# 6 Guarantee of Cryptographic Protocol Security

**Abstract** Some important provable security notions like indistinguishability, match conversation, authentication, etc. are briefly reviewed. The security definitions of UA-Secure, MA-Secure, UK-Secure and MK-Secure are specified based on the trusted freshness principle, and these formalization specifications are proved to be adequate for the intended security goals.

As we have witnessed in cryptographic literature, cryptographic protocols are notoriously error-prone. These protocols can be flawed in very subtle ways. It is widely agreed by researchers with different backgrounds that formal methods should be taken into the security analysis of cryptographic protocols.

The methodology provable security is introduced where the security could be proved under "standard" and well-believed complexity theoretic assumptions (e.g., the assumed intractability of factoring). The provable security method often entails providing (i) a definition of the security goal, (ii) a protocol, and (iii) a proof that the protocol meets its goal, assuming some standard complexity-theoretic assumption holds true.It is the opinion of many researchers that provable security should be in hand for all of the "basic" cryptographic primitives[1, 2].

In provable security field, some novel definitional ideas are achieved: Goldwasser, Micali et al. have suggested probabilistic encryption[3] and digital signatures[4], Blum – Micali and Yao suggested pseudorandom number generation[5, 6], Bellare, Rogaway et al. suggested authentication[2, 7].

The security goals discussed in this book involve unilateral entity authentication secure, mutual entity authentication secure, unilateral authenticated key secure and mutual authenticated key secure. The question of whether the security properties of a cryptographic protocol are adequate for a security goal or not will be answered in this chapter. Particularly we try to raise the security specification guarantees of unilateral entity authentication secure, mutual entity authentication secure, unilateral authenticated key secure and mutual authenticated key secure, similar to those primitives such as encryptions, pseudorandom generators, or digital signatures.

In this chapter, some important provable security notions like indistin-

L. Dong et al., *Cryptographic Protocol*

guishability, authentication, etc., will be briefly reviewed first. Then, we try to formalize the security goals – UA-Secure, MA-Secure, UK-Secure and MK-Secure – of cryptographic protocols based on the trusted freshness principle, and prove that the formalization specifications are adequate for the intended security goals. In deed, the latter security goal MK-Secure, which is well known to applied cryptographers, is very useful to build secure distributed system.

## 6.1 Security definition of authentication

Bellare and Rogaway are the first researchers who propose a computational model for the security of authentication and authenticated key establishment protocols[1]. The idea of the definition of a mutual authentication in this model is simple but strong: any adversary effectively behaves as a trusted wire, if not a broken one. This simple idea is formalized via a notion of matching conversations. The idea of the definition of an authenticated key exchange is to keep the session key remaining protected. The adversary's inability is formalized to gain any helpful information about the session key along the lines of formalizations of security for probabilistic encryption[1, 7].

Four protocols are specifically discussed in the Bellare and Rogaway's model. Protocol MAP1 is a mutual authentication protocol for an arbitrary set of parties. Protocol MAP2 is an extension of MAP1, allowing arbitrary text strings to be authenticated along with its flows. Protocol AKEP1 is a simple authenticated key exchange which uses MAP2 to do the key distribution. Protocol AKEP2 is a particularly efficient authenticated key exchange which introduces the idea of "implicitly" distributing a key; its flows are identical to MAP1, but it accomplishes a key distribution all the same. The primitive required for all of these protocols is a pseudorandom function.

In practice, the pseudo-random function can be practically realized by a message authentication code in cipher-block-chaining mode of operation (CBC-MAC) or by a keyed cryptographic hash function (HMAC). The proof of the security of authentication and authenticated key establishment protocols in the Bellare and Rogaway's model leads from an alleged successful attack on a protocol to the collapse of pseudo-randomness, i.e., the output of a pseudo-random function can be distinguished from that of a truly random function by a polynomial-time distinguisher, in the proof of the adversary; in other words, the existence of pseudo-random functions is denied. This implies that the result of the reduction should be either false or a major breakthrough in the foundations for modern cryptography. As the former is more likely the case, the reduction derives a contradiction as desired[8].

### 6.1.1 Formal modeling of protocols

The protocols considered in Bellare and Rogaway's model are two party ones, formally specified by an efficiently computable function, a polynomial-time function $\Pi$ on the following input values:

$1^k$: the security parameter $k$, $k \in \mathbb{N}$ where $\mathbb{N}$ is a set of natural numbers.

