

## Chapter 1

# Introduction

### 1.1 Overview of Trusted Collaborative Computing

Information and communication technologies, along with society's drive for collaboration in the modern world, make "collaborative computing" (CC) and its applications possible and necessary. Typical CC applications include, but are not limited to, multi-party military actions, teleconferencing, tele-medicine, interactive and collaborative decision making, grid-computing, information distribution, and pay per view services. Trust in such an environment can eventually determine its success and popularity due to the corporate and human's desire for confidentiality and integrity of their personal and/or shared information. The current Internet is not security-oriented by design. Security patches and more powerful computing/storage resources available to hackers may result in more security vulnerabilities. Compared to the two-party interaction model (such as the client-server service model), CC environments are group-oriented, involve a large number of entities and shared resources, are complex, dynamic, distributed, and heterogeneous and may possibly even include hostile elements. Systems experience failures due to internal faults and external attacks from hostile entities. In addition, there is the problem of insider threats, by which attacks are from malicious parties inside the organizations or members of CC groups. Consequently, building a Trusted Collaborative Computing (TCC) environment is very difficult and requires a long term persevering endeavor.

The theme of TCC is to make CC environments and applications highly secure and dependable and be able to not only protect systems against components failures but also defend against external attacks, even the attacks from internal malicious users. TCC will be able to not only migrate

traditional CC applications from untrustworthy environments to a secure and reliable platform, but also provide security guarantee/services for new emerging CC applications. From the technical point of view, TCC would encompass both security and reliability and seek the seamless integration of advanced security and reliability technologies.

TCC environments are characterized by collaborative tasks which require multiple entities to work together and share their resources. The first key issue in this environment is that multiple participating entities must communicate securely among one another. IP multicast provides efficient transmission of messages to a group of users; however, the open nature of IP multicast makes it unable to provide strong confidentiality. Thus, secure group-oriented communication is the first fundamental function for TCC. Another key requirement is related to resource sharing and data exchange. Access to shared resources/data must be finely controlled; otherwise attackers and malicious users can access resources to which they are not entitled to access, abuse, tamper, and even damage the resources. Thus selective data sharing, at different granularity levels and along with access control, becomes another fundamental function. These two classes of fundamental functions should be sufficiently flexible in supporting various possible forms of interactive access relations between the parties and the resources in the system. Consequently, we can identify four fundamental security requirements for TCC: secure group communication (SGC), secure dynamic conferencing (SDC), differential access control (DIF-AC), and hierarchical access control (HAC). Cryptography is a powerful tool to support all these functions.

As is well known, key management is the most important yet difficult issue in such context. How to generate, distribute, update, and revoke keys in large and dynamic environments is an important challenge. This book covers and classifies typical key management schemes for these security functions.

Intrusion is a very serious security problem in computing and communication systems. Intruding attacks, such as Denial of Services (DoS), are easily launched but very difficult to defend. Such attacks exist in CC environments without doubt, moreover, they are more serious in CC environments because the attacks can be launched by internal malicious users and/or the collusion among internal users and/or external attackers. Knowing intrusion attacks and becoming familiar with intrusion detection and defense technologies are crucial for designing and implementing TCC environments. This book presents a comprehensive survey and classification

about intruding attacks, detection and response mechanisms, and trace back technologies.

Reliability is a coherent requirement and feature of TCC. A fault or failure from any part/component of TCC environments would degrade the performance of the systems and affect multiple party collaboration and interaction; furthermore, it may have serious consequences. For example, it could be potentially disastrous if a patient's records fail to be loaded due to system failures and they are unavailable in the event of a life-threatening emergency. Grid computing is a recently developed technology for complex systems with large-scale resource sharing, wide-area communication, and multi-institutional collaboration. It could become a potential platform hosting TCC framework and applications. Therefore, this book will discuss advanced reliability technologies in the context of grid computing environments. Moreover, it also contributes one chapter on the security issues in grid computing.

