2 Background of Cryptographic Protocols

Abstract Some background knowledge including preliminary knowledge, cryptographic primitive knowledge, cryptographic protocol knowledge, cryptographic protocol security knowledge, and communication threat model knowledge are briefly introduced.

A brief introduction of background knowledge on cryptographic protocols is given in this chapter. Readers who are familiar with cryptography knowledge can skip this chapter without causing difficulty for reading the rest parts of this book, while inquisitive readers can refer to Refs. [1, 2] for sufficient reference materials.

2.1 Preliminaries

This section gives some preliminary knowledge of cryptographic protocols including functions, one way transformation, message space, adversary, etc.

2.1.1 Functions

Definition 2.1 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 by $X = \{a, b, c\}$.

Definition 2.2 A function (or transformation) 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 for which the rule Y 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 $f : X \to Y$. 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 by $\text{Im}(f)$.

Definition 2.3 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 $y \in \text{Im}(f)$ it is "computationally infeasible" to find any $x \in X$ so that $f(x) = y$.

Here and elsewhere, the terms easy and computationally infeasible (or hard) are intentionally left without formal definition; it is intended they are intended to be interpreted relative to an understood frame of reference. Easy might mean polynomial time and space, or more practically, within a certain number of machine operations or time units — perhaps seconds or milliseconds. A more specific definition of computationally infeasible might involve super-polynomial effort, require effort far exceeding understood resources, specify a lower bound on the number of operations or memory required in terms of a specified security parameter, or specify that the probability that a security property is violated is exponentially small.

Definition 2.4 A function (or transformation) is injective if each element in the codomain Y is the image of at most one element in the domain X.

Definition 2.5 A function (or transformation) is onto if each element in the codomain Y is the image of at least one element in the domain X. Equivalently, a function $f : X \to Y$ is onto if $\text{Im}(f) = Y$.

Definition 2.6 If a function $f : X \to Y$ is injective and $\text{Im}(f) = Y$, then f is called a bijection.

Definition 2.7 A trapdoor one-way function is a one-way function $f : X \to Y$ with the additional property that gives some extra information (called the trapdoor information) and it becomes feasible to find, for any given $y \in \text{Im}(f)$, an $x \in X$ so that $f(x) = y$.

Definition 2.8 Let S be a finite set and let f be a bijection from S to S (i.e., $f : S \to S$). The function f is called an involution if $f = f^{-1}$. An equivalent way of stating this is $f(f(x)) = x$ for all $x \in S$.

2.1.2 Terminology

Definition 2.9 A denotes a finite set called the alphabet of definition. For example, $A = \{0, 1\}$, the binary alphabet, is a frequently used alphabet of definition.

Definition 2.10 M denotes a set called the message space. M consists of strings of symbols from an alphabet of definition. An element of M is called a plaintext message or simply a plaintext. For example, M may consist of binary strings, English texts, computer codes, etc.

Definition 2.11 C denotes a set called the ciphertext space. C consists of strings of symbols from an alphabet of definition, which may differ from the alphabet of definition for M. An element of C is called a ciphertext.

Definition 2.12 K denotes a set called the key space. An element of K is called a key. In this book, we usually use upper case K as a long-term key, and lower case k as a temporal session key.

Definition 2.13 An encryption scheme consists of a set $E_e : e \in K$ of encryption transformations (encryption algorithm) and a corresponding set $D_d : d \in K$ of decryption transformations (decryption algorithm) with the property that for each $e \in K$ there is a unique key $d \in K$ so that $D_d = E_e^{-1}$, that is, $D_d(E_e(m)) = m$ for all $m \in M$.

Definition 2.14 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.

Public key e, and private key d are paired keys in a public-key system; symmetric keys e and d $(e = d)$ are equal keys in a symmetric-key (single-key) system.

A fundamental premise in cryptography is that the sets $M, C, K, \{E_e : e \in K\}, \{D_d : d \in K\}$ are public knowledge, and the only thing to keep secret is the particular key pair (e, d) that is being used.

The other premise in cryptography a priori is that for point-to-point mechanisms, parties A and B shared a secret key, or A and B each shares a secret key with a trusted party T in a symmetric-key mechanism; A and B know the opponent parties' public key, or A and B both have the knowledge of the public key of a trusted party T in a public-key mechanism.

These shared long-term keys are initially established by non-cryptographic, and out-of-band techniques providing confidentiality and authenticity (e.g., in person, or by trusted courier).

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

Definition 2.15 An adversary is an unauthorized "third" party in a communication who tries to defeat the information security service provided between the sender and receiver. Various other names are synonymous with adversary such as enemy, attacker, opponent, tapper, eavesdropper, intruder, saboteur, and interloper. We usually call the adversary "Malice" in this book.

An adversary will often attempt to play the role of either the legitimate sender or the legitimate receiver.

Definition 2.16 A passive adversary is an adversary who is only capable of reading information from an unsecured channel.

Definition 2.17 An active adversary is an adversary who is capable of not only reading information, but also transmitting, altering, or deleting information on an unsecured channel.

In a practical cryptographic setting, it is prudent to make the assumption that the adversary is an active adversary, and it is very powerful.

Definition 2.18 A passive attack is the one where the adversary only monitors the communication channel. A passive attacker only threatens confidentiality of data.

Passive attacks are in the nature of eavesdropping on, or monitoring of transmitted messages. Two classical types of passive attacks are release of message contents and traffic analysis. Passive attacks are very difficult to detect because they do not involve any alternation of data. However, it is feasible to prevent passive attacks.

Definition 2.19 An active attack is one where the adversary attempts to delete, add, or in some other way alter the transmission on the channel besides the monitoring of transmitted messages. An active attacker threatens data integrity and authentication as well as confidentiality.

Active attacks involve some replay, creation, insertion, deletion or modification of transmissions, masquerade of entity, and denial of service.

2.2 Cryptographic primitives

This section gives a brief introduction to some cryptographic knowledge including primitive, encryption, signature, identification, symmetric-key system etc.

2.2.1 Cryptology

Definition 2.20 An algorithm or a primitive is a well-defined computational procedure that takes a variable input and halts with an output.

Examples of primitives include encryption schemes, hash functions, and digital signature schemes. Primitives, when applied in various ways and with various inputs, will typically exhibit different characteristics.

Definition 2.21 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. An encryption algorithm and a decryption algorithm plus the description of the format of messages and keys form a cryptographic system or a cryptosystem.

Definition 2.22 Cryptanalysis is the study of mathematical techniques for attempting to defeat cryptographic techniques, and, more generally, information security services.