$i$: the identity of the sender ranging over $I$, $i \in I \subseteq \{0,1\}^k$ where $I$ is a set of principals who can participate in the protocol and share a secret long-term key, and $\{0,1\}^k$ denotes the set of finite binary strings of length at most $k$.

$j$: the identity of the (intended) communication partner of the sender, the receiver ranging over $I$. Elements of $I$ will sometimes be denoted as $A$ and $B$ (or Alice and Bob), rather than $i$ and $j$. Note that the adversary is not a partner in the Bellare-Rogaway model and $A = B$ (or $i = j$) is quite possible.

$K$: the long-term symmetric key shared between the sender $i$ and the receiver $j$.

*conv*: the conversation so far, $conv \in \{0,1\}^*$ where $\{0,1\}^*$ denotes the set of finite binary strings. *conv* grows with the protocol run; new string is concatenated to it.

$r$: the random coin inputs of the sender like a nonce generated by the sender.

The function of $\Pi(1^k, i, j, K, conv, r)$ implies that $K, r$ is of size $k$, and $i, j, conv$ is of size polynomial in $k$. The value of $\Pi(1^k, i, j, K, conv, r) = (m, \delta, \alpha)$ specifies:

$m$: the next message to send out, $m \in \{0,1\}^* \cup \{$"no message output"$\}$.

$\alpha$: the privte output, $\alpha \in \{0,1\}^* \cup \{$"no private output"$\}$.

$\delta$: the decision for the sender, $\delta \in \{$Accept, Reject, Undetermined$\}$. An acceptance decision usually does not occur until the end of the protocol, although a rejection decision may occur at any time. For mutual authentication protocol, only acceptance or rejection decision is concerned with. For key exchange protocols, the private output, such as an agreed session key, is concerned with. Once a decision other than "undetermined" is reached, the private output will no longer change.

Oracle $\Pi^s_{i,j}$ models partner $i$ attempting to authenticate partner $j$ in a "session" $s$ for $i, j \in I$ and $s \in \mathbb{N}$.

### 6.1.2 Formal modeling of communications

In Bellare-Rogaway model, all communications among interacting parties are assumed to be under the adversary $I$'s control. Particularly, the adversary can read the messages produced by the parties, provide messages of his own to them, modify messages before they reach their destination, and delay mes-

sages or replay them. Most importantly, the adversary can start up entirely new "instances" of any of the parties, modeling the ability of communicating agents to simultaneously engage in many sessions at once. Formally the adversary $I$ is a probabilistic machine equipped with an infinite collection of oracles $\Pi^s_{i,j}$, $I$ can conduct as many sessions as $I$ pleases among the honest partners, and $I$ can persuade a partner to start a protocol run as if it is run with another honest partner. Each honest party will be modeled by an infinite collection of oracles which the adversary may run. These oracles only interact with the adversary, they never directly interact with one another[7].

*Query*

A query is of the form $(i, j, s, x)$ which means that the adversary $I$ is sending message $x$ to $i$, and the adversary $I$ claims that it is from $j$ in session $s$. A query from $I$ will be answered by an oracle $\Pi^s_{i,j}$. In response to an oracle call, $I$ learns not only the outgoing message but also whether or not the oracle has accepted, but $I$ couldn't learn the oracle's private output. In a particular execution of a protocol, the adversary's $i$-th query to an oracle is said to occur at time $\tau = \tau_i \in \mathbb{R}$ where $\mathbb{R}$ is a set of reals.

*The Benign Adversary*

An adversary is called benign if it is deterministic and it restricts its action to choosing a pair of oracles $\Pi^s_{i,j}$ and $\Pi^s_{j,i}$ and then faithfully conveying each flow from one oracle to the other, with $\Pi^s_{i,j}$ beginning first. In other words, the first query $I$ makes is $(i, j, \tau_1, \texttt{""})$, generating response $m_1$; the second query $I$ makes is $(j, i, \tau_2, m_1)$, generating response $m'_1$, and so forth. While the choice of $i, j, \tau_1, \tau_2$ is up to the adversary, this choice is the same in all executions with security parameter $k$. Therefore, a benign adversary behaves just like a wire between $i$ and $j$. If the adversary wished to have the targeted partners to output the acceptance decision, the adversary's behavior should be restricted to that of a benign adversary.

