all parties know the set of encryption/decryption transformations (i.e., they all know the encryption scheme). As has been emphasized several times the only information which should be required to be kept secret is the key d. However, in symmetric-key encryption, this means that the key e must also be kept secret, as d can be deduced from e. In Figure 1.7 the encryption key e is transported from one entity to the other with the understanding that both can construct the decryption key d.

There are two classes of symmetric-key encryption schemes which are commonly distinguished: *block ciphers* and *stream ciphers*.

1.26 Definition A *block cipher* is an encryption scheme which breaks up the plaintext messages to be transmitted into strings (called *blocks*) of a fixed length t over an alphabet $\mathcal{A}$, and encrypts one block at a time.

Most well-known symmetric-key encryption techniques are block ciphers. A number of examples of these are given in Chapter 7. Two important classes of block ciphers are *substitution ciphers* and *transposition ciphers* (§1.5.2). Product ciphers (§1.5.3) combine these. Stream ciphers are considered in §1.5.4, while comments on the key space follow in §1.5.5.

1.5.2 Substitution ciphers and transposition ciphers

Substitution ciphers are block ciphers which replace symbols (or groups of symbols) by other symbols or groups of symbols.

Simple substitution ciphers

1.27 Definition Let $\mathcal{A}$ be an alphabet of q symbols and $\mathcal{M}$ be the set of all strings of length t over $\mathcal{A}$. Let $\mathcal{K}$ be the set of all permutations on the set $\mathcal{A}$. Define for each $e \in \mathcal{K}$ an encryption transformation E_e as:

$$E_e(m) = (e(m_1)e(m_2)\cdots e(m_t)) = (c_1c_2\cdots c_t) = c,$$

where $m = (m_1m_2\cdots m_t) \in \mathcal{M}$. In other words, for each symbol in a t-tuple, replace (substitute) it by another symbol from $\mathcal{A}$ according to some fixed permutation e. To decrypt $c = (c_1c_2\cdots c_t)$ compute the inverse permutation $d = e^{-1}$ and

$$D_d(c) = (d(c_1)d(c_2)\cdots d(c_t)) = (m_1m_2\cdots m_t) = m.$$

E_e is called a *simple substitution cipher* or a *mono-alphabetic substitution cipher*.

The number of distinct substitution ciphers is $q!$ and is independent of the block size in the cipher. Example 1.25 is an example of a simple substitution cipher of block length five.

Simple substitution ciphers over small block sizes provide inadequate security even when the key space is extremely large. If the alphabet is the English alphabet as in Example 1.25, then the size of the key space is $26! \approx 4 \times 10^{26}$, yet the key being used can be determined quite easily by examining a modest amount of ciphertext. This follows from the simple observation that the distribution of letter frequencies is preserved in the ciphertext. For example, the letter E occurs more frequently than the other letters in ordinary English text. Hence the letter occurring most frequently in a sequence of ciphertext blocks is most likely to correspond to the letter E in the plaintext. By observing a modest quantity of ciphertext blocks, a cryptanalyst can determine the key.

Homophonic substitution ciphers

1.28 Definition To each symbol $a \in \mathcal{A}$, associate a set $H(a)$ of strings of t symbols, with the restriction that the sets $H(a)$, $a \in \mathcal{A}$, be pairwise disjoint. A *homophonic substitution cipher* replaces each symbol a in a plaintext message block with a randomly chosen string from $H(a)$. To decrypt a string c of t symbols, one must determine an $a \in \mathcal{A}$ such that $c \in H(a)$. The key for the cipher consists of the sets $H(a)$.

1.29 Example (*homophonic substitution cipher*) Consider $\mathcal{A} = \{a, b\}$, $H(a) = \{00, 10\}$, and $H(b) = \{01, 11\}$. The plaintext message block ab encrypts to one of the following: 0001, 0011, 1001, 1011. Observe that the codomain of the encryption function (for messages of length two) consists of the following pairwise disjoint sets of four-element bitstrings:

$$\begin{array}{rcl} aa & \longrightarrow & \{0000, 0010, 1000, 1010\} \\ ab & \longrightarrow & \{0001, 0011, 1001, 1011\} \\ ba & \longrightarrow & \{0100, 0110, 1100, 1110\} \\ bb & \longrightarrow & \{0101, 0111, 1101, 1111\} \end{array}$$

Any 4-bitstring uniquely identifies a codomain element, and hence a plaintext message. □

Often the symbols do not occur with equal frequency in plaintext messages. With a simple substitution cipher this non-uniform frequency property is reflected in the ciphertext as illustrated in Example 1.25. A homophonic cipher can be used to make the frequency of occurrence of ciphertext symbols more uniform, at the expense of data expansion. Decryption is not as easily performed as it is for simple substitution ciphers.

Polyalphabetic substitution ciphers

1.30 Definition A *polyalphabetic substitution cipher* is a block cipher with block length t over an alphabet $\mathcal{A}$ having the following properties:

(i) the key space $\mathcal{K}$ consists of all ordered sets of t permutations $(p_1, p_2, \ldots, p_t)$, where each permutation p_i is defined on the set $\mathcal{A}$;
(ii) encryption of the message $m = (m_1 m_2 \cdots m_t)$ under the key $e = (p_1, p_2, \ldots, p_t)$ is given by $E_e(m) = (p_1(m_1) p_2(m_2) \cdots p_t(m_t))$; and
(iii) the decryption key associated with $e = (p_1, p_2, \ldots, p_t)$ is $d = (p_1^{-1}, p_2^{-1}, \ldots, p_t^{-1})$.

1.31 Example (*Vigenère cipher*) Let $\mathcal{A} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \ldots, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$ and $t = 3$. Choose $e = (p_1, p_2, p_3)$, where p_1 maps each letter to the letter three positions to its right in the alphabet, p_2 to the one seven positions to its right, and p_3 ten positions to its right. If

$$m = \text{THI SCI PHE RIS CER TAI NLY NOT SEC URE}$$

then

$$c = E_e(m) = \text{WOS VJS SOO UPC FLB WHS QSI QVD VLM XYO}. \qquad \square$$

Polyalphabetic ciphers have the advantage over simple substitution ciphers that symbol frequencies are not preserved. In the example above, the letter E is encrypted to both O and L. However, polyalphabetic ciphers are not significantly more difficult to cryptanalyze, the approach being similar to the simple substitution cipher. In fact, once the block length t is determined, the ciphertext letters can be divided into t groups (where group i, $1 \le i \le t$, consists of those ciphertext letters derived using permutation p_i), and a frequency analysis can be done on each group.

Transposition ciphers

Another class of symmetric-key ciphers is the simple transposition cipher, which simply permutes the symbols in a block.

1.32 Definition Consider a symmetric-key block encryption scheme with block length t. Let $\mathcal{K}$ be the set of all permutations on the set $\{1, 2, \ldots, t\}$. For each $e \in \mathcal{K}$ define the encryption function

$$E_e(m) = (m_{e(1)} m_{e(2)} \cdots m_{e(t)})$$

where $m = (m_1 m_2 \cdots m_t) \in \mathcal{M}$, the message space. The set of all such transformations is called a *simple transposition cipher.* The decryption key corresponding to e is the inverse permutation $d = e^{-1}$. To decrypt $c = (c_1 c_2 \cdots c_t)$, compute $D_d(c) = (c_{d(1)} c_{d(2)} \cdots c_{d(t)})$.

A simple transposition cipher preserves the number of symbols of a given type within a block, and thus is easily cryptanalyzed.

1.5.3 Composition of ciphers

In order to describe product ciphers, the concept of composition of functions is introduced. Compositions are a convenient way of constructing more complicated functions from simpler ones.

Composition of functions

1.33 Definition Let $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{U}$ be finite sets and let $f\colon \mathcal{S} \longrightarrow \mathcal{T}$ and $g\colon \mathcal{T} \longrightarrow \mathcal{U}$ be functions. The *composition* of g with f, denoted $g \circ f$ (or simply gf), is a function from $\mathcal{S}$ to $\mathcal{U}$ as illustrated in Figure 1.8 and defined by $(g \circ f)(x) = g(f(x))$ for all $x \in \mathcal{S}$.

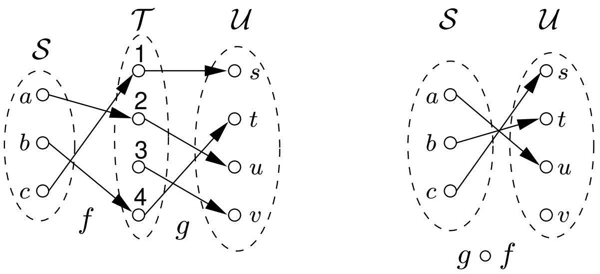

Figure 1.8: *The composition $g \circ f$ of functions g and f.*

Composition can be easily extended to more than two functions. For functions f_1, f_2, $\ldots$, f_t, one can define $f_t \circ \cdots \circ f_2 \circ f_1$, provided that the domain of f_t equals the codomain of f_{t-1} and so on.

Compositions and involutions

Involutions were introduced in §1.3.3 as a simple class of functions with an interesting property: $E_k(E_k(x)) = x$ for all x in the domain of E_k; that is, $E_k \circ E_k$ is the identity function.

1.34 Remark (*composition of involutions*) The composition of two involutions is not necessarily an involution, as illustrated in Figure 1.9. However, involutions may be composed to get somewhat more complicated functions whose inverses are easy to find. This is an important feature for decryption. For example if $E_{k_1}, E_{k_2}, \ldots , E_{k_t}$ are involutions then the inverse of $E_k = E_{k_1} E_{k_2} \cdots E_{k_t}$ is $E_k^{-1} = E_{k_t} E_{k_{t-1}} \cdots E_{k_1}$, the composition of the involutions in the reverse order.

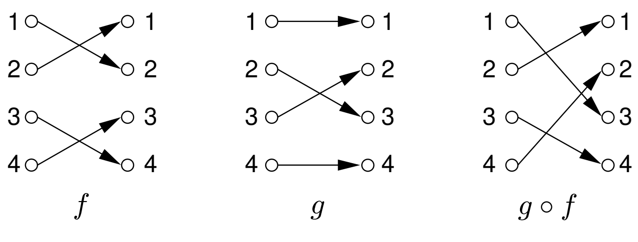

Figure 1.9: *The composition $g \circ f$ of involutions g and f is not an involution.*

Product ciphers

Simple substitution and transposition ciphers individually do not provide a very high level of security. However, by combining these transformations it is possible to obtain strong ciphers. As will be seen in Chapter 7 some of the most practical and effective symmetric-key systems are product ciphers. One example of a *product cipher* is a composition of $t \geq 2$ transformations $E_{k_1} E_{k_2} \cdots E_{k_t}$ where each E_{k_i}, $1 \leq i \leq t$, is either a substitution or a transposition cipher. For the purpose of this introduction, let the composition of a substitution and a transposition be called a *round*.