Computational complexity is one of the most important foundations for modern cryptography.

Definition 2.23 A polynomial-time algorithm, also called efficient algorithm or a good algorithm, is an algorithm whose worst-case running time function could be expressed by a polynomial in the size of the input. That is, the algorithm is deterministic or randomized with polynomial execution time. Thus, informally speaking, a computational problem is said to be easy or tractable if there exists a good or efficient algorithm to solve the problem. Any algorithm whose running time cannot be so bounded by polynomial execution time is called an exponential-time algorithm. Thus, informally speaking, a computational problem is said to be hard or intractable if there doesn't exist an efficient algorithm to solve the problem.

Note that there are, however, some practical situations when this distinction is not appropriate for average-case complexity is more important than worst-case complexity in cryptography – a necessary condition for an encryption scheme to be considered secure is that the corresponding cryptanalysis problem is difficult on average (or more precisely, almost always difficult), and not just for some isolated cases[1].

Definition 2.24 Non-deterministic polynomial-time (NP) problems are those in which we couldn't find an efficient algorithm to solve in polynomial time. Many cryptographic primitives are based on the difficulty of these intractable problems, also called hard problems, such as the integer factorization problem and the discrete logarithm problem. Some hard problems in computational complexity theory can provide a high confidence in the security of a cryptographic algorithm or protocol.

Clearly, a cryptographic algorithm must be designed so that it is tractable for a legitimate user, but is intractable for a non-user or an attacker and so constitutes a difficult problem to solve. Here, "intractable" means that the widely available computational methods cannot effectively handle these problems, that is, they cannot solve these problems in polynomial time.

Definition 2.25 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.

Encryption schemes are not inherent to provide data integrity, and in some cases the encryption scheme may not be compromised while the cryptographic protocol may fail to provide confidentiality adequately.

Cryptographic techniques are typically divided into two generic types: symmetric-key system and public-key system.

Definition 2.26 A symmetric-key system is a system involving two transformations – one for the originator and one for the recipient – both of which make use of either the same secret key or two keys easily computed from each other. Symmetric-key system is also referred to as conventional cryptosystem.

Definition 2.27 An public-key system is a system involving two related transformations – one defined by a public key (the public transformation), and the other defined by a private key — with the property that it is computationally infeasible to determine the private transformation from the public transformation.

Most well-known public-key cryptosystems are based on the difficulty of intractable problems, such as the integer factorization problem, the discrete logarithm problem, and elliptic-curve discrete logarithm problem (ECDLP).

2.2.2 Symmetric-key encryption

Definition 2.28 Symmetric-key encryption is a cryptographic primitive which uses the same encryption and decryption keys, $e = d$, to perform encrypting and decrypting. Other terms used in the literature are single-key encryption, one-key encryption, private key encryption, secret key encryption, and conventional encryption.

In a two-party communication, the key in symmetric-key encryption must remain secret at both ends, and sound cryptographic practice dictates that the key be changed frequently perhaps for each communication session. In symmetric-key case, the only system that has proven secure is the one-time pad.

The symmetric-key primitives can be designed to have much higher rates of data throughput in encrypting and decrypting than public-key encryption.

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

Definition 2.29 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 A, and encrypts one block at a time.

Most well-known symmetric-key encryption techniques are block ciphers. For example, DES, 3DES, IDEA etc.

Definition 2.30 A stream cipher is, in one sense, a very simple block cipher having block length equal to one.

What makes stream ciphers useful is the fact that the encryption transformation can change for each symbol of plaintext encrypted. They are especially useful in some situations, such as highly probable error transmission channel, or buffering of data is limited.

2.2.3 Public-key encryption

Definition 2.31 A public-key encryption is a cryptographic primitive which uses different encryption and decryption keys, the public-key e and the private key d and the two keys match each other; the encryption key e needn't be kept secret, and only the party who is the owner of e can decrypt a ciphertext encrypted under e using the matching private key d. Other terms used in the literature are asymmetric cryptosystems.

Each entity in the network has a public/private encryption key pair (e, d). The public-key e along with the identity of the entity is usually stored in a central repository. In a large network, the number of keys necessary may be considerably smaller than in the symmetric-key scenario. Per-communication interaction with the central repository can be eliminated if entities store certificates locally. Depending on the mode of usage, a private/public-key pair (e, d) may remain unchanged for considerable periods of time, e.g., many communication sessions (even several years).

Throughput rates for the most popular public-key encryption methods are several orders of magnitude slower than the best known symmetric-key schemes. That is, the computational performance of a public-key encryption is inferior to that of a symmetric-key encryption.

2.2.4 Digital signatures

Definition 2.32 A digital signature is a cryptographic primitive which entails transforming the message and some secret information held by the entity into a tag called a signature in digital form, contrary to handwritten signature in previous centuries. Digital signature is fundamental in authentication, authorization, and nonrepudiation.

A digital signature of a message is a number dependent on some secret known only to the signer, and additionally, the content of the message being signed. Signatures must be verifiable; if a dispute arises as to whether a party signed a document (caused by either a lying signer trying to repudiate a signature it did create, or a fraudulent claimant), an unbiased third party should be able to resolve the matter equitably, without requiring access to the signer's secret information (private key).

A digital signature scheme (or digital signature mechanism) consists of a signature generation algorithm and an associated verification algorithm.

Schemes for unforgeable digital signature can be constructed using the same computational assumptions as used in the constructing of private-key encryption schemes, where the encryption key d should be kept secret, and all users including the adversary can decrypt a ciphertext encrypted under d using the matching public-key e.

Digital signatures have many applications in information security, including authentication, data integrity, and non-repudiation. One of the most significant applications of digital signatures is the certification of public-keys in large networks. Certification is a means for a trusted third party to bind the identity of a user to a public-key, so that at some later time, other entities can authenticate a public-key without assistance from a trusted third party.

Public-key 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 efficiency of the symmetric-key scheme. Note that data encryption is frequently the most time consuming part of the encryption process, and the public-key scheme for key establishment is a small fraction of the total encryption process between A and B[1].

The size of public-key signatures is larger than that of tags providing data origin authentication from symmetric-key techniques.

2.2.5 Hash Functions

Definition 2.33 A hash function is a cryptographic primitive which is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed n bits length, called hash values. For any given $x \in X$, the corresponding hash value y is a binary string of some fixed n bits length, $y = h(x)$. Hash value is also referred to as a hashcode, hash-result, or simply hash.