*Time*

Conforming notions of time include "abstract time", where $\tau_i = i$, and "Turing machine time", where $\tau_i =$ the $i$-th step in $I$'s computation, when parties are realized by interacting Turing machines. Another conforming notion of time (but a harder one to formalize) is "real time", where $\tau_i$ is the exact time when the $i$-th query is made, when parties are realized by interacting computers. For the $i$-th query and the $j$-th query, if $i < j$, we demand that $\tau_i < \tau_j$.

### 6.1.3 Formal modeling of entity authentication

A central notion in formalizing entity authentication goals is that of a matching conversation.

Bellare and Rogaway defined authenticity security as the matching conversation by an experiment involving the running of the adversary $I$ with security parameter $k$. When $I$ terminates, each oracle $\Pi_{i,j}^{s}$ has had a certain conversation $conv_{i,j}^{s}$ with $I$, and it has reached a certain decision $\delta \in \{\text{Accept, Reject, Undetermined}\}$.

*Conversation*

A conversation of oracle $\Pi_{i,j}^{s}$ is a sequence of timely ordered messages that a partner sent out (respectively, received), and as consequent responses, received (respectively, sent). Let $\tau_1 < \tau_2 < \cdots \tau_n$ be a time sequence, for any oracle $\Pi_{i,j}^{s}$, the conversation (for this execution) can be denoted by the following sequence:

$$conv = (\tau_1, m_1, m_1'), (\tau_2, m_2, m_2'), \cdots, (\tau_n, m_n, m_n').$$

This sequence encodes that at time $\tau_1$, the participant was asked $m_1$ and responded with $m_1'$, and then, at some later time $\tau_2 > \tau_1$, oracle $\Pi_{i,j}^{s}$ was asked $m_2$, and answered $m_2'$; and so forth, until, finally, at time $\tau_n$ it was asked $m_n$, and answered $m_n'$. Adversary $I$ terminates without asking oracle $\Pi_{i,j}^{s}$ any more questions.

Suppose oracle $\Pi_{i,j}^{s}$ has conversation prefixed by $(\tau_1, m_1, m_1')$. Then if $m_1 =$"", we call oracle $\Pi_{i,j}^{s}$ an *initiator oracle*; if $m_1$ is any other string, we call $\Pi_{i,j}^{s}$ a *responder oracle*. If $m_n =$"no message output", $\Pi_{i,j}^{s}$ ends the conversation. At the end of a protocol run, each participant makes a decision about the authentication of the intended partner: accept, reject, or undetermined.

*Matching conversation*

Give a protocol $\Pi$ between partners $i$ and $j$. Run $\Pi$ in the presence of a benign adversary $I$ and consider two oracles $\Pi_{i,j}^{s}$ and $\Pi_{j,i}^{s'}$ that engage in conversations *conv* and *conv*$'$ in sessions $s$ and $s'$, respectively.

1) We say that *conv*$'$ is a matching conversation to *conv* if there exist time sequences $\tau_0 < \tau_1 < \tau_2 < \cdots < \tau_n$ and $m_1, m_1', m_2, m_2', m_3, \cdots, m_{t-1}', m_t, m_t'$ so that *conv* is prefixed by

$$conv = (\tau_0, \text{""}, m_1), (\tau_2, m_1', m_2), (\tau_4, m_2', m_3), \cdots, \left(\tau_{2t-2}, m_{t-1}', m_t\right)$$

and *conv*$'$ is prefixed by

$$conv' = (\tau_1, m_1, m_1'), (\tau_3, m_2, m_2'), (\tau_5, m_3, m_3'), \cdots, \left(\tau_{2t-3}, m_{t-1}, m_{t-1}'\right).$$

2) We say that *conv* is a matching conversation to *conv*$'$ if there exists time sequence $\tau_0 < \tau_1 < \tau_2 < \cdots \tau_n$ and $m_1, m_1', m_2, m_2', m_3, \cdots, m_{t-1}', m_t, m_t'$ so that *conv*$'$ is prefixed by

$$conv' = (\tau_1, m_1, m_1'), (\tau_3, m_2, m_2'), (\tau_5, m_3, m_3'), \cdots, (\tau_{2t-1}, m_t, *)$$

and *conv* is prefixed by

$$conv = (\tau_0, \text{""}, m_1), (\tau_2, m_1', m_2), (\tau_4, m_2', m_3), \cdots, (\tau_{2t-2}, m_{t-1}', m_t).$$

Explanation. Case (1) defines when the conversation of a responder oracle matches the conversation of an initiator oracle. Case (2) defines when the conversation of an initiator oracle matches the conversation of a responder oracle.