The medical information system (MIS) is a typical collaborative computing application in which physicians, nurses, professors, researchers, health insurance personnel, etc. share patient information (including text, images, multimedia data) and collaboratively conduct critical tasks via the networked system. On one hand, people would be willing to step into the MIS age only when their privacy and integrity can be protected and guaranteed within MIS systems. On the other hand, only secure and reliable MIS systems would provide safe and solid medical and health care services to people. This book devotes one chapter to discuss trusted and seamless MIS systems, including why security and reliability technologies are necessary and how they can be integrated with the existing MIS systems to make the systems highly secure and dependable.

In the rest of this chapter, we present basic concepts and preliminaries used in the remainder of the book. These include: one-way functions, (one-way) hash function, the Chinese Remainder Theorem (CRT), the Discrete Logarithm Problem (DLP), and secret sharing. Included are also universal generating function, graph theory, and Bayesian approach, fault-tree analysis, binary decision diagram, and reliability block diagram.

## 1.2 Basic Concepts in Terms of Security

**Definition 1.1.** A function  $F = f(x)$  is called a one-way function if it is easy to compute  $F$  given  $x$  however it is computationally difficult to compute a preimage  $x$  given  $F$ .

**Definition 1.2.** A function  $F = f(x)$  is called a (cryptographic) hash function if it is a mapping:  $\{0, 1\}^* \rightarrow \{0, 1\}^n$  (i.e., from a bit-string of arbitrary length to a bit-string of fixed length) and satisfies the following three properties [Stinson (1995)]:

- (1) One-way, that is, given  $F \in \{0, 1\}^n$ , it is difficult to find an  $x \in \{0, 1\}^*$  such that  $F = f(x)$ .
- (2) Matching resistant, that is: given  $x \in \{0, 1\}^*$ , it is difficult to find a  $x' \in \{0, 1\}^*$  such that  $f(x') = f(x)$
- (3) Collision resistant, that is, it is difficult to find  $x, x' \in \{0, 1\}^*$  such that  $f(x') = f(x)$ .

It is worthy to mention that in many situations, the one-way function is utilized in the form of  $F = f(x, y)$ . It has two inputs  $x, y$  and the two inputs can form a binary string by some operations such as concatenation, exclusive-OR, etc.

The most important feature of a cryptosystem or secure group communication scheme is its strength in preventing opponents (attackers) from breaking the system. There are two basic mechanisms to discuss the security of a cryptosystem: *computational security* and *unconditional security* [Stinson (1995)].

**Definition 1.3.** A cryptosystem is defined to be computationally secure if the best algorithm for breaking it requires at least  $N$  operations, where  $N$  is some specified, very large number. On the other hand, a cryptosystem is defined to be unconditionally secure if it cannot be broken by attackers, even if the attackers collude and have infinite computational resources [Stinson (1995)].

Unconditional security implies that when an opponent does not know the secret key  $k$ , then  $k$  appears to him as a random element in the key space; that is, the opponent has no way of distinguishing the key  $k$  from any other element in the key space. As for computational security, in general, a computationally secure cryptosystem utilizes one-way functions. A prominent example is a public-key based system, which is always computationally secure, but never unconditionally secure. On the contrary, secret-key based systems generally belong to the category of unconditional security.

Another approach used to discuss the security of secure group communications and hierarchical access control, is that of *k-resilient security*.

**Definition 1.4.** A scheme (system) is said to have *k-resilient security* if it is secure against the collusion of any subset of up-to  $k$  opponents but cannot defend against the collusion of some  $k + 1$  or more opponents.

For  $k$ -resilient security, the value  $k$  is a system security parameter and can be set/selected during the system setup. In general, the larger the  $k$  is, the more secure a system will be. In contrast, the system will become more inefficient in terms of both space and time complexity.

The Discrete Logarithm Problem (DLP) is the basis of many cryptographic primitives including the ElGamal public-key cryptosystem [ElGamal (1985); Stinson (1995)], the Diffie-Hellman key exchange protocol, and the ElGamal signature scheme [ElGamal (1985); Stinson (1995)]. The following is its definition [Stinson (1995)].