1.35 Example (*product cipher*) Let $\mathcal{M} = \mathcal{C} = \mathcal{K}$ be the set of all binary strings of length six. The number of elements in $\mathcal{M}$ is $2^6 = 64$. Let $m = (m_1 m_2 \cdots m_6)$ and define

$$\begin{aligned} E_k^{(1)}(m) &= m \oplus k, \text{ where } k \in \mathcal{K}, \\ E^{(2)}(m) &= (m_4 m_5 m_6 m_1 m_2 m_3). \end{aligned}$$

Here, $\oplus$ is the *exclusive-OR* (XOR) operation defined as follows: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$. $E_k^{(1)}$ is a polyalphabetic substitution cipher and $E^{(2)}$ is a transposition cipher (not involving the key). The product $E_k^{(1)} E^{(2)}$ is a round. While here the transposition cipher is very simple and is not determined by the key, this need not be the case. □

1.36 Remark (*confusion and diffusion*) A substitution in a round is said to add *confusion* to the encryption process whereas a transposition is said to add *diffusion*. Confusion is intended to make the relationship between the key and ciphertext as complex as possible. Diffusion refers to rearranging or spreading out the bits in the message so that any redundancy in the plaintext is spread out over the ciphertext. A round then can be said to add both confusion and diffusion to the encryption. Most modern block cipher systems apply a number of rounds in succession to encrypt plaintext.

1.5.4 Stream ciphers

Stream ciphers form an important class of symmetric-key encryption schemes. They are, in one sense, very simple block ciphers having block length equal to one. What makes them useful is the fact that the encryption transformation can change for each symbol of plaintext being encrypted. In situations where transmission errors are highly probable, stream ciphers are advantageous because they have no error propagation. They can also be used when the data must be processed one symbol at a time (e.g., if the equipment has no memory or buffering of data is limited).

1.37 Definition Let $\mathcal{K}$ be the key space for a set of encryption transformations. A sequence of symbols $e_1 e_2 e_3 \cdots e_i \in \mathcal{K}$, is called a *keystream*.

1.38 Definition Let $\mathcal{A}$ be an alphabet of q symbols and let E_e be a simple substitution cipher with block length 1 where $e \in \mathcal{K}$. Let $m_1 m_2 m_3 \cdots$ be a plaintext string and let $e_1 e_2 e_3 \cdots$ be a keystream from $\mathcal{K}$. A *stream cipher* takes the plaintext string and produces a ciphertext string $c_1 c_2 c_3 \cdots$ where $c_i = E_{e_i}(m_i)$. If d_i denotes the inverse of e_i, then $D_{d_i}(c_i) = m_i$ decrypts the ciphertext string.

A stream cipher applies simple encryption transformations according to the keystream being used. The keystream could be generated at random, or by an algorithm which generates the keystream from an initial small keystream (called a *seed*), or from a seed and previous ciphertext symbols. Such an algorithm is called a *keystream generator*.

The Vernam cipher

A motivating factor for the Vernam cipher was its simplicity and ease of implementation.

1.39 Definition The *Vernam Cipher* is a stream cipher defined on the alphabet $\mathcal{A} = \{0, 1\}$. A binary message $m_1 m_2 \cdots m_t$ is operated on by a binary key string $k_1 k_2 \cdots k_t$ of the same length to produce a ciphertext string $c_1 c_2 \cdots c_t$ where

$$c_i = m_i \oplus k_i, \;\; 1 \leq i \leq t.$$

If the key string is randomly chosen and never used again, the Vernam cipher is called a *one-time system* or a *one-time pad*.

To see how the Vernam cipher corresponds to Definition 1.38, observe that there are precisely two substitution ciphers on the set $\mathcal{A}$. One is simply the identity map E_0 which sends 0 to 0 and 1 to 1; the other E_1 sends 0 to 1 and 1 to 0. When the keystream contains a 0, apply E_0 to the corresponding plaintext symbol; otherwise, apply E_1.

If the key string is reused there are ways to attack the system. For example, if $c_1 c_2 \cdots c_t$ and $c'_1 c'_2 \cdots c'_t$ are two ciphertext strings produced by the same keystream $k_1 k_2 \cdots k_t$ then

$$c_i = m_i \oplus k_i, \quad c'_i = m'_i \oplus k_i$$

and $c_i \oplus c'_i = m_i \oplus m'_i$. The redundancy in the latter may permit cryptanalysis.

The one-time pad can be shown to be theoretically unbreakable. That is, if a cryptanalyst has a ciphertext string $c_1 c_2 \cdots c_t$ encrypted using a random key string which has been used only once, the cryptanalyst can do no better than guess at the plaintext being any binary string of length t (i.e., t-bit binary strings are equally likely as plaintext). It has been proven that to realize an unbreakable system requires a random key of the same length as the message. This reduces the practicality of the system in all but a few specialized situations. Reportedly until very recently the communication line between Moscow and Washington was secured by a one-time pad. Transport of the key was done by trusted courier.

1.5.5 The key space

The size of the key space is the number of encryption/decryption key pairs that are available in the cipher system. A key is typically a compact way to specify the encryption transformation (from the set of all encryption transformations) to be used. For example, a transposition cipher of block length t has $t!$ encryption functions from which to select. Each can be simply described by a permutation which is called the key.

It is a great temptation to relate the security of the encryption scheme to the size of the key space. The following statement is important to remember.

1.40 Fact A necessary, but usually not sufficient, condition for an encryption scheme to be secure is that the key space be large enough to preclude exhaustive search.

For instance, the simple substitution cipher in Example 1.25 has a key space of size $26! \approx 4 \times 10^{26}$. The polyalphabetic substitution cipher of Example 1.31 has a key space of size $(26!)^3 \approx 7 \times 10^{79}$. Exhaustive search of either key space is completely infeasible, yet both ciphers are relatively weak and provide little security.

1.6 Digital signatures

A cryptographic primitive which is fundamental in authentication, authorization, and non-repudiation is the *digital signature*. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of *signing* entails transforming the message and some secret information held by the entity into a tag called a *signature*. A generic description follows.

Nomenclature and set-up

- $\mathcal{M}$ is the set of messages which can be signed.
- $\mathcal{S}$ is a set of elements called *signatures*, possibly binary strings of a fixed length.
- S_A is a transformation from the message set $\mathcal{M}$ to the signature set $\mathcal{S}$, and is called a *signing transformation* for entity A.[3] The transformation S_A is kept secret by A, and will be used to create signatures for messages from $\mathcal{M}$.
- V_A is a transformation from the set $\mathcal{M} \times \mathcal{S}$ to the set $\{$*true*,*false*$\}$.[4] V_A is called a *verification transformation* for A's signatures, is publicly known, and is used by other entities to verify signatures created by A.

1.41 Definition The transformations S_A and V_A provide a *digital signature scheme* for A. Occasionally the term *digital signature mechanism* is used.

1.42 Example (*digital signature scheme*) $\mathcal{M} = \{m_1, m_2, m_3\}$ and $\mathcal{S} = \{s_1, s_2, s_3\}$. The left side of Figure 1.10 displays a signing function S_A from the set $\mathcal{M}$ and, the right side, the corresponding verification function V_A. □

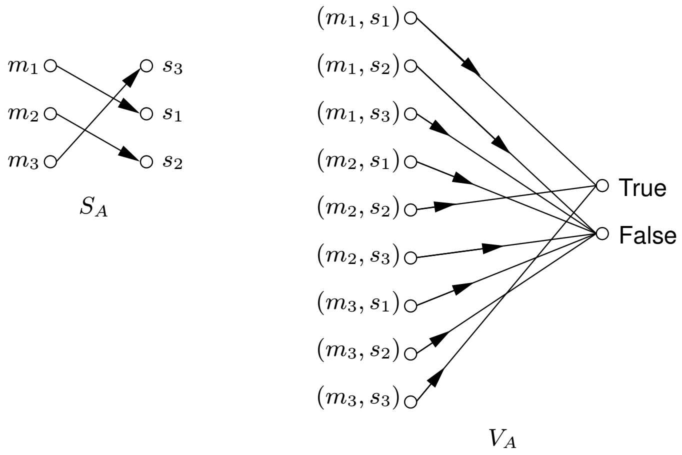

Figure 1.10: *A signing and verification function for a digital signature scheme.*

[3]The names of Alice and Bob are usually abbreviated to A and B, respectively.
[4]$\mathcal{M} \times \mathcal{S}$ consists of all pairs (m, s) where $m \in \mathcal{M}$, $s \in \mathcal{S}$, called the *Cartesian product* of $\mathcal{M}$ and $\mathcal{S}$.

Signing procedure

Entity A (the *signer*) creates a signature for a message $m \in \mathcal{M}$ by doing the following:

1. Compute $s = S_A(m)$.
2. Transmit the pair (m, s). s is called the *signature* for message m.

Verification procedure

To verify that a signature s on a message m was created by A, an entity B (the *verifier*) performs the following steps:

1. Obtain the verification function V_A of A.
2. Compute $u = V_A(m, s)$.
3. Accept the signature as having been created by A if $u =$ *true*, and reject the signature if $u =$ *false*.

1.43 Remark (*concise representation*) The transformations S_A and V_A are typically characterized more compactly by a key; that is, there is a class of signing and verification algorithms publicly known, and each algorithm is identified by a key. Thus the signing algorithm S_A of A is determined by a key k_A and A is only required to keep k_A secret. Similarly, the verification algorithm V_A of A is determined by a key l_A which is made public.

1.44 Remark (*handwritten signatures*) Handwritten signatures could be interpreted as a special class of digital signatures. To see this, take the set of signatures $\mathcal{S}$ to contain only one element which is the handwritten signature of A, denoted by s_A. The verification function simply checks if the signature on a message purportedly signed by A is s_A.

An undesirable feature in Remark 1.44 is that the signature is not message-dependent. Hence, further constraints are imposed on digital signature mechanisms as next discussed.

Properties required for signing and verification functions

There are several properties which the signing and verification transformations must satisfy.

(a) s is a valid signature of A on message m if and only if $V_A(m, s) =$ *true*.

(b) It is computationally infeasible for any entity other than A to find, for any $m \in \mathcal{M}$, an $s \in \mathcal{S}$ such that $V_A(m, s) =$ *true*.

Figure 1.10 graphically displays property (a). There is an arrowed line in the diagram for V_A from (m_i, s_j) to *true* provided there is an arrowed line from m_i to s_j in the diagram for S_A. Property (b) provides the security for the method – the signature uniquely binds A to the message which is signed.