Consider an execution in which $\Pi_{i,j}^{s}$ is an initiator oracle and $\Pi_{j,i}^{s'}$ is a responder oracle. If every message that $\Pi_{i,j}^{s}$ sends out, except possibly the last, is subsequently delivered to $\Pi_{j,i}^{s'}$, with the response to this message being returned to $\Pi_{i,j}^{s}$ as its own next message, then we say that the conversation of $\Pi_{j,i}^{s'}$ matches that of $\Pi_{i,j}^{s}$. Similarly, if every message that $\Pi_{j,i}^{s'}$ receives was previously generated by $\Pi_{i,j}^{s}$, and each message that $\Pi_{j,i}^{s'}$ sends out is subsequently delivered to $\Pi_{i,j}^{s}$, with the response that this message generates being returned to $\Pi_{j,i}^{s'}$ as its own next message, then it is said that the conversation of $\Pi_{i,j}^{s}$ matches the one of $\Pi_{j,i}^{s'}$.

It is said that oracle $\Pi_{j,i}^{s}$ has a matching conversation with oracle $\Pi_{i,j}^{s}$ if $\Pi_{j,i}^{s}$ has conversation $conv'$, $\Pi_{i,j}^{s}$ has conversation *conv*, meanwhile $conv'$ matches *conv*.

*Entity Authentication*

Any mutual authentication protocol must have at least 3 round message exchanges. The definition of a mutual authentication is similar to [1] as follows:

**Definition 6.1** (Secure mutual authentication) Give a protocol $\Pi$ between partners $A$ and $B$. We say that $\Pi$ is a secure mutual authentication protocol in the presence of any polynomial time adversary $I$ if

1) (Matching conversations $\Rightarrow$ acceptance.) The oracles $\Pi_{A,B}^{s}$ and $\Pi_{B,A}^{s'}$ have matching conversations, then both oracles accept.

2) (Acceptance $\Rightarrow$ matching conversations.) The adversary $I$ cannot win with a non-negligible probability in $k$. Here the adversary wins if the oracles $\Pi_{A,B}^{s}$ and $\Pi_{B,A}^{s'}$ both reach the "accept" decision while they do not have matching conversations in oracles.

Explanation. The first condition says that if one party's messages are faithfully relayed to one another, then the party accepts the authentication of one another. The second condition says that if oracles $\Pi_{A,B}^{s}$ and $\Pi_{B,A}^{s'}$ have reached the "accept" decision, then $\Pi_{A,B}^{s}$ or $\Pi_{B,A}^{s'}$ must have a matching conversation in both oracles.

Note that an oracle's matching partner is unique based on the Definition 6.1. More formally, let $\textbf{Multiple-Match}^{E}(k)$ be the event that some $\Pi_{A,B}^{s}$ accepts in the presence of any polynomial time adversary $I$, and there are at least two distinct oracles $\Pi_{B,A}^{s'}$ and $\Pi_{B,A}^{s''}$ which have had matching conversations with $\Pi_{A,B}^{s}$.

**Proposition 6.1** Suppose the protocol $\Pi$ between partners $A$ and $B$ is a secure mutual authentication protocol. Let $I$ be any polynomial time adversary. Then the probability of Multiple-Match$^E(k)$ is negligible.

The probability of Multiple-Match$^E(k)$ is at most $l^2 * 2^{-k}$ where $l$ is the (polynomial) number of oracle calls of the adversary $I$ and $k$ is the security parameter.

Bellare and Rogaway demonstrate their formal proof technique by providing a simple mutual entity authentication protocol named MAP1 and conducting its proof of security. They also consider the correctness for authenticated key establishment protocols. For more details, please refer to references [1, 7].

## 6.2 Security definition of SK-security

Bellare and Rogaway's idea of provable security under a computational model originates from the seminal work of Goldwasser and Micali[3]. Goldwasser and Micali propose a well-known public-key probabilistic encryption scheme named the Goldwasser-Micali cryptosystem, GM cryptosystem, which possesses the property of semantic security assuming the intractability of the quadratic residuosity problem. The semantic security is a stronger security notion: Whatever is efficiently computable about the plaintext given the ciphertext, is also efficiently computable without the ciphertext[8].