The cryptographic hash function is often informally called a one-way hash function. The basic idea of cryptographic hash functions is that a hash value serves as a compact representative image (sometimes called an imprint, digital fingerprint, or message digest) of an input string, and can be used as if it were uniquely identifiable with that string.

The main properties of cryptographic hash functions include:

1) Preimage resistance — one-way, for essentially all pre-specified outputs, is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage x' so that $h(x') = y$ when giving any y for which a corresponding input is not known. A hash function with this property is called a one-way hash function.

2) 2nd-preimage resistance — weak collision resistance is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2nd-preimage $x' \neq x$ so that $h(x') = h(x)$.

3) Collision resistance — strong collision resistance is computationally infeasible to find any two distinct inputs x, x' which hash to the same output so that $h(x) = h(x')$ (Note that here there is free choice of both inputs contrary to 2nd-preimage resistance). A hash function with this property is called a collision resistance hash function.

At the highest level, hash functions may be split into two generic classes:

1) An unkeyed hash function is a specific hash function whose specification dictates a single input parameter, a message.

2) A keyed hash functions is a specific hash function whose specification dictates two distinct inputs, a message and a secret key.

The most common cryptographic uses of hash functions are with digital signatures and are 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 in place of the original message. For data integrity, the hash value corresponding to a particular input is computed at some point in time by the sender, and later the hash value is recomputed by the receiver using the particular input at hand which is the same as the sender, and then the recomputed hash value is compared for equality with the original hash value. The problem of preserving the integrity of a potentially large message is thus reduced to that of a small fixed-size hash value.

Specific applications, which hash functions are used for data integrity in conjunction with digital signature schemes, include virus protection and software distribution.

Keyed hash functions can be used to provide data origin authentication as conventional encryption schemes do. Keyed hash functions could not directly recover the input binary strings from a computed hash value while conventional encryption schemes could recover the plaintext from a ciphertext with a common secret key at hand. When they are used to provide integrity and data origin authentication information security services, they behave almost the same. Hence, in this book, we do not distinguish keyed hash functions from conventional encryption schemes accurately when they are used to provide data origin authentication, and we take keyed hash functions as a sort of symmetric-key primitives in this case since they both require a common secret key shared by some specific identities to complete computing.

2.2.6 Message authentication

Definition 2.34 Manipulation detection codes (MDCs) are a subclass of unkeyed hash functions to provide a representative image or hash for message manipulation detection. MDCs is also called modification detection codes

(MDCs), and less commonly known as message integrity codes (MICs).

MDCs may be further classified, the specific two primary classes of MDCs are:

1) one-way hash functions (OWHFs): for these hash functions, finding an input which hashes to a pre-specified hash value is difficult;

2) collision resistant hash functions (CRHFs): for these hash functions, finding any two inputs having the same hash value is difficult.

Definition 2.35 Message authentication codes (MACs) are a distinct class of hash functions, parameterized by a secret key k, which provide the detection of message alternation on an insecure channel for encryption schemes.

MAC algorithms are also a subclass of keyed hash functions which take two functionally distinct inputs, a message and a secret key k, and produce a fixed-size (say n-bit) output or hash value, with the design intent that it is infeasible in practice to produce the same output without knowledge of the key k. That is, a hash value should be uniquely identifiable with a single input in practice, and collisions should be computationally difficult to find (essentially never occurring in practice).

MACs can be used, without the use of any additional mechanisms, to provide data integrity and symmetric data origin authentication (the source of a message), as well as identification in symmetric-key schemes. That is, MAC allows message authentication by symmetric techniques.

Distinction should be made between the use of a MAC algorithm and that of a MDC, where MAC is with a secret key included as part of its message input.

It is generally assumed that the algorithmic specification of a hash function is public knowledge. Thus in the case of MDCs, given a message as input, anyone may compute the hash value, and in the case of MACs, given a message as input, anyone with knowledge of the key may compute the hash value.

A message authentication scheme does not necessarily constitute a digital signature scheme. In some sense, message authentication is similar to a digital signature. The difference between these two is that in the setting of message authentication it is not required that a "third" party (who may be the dishonest adversaries) should be able to verify the validity of authentication tags produced by the designed users[2].

Table 2.1 indicates various types of algorithms commonly used to achieve the specified cryptographic objectives. The classification given requires specification of both the type of algorithm (e.g., encryption vs. signature) and the intended use (e.g., confidentiality vs. entity authentication).

Data origin authentication is a type of authentication whereby a party is corroborated as the (original) source of specified data created at some (typically unspecified) time in the past.

Table 2.1 Types of algorithms commonly used to meet specified objectives

Cryptographic objective (usage)	symmetric-key	public-key
confidentiality	encryption	encryption
data integrity	MDC	signature
message authentication	MAC	signature
data origin authentication	MAC	signature
non-repudiation	/	signature
key transport	encryption	encryption
key agreement	various methods	Diffie-Hellman
entity authentication (by challenge-response protocols)	1. MAC 2. encryption	1. signature 2. decryption 3. customized

Message authentication is a term used analogously with data origin authentication. It provides data origin authentication with respect to the original message source (and data integrity, but no uniqueness and timeliness guarantees).

In deed, data origin authentication implicitly provides data integrity since, if the message was modified during transmission, the message sender would no longer be the originator. Message authentication itself provides no timeliness guarantees, while entity authentication involves corroboration of a claimant's identity at the current instant in time through actual communications with an associated verifier during execution of the protocol itself.

Encryption may provide privacy of keying material in key establishment protocols, while signature and MAC may provide data integrity, message authentication and data origin authentication.

We use the terms identification and entity authentication synonymously throughout this book. Elsewhere in the literature of cryptography, identification sometimes implies only a claimed or stated identity whereas entity authentication suggests a corroborated identity. In many applications, the motivation for identification is to provide access service allowing resource usage to be tracked to identified entities, and to facilitate appropriate billing. In general, an authentication is considered flawed if a principal concludes a normal run of the protocol with its intended communication partner while the intended partner would have a different conclusion.

Identification schemes are closely related to, but simpler than digital signature schemes, which involve a variable message and typically provide a non-repudiation feature allowing disputes to be resolved by judging after the fact. Identifications do not have "lifetimes" as signatures do and a claimed identity is either corroborated or rejected immediately, with associated privileges or access either granted or denied in real time.

One of the primary purposes of entity authentication is to facilitate access control to a resource, when an access privilege is linked to a particular identity.

Entity authentication techniques may be divided into three main categories, depending on which of the following securities is based on[1]:

1) Something known. Examples include standard passwords, Personal Identification Numbers (PINs), and the secret or private keys.