**Definition 1.5.** Let  $G$  be a cyclic group of very large finite order  $n$ , which without loss of generality we denote multiplicatively. Let  $\alpha \in G$  be a generator of  $G$ , thus, for any element  $\beta \in G$ , there is a unique positive integer  $a$ ,  $0 \leq a \leq n - 1$  such that  $\beta = \alpha^a$ . The related Discrete Logarithm Problem (DLP) can be stated as follows: Given the group  $G$ , the generator  $\alpha$  and an element  $\beta \in G$ , determine the unique  $a \in [0, n - 1]$  such that  $\beta = \alpha^a$ . We denote the integer  $a$  by  $\log_{\alpha}\beta$ .

We wish to point out that the intractability (or otherwise) of DLP is not intrinsically a group-theoretic property, but rather depends on the representation of the particular cyclic group  $G$ . Note for example that every cyclic group of order  $n$  is isomorphic to the additive group  $\mathbb{Z}_n$  under addition modulo  $n$ . However, if  $\alpha$  is a generator of  $\mathbb{Z}_n$  then  $\gcd(\alpha, n) = 1$ , and if  $\beta$  is an arbitrary element of  $\mathbb{Z}_n$ , the discrete logarithm problem now becomes  $a\alpha = \beta$ . However, since  $\gcd(\alpha, n) = 1$ ,  $\alpha$  has a multiplicative inverse, say  $\gamma$ , in the ring  $(\mathbb{Z}_n, +, \cdot)$ , which can be efficiently computed by means of the Euclidean algorithm. We then have that  $a = \beta\gamma \bmod n$ , and the DLP has a trivial solution, independently of  $n$ .

In general, DLP is intractable in the cyclic multiplicative group  $\mathbb{F}^*$  of a finite field  $\mathbb{F} = \mathbb{F}_q = GF(q)$  of order  $q = p^m$ . DLP is also generally intractable in very large cyclic components of a chosen elliptic curve  $\mathcal{E}$  over a finite field  $\mathbb{F}_q$ . In both the finite field and elliptic curve case the order of the cyclic group must be such that certain well known attacks cannot be mounted.

In the case where we work with a field  $\mathbb{Z}_p$  of prime order  $p$ , The DLP is described in a simpler way as follows:

**Definition 1.6.** Let  $p$  be a large prime and  $\alpha \in \mathbb{Z}_p$  a primitive element (i.e., generator) of  $\mathbb{Z}_p^*$ . The question is: given any  $\beta \in \mathbb{Z}_p^*$ , find the unique integer  $a$ ,  $0 \leq a \leq p - 2$  such that  $\alpha^a \equiv \beta \pmod{p}$ . As before we denote the integer  $a$  by  $\log_\alpha \beta$ .

In the latter case also, care should be exercised in the choice of  $p$  so that certain known attacks will not compromise the DLP. It is rather clear that for secure choices of the parameters, the DLP problem induces, in fact, a one-way function. Thus, any system or protocol based on DLP is *computationally secure*.

The Chinese Remainder Theorem (CRT) is used as a theoretical foundation and practical tool in many cryptosystems as well as schemes for secure group communications, such as the Secure Lock [Chiou and W.T.Chen (1989)]. We review the CRT below (See also [Stinson (1995)].)

**Theorem 1.1.** Suppose  $N_1, \dots, N_r$  are pairwise relatively prime positive integers, i.e.,  $\gcd(N_i, N_j) = 1$  if  $i \neq j$ , and let  $N = N_1 \cdot N_2 \cdots N_r$ . For any given integers  $a_1, \dots, a_r$ , consider the following system of congruences:

$$\begin{aligned} x &\equiv a_1 \pmod{N_1} \\ x &\equiv a_2 \pmod{N_2} \\ &\vdots \\ x &\equiv a_r \pmod{N_r}. \end{aligned}$$

Then the system has a unique solution modulo  $N$ , namely :

$$x \equiv \sum_{i=1}^r a_i n_i y_i \pmod{N}$$

where  $n_i = N/N_i$  and  $y_i = n_i^{-1} \pmod{N_i}$  (i.e.  $y_i$  is the multiplicative inverse of  $n_i$  modulo  $N_i$ ). On the other hand, given any  $x \in \mathbb{Z}_N$ ,  $a_1, \dots, a_r$  can be uniquely computed as  $a_i \equiv x \pmod{N_i}$ .