The semantic security, which is also known as IND-CPA security or polynomial-time indistinguishability, means that a ciphertext does not leak any useful information about the plaintext to any attacker whose computational power is polynomially bounded. They observed that in many applications, messages may contain certain apriori information which may be useful for an attack. Goldwasser and Micali point out that public-key cryptosystems which are based on direct applications of one-way trapdoor functions are in general very weak for hiding such messages[3, 8]. The semantic security notion tries to meet the need for a general fix of this much bigger problem.

There, a security property (one of several confidentiality qualities) is argued under a given attacking scenario (one of several attacking games each of which models, with sufficient generality and precision, one of some typical behavior of a real-world attacker against public-key encryption schemes). A proof of security for public-key encryption schemes with respect to an alleged attack involves demonstrating an efficient transformation (called a polynomial-time reduction) leading from the alleged attack to a major breakthrough to a well-believed hard problem in computational complexity. It is the wide belief on the unlikelihood of the major breakthrough that should refute the existence of the alleged attack, that is, a proof is given by contradiction[8].

The Goldwasser-Micali scheme[3] can be described in a general setting by using the notion of a trapdoor predicate. Briefly, a trapdoor predicate is

a Boolean function $B : \{0,1\}^* \rightarrow \{0,1\}$ so that, given a bit $v$, it is easy to choose an $x$ at random satisfying $B(x) = v$. Moreover, given a bitstring $x$, computing $B(x)$ correctly with probability significantly greater than $\frac{1}{2}$ is difficult; however, if certain trapdoor information is known, then it is easy to compute $B(x)$. Suppose an entity $A$'s public-key is a trapdoor predicate $B$, any other entity encrypts a message bit $m_i$ by randomly selecting an $x_i$ so that $B(x_i) = m_i$, and then sends $x_i$ to $A$. Since $A$ knows the trapdoor information, $A$ can compute $B(x_i)$ to recover $m_i$, but an adversary can do no better than guess the value of $m_i$. Goldwasser and Micali proved that if trapdoor predicates exist, then this probabilistic encryption scheme is polynomially secure[3, 9].

In the Bellare-Rogaway model, they also consider the correctness for authenticated session key establishment (session key transport) protocols (for the two-party case[1], also for the three-party case which uses a trusted third party as an authentication server[7]). The security notion of key establishment originates from the Goldwasser-Micali probabilistic encryption[3]. For key establishment protocols, "Malice wins" means a successful guess of the new session key. Since the new session key is randomly chosen by a pseudo-random function, and the transported key is encrypted under the shared long-term key, successful guessing of the session key is similarly hard as making distinction between a pseudo-random function and a truly random function.

SK-Security notation is an important notion in the authentication protocol field. Canetti and Krawczyk put forward the CK model[10], including the SK-Security notation, for the analysis of key establishment protocols that results from the combination of two previous works in this area: [1] by Bellare and Rogaway and [2] by Bellare, Canetti and Krawczyk.

### 6.2.1 Protocol and adversary models in CK model

Canetti and Krawczyk extend and refine the formalism in the approach of [10], where a general framework for studying the security of session-based multi-party protocols over insecure channels is introduced. Let's review the CK model.

#### 6.2.1.1 Protocol notations

$P_1, P_2, ..., P_n$: a set of parties (probabilistic polynomial-time machines) interconnected by point-to-point links over which messages can be exchanged.

Protocols: collections of interactive procedures, run concurrently by parties $P_1, P_2, ..., P_n$, which specify a particular processing of incoming messages and the generation of outgoing messages.

Message-driven protocols: Protocols are 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.

Session: Each copy of a protocol run at a party is a session. 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 by a session-identifier (session ID). Several copies of protocols may be simultaneously run by each party. Each invocation of a protocol (or session) at a given party creates a local state for that session during execution, and produces local outputs by that party. When a session ends its run we call it complete and assume that its local state is erased.

Session key: Key-Establishment (KE) protocols are message-driven protocols where the communication takes place between pairs of parties and which return, upon completion, a secret key called a session key. It is required that the calling protocol makes sure that the session ID's of no two KE sessions in which the party participates are identical. Furthermore, we leave it to the calling protocol to make sure that two parties that wish to exchange a key will activate matching sessions.