2) Something possessed. It is typically a physical accessory, resembling a passport in function. Examples include magnetic-striped cards, smart cards (also called chipcards or IC cards), and hand-held customized calculators (also called password generators, or Tokencard, such as RSA Secure ID) which provide time-variant passwords or passcodes. The generator usually contains a device-specific secret key.

3) Something inherent (to a human individual). This category includes methods which make use of human physical characteristics and involuntary actions (biometrics), such as handwritten signatures, and fingerprints.

The important points in practice about cryptographic primitives are:

1) Public-key cryptography facilitates efficient signatures (particularly non-repudiation) and key management.

2) Symmetric-key cryptography is efficient for encryption and some data integrity applications.

2.3 Cryptographic protocols

This section gives some background knowledge about the cryptographic protocols including Message-driven protocol, secured channel, etc.

A communication protocol or protocol is a collection of interactive communication procedures that specify a particular processing of incoming messages and the generation of outgoing messages. A communication procedure including a sequence of determined steps runs between or among co-operative parties.

A Message-driven protocol is initially triggered at a party by an external "call" and later by the arrival of messages. Upon each of these events, and according to the protocol specification, the protocol processes information and may generate and transmit a message, and/or wait for the next message to arrive.

Note that message-driven protocols are asynchronous in nature, and this reflects the prevalent form of communication in today's networks.

Recall that a cryptographic 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.

A mechanism is a more general term encompassing protocols, including 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.

2.3.1 Secure channel

Definition 2.36 A channel is a means of conveying information from one entity to another.

1) A physically secure channel (or secure channel) is the one that is not physically accessible to the adversary. Some channels are assumed to be physically secure, and these include trusted couriers, personal contact between communicating parties, and a dedicated communication link, for example.

2) An unsecured channel (or open channel, or unprotected channel) is the one from which parties other than those for which the information is intended can reorder, delete, insert, or read.

3) A secured channel (or protected channel,or authenticated channel) is the one from which an adversary does not have the ability to reorder, delete, insert, or read. A secured channel may be secured by physical or cryptographic techniques. Secure channel is a type of secured channel.

Definition 2.37 A session is a copy of a protocol run or a protocol instance at a party. Several copies of protocol run (i.e., interactive subroutines or other protocols) may be simultaneously instantiated by one party.

Technically, a session is an interactive subroutine executed inside a party. Each session is identified by the party that runs it, the parties with whom the session communicates and a session-identifier. The session-identifiers are used in practice to bind transmitted messages to their corresponding sessions.

A session could be completed at one party but not necessarily at the other. Each run of a protocol at a given party creates a local state for that session during execution. Typically, the local state of a session is mostly independent of local states of other sessions. When a session ends its run and outputs a key in a key establishment protocol, we call this session a complete one and assume that its local state is erased.

A session key is an ephemeral secret whose use is restricted to a short time period such as a single telecommunications connection (or a session), after which all trace of it is eliminated.

Key establishment protocols result in a shared ephemeral session key or shared ephemeral session key parts for subsequent communications. Motivation for using ephemeral keys includes the following:

1) to limit available ciphertext (under a fixed key) for cryptanalytic attack;

2) to limit exposure, with respect to both time period and quantity of data, in the event of (session) key compromise;

3) to avoid long-term storage of a large number of distinct secret keys (in the case where one terminal communicates with a large number of others) by creating keys only when actually required;

4) to create independence across communication sessions or applications. It is also desirable in practice to avoid the requirement of maintaining state information across sessions.

2.3.2 Principals

An entity (or a party) is someone or something that sends, receives, or manipulates information over point-to-point channels. An entity may be a person, a computer terminal, etc., or even an adversary.

Definition 2.38 A communicating entity or party is formally called principal in key establishment protocol with a unique name. They are probabilistic polynomial-time users, processes, or machines in current cryptographic literature, and we usually denote principals by $P_1, P_2, ..., P_n$, etc.

Definition 2.39 A trusted third party (TTP) is an entity in the network who is trusted by all other entities. This party is referred to by a variety of names depending on the role it plays: trusted third party, trusted server, authentication server, key distribution center (KDC), key translation center (KTC), and certification authority (CA).

Definition 2.40 A key translation center (KTC) is a trusted server which allows two parties A and B, which do not directly shared keying material, to establish a secure communication through use of the long-term keys K_{AT} and K_{BT} they respectively shared with TTP T.

In symmetric-key case, each entity shares a distinct symmetric key with the TTP. These keys are assumed to have been distributed over a secured channel. In public-key case, each entity knows the public-key of the TTP. The TTP carefully verifies the identity of each entity, and signs a message consisting of an identifier and the entity's authentic public-key.

From a communication viewpoint, TTPs may be classified based on their real-time interactions with other communicating entities such as A and/or B.

1) In-line TTP: it is an intermediary TTP, serving as the real-time means of communication between A and B.

2) On-line TTP: it is a TTP involved in real-time communication during each protocol instance (communicating with A and/or B), but A and B communicate directly rather than through T.

3) Off-line TTP: it is a TTP not involved in real-time communication during a protocol instance, but T provides or prepares information a priori, which is available to A and/or B and the priori is used during protocol execution.

In-line third parties are of particular interest when A and B belong to different security domains or cannot otherwise interact directly due to non-interoperable security mechanisms. Examples of in-line third parties include KDCs or KTCs which provide the communication paths between A and B, as in IEEE 802.11i[3].

On-line third parties are widely used in key establishment protocols. Examples include Needham-Schroeder shared key protocol[4], Woo-Lam protocol[5] etc.

A typical example of an off-line third party is a certification authority (CA) producing public-key certificates and placing them in a public directory; here, the directory owner may be an on-line third party, but the certification authority is not. Protocols with off-line third parties usually involve fewer real-time message exchanges, and do not require real-time availability of the third parties.

Definition 2.41 A certificate is a TTP singed message, which binds the identity of an entity to its public-key. If the signing key (private key) of the TTP is compromised, all communications become insecure.

In this book, we suppose the honest entity TTP itself is unconditionally secure, while the TTP could be impersonated by an adversary with the requirement that the TTP machine itself has not been crashed.

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, in this book, we assume that the authenticity of this verification function is provided to B by noncryptographic means, for example by B obtaining it from the TTP in person.

Definition 2.42 Certificate Authority (CA) is a trusted third party who is responsible for establishing and vouching for the authenticity of the public-key bound to the subject entity. CA's responsibility includes binding public-keys to distinguished names through signed certificates, managing certificate serial numbers, and certificate revocation.