There are two kinds of cryptosystems: *secret-key* (or *symmetric*) cryptosystems and *public-key* (or *asymmetric*) cryptosystems.

In a secret-key cryptosystem, the sender and receiver share a common secret key  $k$ , belonging to a very large set of possible keys, the *key space*  $\mathcal{K}$ . Each key  $k \in \mathcal{K}$  uniquely determines functions  $e_k$  (the *encryption transformation*) and  $d_k$  (the *decryption transformation*). The bijections  $e_k$  and  $d_k$  are inverses of each other under composition of functions, and each can easily be derived from the other. Therefore secret-key cryptosystems are

also called *symmetric cryptosystems*. The domain  $\mathcal{P}$  and co-domain  $\mathcal{C}$  of  $e_k$  are very large sets, and in many applications,  $\mathcal{P} = \mathcal{C}$ , in which case  $e_k$  and  $d_k$  are permutations on the *message space*  $\mathcal{M} = \mathcal{P} = \mathcal{C}$ .

In *public-key* cryptosystems, two transformations  $e_A$  (*encryption*) and  $d_A$  (*decryption*) are determined by each communicant  $A$ . These transformations are bijections and inverses of each other as in the case of secret-key cryptosystems above. However, in this case it is computationally infeasible to determine  $d_A$  given  $e_A$ , and equally infeasible to determine  $e_A$  from  $d_A$ . The transformation  $e_A$  is made public, while  $d_A$  is kept secret, known only by  $A$ . When participant  $B$  wishes to send a secure message  $x$  to  $A$ , he/she looks up  $e_A$  from a public directory, computes  $y = e_A(x)$  and sends  $y$  to  $A$ . To recover the message,  $A$  computes  $d_A(y) = d_A(e_A(x)) = x$ .

*RSA* is one of the most widely-used public-key cryptosystems. *RSA* is based on the fact that no efficient algorithms are known (except on a quantum computer) to find the factorization of a large integer as the product of prime factors. In contrast, the problem of determining whether a given integer is prime (or composite) can be solved efficiently by both probabilistic algorithms [Koblitz (1994); Menezes *et al.* (1996)], and recently by a deterministic algorithm [Agrawal *et al.* (2002)]. In particular, if  $n = pq$  is the product of two primes, the multiplicative group of units  $\mathbb{Z}_n^*$ , of the ring  $\mathbb{Z}_n$ , has order  $\phi(n) = (p - 1)(q - 1)$ , where  $\phi(\cdot)$  denotes Euler's  $\phi$  function. It is now easy to see that knowledge of  $\phi(n)$  is equivalent to knowing the factorization of  $n$ . If  $x \in \mathbb{Z}_n^*$  then  $x^{\phi(n)} \equiv 1 \pmod n$ , (Fermat's little theorem), and *RSA* is based on this fact.

Because the *RSA* cryptosystem involves arithmetic with very large integers, the encryption and decryption operations are rather slow and the system is used mostly in a large number of key management and other cryptographic protocols rather than for message encryption. It is important to note that from a theoretical point of view *RSA* can be seen as a system based on the fact that the order of the underlying group ( $\mathbb{Z}_n^*$ ) is not publicly known.

The formal definition of *RSA* is as follows.

**Definition 1.7.** Let  $n = pq$  where  $p$  and  $q$  are large primes. Let the ciphertext space and the plaintext space be both  $\mathbb{Z}_n$ , and define

$$\mathcal{K} = \{(n, p, q, a, b) : ab \equiv 1 \pmod{\phi(n)}\}.$$

For  $K = (n, p, q, a, b)$ , define

$$\text{the encryption rule as } e_K(x) = x^b \pmod n$$

and

the decryption rule as  $d_K(y) = y^a \bmod n$

$(x, y \in \mathbb{Z}_n)$ . The values  $n, b$  comprise the public key, and the values  $p, q, a$  form the private key.