No one has yet formally proved that digital signature schemes satisfying (b) exist (although existence is widely believed to be true); however, there are some very good candidates. §1.8.3 introduces a particular class of digital signatures which arise from public-key encryption techniques. Chapter 11 describes a number of digital signature mechanisms which are believed to satisfy the two properties cited above. Although the description of a digital signature given in this section is quite general, it can be broadened further, as presented in §11.2.

1.7 Authentication and identification

Authentication is a term which is used (and often abused) in a very broad sense. By itself it has little meaning other than to convey the idea that some means has been provided to guarantee that entities are who they claim to be, or that information has not been manipulated by unauthorized parties. Authentication is specific to the security objective which one is trying to achieve. Examples of specific objectives include access control, entity authentication, message authentication, data integrity, non-repudiation, and key authentication. These instances of authentication are dealt with at length in Chapters 9 through 13. For the purposes of this chapter, it suffices to give a brief introduction to authentication by describing several of the most obvious applications.

Authentication is one of the most important of all information security objectives. Until the mid 1970s it was generally believed that secrecy and authentication were intrinsically connected. With the discovery of hash functions (§1.9) and digital signatures (§1.6), it was realized that secrecy and authentication were truly separate and independent information security objectives. It may at first not seem important to separate the two but there are situations where it is not only useful but essential. For example, if a two-party communication between Alice and Bob is to take place where Alice is in one country and Bob in another, the host countries might not permit secrecy on the channel; one or both countries might want the ability to monitor all communications. Alice and Bob, however, would like to be assured of the identity of each other, and of the integrity and origin of the information they send and receive.

The preceding scenario illustrates several independent aspects of authentication. If Alice and Bob desire assurance of each other's identity, there are two possibilities to consider.

1. Alice and Bob could be communicating with no appreciable time delay. That is, they are both active in the communication in "real time".
2. Alice or Bob could be exchanging messages with some delay. That is, messages might be routed through various networks, stored, and forwarded at some later time.

In the first instance Alice and Bob would want to verify identities in real time. This might be accomplished by Alice sending Bob some challenge, to which Bob is the only entity which can respond correctly. Bob could perform a similar action to identify Alice. This type of authentication is commonly referred to as *entity authentication* or more simply *identification*.

For the second possibility, it is not convenient to challenge and await response, and moreover the communication path may be only in one direction. Different techniques are now required to authenticate the originator of the message. This form of authentication is called *data origin authentication*.

1.7.1 Identification

1.45 Definition An *identification* or *entity authentication* technique assures one party (through acquisition of corroborative evidence) of both the identity of a second party involved, and that the second was active at the time the evidence was created or acquired.

Typically the only data transmitted is that necessary to identify the communicating parties. The entities are both active in the communication, giving a timeliness guarantee.

1.46 Example (*identification*) A calls B on the telephone. If A and B know each other then entity authentication is provided through voice recognition. Although not foolproof, this works effectively in practice. □

1.47 Example (*identification*) Person A provides to a banking machine a personal identification number (PIN) along with a magnetic stripe card containing information about A. The banking machine uses the information on the card and the PIN to verify the identity of the card holder. If verification succeeds, A is given access to various services offered by the machine. □

Example 1.46 is an instance of *mutual authentication* whereas Example 1.47 only provides *unilateral authentication*. Numerous mechanisms and protocols devised to provide mutual or unilateral authentication are discussed in Chapter 10.

1.7.2 Data origin authentication

1.48 Definition *Data origin authentication* or *message authentication* techniques provide to one party which receives a message assurance (through corroborative evidence) of the identity of the party which originated the message.

Often a message is provided to B along with additional information so that B can determine the identity of the entity who originated the message. This form of authentication typically provides no guarantee of timeliness, but is useful in situations where one of the parties is not active in the communication.

1.49 Example (*need for data origin authentication*) A sends to B an electronic mail message (e-mail). The message may travel through various network communications systems and be stored for B to retrieve at some later time. A and B are usually not in direct communication. B would like some means to verify that the message received and purportedly created by A did indeed originate from A. □

Data origin authentication implicitly provides data integrity since, if the message was modified during transmission, A would no longer be the originator.

1.8 Public-key cryptography

The concept of public-key encryption is simple and elegant, but has far-reaching consequences.

1.8.1 Public-key encryption

Let $\{E_e\colon e \in \mathcal{K}\}$ be a set of encryption transformations, and let $\{D_d\colon d \in \mathcal{K}\}$ be the set of corresponding decryption transformations, where $\mathcal{K}$ is the key space. Consider any pair of associated encryption/decryption transformations (E_e, D_d) and suppose that each pair has the property that knowing E_e it is computationally infeasible, given a random ciphertext $c \in \mathcal{C}$, to find the message $m \in \mathcal{M}$ such that $E_e(m) = c$. This property implies that given e it is infeasible to determine the corresponding decryption key d. (Of course e and d are simply means to describe the encryption and decryption functions, respectively.) E_e is being viewed here as a trapdoor one-way function (Definition 1.16) with d being the trapdoor information necessary to compute the inverse function and hence allow decryption. This is unlike symmetric-key ciphers where e and d are essentially the same.

Under these assumptions, consider the two-party communication between Alice and Bob illustrated in Figure 1.11. Bob selects the key pair (e, d). Bob sends the encryption key e (called the *public key*) to Alice over any channel but keeps the decryption key d (called the *private key*) secure and secret. Alice may subsequently send a message m to Bob by applying the encryption transformation determined by Bob's public key to get $c = E_e(m)$. Bob decrypts the ciphertext c by applying the inverse transformation D_d uniquely determined by d.

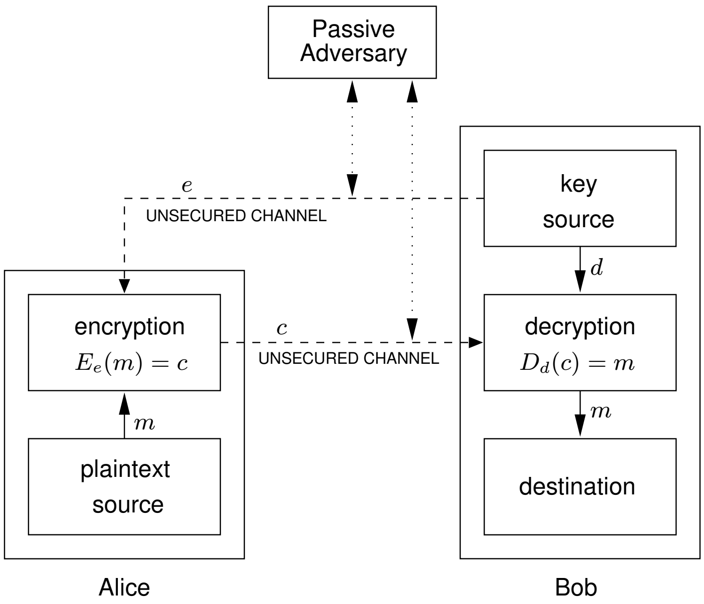

Figure 1.11: *Encryption using public-key techniques.*

Notice how Figure 1.11 differs from Figure 1.7 for a symmetric-key cipher. Here the encryption key is transmitted to Alice over an unsecured channel. This unsecured channel may be the same channel on which the ciphertext is being transmitted (but see §1.8.2).

Since the encryption key e need not be kept secret, it may be made public. Any entity can subsequently send encrypted messages to Bob which only Bob can decrypt. Figure 1.12 illustrates this idea, where A_1, A_2, and A_3 are distinct entities. Note that if A_1 destroys message m_1 after encrypting it to c_1, then even A_1 cannot recover m_1 from c_1.

As a physical analogue, consider a metal box with the lid secured by a combination lock. The combination is known only to Bob. If the lock is left open and made publicly available then anyone can place a message inside and lock the lid. Only Bob can retrieve the message. Even the entity which placed the message into the box is unable to retrieve it.

Public-key encryption, as described here, assumes that knowledge of the public key e does not allow computation of the private key d. In other words, this assumes the existence of trapdoor one-way functions (§1.3.1(iii)).

1.50 Definition Consider an encryption scheme consisting of the sets of encryption and decryp-

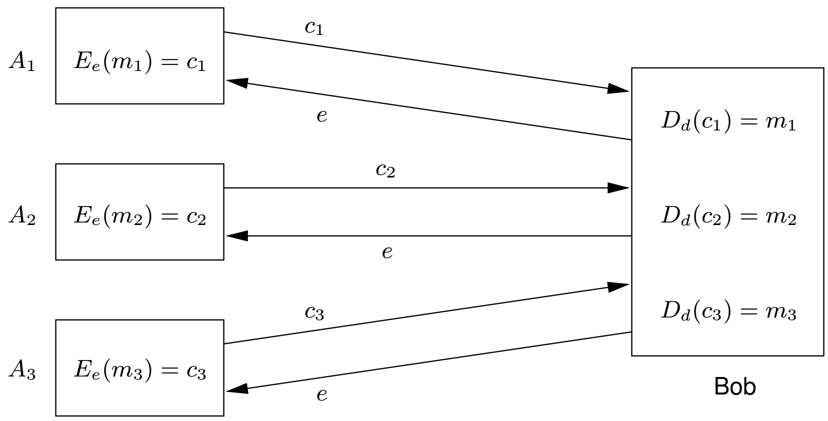

Figure 1.12: *Schematic use of public-key encryption.*

tion transformations $\{E_e : e \in \mathcal{K}\}$ and $\{D_d : d \in \mathcal{K}\}$, respectively. The encryption method is said to be a *public-key encryption scheme* if for each associated encryption/decryption pair (e, d), one key e (the *public key*) is made publicly available, while the other d (the *private key*) is kept secret. For the scheme to be *secure*, it must be computationally infeasible to compute d from e.

1.51 Remark (*private key vs. secret key*) To avoid ambiguity, a common convention is to use the term *private key* in association with public-key cryptosystems, and *secret key* in association with symmetric-key cryptosystems. This may be motivated by the following line of thought: it takes two or more parties to *share* a secret, but a key is truly *private* only when one party alone knows it.

There are many schemes known which are widely believed to be secure public-key encryption methods, but none have been mathematically proven to be secure independent of qualifying assumptions. This is not unlike the symmetric-key case where the only system which has been proven secure is the one-time pad (§1.5.4).

1.8.2 The necessity of authentication in public-key systems

It would appear that public-key cryptography is an ideal system, not requiring a secure channel to pass the encryption key. This would imply that two entities could communicate over an unsecured channel without ever having met to exchange keys. Unfortunately, this is not the case. Figure 1.13 illustrates how an active adversary can defeat the system (decrypt messages intended for a second entity) without breaking the encryption system. This is a type of *impersonation* and is an example of *protocol failure* (see §1.10). In this scenario the adversary impersonates entity B by sending entity A a public key e' which A assumes (incorrectly) to be the public key of B. The adversary intercepts encrypted messages from A to B, decrypts with its own private key d', re-encrypts the message under B's public key e, and sends it on to B. This highlights the necessity to *authenticate* public keys to achieve data origin authentication of the public keys themselves. A must be convinced that she is encrypting under the legitimate public key of B. Fortunately, public-key techniques also allow an elegant solution to this problem (see §1.11).