The authenticity of the CA's public-key may be originally provided by non-cryptographic means including personal acquisition, or through trusted couriers; authenticity of the CA's public-key is required, but not secrecy. This CA public-key allows any system user, through certificate acquisition and verification, to transitively acquire trust in the authenticity of the public-key in any certificate signed by that CA.

Definition 2.43 A public-key certificate is a data structure consisting of a data part and a signature part. The data part contains cleartext data including, as a minimum, a public-key and a string identifying the subject party to be associated therewith. The signature part consists of the digital signature of a certification authority over the data part, thereby binding the subject entity's identity to the specified public-key.

2.3.3 Time-variant parameters

Definition 2.44 A time-variant parameter (TVP), also called freshness component, is a value used no more than once for the same purpose. It typically serves to prevent (undetectable) replay. It is sometimes called nonce, unique number, or non-repeating value.

Often, to ensure protocol security, the integrity of such TVPs must be guaranteed (e.g., by cryptographically binding them with other data via a MAC or digital signature algorithm).

Uniqueness means to be unique, and it is often required only within a given key lifetime or time window.

A uniqueness guarantee may be provided by a TVP such as a timestamp or a never-repeated sequential counter. The never-repeated sequential counter may not provide (real-time) timeliness, and thus are not appropriate to entity authentication.

Definition 2.45 Timeliness (or freshness) typically means to be recent, it is in the sense of having originated subsequent to the beginning of the current protocol instance.

A timeliness guarantee may also be provided using TVPs such as timestamps. Note that timeliness alone does not rule out interleaving attacks using parallel sessions.

Three main classes of time-variant parameters are below:

1) Random number. Random numbers may be used to provide uniqueness and timeliness assurances, and to preclude certain replay and interleaving attacks on protocols. In a challenge-response mechanism, the challenger must temporarily maintain per-connection short-term state information, but only until the response is verified. The term nonce is most often used to refer to a "random" number in a challenge-response mechanism, but the required randomness properties vary.

In protocol descriptions, "choose a random number" is intended to express "pick a number with uniform distribution from a specified sample space" or "select from a uniform distribution".

2) Sequence number (or serial number, or counter value). A sequence number serves as a unique number identifying a message, and is typically used to detect message replays. Forced delays of messages with sequence number are not detectable in general. It may also be used to provide timeliness in conjunction with the maintenance of pairwise (sender, receiver) state information. As a consequence of the overhead and synchronization maintenance necessary, sequence numbers are most appropriate for smaller, and closed groups.

3) Timestamp. Timestamps may be used to provide timeliness and uniqueness guarantees, to detect message replays and forced delays. It provides timeliness in conjunction with distributed timeclocks. Timestamp-based protocols require that timeclocks be both loosely synchronized (fixing clock drift) and secured from modification.

2.3.4 Challenge and response

A challenge is typically a nonce chosen by one entity at the outset of the protocol and subsequent challenges will differ from each other.

Definition 2.46 A challenge-response protocol (or Challenge-response mechanism) is that one entity (the claimant, or the sender) "proves" its identity to another entity (the verifier, the receiver) by demonstrating one's possession of a secret to be associated with the claimant, without revealing the secret itself to the verifier during the protocol.

This is done by providing a response to a time-variant challenge, where the response depends on both the claimant's secret and the time-variant challenge. Answering a challenge in challenge-response protocols requires some type of computing device and secure storage for long-term keying material.

Challenge-response mechanisms may be implemented via symmetric-key techniques, public-key techniques, and zero-knowledge techniques. Examples of challenge-response protocols based on symmetric-key encryption are the Kerberos protocol[6] and the Needham-Schroeder shared-key protocol[4]. Challenge-response mechanisms may also be implemented by keyed hash functions which could provide data origin authentication security service similar to a symmetric-key block cipher MAC, e.g., IEEE 802.11i[3].

2.3.5 Other classes of cryptographic protocols

Besides the classification of cryptographic protocols in Chapter 1, there exist other classifications. For example, cryptographic protocols are classified by time-variant parameters (TVPs) used:

1) Challenge-response protocols: One entity includes a (new) time-variant challenge, mostly a random number, in an outgoing message, then other entity provides a response to this challenge in the next protocol message, where the response depends on both the entity's secret and the challenge. This protocol instance is then deemed to be fresh based on the reasoning that the random number links the two messages. Typically the protocol involves one additional message compared to timestamp-based protocols, and the challenger must temporarily maintain state information, but only until the response is verified.

2) Timestamp-based protocols: timestamps are in conjunction with distributed timeclocks in a timestamp-based cryptographic protocol. Timestamp-based protocols require that timeclocks be both synchronized and secured. Timestamps in protocols may typically be replaced by a random number challenge plus a return message.

3) Sequence number-based protocols: sequence numbers are in conjunction with the maintenance of pairwise (claimant, verifier) state information in

a sequence number-based protocol. Sequence number should be guaranteed to be increasing and unique.

In addition, classification of cryptographic protocols can be also based on the use of trusted third party (cryptographic protocols with trusted third party; cryptographic protocols without trusted third party), or based on the cryptosystems used (symmetric-key based protocols and public-key based protocols).

2.4 Security of cryptographic protocols

This section gives some background knowledge about the security of cryptographic protocols including attack models, security models, analysis methods for protocol security, etc.

2.4.1 Attacks on primitives

Some generic types of attacks on encryption schemes, signature schemes, message authentication code schemes are given below.

1. Classification of attacks on encryption schemes

The following are some generic types of attacks on encryption schemes:

1) Ciphertext-only attack. 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) Known-plaintext attack. The adversary tries to deduce the decryption key or plaintext in possession of a quantity of plaintext and corresponding ciphertext.

3) Chosen-plaintext attack. The adversary chooses plaintext and is then given corresponding ciphertext. Subsequently, the adversary tries to recover plaintext corresponding to previously unseen ciphertext by any information deduced.

4) Adaptive chosen-plaintext attack. It is a chosen-plaintext attack wherein the choice of plaintext may depend on the ciphertext received from previous requests.

5) Chosen-ciphertext attack. The adversary chooses ciphertext and is then given corresponding plaintext. Subsequently, the adversary tries to recover plaintexts corresponding to different ciphertext by any information deduced.

6) Adaptive chosen-ciphertext attack. It is a chosen-ciphertext attack where the choice of ciphertext may depend on the plaintext received from previous requests.

2. Classification of attacks on signature schemes

The objective of the attacker on signature schemes is to compute the private key information of the signer, or to forge signatures, that is, to produce signatures which will be accepted as those of some other entity. The following are some generic types of attacks on signature schemes:

1) Key-only attack. The adversary tries to produce signatures which will be accepted as those of some other entity while the adversary knows only the signer's public-key.