It is rather interesting to note that  $x^{ab} \equiv x \bmod n$  for all  $x \in \mathbb{Z}_n$  rather than just the units of  $\mathbb{Z}_n$ .

One of the most difficult problems in secret-key cryptosystems is to establish a secret key between two communicants. In 1976, Diffie and Hellman proposed an elegant protocol, based on DLP, for two parties to achieve a shared secret key over an insecure channel. This is the well known *Diffie-Hellman key exchange* protocol [Diffie and Hellman (1976)]. We give its formal definition as follows.

**Definition 1.8.** Suppose  $p$  is a large prime and  $g$  is a generator of  $\mathbb{Z}_p^*$  such that the DLP problem in  $\mathbb{Z}_p^*$  is intractable. Parameters  $p$  and  $g$  are publicly known to members  $m_1$  and  $m_2$ . Member  $m_1$  selects a random number  $a_1 \in [1, p-2]$ , computes<sup>1</sup>  $b_1 = g^{a_1} \bmod p$ , and sends  $b_1$  to  $m_2$  over the insecure channel. Similarly,  $m_2$  selects a random number  $a_2 \in [1, p-2]$ , computes  $b_2 = g^{a_2} \bmod p$ , and sends  $b_2$  to  $m_1$ . As a result, each of  $m_1, m_2$  is able to compute the shared secret key  $SK = b_2^{a_1} = g^{a_1 a_2} = b_1^{a_2} \bmod p$ .

However any individual other than  $m_1$  or  $m_2$  cannot compute  $SK$  even if he/she knows the public parameters  $p, g$  and has intercepted the public components  $b_1$  and  $b_2$ . For  $i \in \{1, 2\}$ ,  $a_i$  is called the *Diffie-Hellman private share* or simply *private share* of  $m_i$ , and  $b_i$  the *Diffie-Hellman disguised public share* or simply *public share* of  $m_i$ . We call the derived secret key  $SK = g^{a_1 a_2}$  a DH key.

*Secret sharing schemes* are secure schemes which recover a *secret* (the key) from an appropriate number of *shares* belonging to particular privileged subsets  $S_i$  of participants. Combining the shares of all the members of such a privileged subset  $S_i$  yields complete knowledge of the secret, but combining shares of members of any proper subset  $T \subset S_i$  yields absolutely no information about the secret. An elegant implementation of secret sharing is Shamir's  $(k, n)$  Threshold scheme [Shamir (1979a)], described below.

This scheme is based on the polynomial interpolation. Given  $k$  points  $(x_1, y_1), \dots, (x_k, y_k)$  with distinct  $x_i$  in a two-dimension plane, there is one and only one polynomial  $q(x)$  of degree  $k-1$  such that  $q(x_i) = y_i$  for all  $i$ .

<sup>1</sup>In a Diffie-Hellman based system, any operation will be performed by mod  $p$ . We often omit the mod  $p$  for simplicity.

To divide the secret data  $D$  into  $n$  pieces  $D_i$  ( $i = 1, \dots, n$ ), we pick a random  $k-1$  degree polynomial,  $q(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1}$  in which  $a_0 = D$ , and evaluate:  $D_1 = q(1), \dots, D_i = q(i), \dots, D_n = q(n)$ .

Given any subset of  $k$  of these  $D_i$  values (together with their identifying indices), we can find the coefficients of  $q(x)$  by interpolation, and then evaluate  $D = q(0)$ . Knowledge of  $k-1$  of these values is not enough to compute  $D$ . The formal definition is as follows.

**Definition 1.9.** Given any secret  $D$  to share (suppose  $D$  is an integer), we pick a prime  $p$  which is larger than both  $D$  and  $n$ , where  $n$  is the number of users. The coefficients  $a_0, \dots, a_{k-1}$  in  $q(x)$  are randomly chosen from a uniform distribution over the integers in  $[0, p)$ , and the values  $D_1, \dots, D_n$  are computed  $D_1 = q(1), \dots, D_i = q(i), \dots, D_n = q(n)$  modulo  $p$ . Finally, each user  $U_i$  is given its share  $D_i$ .