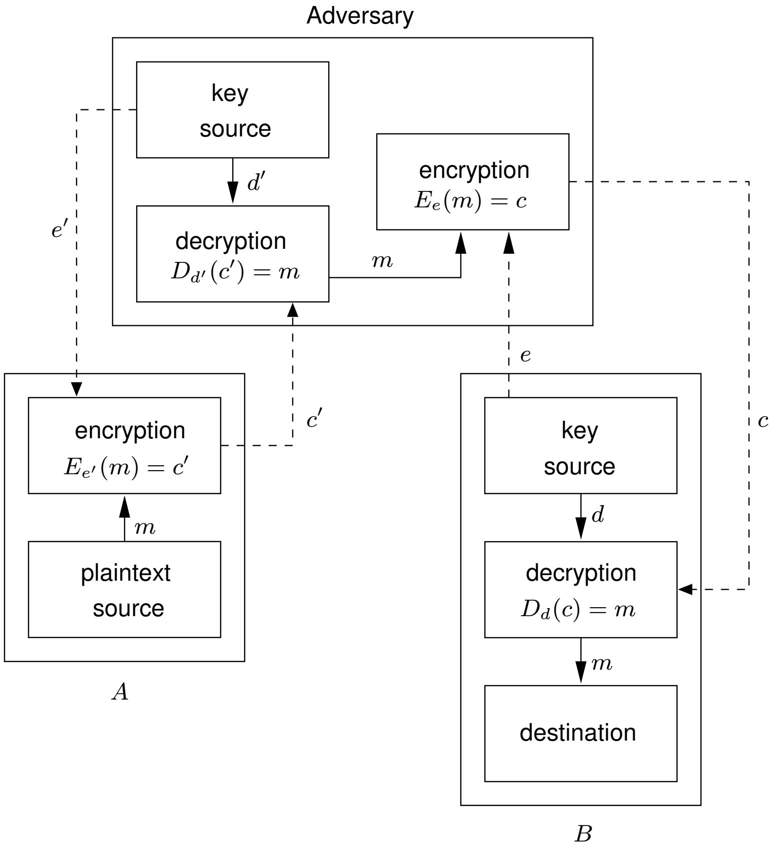

Figure 1.13: *An impersonation attack on a two-party communication.*

1.8.3 Digital signatures from reversible public-key encryption

This section considers a class of digital signature schemes which is based on public-key encryption systems of a particular type.

Suppose E_e is a public-key encryption transformation with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Suppose further that $\mathcal{M} = \mathcal{C}$. If D_d is the decryption transformation corresponding to E_e then since E_e and D_d are both permutations, one has

$$D_d(E_e(m)) = E_e(D_d(m)) = m, \text{ for all } m \in \mathcal{M}.$$

A public-key encryption scheme of this type is called *reversible*.[5] Note that it is essential that $\mathcal{M} = \mathcal{C}$ for this to be a valid equality for all $m \in \mathcal{M}$; otherwise, $D_d(m)$ will be meaningless for $m \notin \mathcal{C}$.

[5]There is a broader class of digital signatures which can be informally described as arising from *irreversible* cryptographic algorithms. These are described in §11.2.

Construction for a digital signature scheme

1. Let $\mathcal{M}$ be the message space for the signature scheme.
2. Let $\mathcal{C} = \mathcal{M}$ be the signature space $\mathcal{S}$.
3. Let (e, d) be a key pair for the public-key encryption scheme.
4. Define the signing function S_A to be D_d. That is, the signature for a message $m \in \mathcal{M}$ is $s = D_d(m)$.
5. Define the verification function V_A by

$$V_A(m, s) = \begin{cases} \textit{true}, & \text{if } E_e(s) = m, \\ \textit{false}, & \text{otherwise.} \end{cases}$$

The signature scheme can be simplified further if A only signs messages having a special structure, and this structure is publicly known. Let $\mathcal{M}'$ be a subset of $\mathcal{M}$ where elements of $\mathcal{M}'$ have a well-defined special structure, such that $\mathcal{M}'$ contains only a negligible fraction of messages from the set. For example, suppose that $\mathcal{M}$ consists of all binary strings of length $2t$ for some positive integer t. Let $\mathcal{M}'$ be the subset of $\mathcal{M}$ consisting of all strings where the first t bits are replicated in the last t positions (e.g., 101101 would be in $\mathcal{M}'$ for $t = 3$). If A only signs messages within the subset $\mathcal{M}'$, these are easily recognized by a verifier.

Redefine the verification function V_A as

$$V_A(s) = \begin{cases} \textit{true}, & \text{if } E_e(s) \in \mathcal{M}', \\ \textit{false}, & \text{otherwise.} \end{cases}$$

Under this new scenario A only needs to transmit the signature s since the message $m = E_e(s)$ can be recovered by applying the verification function. Such a scheme is called a *digital signature scheme with message recovery*. Figure 1.14 illustrates how this signature function is used. The feature of selecting messages of special structure is referred to as selecting messages with *redundancy*.

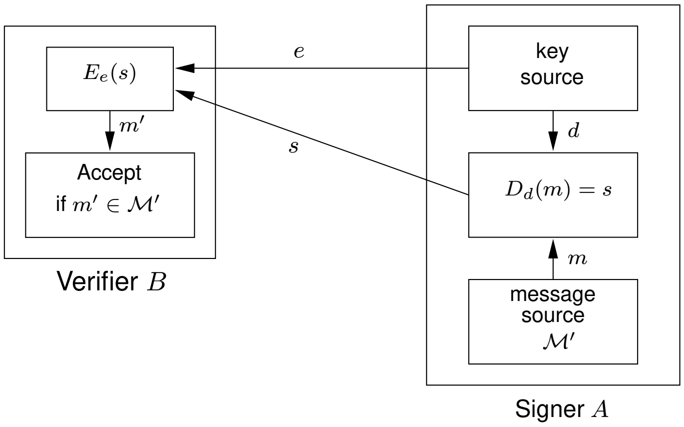

Figure 1.14: *A digital signature scheme with message recovery.*

The modification presented above is more than a simplification; it is absolutely crucial if one hopes to meet the requirement of property (b) of signing and verification functions (see page 23). To see why this is the case, note that any entity B can select a random element $s \in \mathcal{S}$ as a signature and apply E_e to get $u = E_e(s)$, since $\mathcal{S} = \mathcal{M}$ and E_e is public knowledge. B may then take the message $m = u$ and the signature on m to be s and transmits (m, s). It is easy to check that s will verify as a signature created by A for m but in which A has had no part. In this case B has *forged* a signature of A. This is an example of what is called *existential forgery*. (B has produced A's signature on some message likely not of B's choosing.)

If $\mathcal{M}'$ contains only a negligible fraction of messages from $\mathcal{M}$, then the probability of some entity forging a signature of A in this manner is negligibly small.

1.52 Remark (*digital signatures vs. confidentiality*) Although digital signature schemes based on reversible public-key encryption are attractive, they require an encryption method as a primitive. There are situations where a digital signature mechanism is required but encryption is forbidden. In such cases these digital signature schemes are inappropriate.

Digital signatures in practice

For digital signatures to be useful in practice, concrete realizations of the preceding concepts should have certain additional properties. A digital signature must

1. be easy to compute by the signer (the signing function should be easy to apply);
2. be easy to verify by anyone (the verification function should be easy to apply); and
3. have an appropriate lifespan, i.e., be computationally secure from forgery until the signature is no longer necessary for its original purpose.

Resolution of disputes

The purpose of a digital signature (or any signature method) is to permit the resolution of disputes. For example, an entity A could at some point deny having signed a message or some other entity B could falsely claim that a signature on a message was produced by A. In order to overcome such problems a *trusted third party* (TTP) or *judge* is required. The TTP must be some entity which all parties involved agree upon in advance.

If A denies that a message m held by B was signed by A, then B should be able to present the signature s_A for m to the TTP along with m. The TTP rules in favor of B if $V_A(m, s_A) =$ *true* and in favor of A otherwise. B will accept the decision if B is confident that the TTP has the same verifying transformation V_A as A does. A will accept the decision if A is confident that the TTP used V_A and that S_A has not been compromised. Therefore, fair resolution of disputes requires that the following criteria are met.

Requirements for resolution of disputed signatures

1. S_A and V_A have properties (a) and (b) of page 23.
2. The TTP has an authentic copy of V_A.
3. The signing transformation S_A has been kept secret and remains secure.

These properties are necessary but in practice it might not be possible to guarantee them. For example, the assumption that S_A and V_A have the desired characteristics given in property 1 might turn out to be false for a particular signature scheme. Another possibility is that A claims falsely that S_A was compromised. To overcome these problems requires an agreed method to validate the time period for which A will accept responsibility for the verification transformation. An analogue of this situation can be made with credit card revocation. The holder of a card is responsible until the holder notifies the card issuing company that the card has been lost or stolen. §13.8.2 gives a more indepth discussion of these problems and possible solutions.

1.8.4 Symmetric-key vs. public-key cryptography

Symmetric-key and public-key encryption schemes have various advantages and disadvantages, some of which are common to both. This section highlights a number of these and summarizes features pointed out in previous sections.

(i) Advantages of symmetric-key cryptography

1. Symmetric-key ciphers can be designed to have high rates of data throughput. Some hardware implementations achieve encrypt rates of hundreds of megabytes per second, while software implementations may attain throughput rates in the megabytes per second range.
2. Keys for symmetric-key ciphers are relatively short.
3. Symmetric-key ciphers can be employed as primitives to construct various cryptographic mechanisms including pseudorandom number generators (see Chapter 5), hash functions (see Chapter 9), and computationally efficient digital signature schemes (see Chapter 11), to name just a few.
4. Symmetric-key ciphers can be composed to produce stronger ciphers. Simple transformations which are easy to analyze, but on their own weak, can be used to construct strong product ciphers.
5. Symmetric-key encryption is perceived to have an extensive history, although it must be acknowledged that, notwithstanding the invention of rotor machines earlier, much of the knowledge in this area has been acquired subsequent to the invention of the digital computer, and, in particular, the design of the Data Encryption Standard (see Chapter 7) in the early 1970s.

(ii) Disadvantages of symmetric-key cryptography

1. In a two-party communication, the key must remain secret at both ends.
2. In a large network, there are many key pairs to be managed. Consequently, effective key management requires the use of an unconditionally trusted TTP (Definition 1.65).
3. In a two-party communication between entities A and B, sound cryptographic practice dictates that the key be changed frequently, and perhaps for each communication session.
4. Digital signature mechanisms arising from symmetric-key encryption typically require either large keys for the public verification function or the use of a TTP (see Chapter 11).