2) Known-message attack. The adversary tries to forge signatures in possession of a quantity of signatures for a set of messages which are known to the adversary but not chosen by him.

3) Chosen-message attack. The adversary obtains valid signatures from a chosen list of messages before attempting to break the signature scheme. This attack is non-adaptive in the sense that messages are chosen before any signatures are seen.

4) Adaptive chosen-message attack. The adversary uses the signer as an oracle, and he may even request signatures of messages which depend on previously obtained signatures or messages.

3. Classification of attacks on message authentication codes

$h_k(x)$ is the message authentication code given the input value key k and some text x. To attack a MAC means: given one or more pairs $(x_i, h_k(x_i))$, without prior knowledge of a key k, the adversary can compute a new text-MAC pair $(x, h_k(x))$ for some text $x \neq x_i$. The following are some generic types of attacks on message authentication codes:

1) Known-text attack. The adversary obtains one or more text-MAC pairs $(x_i, h_k(x_i))$.

2) Chosen-text attack. The adversary chooses x_i and obtains more text-MAC pairs $(x_i, h_k(x_i))$.

3) Adaptive chosen-text attack. The adversary may choose x_i to obtain new text-MAC pair $(x_i, h_k(x_i))$ based on the results of prior queries.

Some practical applications may limit the number of interactions allowed for text-MAC pair queries over a fixed period of time, or may be designed so as to compute MACs only for inputs created within the application itself; but it is practical to allow access to an unlimited number of text-MAC pairs, or to allow MAC verification of an unlimited number of messages and to accept any with a correct MAC for further processing.

2.4.2 Attacks on protocols

Definition 2.47 A protocol failure (or mechanism failure) occurs when a mechanism fails to meet the goals for which it is 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.

That is, the underlying primitive has not been compromised but the protocol has failed to provide the intended security service adequately.

An adversary in a key establishment protocol may pursue many strategies, including attempting to:

1) Deduce a session key using information gained by eavesdropping.

2) Participate covertly in a protocol initiated by one party with another, and influence it, e.g., by altering messages so as to be able to deduce the key.

3) Initiate one or more protocol executions (possibly simultaneously), and combine (interleave) messages from one with another, so as to masquerade as some party or carry out one of the above attacks.

4) Without being able to deduce the session key itself, deceive a legitimate party regarding the adversary as the identity of intended party with which it shares a key.

In a unauthenticated key establishment protocol, impersonation is usually possible. In an entity authentication protocol, where there is no session key to attack, an adversary's objective is to deceive a legitimate party to believe that the protocol has been run successfully with a party other than the adversary.

The following are some generic types of attacks on cryptographic protocols:

1) Known-key attack. The adversary obtains some keys used previously and then uses this information to determine new keys.

2) Replay. The adversary records a communication session and replays the entire session, or a portion thereof, at some later point in time.

3) Impersonation. The adversary personates 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 with the stored value. An adversary can take a list of probable passwords, hash all entries in this list, and then compare this with the list of true encrypted passwords in file 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.

6) Interleaving attack. This type of attack usually involves parallel sessions and impersonation in an authentication protocol.

2.4.3 Security of protocols

It is typically assumed that protocol messages are transmitted over unsecured channels and the adversary has complete control of the data therein, with the ability to record, alter, delete, insert, redirect, and reuse the past or current messages, and inject new messages. In general a protocol has the following properties:

1) Operational property. In the absence of active adversaries and communication errors, honest participants who comply with its specification always complete the protocol. For example, for a key establishment protocol, the honest participants always compute a common key and corroborate the identities of the parties with whom the key is shared.

2) Completeness property. In the absence of active adversaries and communication errors, a protocol is complete if, given an honest claimant and an honest verifier, the protocol succeeds with overwhelming probability (i.e., the verifier accepts the claimant's claim or they both have the knowledge of the new session key). The definition of overwhelming generally implies that the probability of failure is not of practical significance.

3) Soundness property. An interactive proof of a protocol is sound if there exists an expected polynomial-time algorithm M with the following property: if a dishonest prover (impersonating A) can, with non-negligible probability, successfully execute the protocol with B, then M can be used to extract A's secret from this prover's knowledge with overwhelming probability.

Suppose all protocols are operational and complete in this book, which is the basic correctness requirement of protocols.

1. Attack models

The following are some types of attack models:

1) Passive attack. The adversary is passive when facing a ciphertext, i.e., all that the adversary could do about a ciphertext is eavesdropping.

2) Indistinguishable chosen-plaintext attack (IND-CPA). In this attack model, the adversary is allowed to obtain an assistance in the encryption mode to break the target cryptosystems.

3) Indistinguishable chosen-ciphertext attack (IND-CCA). In this attack model, the adversary is allowed to obtain a conditional assistance in the decryption mode to break the target cryptosystems. Other synonymous names are lunchtime attack, midnight attack or indifferent chosen-ciphertext attack.

4) Indistinguishable adaptive chosen-ciphertext attack (IND-CCA2). In this attack model, the adversary is allowed to obtain an assistance in the decryption mode to break the target cryptosystems. Other synonymous names are small-hours attack.

2. Security models

Definition 2.48 (All-or-nothing security) For a given ciphertext output from a given encryption algorithm, the adversary either succeeds with obtaining the whole block of the targeted secret, or fails with nothing. Here "all" means to find the whole plaintext block which in general has a size stipulated by a security parameter of the cryptosystem; "nothing" means that the adversary does not have any knowledge about the targeted secret before or after its attacking attempt.

All-or-nothing secrecy is unfit for the real world since the guarantee of the secrecy is valid only if the attacker is passive, i.e., all that the attacker could do about a ciphertext is eavesdropping. Hence, if a cryptosystem is all-or-nothing secure, then it is a "textbook crypto". Numerous attacks have been discovered in practice on the textbook cryptos.

Definition 2.49 (Semantic security, IND-CPA security) A cryptosystem with a security parameter k is said to be semantically secure: after the IND-CPA attack game being played with any polynomially bounded adversary, the advantage *Adv* for the adversary Malice to distinguish the two plaintexts chosen by the adversary is negligible. The semantic security is also called the security for indistinguishable chosen-plaintext attack, for short IND-CPA security.

Informally speaking, semantic security means that whatever is efficiently computable about the plaintext given the ciphertext, is also efficiently computable without the ciphertext. The notion of the IND-CPA security captures the intuition that any polynomially bounded adversary should not be able to obtain any apriori information about a plaintext.