This  $(k, n)$  threshold scheme has some interesting properties:

- The size of each share does not exceed that of the original secret.
- When  $k$  is kept fixed,  $D_i$  pieces can be dynamically added or removed without affecting the other  $D_i$  pieces.
- It is easy to change the  $D_i$  pieces without changing the original secret  $D$  - all we need is a new polynomial  $q(x)$  with the same  $a_0$ . This method would enhance the security because the pieces exposed by security breaches cannot be accumulated unless all of them are values of the same version of the  $q(x)$  polynomial.
- A hierarchical scheme can be obtained in which the number of pieces needed to determine  $D$  depends on an individual's importance. For example, a person of the higher importance can be given two or more shares.

### 1.3 Basic Concepts in Terms of Reliability

Reliability is the analysis of failures, their causes and consequences. It is the most important characteristic of product quality, as things must work satisfactorily before considering other quality attributes. Usually, specific performance measures can be embedded into reliability analysis by the fact that if the performance is below a certain level, a failure can be said to have occurred.

The commonly used definition of reliability is the following.

**Definition 1.10.** *Reliability* is the probability that the system will perform its intended function under specified working condition for a specified period of time.

Mathematically, the reliability function  $R(t)$  is the probability that a system will be successfully operating without failure in the interval from time 0 to time  $t$ :

$$R(t) = P(T > t), \quad t \geq 0 \quad (1.1)$$

where  $T$  is a random variable representing the failure time or time-to-failure.

The failure probability, or unreliability, is then

$$F(t) = 1 - R(t) = P(T \leq t)$$

which is known as the distribution function of  $T$ .

If the time-to-failure random variable  $T$  has a density function  $f(t)$ , then

$$R(t) = \int_t^{\infty} f(x) dx$$

The density function can be mathematically described as  $\lim_{\Delta t \rightarrow 0} P(t < T \leq t + \Delta t)$ . This can be interpreted as the probability that the failure time  $T$  will occur between time  $t$  and the next interval of operation,  $t + \Delta t$ . The three functions,  $R(t)$ ,  $F(t)$  and  $f(t)$  are closely related to one another. If any of them is known, all the others can be determined.

Usually we are interested in the expected time to next failure, and this is termed mean time to failure.

**Definition 1.11.** The *mean time to failure (MTTF)* is defined as the expected value of the lifetime before a failure occurs.

Suppose that the reliability function for a system is given by  $R(t)$ , the MTTF can be computed as

$$MTTF = \int_0^{\infty} t \cdot f(t) dt = \int_0^{\infty} R(t) dt \quad (1.2)$$

**Example 1.1.** If the lifetime distribution function follows an exponential distribution with parameter  $\lambda$ , that is,  $F(t) = 1 - \exp(-\lambda t)$ , the MTTF is

$$MTTF = \int_0^{\infty} R(t)dt = \int_0^{\infty} \exp(-\lambda t)dt = \frac{1}{\lambda} \quad (1.3)$$

This is an important result as for exponential distribution. MTTF is related to a single model parameter in this case. Hence, if MTTF is known, the distribution is specified.

The failure rate function, or hazard function, is very important in reliability analysis because it specifies the rate of the system aging. The definition of failure rate function is given here.

**Definition 1.12.** The *failure rate function*  $\lambda(t)$  is defined as

$$\lambda(t) = \lim_{\Delta t \rightarrow 0} \frac{R(t) - R(t + \Delta t)}{\Delta t R(t)} = \frac{f(t)}{R(t)} \quad (1.4)$$

The quantity  $\lambda(t)dt$  represents the probability that a device of age  $t$  will fail in the small interval from time  $t$  to  $t + dt$ . The importance of the failure rate function is that it indicates the changing rate in the aging behavior over the life of a population of components. For example, two designs may provide the same reliability at a specific point in time, but the failure rate curves can be very different.