(iii) Advantages of public-key cryptography

1. Only the private key must be kept secret (authenticity of public keys must, however, be guaranteed).
2. The administration of keys on a network requires the presence of only a functionally trusted TTP (Definition 1.66) as opposed to an unconditionally trusted TTP. Depending on the mode of usage, the TTP might only be required in an "off-line" manner, as opposed to in real time.
3. Depending on the mode of usage, a private key/public key pair may remain unchanged for considerable periods of time, e.g., many sessions (even several years).
4. Many public-key schemes yield relatively efficient digital signature mechanisms. The key used to describe the public verification function is typically much smaller than for the symmetric-key counterpart.

5. In a large network, the number of keys necessary may be considerably smaller than in the symmetric-key scenario.

(iv) Disadvantages of public-key encryption

1. Throughput rates for the most popular public-key encryption methods are several orders of magnitude slower than the best known symmetric-key schemes.
2. Key sizes are typically much larger than those required for symmetric-key encryption (see Remark 1.53), and the size of public-key signatures is larger than that of tags providing data origin authentication from symmetric-key techniques.
3. No public-key scheme has been proven to be secure (the same can be said for block ciphers). The most effective public-key encryption schemes found to date have their security based on the presumed difficulty of a small set of number-theoretic problems.
4. Public-key cryptography does not have as extensive a history as symmetric-key encryption, being discovered only in the mid 1970s.[6]

Summary of comparison

Symmetric-key and public-key encryption have a number of complementary advantages. Current cryptographic systems exploit the strengths of each. An example will serve to illustrate.

Public-key encryption techniques may be used to establish a key for a symmetric-key system being used by communicating entities A and B. In this scenario A and B can take advantage of the long term nature of the public/private keys of the public-key scheme and the performance efficiencies of the symmetric-key scheme. Since data encryption is frequently the most time consuming part of the encryption process, the public-key scheme for key establishment is a small fraction of the total encryption process between A and B.

To date, the computational performance of public-key encryption is inferior to that of symmetric-key encryption. There is, however, no proof that this must be the case. The important points in practice are:

1. public-key cryptography facilitates efficient signatures (particularly non-repudiation) and key mangement; and
2. symmetric-key cryptography is efficient for encryption and some data integrity applications.

1.53 Remark (*key sizes: symmetric key vs. private key*) Private keys in public-key systems must be larger (e.g., 1024 bits for RSA) than secret keys in symmetric-key systems (e.g., 64 or 128 bits) because whereas (for secure algorithms) the most efficient attack on symmetric-key systems is an exhaustive key search, all known public-key systems are subject to "short-cut" attacks (e.g., factoring) more efficient than exhaustive search. Consequently, for equivalent security, symmetric keys have bitlengths considerably smaller than that of private keys in public-key systems, e.g., by a factor of 10 or more.

[6]It is, of course, arguable that some public-key schemes which are based on hard mathematical problems have a long history since these problems have been studied for many years. Although this may be true, one must be wary that the mathematics was not studied with this application in mind.

1.9 Hash functions

One of the fundamental primitives in modern cryptography is the cryptographic hash function, often informally called a one-way hash function. A simplified definition for the present discussion follows.

1.54 Definition A *hash function* is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called *hash-values*.

For a hash function which outputs n-bit hash-values (e.g., $n = 128$ or 160) and has desirable properties, the probability that a randomly chosen string gets mapped to a particular n-bit hash-value (image) is 2^{-n}. The basic idea is that a hash-value serves as a compact representative of an input string. To be of cryptographic use, a hash function h is typically chosen such that it is computationally infeasible to find two distinct inputs which hash to a common value (i.e., two *colliding* inputs x and y such that $h(x) = h(y)$), and that given a specific hash-value y, it is computationally infeasible to find an input (pre-image) x such that $h(x) = y$.

The most common cryptographic uses of hash functions are with digital signatures and for data integrity. With digital signatures, a long message is usually hashed (using a publicly available hash function) and only the hash-value is signed. The party receiving the message then hashes the received message, and verifies that the received signature is correct for this hash-value. This saves both time and space compared to signing the message directly, which would typically involve splitting the message into appropriate-sized blocks and signing each block individually. Note here that the inability to find two messages with the same hash-value is a security requirement, since otherwise, the signature on one message hash-value would be the same as that on another, allowing a signer to sign one message and at a later point in time claim to have signed another.

Hash functions may be used for data integrity as follows. The hash-value corresponding to a particular input is computed at some point in time. The integrity of this hash-value is protected in some manner. At a subsequent point in time, to verify that the input data has not been altered, the hash-value is recomputed using the input at hand, and compared for equality with the original hash-value. Specific applications include virus protection and software distribution.

A third application of hash functions is their use in protocols involving a priori commitments, including some digital signature schemes and identification protocols (e.g., see Chapter 10).

Hash functions as discussed above are typically publicly known and involve no secret keys. When used to detect whether the message input has been altered, they are called *modification detection codes* (MDCs). Related to these are hash functions which involve a secret key, and provide data origin authentication (§9.76) as well as data integrity; these are called *message authentication codes* (MACs).

1.10 Protocols and mechanisms

1.55 Definition A *cryptographic protocol* (*protocol*) is a distributed algorithm defined by a sequence of steps precisely specifying the actions required of two or more entities to achieve a specific security objective.

1.56 Remark (*protocol vs. mechanism*) As opposed to a protocol, a *mechanism* is a more general term encompassing protocols, algorithms (specifying the steps followed by a single entity), and non-cryptographic techniques (e.g., hardware protection and procedural controls) to achieve specific security objectives.

Protocols play a major role in cryptography and are essential in meeting cryptographic goals as discussed in §1.2. Encryption schemes, digital signatures, hash functions, and random number generation are among the primitives which may be utilized to build a protocol.

1.57 Example (*a simple key agreement protocol*) Alice and Bob have chosen a symmetric-key encryption scheme to use in communicating over an unsecured channel. To encrypt information they require a key. The communication protocol is the following:

1. Bob constructs a public-key encryption scheme and sends his public key to Alice over the channel.
2. Alice generates a key for the symmetric-key encryption scheme.
3. Alice encrypts the key using Bob's public key and sends the encrypted key to Bob.
4. Bob decrypts using his private key and recovers the symmetric (secret) key.
5. Alice and Bob begin communicating with privacy by using the symmetric-key system and the common secret key.

This protocol uses basic functions to attempt to realize private communications on an unsecured channel. The basic primitives are the symmetric-key and the public-key encryption schemes. The protocol has shortcomings including the impersonation attack of §1.8.2, but it does convey the idea of a protocol. □

Often the role of public-key encryption in privacy communications is exactly the one suggested by this protocol – public-key encryption is used as a means to exchange keys for subsequent use in symmetric-key encryption, motivated by performance differences between symmetric-key and public-key encryption.

Protocol and mechanism failure

1.58 Definition A *protocol failure* or *mechanism failure* occurs when a mechanism fails to meet the goals for which it was intended, in a manner whereby an adversary gains advantage not by breaking an underlying primitive such as an encryption algorithm directly, but by manipulating the protocol or mechanism itself.

1.59 Example (*mechanism failure*) Alice and Bob are communicating using a stream cipher. Messages which they encrypt are known to have a special form: the first twenty bits carry information which represents a monetary amount. An active adversary can simply XOR an appropriate bitstring into the first twenty bits of ciphertext and change the amount. While the adversary has not been able to read the underlying message, she has been able to alter the transmission. The encryption has not been compromised but the protocol has failed to perform adequately; the inherent assumption that encryption provides data integrity is incorrect. □

1.60 Example (*forward search attack*) Suppose that in an electronic bank transaction the 32-bit field which records the value of the transaction is to be encrypted using a public-key scheme. This simple protocol is intended to provide privacy of the value field – but does it? An adversary could easily take all 2^{32} possible entries that could be plaintext in this field and encrypt them using the public encryption function. (Remember that by the very nature of public-key encryption this function must be available to the adversary.) By comparing each of the 2^{32} ciphertexts with the one which is actually encrypted in the transaction, the adversary can determine the plaintext. Here the public-key encryption function is not compromised, but rather the way it is used. A closely related attack which applies directly to authentication for access control purposes is the dictionary attack (see §10.2.2). □

1.61 Remark (*causes of protocol failure*) Protocols and mechanisms may fail for a number of reasons, including:

1. weaknesses in a particular cryptographic primitive which may be amplified by the protocol or mechanism;
2. claimed or assumed security guarantees which are overstated or not clearly understood; and
3. the oversight of some principle applicable to a broad class of primitives such as encryption.

Example 1.59 illustrates item 2 if the stream cipher is the one-time pad, and also item 1. Example 1.60 illustrates item 3. See also §1.8.2.

1.62 Remark (*protocol design*) When designing cryptographic protocols and mechanisms, the following two steps are essential:

1. identify *all* assumptions in the protocol or mechanism design; and
2. for each assumption, determine the effect on the security objective if that assumption is violated.

1.11 Key establishment, management, and certification

This section gives a brief introduction to methodology for ensuring the secure distribution of keys for cryptographic purposes.

1.63 Definition *Key establishment* is any process whereby a shared secret key becomes available to two or more parties, for subsequent cryptographic use.

1.64 Definition *Key management* is the set of processes and mechanisms which support key establishment and the maintenance of ongoing keying relationships between parties, including replacing older keys with new keys as necessary.

Key establishment can be broadly subdivided into *key agreement* and *key transport*. Many and various protocols have been proposed to provide key establishment. Chapter 12 describes a number of these in detail. For the purpose of this chapter only a brief overview of issues related to key management will be given. Simple architectures based on symmetric-key and public-key cryptography along with the concept of certification will be addressed.

As noted in §1.5, a major issue when using symmetric-key techniques is the establishment of pairwise secret keys. This becomes more evident when considering a network of entities, any two of which may wish to communicate. Figure 1.15 illustrates a network consisting of 6 entities. The arrowed edges indicate the 15 possible two-party communications which could take place. Since each pair of entities wish to communicate, this small network requires the secure exchange of $\binom{6}{2} = 15$ key pairs. In a network with n entities, the number of secure key exchanges required is $\binom{n}{2} = \frac{n(n-1)}{2}$.