Definition 2.50 (IND-CCA security) A cryptosystem with a security parameter k is said to be secure against an indistinguishable chosen-ciphertext attack (IND-CCA security): after the IND-CCA attack game being played with any polynomially bounded adversary, the advantage *Adv* for the adversary Malice to distinguish the two plaintexts chosen by the adversary is negligible.

Lunchtime attack is a quite restrictive attack model in that the decryption assistance provided to Malice is only available in a short period of time, which is not a reasonable or realistic scenario. In reality, naive users will remain permanently naive, and Malice will definitely strike back, probably even in the afternoon tea-break time[2]!

Definition 2.51 (IND-CCA2 security) A cryptosystem with a security parameter k is said to be secure against an indistinguishable adaptive chosen-ciphertext attack (IND-CCA2 security): after the CCA2 attack game being played with any polynomially bounded adversary, the advantage *Adv* for the adversary Malice to distinguish the two plaintexts chosen by the adversary is negligible.

Most cryptosystems which are IND-CPA secure may be particularly vulnerable in IND-CCA (or IND-CCA2) model. In CCA and CCA2 models, an adversary (now he is Malice) may get decryption assistance, that is, he may be in a certain level of control of a "decryption box" and so may have some ciphertext of his choice to be decrypted for him even though he does not have possession of the decryption key. Such an assistance is treated as a "cryptanalysis training course" provided to Malice in order to ease his attack job. These modes of attacks, particularly CCA2 model, are realistic in many applications of public-key cryptography[7]. Nowadays, IND-CCA2 is becoming the standard and fit-for-application security notion for public-key cryptosystems[2]. New public-key encryption schemes need to have this security quality for general purpose applications in real world setting.

2.4.4 Analysis methods for protocol security

We identify two distinct approaches for analyzing cryptographic protocols: Informal approaches and formal approaches (or formalisms). Formal approaches are a natural extension to informal ones and they are more important in protocol security analysis field. The following are some types of analysis methods for protocol security[1, 2, 7, 8].

1. Ad hoc and practical analysis

It is also called heuristic security. Protocols are typically designed to counter standard attacks, and shown to follow accepted principles. This approach, perhaps, is the most commonly used and practical one, but it may provide least satisfying of protocol security. Claims of security in this class generally remain questionable, and unforeseen attacks remain a threat.

2. Complexity-theoretic analysis

It is also called computational security or computationally secure. An appropriate model of computation is defined, and adversaries are modeled as having polynomial computational power (they may mount attacks involving time and space polynomial in the size of appropriate security parameters). Security analysis of this mathematical type helps a protocol designer or analyzer to consider using correct or more precise cryptographic services, and so protocol flaws can be avoided.

Provable security, also called provably secure, may be considered as part of a special sub-class of the computational security. Provable Security is a formal method for proving the security of cryptographic schemes, in which the difficulty of breaking a particular scheme is formally related to that of solving a widely believed computational hard problem, such as integer factorization or the computation of discrete logarithms. Random oracle is a very powerful and imaginary hash function with the "mixing-transformation" property: for any input, the distribution of the output hash values is uniform in the function's output space. In provable security, random oracle is used to construct public-key encryption schemes out of using the basic and popular public-key cryptographic primitives.

Provable security is the most commonly used analysis method in computational security. Hence, in this book, we typically use provable security to refer to computational security. It is often required for a scheme to be secure in this class, and most of the best known public-key and symmetric key schemes in current use are in it.

3. Information-theoretic analysis

This approach uses mathematical proofs involving entropy relationships to prove that protocols are unconditionally secure. 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 also called perfect secrecy. While unconditional security is ultimately desirable, this approach is not applicable to most practical schemes. This approach cannot be combined with computational complexity arguments because it allows unlimited computation, while computational complexity requires that the adversaries should only have polynomial computational power.

4. Symbolic manipulation analysis

It is also called formal methods, verification methods or formalisms. This approach uses a set of abstract symbols to express security properties and these abstract symbols can be manipulated. The so-called approaches include formal logic systems, term re-writing systems, expert systems, and various other methods which combine algebraic and state-transition techniques.

On one hand, symbolic formal analysis methods are simple but have proven to be of utility in finding flaws and redundancies in protocols, and some are automatable to varying degrees. On the other hand, the "proofs" provided are proofs within the specified formal system, and cannot be interpreted as absolute proofs of security. For example, the security of symbolic view regards an encryption as a deterministic function. The foundations for formal cryptology need to be strengthened.

2.5 Communication threat model

This section gives a brief introduction to the communication threat model.

2.5.1 Dolev-Yao threat model

Dolev and Yao propose a communication threat model[9], which has been widely accepted as the standard threat model for cryptographic protocols.

The Dolev-Yao threat model supposes that Malice, the attacker, controls the entire communication network, so Malice is able to observe all message traffic over the network, to intercept, read, modify or destroy messages. Further more, Malice may perform transformation operations on the intercepted messages (such as encryption or decryption as long as he has in his possession of the correct keys), and send his messages to other principals by masquerading as some principal.

The Dolev-Yao threat model requires very few quantity assumptions on the behavior of the adversary. Here are the basic assumptions for the protocol environment in Dolev-Yao threat model.

1) In a perfect public-key system, as long as

— the one-way functions used are unbreakable;
— the public directory, including all the text $m \in M$ and its corresponding ciphertext $E_e(m)$ pairs, is secure and cannot be tampered with;
— everyone has access to the encryptions under public-key e;
— only the public-key owner has the corresponding private key d.

2) In a two-party protocol, only the two users who wish to communicate are involved in the transmission process; the assistance of a third party in decryption or encryption is not needed.

3) In a uniform protocol, the same format is used by every pair of users who wish to communicate.

4) The adversaries are "active" eavesdroppers: someone who first taps the communication line to obtain messages will try everything he can to discover the plaintext. More precisely, the followings are assumed:

— He can obtain any message passing through the network.
— He is a legitimate user of the network, and thus particularly can initiate a conversation with any other user.
— He will have the opportunity to become a receiver to any user.
— He can send messages to any principal by impersonating any other principal.

Recall that Malice only has polynomial computational power, so he has the following characteristics:

— Malice cannot guess a random number which is chosen from a sufficiently large space.
— Without the correct secret (or private) key, Malice cannot retrieve plaintext from given ciphertext, and cannot create valid ciphertext from given plaintext.
— Malice is not in control of many private areas of the computing environment, such as accessing the memory of a entity's offline computing device.