**Example 1.2.** If the failure distribution function follows an exponential distribution with parameter  $\lambda$ , then the failure rate function is

$$\lambda(t) = \frac{f(t)}{R(t)} = \frac{\lambda \cdot \exp(-\lambda t)}{\exp(-\lambda t)} = \lambda \quad (1.5)$$

This means that the failure rate function of the exponential distribution is a constant. In this case, the system does not have any aging property. This assumption is usually valid for software systems. However, for hardware systems, the failure rate could have other shapes.

When a system fails to perform satisfactorily, repair is normally carried out to locate and correct the fault. The system is restored to operational effectiveness by making an adjustment or by replacing a component.

**Definition 1.13.** *Maintainability* is defined as the probability that a failed system will be restored to a functioning state within a given period of time when maintenance is performed according to prescribed procedures and resources.

Generally, maintainability is the probability of isolating and repairing a fault in a system within a given time. Maintenance personnel have to work

with system designers to ensure that the system product can be maintained cost effectively.

Let  $T$  denote the time to repair or the total downtime. If the repair time  $T$  has a density function  $g(t)$ , then the maintainability,  $V(t)$ , is defined as the probability that the failed system will be back in service by time  $t$ , i.e.,

$$V(t) = P(T \leq t) = \int_0^t g(x)dx$$

An important measure often used in maintenance studies is the mean time to repair (MTTR) or the mean downtime. MTTR is the expected value of the repair time.

System repair time usually consists of two separate intervals: passive repair time and active repair time. Passive repair time is mainly determined by the time taken by service engineers in preparation, such as traveling to the customer site. In many cases, the cost of travel time could exceed the cost of the actual repair. Active repair time is directly affected by the system design. The active repair time can be reduced significantly by designing the system in such a way that faults may be quickly detected and isolated. As more complex systems are designed, it becomes more difficult to isolate the faults.

Another important reliability related concept is system availability. This is a measure that takes both reliability and maintainability into account.

**Definition 1.14.** The *availability* function of a system, denoted by  $A(t)$ , is defined as the probability that the system is available at time  $t$ .

Different from the reliability that focuses on a period of time when the system is free of failures, availability concerns a time point in which the system does not stay at the failed state. Mathematically,

$$A(t) = \Pr(\text{System is up or available at time instant } t)$$

The availability function, which is a complex function of time, has a simple steady-state or asymptotic expression. In fact, usually we are mainly concerned with systems running for a long time. The steady-state or asymptotic availability is given by

$$A = \lim_{t \rightarrow \infty} A(t) = \frac{\text{System up time}}{\text{System up time} + \text{System down time}} \\ = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

The mean time between failures (MTBF) is another important measure in repairable systems. This implies that the system has failed and has

$z$  A random integer which is changed and made public every time

$A(x)$  Access control polynomial (ACP)

$P(x)$  Public polynomial sent to users for key distribution,  $P(x) = A(x) + K$

$P$  The system prime used for modular computation

$U_i$  A group member in a certain group

$v_j$  A certain vertex in an access hierarchy

$\hat{k}_i$  A secret node key

$k_i$  A private node key

$f(x, y)$  or  $h(x)$  The public one-way hash function

$ID_i$  A unique public identity assigned to each vertex in an access hierarchy

$m$  The number of users in a certain node

$n$  The number of vertices in an access hierarchy

$e_{i,j}$  The public edge value on the edge from  $v_i$  to  $v_j$ .

$\%$  Modular operation

## 1.5 Outline

This book consists of seven chapters, which are listed and described, as follows:

**Chapter 1.** This chapter presents an introductory overview of TCC. The chapter begins by introducing central problems of TCC, such as security and dependability in CC environments. It proceeds to introduce preliminaries for TCC and notations used in the rest of the book. Finally, an outline of the subsequent chapters of the book is presented.

**Chapter 2.** This chapter discusses the first one of the main security requirements/functions in TCC environments: secure group communication (SGC) and interaction. The primary issue for SGC is group key management. This chapter gives a comprehensive overview over the state-of-art group key management schemes. The chapter also contains a discussion on secure dynamic conferences (SDC).