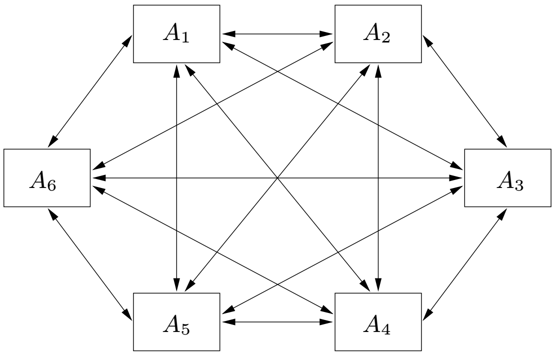

Figure 1.15: *Keying relationships in a simple 6-party network.*

The network diagram depicted in Figure 1.15 is simply the amalgamation of 15 two-party communications as depicted in Figure 1.7. In practice, networks are very large and the key management problem is a crucial issue. There are a number of ways to handle this problem. Two simplistic methods are discussed; one based on symmetric-key and the other on public-key techniques.

1.11.1 Key management through symmetric-key techniques

One solution which employs symmetric-key techniques involves an entity in the network which is trusted by all other entities. As in §1.8.3, this entity is referred to as a *trusted third party* (TTP). Each entity A_i shares a distinct symmetric key k_i with the TTP. These keys are assumed to have been distributed over a secured channel. If two entities subsequently wish to communicate, the TTP generates a key k (sometimes called a *session key*) and sends it encrypted under each of the fixed keys as depicted in Figure 1.16 for entities A_1 and A_5.

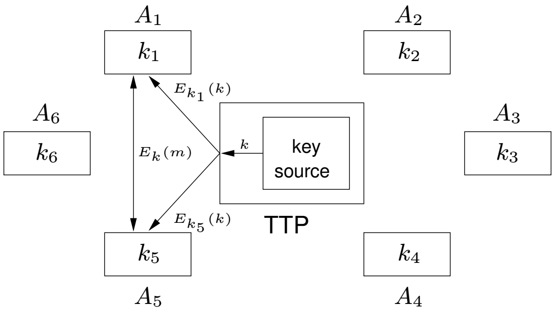

Figure 1.16: *Key management using a trusted third party (TTP).*

Advantages of this approach include:

1. It is easy to add and remove entities from the network.
2. Each entity needs to store only one long-term secret key.

Disadvantages include:

1. All communications require initial interaction with the TTP.
2. The TTP must store n long-term secret keys.

3. The TTP has the ability to read all messages.
4. If the TTP is compromised, all communications are insecure.

1.11.2 Key management through public-key techniques

There are a number of ways to address the key management problem through public-key techniques. Chapter 13 describes many of these in detail. For the purpose of this chapter a very simple model is considered.

Each entity in the network has a public/private encryption key pair. The public key along with the identity of the entity is stored in a central repository called a *public file*. If an entity A_1 wishes to send encrypted messages to entity A_6, A_1 retrieves the public key e_6 of A_6 from the public file, encrypts the message using this key, and sends the ciphertext to A_6. Figure 1.17 depicts such a network.

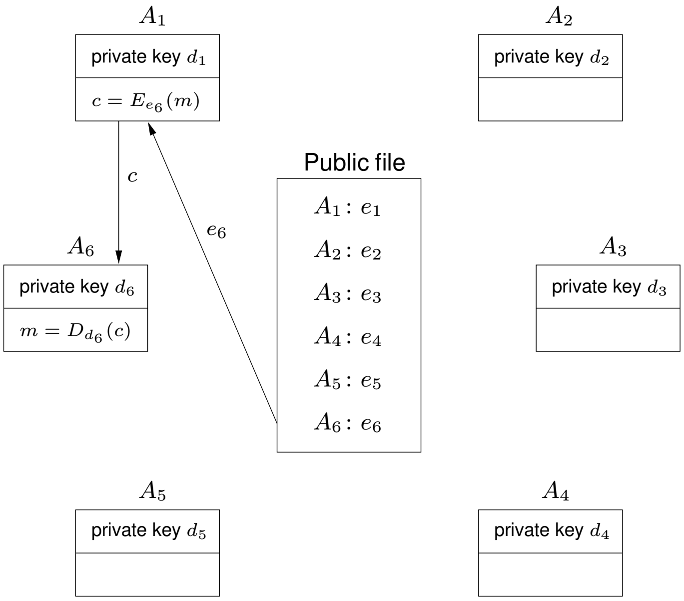

Figure 1.17: *Key management using public-key techniques.*

Advantages of this approach include:

1. No trusted third party is required.
2. The public file could reside with each entity.
3. Only n public keys need to be stored to allow secure communications between any pair of entities, assuming the only attack is that by a passive adversary.

The key management problem becomes more difficult when one must take into account an adversary who is *active* (i.e. an adversary who can alter the public file containing public keys). Figure 1.18 illustrates how an active adversary could compromise the key management scheme given above. (This is directly analogous to the attack in §1.8.2.) In the figure, the adversary alters the public file by replacing the public key e_6 of entity A_6 by the adversary's public key e^*. Any message encrypted for A_6 using the public key from the public file can be decrypted by only the adversary. Having decrypted and read the message, the adversary can now encrypt it using the public key of A_6 and forward the ciphertext to A_6. A_1 however believes that only A_6 can decrypt the ciphertext c.

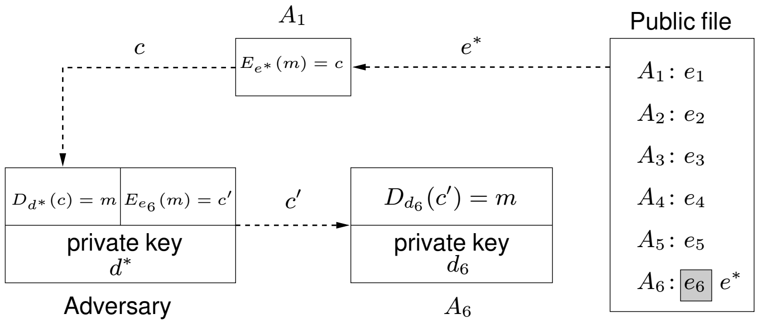

Figure 1.18: *An impersonation of A_6 by an active adversary with public key e^*.*

To prevent this type of attack, the entities may use a TTP to *certify* the public key of each entity. The TTP has a private signing algorithm S_T and a verification algorithm V_T (see §1.6) assumed to be known by all entities. The TTP carefully verifies the identity of each entity, and signs a message consisting of an identifier and the entity's authentic public key. This is a simple example of a *certificate*, binding the identity of an entity to its public key (see §1.11.3). Figure 1.19 illustrates the network under these conditions. A_1 uses the public key of A_6 only if the certificate signature verifies successfully.

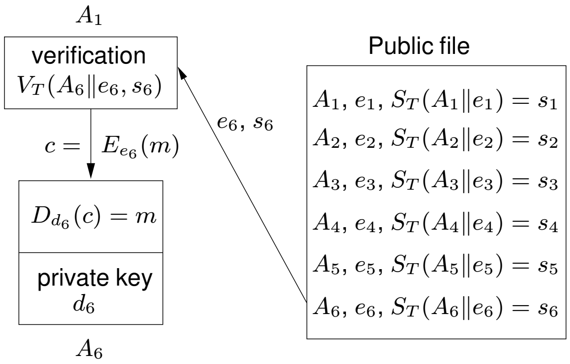

Figure 1.19: *Authentication of public keys by a TTP. $\|$ denotes concatenation.*

Advantages of using a TTP to maintain the integrity of the public file include:

1. It prevents an active adversary from impersonation on the network.
2. The TTP cannot monitor communications. Entities need trust the TTP only to bind identities to public keys properly.
3. Per-communication interaction with the public file can be eliminated if entities store certificates locally.

Even with a TTP, some concerns still remain:

1. If the signing key of the TTP is compromised, all communications become insecure.
2. All trust is placed with one entity.

1.11.3 Trusted third parties and public-key certificates

A trusted third party has been used in §1.8.3 and again here in §1.11. The trust placed on this entity varies with the way it is used, and hence motivates the following classification.

1.65 Definition A TTP is said to be *unconditionally trusted* if it is trusted on all matters. For example, it may have access to the secret and private keys of users, as well as be charged with the association of public keys to identifiers.

1.66 Definition A TTP is said to be *functionally trusted* if the entity is assumed to be honest and fair but it does not have access to the secret or private keys of users.

§1.11.1 provides a scenario which employs an unconditionally trusted TTP. §1.11.2 uses a functionally trusted TTP to maintain the integrity of the public file. A functionally trusted TTP could be used to register or certify users and contents of documents or, as in §1.8.3, as a judge.

Public-key certificates

The distribution of public keys is generally easier than that of symmetric keys, since secrecy is not required. However, the integrity (authenticity) of public keys is critical (recall §1.8.2).

A *public-key certificate* consists of a *data part* and a *signature part*. The data part consists of the name of an entity, the public key corresponding to that entity, possibly additional relevant information (e.g., the entity's street or network address, a validity period for the public key, and various other attributes). The signature part consists of the signature of a TTP over the data part.

In order for an entity B to verify the authenticity of the public key of an entity A, B must have an authentic copy of the public signature verification function of the TTP. For simplicity, assume that the authenticity of this verification function is provided to B by non-cryptographic means, for example by B obtaining it from the TTP in person. B can then carry out the following steps:

1. Acquire the public-key certificate of A over some unsecured channel, either from a central database of certificates, from A directly, or otherwise.
2. Use the TTP's verification function to verify the TTP's signature on A's certificate.
3. If this signature verifies correctly, accept the public key in the certificate as A's authentic public key; otherwise, assume the public key is invalid.

Before creating a public-key certificate for A, the TTP must take appropriate measures to verify the identity of A and the fact that the public key to be certificated actually belongs to A. One method is to require that A appear before the TTP with a conventional passport as proof of identity, and obtain A's public key from A in person along with evidence that A knows the corresponding private key. Once the TTP creates a certificate for a party, the trust that all other entities have in the authenticity of the TTP's public key can be used transitively to gain trust in the authenticity of that party's public key, through acquisition and verification of the certificate.

1.12 Pseudorandom numbers and sequences

Random number generation is an important primitive in many cryptographic mechanisms. For example, keys for encryption transformations need to be generated in a manner which is unpredictable to an adversary. Generating a random key typically involves the selection of random numbers or bit sequences. Random number generation presents challenging issues. A brief introduction is given here with details left to Chapter 5.

Often in cryptographic applications, one of the following steps must be performed:

(i) From a finite set of n elements (e.g., $\{1, 2, \ldots, n\}$), select an element at random.
(ii) From the set of all sequences (strings) of length m over some finite alphabet $\mathcal{A}$ of n symbols, select a sequence at random.
(iii) Generate a random sequence (string) of symbols of length m over a set of n symbols.

It is not clear what exactly it means to *select at random* or *generate at random*. Calling a number random without a context makes little sense. Is the number 23 a random number? No, but if 49 identical balls labeled with a number from 1 to 49 are in a container, and this container mixes the balls uniformly, drops one ball out, and this ball happens to be labeled with the number 23, then one would say that 23 was generated randomly from a uniform distribution. The *probability* that 23 drops out is 1 in 49 or $\frac{1}{49}$.