2.5.2 Assumptions of protocol environment

The Dolev-Yao-like threat model will be applied to the protocol analysis in this book, where Malice is assumed to have the entire control of the vulnerable network and his computational capability is polynomially bounded.

To clarify the threats that cryptographic protocols may be subject to, and to motivate the need for specific protocol characteristics, precise assumptions of protocol environment in our protocol analysis are given.

1. Assumptions of cryptosystems

1) Suppose cryptographic primitives are perfect. That is, when examining the security of protocols, it is assumed that the underlying cryptographic mechanisms used, such as encryption algorithms and digital signatures schemes, are secure in the protocol run. An adversary is hypothesized to be not a cryptanalyst attacking the underlying primitives such as encryption algorithms directly, or rather the one attempting to subvert the protocol objectives by manipulating the protocol or mechanism itself.

Suppose cryptographic primitives are unbreakable in protocol analysis, so an adversary gains advantages not by breaking an underlying primitive such as an encryption algorithm directly, but by manipulating the protocol or mechanism itself.

Recall that security for indistinguishable adaptive chosen-ciphertext attack (IND-CCA2) is fit-for-application in a practical cryptographic setting, and it can be mathematically specified and proved independent of qualifying assumptions. Hence, we suppose our "perfect" cryptographic primitives are practical primitives which are IND-CCA2 secure. Namely, under IND-CCA2, the failures in a cryptographic protocol are not in any way related to the strength or weakness of the primitive used, but related to the protocol logic flaws.

2) The secret (or private) key d of the particular key pair (e, d), in which the communication parties are being used, should be kept secret while communicating securely. Without the correct private key, Malice cannot be able to retrieve plaintext from given ciphertext, and cannot create valid ciphertext from given plaintext.

History has shown that maintaining the secrecy, the secret (or private) key, of the encryption and decryption transformations is very difficult indeed.

3) Suppose that a legitimate party is either totally corrupted or totally secure. That is, the secrecy or the local state for a session during execution is either totally known by the adversary or totally not.

In practice, it is a common case that the adversary finds part of the local state or other information, and then he can deduce the decryption key or recover the desired plaintext. However, we omit this information exposure case for it is out of the scope of this book: analyzing the security of the protocol itself.

4) Suppose that each party has his own private key and the public-keys of other parties (including the adversary) in public-key case; Suppose that each party shares long-term keys with co-operative principals or the trusted third party in symmetric-key case. Furthermore, private keys and shared keys are commonly assumed to be too long to guess in a computationally feasible way.

In common, public-keys or long-term keys are deployed safely before the protocol run via authenticated channel or even traditional communication means. However, in some special cases, public-keys or long-term keys may also be transmitted in a cryptographic protocol.

5) Malice cannot guess a random number which is chosen from a sufficiently large space.

2. Assumptions of receiving messages

The receiving information of a participant is referred to as bit string. Further decision should be made to determine whether this bit string is a message or not. Suppose each entity can distinguish the sentence structure of a message from random bit strings and has the ability to recognize a basic message.

3. Assumptions of participants

Suppose each participant, given an honest claimant and an honest verifier, will comply with the determined protocol steps, and the protocol will succeed with overwhelming probability (i.e., for key establishment protocol, the verifier accepts the claimant's claim or they both have the knowledge of the new session key).

4. Assumptions of encryption and decryption

Suppose the encryption and decryption manipulation on a message are inverse functions, that is, they obey a set of term-rewriting rules. For example, $D_d(E_e(m)) = m$ for all $m \in M$, where public-key e and private key d are paired keys in a public-key system, and symmetric keys e and d are equal keys ($e = d$) in a symmetric-key system.

5. Assumptions of adversary

In general, it is assumed that Malice is very clever in manipulating communications over the open network. His manipulation techniques are unpredictable because they are unspecified.

Malice is able to observe all message traffic over the network, to intercept, read, modify or destroy messages at will, to perform transformation operations on the intercepted messages, and to send his messages to other principals by masquerading as some principal. However, Malice's computational capability is polynomially bounded, therefore there is a set of "words" that Malice does not know naturally at the beginning of a protocol run, this set of words can be secret messages or cryptographic keys for which a protocol is meant to protect.

Malice represents a coalition of bad guys and thereby may use a large number of computers across the open network in parallel. Hence, an adversary usually means a group of attackers in this literature.

Furthermore, suppose the attacker can perform a kind of cryptanalysis training course that helps him to obtain a conditional assistance, in the decryption mode or encryption mode, and makes him more experienced in the future.

2.5.3 Expressions of cryptographic protocols

We adopt the formal notation expressions in the Dolev-Yao-like threat model, which distinguish the underlying primitives from the cryptographic protocols explicitly and the security of a cryptographic protocol is discussed under a "perfect" primitive.

One message procedure step usually is

$$\text{Message 2} \quad A \rightarrow B \quad \{m\}_{K_{AB}}.$$

"Message 2" means that this is the second message exchanged in a protocol. "$A \rightarrow B : \{m\}_{K_{AB}}$" indicates that the protocol participant A has sent the message "$\{m\}_{K_{AB}}$" to the opponent protocol participant B. "$\rightarrow$" shows that the message is sent from the arrow end terminal A to the arrow top terminal B. m is a text where $m \in M$, K_{AB} is a long-term key between the principals A and B. $\{m\}_{K_{AB}}$ is an encryption of m under key K_{AB}.

$I(A)$ (or $I(B)$) is the adversary Malice I impersonating A (or B respectively).

References

[1] Menezes A, van Oorschot P, Vanstone S (1996) Handbook of Applied Cryptography. CRC Press, New York

[2] Mao W (2004) Modern Cryptography: Theory and Practice. Prentice Hall, New Jersey

[3] IEEE Std 802.11i-2004. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Security Enhancements. July 2004

[4] Needham RM, Schroeder MD (1978) Using Encryption for Authentication in Large Network of Computers. Communication of the ACM 21(12): 993 – 999

[5] Woo TYC, Lam SS (1992) Authentication for Distributed Systems. Computer 25(1): 39 – 52

[6] Miller SP, Neuman BC, Schiller JI, Saltzer JH (1987) Kerberos Authentication and Authorization System. Paper Presented at the Project Athena Technical Plan Section E.2.1. MIT, Boston

[7] Stallings W (2006) Cryptography and Network Security: Principles and Practice, 4th edn. Prentice Hall, New Jersey

[8] Goldreich O (2003) Foundations of Cryptography. Cambridge University Press, New York

[9] Dolev D, Yao AC (1983) On the Security of Public-key Protocols. IEEE Transactions on Information Theory 29(2): 198 – 208