**Chapter 3.** This chapter discusses another important security requirement and function in TCC environments: data sharing/exchange and access control. The chapter mainly focuses on cryptography based hierarchical access control (CHAC) techniques.

**Chapter 4.** In this chapter, we discuss and classify intrusion attacks, their corresponding detection and response technologies. In particular, we discuss typical traceback techniques and introduce a new DoS/DDoS de-

fendant architecture, called Secure Overlay Services (SoS) and proposed by Dr. Keromytis's research group.

**Chapter 5.** As a representative of collaborative computing, grid computing is a newly developed technology for complex systems with large-scale resource sharing, wide-area communication, and multi-institutional collaboration. Although the developmental tools and infrastructures for the grid have been widely studied, grid reliability analysis and modeling are not easy because of its largeness, complexity and stiffness. This chapter introduces the Grid computing technology, analyzes different types of failures in the grid system and their influence on its reliability and performance. The chapter then presents models for star-topology grid considering data dependence and tree-structure grid considering failure correlation. Evaluation tools and algorithms are developed, based on Universal Generating function, Graph theory, and Bayesian approach.

**Chapter 6.** Grid computing environments are typical CC paradigms that users offer and share their resources. In this omnipresent environment, security becomes a critical factor to its success. This chapter introduces the Grid Security problem, its challenges and requirements. Moreover, we present a dual level key management scheme which can offer secure grid communication and fine-grained access control to shared grid resources. In the remainder of the chapter, some examples of typical grid services present how grid services are supported by this scheme.

**Chapter 7.** Medical Information Systems (MIS) help medical practice and health care significantly. Security and dependability are two increasingly important factors for MIS nowadays. On one hand, people would be willing to step into the MIS age only when their privacy and integrity can be protected and guaranteed within MIS systems. On the other hand, only secure and reliable MIS systems would provide safe and solid medical and health care service to people. In this chapter, we discuss why the security and reliability technologies presented in the previous chapters are necessary and how they can be integrated with the existing MIS systems. We also present a Middleware architecture which has been implemented and integrated with the existing VISTA (Veterans Health Information Systems and Technology Architecture) and CPRS (Computerized Patient Record System) in the U.S. Department of Veterans Affairs seamlessly, and does not modify/replace any current components.

been repaired. Like MTTF and MTTR, MTBF is an expected value of the random variable time between failures. Mathematically,  $MTBF = MTTR + MTTF$ .

**Example 1.3.** If a system has a lifetime distribution function  $F(t) = 1 - \exp(-\lambda t)$  and a maintainability function  $V(t) = 1 - \exp(-\mu t)$ , then  $MTTF = 1/\lambda$  and  $MTTR = 1/\mu$ . The MTBF is the sum of MTTF and MTTR and the steady-state availability is

$$A = \frac{MTTF}{MTTR + MTTF} = \frac{1/\lambda}{1/\lambda + 1/\mu} = \frac{\mu}{\lambda + \mu}$$

## 1.4 Abbreviations and Notations

### Abbreviations:

ACP – Access Control Polynomial

CA – Central Authenticator

CPRS – Computerized Patient Record System

DLKM – Dual-Level Key Management

DoS – Denial of Service

DDoS – Distributed Denial of Service

DIDS – Distributed Intrusion Detection System

GC – Group Controller

GSI – Globus Security Infrastructure

HAC – Hierarchical Access Control

HIDS – Host based Intrusion Detection System

IDS – Intrusion Detection System

KMS – Key Management Server

LCM – Least Common Multiple

MANET – Mobile Ad hoc NETWORKS

MTST – Minimal Task Spanning Tree

NIDS – Network based Intrusion Detection System

RMS – Resource Management System

PKI – Public Key Infrastructure

SOS – Secure Overlay Service

VISTA – Veterans Health Information Systems and Technology Architecture

VO – Virtual Organization

### Notations:

$SID_i$  Every valid user/machine is assigned a permanent secret key  $SID_i$