If the number on the ball which was dropped from the container is recorded and the ball is placed back in the container and the process repeated 6 times, then a random sequence of length 6 defined on the alphabet $\mathcal{A} = \{1, 2, \ldots, 49\}$ will have been generated. What is the chance that the sequence $17, 45, 1, 7, 23, 35$ occurs? Since each element in the sequence has probability $\frac{1}{49}$ of occuring, the probability of the sequence $17, 45, 1, 7, 23, 35$ occurring is

$$\frac{1}{49} \times \frac{1}{49} \times \frac{1}{49} \times \frac{1}{49} \times \frac{1}{49} \times \frac{1}{49} = \frac{1}{13841287201}.$$

There are precisely 13841287201 sequences of length 6 over the alphabet $\mathcal{A}$. If each of these sequences is written on one of 13841287201 balls and they are placed in the container (first removing the original 49 balls) then the chance that the sequence given above drops out is the same as if it were generated one ball at a time. Hence, (ii) and (iii) above are essentially the same statements.

Finding good methods to generate random sequences is difficult.

1.67 Example (*random sequence generator*) To generate a random sequence of 0's and 1's, a coin could be tossed with a head landing up recorded as a 1 and a tail as a 0. It is assumed that the coin is *unbiased*, which means that the probability of a 1 on a given toss is exactly $\frac{1}{2}$. This will depend on how well the coin is made and how the toss is performed. This method would be of little value in a system where random sequences must be generated quickly and often. It has no practical value other than to serve as an example of the idea of random number generation. □

1.68 Example (*random sequence generator*) A *noise diode* may be used to produce random binary sequences. This is reasonable if one has some way to be convinced that the probability that a 1 will be produced on any given trial is $\frac{1}{2}$. Should this assumption be false, the sequence generated would not have been selected from a uniform distribution and so not all sequences of a given length would be equally likely. The only way to get some feeling for the reliability of this type of random source is to carry out statistical tests on its output. These are considered in Chapter 5. If the diode is a source of a uniform distribution on the set of all binary sequences of a given length, it provides an effective way to generate random sequences. □

Since most *true sources* of random sequences (if there is such a thing) come from *physical means*, they tend to be either costly or slow in their generation. To overcome these problems, methods have been devised to construct *pseudorandom sequences* in a deterministic manner from a shorter random sequence called a *seed*. The pseudorandom sequences appear to be generated by a truly random source to anyone not knowing the method of generation. Often the generation algorithm is known to all, but the seed is unknown except by the entity generating the sequence. A plethora of algorithms has been developed to generate pseudorandom bit sequences of various types. Many of these are completely unsuitable for cryptographic purposes and one must be cautious of claims by creators of such algorithms as to the random nature of the output.

1.13 Classes of attacks and security models

Over the years, many different types of attacks on cryptographic primitives and protocols have been identified. The discussion here limits consideration to attacks on encryption and protocols. Attacks on other cryptographic primitives will be given in appropriate chapters.

In §1.11 the roles of an active and a passive adversary were discussed. The attacks these adversaries can mount may be classified as follows:.

1. A *passive attack* is one where the adversary only monitors the communication channel. A passive attacker only threatens confidentiality of data.
2. An *active attack* is one where the adversary attempts to delete, add, or in some other way alter the transmission on the channel. An active attacker threatens data integrity and authentication as well as confidentiality.

A passive attack can be further subdivided into more specialized attacks for deducing plaintext from ciphertext, as outlined in §1.13.1.

1.13.1 Attacks on encryption schemes

The objective of the following attacks is to systematically recover plaintext from ciphertext, or even more drastically, to deduce the decryption key.

1. A *ciphertext-only attack* is one where the adversary (or cryptanalyst) tries to deduce the decryption key or plaintext by only observing ciphertext. Any encryption scheme vulnerable to this type of attack is considered to be completely insecure.
2. A *known-plaintext attack* is one where the adversary has a quantity of plaintext and corresponding ciphertext. This type of attack is typically only marginally more difficult to mount.
3. A *chosen-plaintext attack* is one where the adversary chooses plaintext and is then given corresponding ciphertext. Subsequently, the adversary uses any information deduced in order to recover plaintext corresponding to previously unseen ciphertext.
4. An *adaptive chosen-plaintext attack* is a chosen-plaintext attack wherein the choice of plaintext may depend on the ciphertext received from previous requests.
5. A *chosen-ciphertext attack* is one where the adversary selects the ciphertext and is then given the corresponding plaintext. One way to mount such an attack is for the adversary to gain access to the equipment used for decryption (but not the decryption key, which may be securely embedded in the equipment). The objective is then to be able, without access to such equipment, to deduce the plaintext from (different) ciphertext.

6. An *adaptive chosen-ciphertext attack* is a chosen-ciphertext attack where the choice of ciphertext may depend on the plaintext received from previous requests.

Most of these attacks also apply to digital signature schemes and message authentication codes. In this case, the objective of the attacker is to forge messages or MACs, as discussed in Chapters 11 and 9, respectively.

1.13.2 Attacks on protocols

The following is a partial list of attacks which might be mounted on various protocols. Until a protocol is proven to provide the service intended, the list of possible attacks can never be said to be complete.

1. *known-key attack*. In this attack an adversary obtains some keys used previously and then uses this information to determine new keys.
2. *replay*. In this attack an adversary records a communication session and replays the entire session, or a portion thereof, at some later point in time.
3. *impersonation*. Here an adversary assumes the identity of one of the legitimate parties in a network.
4. *dictionary*. This is usually an attack against passwords. Typically, a password is stored in a computer file as the image of an unkeyed hash function. When a user logs on and enters a password, it is hashed and the image is compared to the stored value. An adversary can take a list of probable passwords, hash all entries in this list, and then compare this to the list of true encrypted passwords with the hope of finding matches.
5. *forward search*. This attack is similar in spirit to the dictionary attack and is used to decrypt messages. An example of this method was cited in Example 1.60.
6. *interleaving attack*. This type of attack usually involves some form of impersonation in an authentication protocol (see §12.9.1).

1.13.3 Models for evaluating security

The security of cryptographic primitives and protocols can be evaluated under several different models. The most practical security metrics are computational, provable, and ad hoc methodology, although the latter is often dangerous. The confidence level in the amount of security provided by a primitive or protocol based on computational or ad hoc security increases with time and investigation of the scheme. However, time is not enough if few people have given the method careful analysis.

(i) Unconditional security

The most stringent measure is an information-theoretic measure – whether or not a system has *unconditional security*. An adversary is assumed to have unlimited computational resources, and the question is whether or not there is enough information available to defeat the system. Unconditional security for encryption systems is called *perfect secrecy*. For perfect secrecy, the uncertainty in the plaintext, after observing the ciphertext, must be equal to the a priori uncertainty about the plaintext – observation of the ciphertext provides no information whatsoever to an adversary.

A necessary condition for a symmetric-key encryption scheme to be unconditionally secure is that the key be at least as long as the message. The one-time pad (§1.5.4) is an example of an unconditionally secure encryption algorithm. In general, encryption schemes do not offer perfect secrecy, and each ciphertext character observed decreases the theoretical uncertainty in the plaintext and the encryption key. Public-key encryption schemes cannot be unconditionally secure since, given a ciphertext c, the plaintext can in principle be recovered by encrypting all possible plaintexts until c is obtained.

(ii) Complexity-theoretic security

An appropriate model of computation is defined and adversaries are modeled as having polynomial computational power. (They mount attacks involving time and space polynomial in the size of appropriate security parameters.) A proof of security relative to the model is then constructed. An objective is to design a cryptographic method based on the weakest assumptions possible anticipating a powerful adversary. Asymptotic analysis and usually also worst-case analysis is used and so care must be exercised to determine when proofs have practical significance. In contrast, polynomial attacks which are feasible under the model might, in practice, still be computationally infeasible.

Security analysis of this type, although not of practical value in all cases, may nonetheless pave the way to a better overall understanding of security. Complexity-theoretic analysis is invaluable for formulating fundamental principles and confirming intuition. This is like many other sciences, whose practical techniques are discovered early in the development, well before a theoretical basis and understanding is attained.

(iii) Provable security

A cryptographic method is said to be *provably secure* if the difficulty of defeating it can be shown to be essentially as difficult as solving a well-known and *supposedly* difficult (typically number-theoretic) problem, such as integer factorization or the computation of discrete logarithms. Thus, "provable" here means provable subject to assumptions.

This approach is considered by some to be as good a practical analysis technique as exists. Provable security may be considered part of a special sub-class of the larger class of computational security considered next.

(iv) Computational security

This measures the amount of computational effort required, by the best currently-known methods, to defeat a system; it must be assumed here that the system has been well-studied to determine which attacks are relevant. A proposed technique is said to be *computationally secure* if the perceived level of computation required to defeat it (using the best attack known) exceeds, by a comfortable margin, the computational resources of the hypothesized adversary.

Often methods in this class are related to hard problems but, unlike for provable security, no proof of equivalence is known. Most of the best known public-key and symmetric-key schemes in current use are in this class. This class is sometimes also called *practical security*.

(v) Ad hoc security

This approach consists of any variety of convincing arguments that every successful attack requires a resource level (e.g., time and space) greater than the fixed resources of a perceived adversary. Cryptographic primitives and protocols which survive such analysis are said to have *heuristic security*, with security here typically in the computational sense.

Primitives and protocols are usually designed to counter standard attacks such as those given in §1.13. While perhaps the most commonly used approach (especially for protocols), it is, in some ways, the least satisfying. Claims of security generally remain questionable and unforeseen attacks remain a threat.

1.13.4 Perspective for computational security

To evaluate the security of cryptographic schemes, certain quantities are often considered.

1.69 Definition The *work factor* W_d is the minimum amount of work (measured in appropriate units such as elementary operations or clock cycles) required to compute the private key d given the public key e, or, in the case of symmetric-key schemes, to determine the secret key k. More specifically, one may consider the work required under a ciphertext-only attack given n ciphertexts, denoted $W_d(n)$.

If W_d is t years, then for sufficiently large t the cryptographic scheme is, for all practical purposes, a secure system. To date no public-key system has been found where one can prove a sufficiently large lower bound on the work factor W_d. The best that is possible to date is to rely on the following as a basis for security.

1.70 Definition The *historical work factor* $\overline{W_d}$ is the minimum amount of work required to compute the private key d from the public key e using the best known algorithms at a given point in time.

The historical work factor $\overline{W_d}$ varies with time as algorithms and technology improve. It corresponds to computational security, whereas W_d corresponds to the true security level, although this typically cannot be determined.

How large is large?

§1.4 described how the designer of an encryption system tries to create a scheme for which the best approach to breaking it is through exhaustive search of the key space. The key space must then be large enough to make an exhaustive search completely infeasible. An important question then is "How large is large?". In order to gain some perspective on the magnitude of numbers, Table 1.2 lists various items along with an associated magnitude.

Reference	Magnitude
Seconds in a year	$\approx 3 \times 10^7$
Age of our solar system (years)	$\approx 6 \times 10^9$
Seconds since creation of solar system	$\approx 2 \times 10^{17}$
Clock cycles per year, 50 MHz computer	$\approx 1.6 \times 10^{15}$
Binary strings of length 64	$2^{64} \approx 1.8 \times 10^{19}$
Binary strings of length 128	$2^{128} \approx 3.4 \times 10^{38}$
Binary strings of length 256	$2^{256} \approx 1.2 \times 10^{77}$
Number of 75-digit prime numbers	$\approx 5.2 \times 10^{72}$
Electrons in the universe	$\approx 8.37 \times 10^{77}$

Table 1.2: *Reference numbers comparing relative magnitudes.*

Some powers of 10 are referred to by prefixes. For example, high-speed modern computers are now being rated in terms of *teraflops* where a teraflop is 10^{12} floating point operations per second. Table 1.3 provides a list of commonly used prefixes.

Prefix	Symbol	Magnitude
exa	E	10^{18}
peta	P	10^{15}
tera	T	10^{12}
giga	G	10^{9}
mega	M	10^{6}
kilo	k	10^{3}
hecto	h	10^{2}
deca	da	10

Prefix	Symbol	Magnitude
deci	d	10^{-1}
centi	c	10^{-2}
milli	m	10^{-3}
micro	μ	10^{-6}
nano	n	10^{-9}
pico	p	10^{-12}
femto	f	10^{-15}
atto	a	10^{-18}

Table 1.3: *Prefixes used for various powers of 10.*

1.14 Notes and further references

§1.1

Kahn [648] gives a thorough, comprehensive, and non-technical history of cryptography, published in 1967. Feistel [387] provides an early exposition of block cipher ideas. The original specification of DES is the 1977 U.S. Federal Information Processing Standards Publication 46 [396]. Public-key cryptography was introduced by Diffie and Hellman [345]. The first concrete realization of a public-key encryption scheme was the knapsack scheme by Merkle and Hellman [857]. The RSA public-key encryption and signature scheme is due to Rivest, Shamir, and Adleman [1060], while the ElGamal public-key encryption and signature schemes are due to ElGamal [368]. The two digital signature standards, ISO/IEC 9796 [596] and the Digital Signature Standard [406], are discussed extensively in Chapter 11.

Cryptography has used specialized areas of mathematics such as number theory to realize very practical mechanisms such as public-key encryption and digital signatures. Such usage was not conceived as possible a mere twenty years ago. The famous mathematician, Hardy [539], went as far as to boast about its lack of utility:

> " . . . both Gauss and lesser mathematicians may be justified in rejoicing that there is one science at any rate, and that their own, whose very remoteness from ordinary human activities should keep it gentle and clean."

§1.2

This section was inspired by the foreword to the book *Contemporary Cryptology, The Science of Information Integrity*, edited by Simmons [1143]. The handwritten signature came into the British legal system in the seventeenth century as a means to provide various functions associated with information security. See Chapter 9 of Meyer and Matyas [859] for details.

This book only considers cryptography as it applies to information in digital form. Chapter 9 of Beker and Piper [84] provides an introduction to the encryption of analogue signals, in particular, speech. Although in many cases physical means are employed to facilitate privacy, cryptography plays the major role. Physical means of providing privacy include fiber optic communication links, spread spectrum technology, TEMPEST techniques, and tamper-resistant hardware. *Steganography* is that branch of information privacy which attempts to obscure the existence of data through such devices as invisible inks, secret compartments, the use of subliminal channels, and the like. Kahn [648] provides an historical account of various steganographic techniques.

Excellent introductions to cryptography can be found in the articles by Diffie and Hellman [347], Massey [786], and Rivest [1054]. A concise and elegant way to describe cryptography was given by Rivest [1054]: *Cryptography is about communication in the presence of adversaries*. The taxonomy of cryptographic primitives (Figure 1.1) was derived from the classification given by Bosselaers, Govaerts, and Vandewalle [175].

§1.3

The theory of functions is fundamental in modern mathematics. The term *range* is often used in place of image of a function. The latter, being more descriptive, is preferred. An alternate term for one-to-one is *injective*; an alternate term for onto is *surjective*.

One-way functions were introduced by Diffie and Hellman [345]. A more extensive history is given on page 377. Trapdoor one-way functions were first postulated by Diffie and Hellman [345] and independently by Merkle [850] as a means to obtain public-key encryption schemes; several candidates are given in Chapter 8.

§1.4

The basic concepts of cryptography are treated quite differently by various authors, some being more technical than others. Brassard [192] provides a concise, lucid, and technically accurate account. Schneier [1094] gives a less technical but very accessible introduction. Salomaa [1089], Stinson [1178], and Rivest [1054] present more mathematical approaches. Davies and Price [308] provide a very readable presentation suitable for the practitioner.

The comparison of an encryption scheme to a resettable combination lock is from Diffie and Hellman [347]. Kerckhoffs' desiderata [668] were originally stated in French. The translation stated here is given in Kahn [648]. Shannon [1121] also gives desiderata for encryption schemes.

§1.5

Symmetric-key encryption has a very long history, as recorded by Kahn [648]. Most systems invented prior to the 1970s are now of historical interest only. Chapter 2 of Denning [326] is also a good source for many of the more well known schemes such as the Caesar cipher, Vigenère and Beaufort ciphers, rotor machines (Enigma and Hagelin), running key ciphers, and so on; see also Davies and Price [308] and Konheim [705]. Beker and Piper [84] give an indepth treatment, including cryptanalysis of several of the classical systems used in World War II. Shannon's paper [1121] is considered the seminal work on secure communications. It is also an excellent source for descriptions of various well-known historical symmetric-key ciphers.

Simple substitution and transposition ciphers are the focus of §1.5. Hill ciphers [557], a class of substitution ciphers which substitute blocks using matrix methods, are covered in Example 7.52. The idea of confusion and diffusion (Remark 1.36) was introduced by Shannon [1121].

Kahn [648] gives 1917 as the date when Vernam discovered the cipher which bears Vernam's name, however, Vernam did not publish the result until 1926 [1222]; see page 274 for further discussion. Massey [786] states that reliable sources have suggested that the Moscow-Washington hot-line (channel for very high level communications) is no longer secured with a one-time pad, which has been replaced by a symmetric-key cipher requiring a much shorter key. This change would indicate that confidence and understanding in the ability to construct very strong symmetric-key encryption schemes exists. The one-time pad seems to have been used extensively by Russian agents operating in foreign countries. The highest ranking Russian agent ever captured in the United States was Rudolph Abel. When apprehended in 1957 he had in his possession a booklet the size of a postage stamp ($1\frac{7}{8} \times \frac{7}{8} \times \frac{7}{8}$ inches) containing a one-time key; see Kahn [648, p.664].

§1.6

The concept of a digital signature was introduced by Diffie and Hellman [345] and independently by Merkle [850]. The first practical realization of a digital signature scheme appeared in the paper by Rivest, Shamir, and Adleman [1060]. Rabin [1022] (see also [1023]) also claims to have independently discovered RSA but did not publish the result.

Most introductory sources for digital signatures stress digital signatures with message recovery coming from a public-key encryption system. Mitchell, Piper, and Wild [882] give a good general treatment of the subject. Stinson [1178] provides a similar elementary but general introduction. Chapter 11 generalizes the definition of a digital signature by allowing randomization. The scheme described in §1.8 is referred to as *deterministic*. Many other types of digital signatures with specific properties have been created, such as blind signatures, undeniable signatures, and failstop signatures (see Chapter 11).

§1.7

Much effort has been devoted to developing a theory of authentication. At the forefront of this is Simmons [1144], whose contributions are nicely summarized by Massey [786]. For a more concrete example of the necessity for authentication without secrecy, see the article by Simmons [1146].

§1.8

1976 marked a major turning point in the history of cryptography. In several papers that year, Diffie and Hellman introduced the idea of public-key cryptography and gave concrete examples of how such a scheme might be realized. The first paper on public-key cryptography was "Multiuser cryptographic techniques" by Diffie and Hellman [344], presented at the National Computer Conference in June of 1976. Although the authors were not satisfied with the examples they cited, the concept was made clear. In their landmark paper, Diffie and Hellman [345] provided a more comprehensive account of public-key cryptography and described the first viable method to realize this elegant concept. Another good source for the early history and development of the subject is Diffie [343]. Nechvatal [922] also provides a broad survey of public-key cryptography.

Merkle [849, 850] independently discovered public-key cryptography, illustrating how this concept could be realized by giving an elegant and ingenious example now commonly referred to as the *Merkle puzzle scheme*. Simmons [1144, p.412] notes the first reported application of public-key cryptography was fielded by Sandia National Laboratories (U.S.) in 1978.

§1.9

Much of the early work on cryptographic hash functions was done by Merkle [850]. The most comprehensive current treatment of the subject is by Preneel [1004].

§1.10

A large number of successful cryptanalytic attacks on systems claiming security are due to protocol failure. An overview of this area is given by Moore [899], including classifications of protocol failures and design principles.

§1.11

One approach to distributing public-keys is the so-called *Merkle channel* (see Simmons [1144, p.387]). Merkle proposed that public keys be distributed over so many independent public channels (newspaper, radio, television, etc.) that it would be improbable for an adversary to compromise all of them.

In 1979 Kohnfelder [702] suggested the idea of using *public-key certificates* to facilitate the distribution of public keys over unsecured channels, such that their authenticity can be verified. Essentially the same idea, but by on-line requests, was proposed by Needham and Schroeder (ses Wilkes [1244]).

A provably secure key agreement protocol has been proposed whose security is based on the Heisenberg uncertainty principle of quantum physics. The security of so-called *quantum cryptography* does not rely upon any complexity-theoretic assumptions. For further details on quantum cryptography, consult Chapter 6 of Brassard [192], and Bennett, Brassard, and Ekert [115].

§1.12

For an introduction and detailed treatment of many pseudorandom sequence generators, see Knuth [692]. Knuth cites an example of a complex scheme to generate random numbers which on closer analysis is shown to produce numbers which are far from random, and concludes: *...random numbers should not be generated with a method chosen at random.*

§1.13

The seminal work of Shannon [1121] on secure communications, published in 1949, remains as one of the best introductions to both practice and theory, clearly presenting many of the fundamental ideas including redundancy, entropy, and unicity distance. Various models under which security may be examined are considered by Rueppel [1081], Simmons [1144], and Preneel [1003], among others; see also Goldwasser [476].