Upon activation, the partners $P_i$ and $P_j$ of two matching sessions exchange messages, and eventually generate local outputs that include the name of the partners of the session, the session identifier (session ID), and the value of the computed session key. A key establishment event is recorded only when the exchange is completed. Note that a session can be completed at one partner but not necessarily at the other.

#### 6.2.1.2 Adversary models

Adversarial setting determines the capabilities and possible actions of the attacker. In CK model, the adversary model is given as generic as possible (as opposed to, say, merely representing a list of possible attacks). The CK model follows the general adversary formalism of [2] but specializes and extends the adversarial model here for the case of KE protocols.

1. The unauthenticated-links adversarial model (UM)

Basic attacker capabilities Consider a probabilistic polynomial-time (PPT) attacker that has full control of the communication links. The formalism represents this ability of the attacker by letting the attacker be the one in charge of passing messages from one party to another. The attacker also controls the scheduling of all protocol events including the initiation of protocols and message delivery.

Obtaining secret information All the secret information stored at a party is potentially vulnerable to break-ins or to other forms of leakage. The attacker may obtain secret information stored in the party's memories via explicit attacks. However, when defining security of a protocol, it is important to guarantee that the leakage of some form of secret information has the least possible effect on the security of other secrets. In order to be able to differentiate between various vulnerabilities and to be able to guarantee as

much security as possible in the event of information exposures, three attacks categories are classified depending on the type of information accessed by the adversary:

1) *Session-state reveal.* The attacker provides the name of a party and a session identifier of a yet incomplete session at that party and receives the internal state of that session. What information is included in the local states of a session is to be specified by each KE protocol. Typically, the revealed information will include all the local state of the session and its subroutines, except for the local state of the subroutines that directly access the long-term secret information.

2) *Session-key query.* The attacker provides a party's name and a session identifier of a completed session at that party and receives the value of the key generated by the named session. This attack provides the formal modeling for leakage of information on specific session keys that may result from events such as break-ins, cryptanalysis, and careless disposal of keys.

3) *Party corruption.* The attacker can decide at any point to corrupt a party, in which case the attacker learns *all* the internal memory of that party including long-term secrets (such as private keys or master shared keys used across different sessions) and session-specific information contained in the party's memory (such as internal state of incomplete sessions and session-keys corresponding to completed sessions).

If a session is subject to any of the above three attacks then the session is called locally exposed. If a session or its matching session is locally exposed then we call the session exposed.

**Session expiration.** Session expiration means that a session key (and any related session state) is erased from that party's memory. The value of an expired session key cannot be found via any of the above session-state reveal, session-key query and party corruption attacks if these attacks are performed after the session expired.

2. The authenticated-links adversarial model (AM)

Authenticated-links adversarial model The attacker is restricted to only deliver messages truly generated by the parties without any change or addition to them. This is the fundamental difference between UM adversarial model and AM adversarial model.

Emulation A notion introduced in order to capture the equivalence of functionality between protocols in different adversarial models, particularly between the UM and AM adversarial models.

Authenticator A special algorithm acts as an automatic "compiler" that translates protocols in the AM adversarial model into equivalent (or "as secure as") protocols in the UM adversarial model.

### 6.2.2 SK-security in CK model

The security of a key-exchange protocol was defined by Canetti and Krawczyk, it's also called session-key security in CK model.

Session-key security (or SK-security), focus on ensuring the security of an individual session-key as long as the session key value is not obtained by the attacker via an explicit key exposure. To capture the idea that the attacker "does not learn anything about the value of the key" from interacting with the key-establishment protocol and attacking other sessions and parties, the CK model formalizes SK-security via the infeasibility to distinguish between the real value of the key and an independent random value as that in the semantic-security approach[1]. The formulation of SK-security is very careful about tuning the definition to offer enough strength as required for the use of key-establishment protocols to realize secure channels, as well as being realistic enough to avoid over-kill requirements which would prevent researchers from proving the security of very useful protocols[10].

First, define an "experiment" where the attacker $I$ chooses a session which is "tested" about information it learned on the session-key; specifically, ask the attacker to differentiate the real value of the chosen session key from a random value.

For the sake of this experiment we extend the usual capabilities of the adversary $I$ in the UM by allowing it to perform a test-session query. That is, in addition to the regular actions of $I$ against a key-exchange protocol $\Pi$, we let $I$ choose, at any time during its run, a test-session among the sessions that have been completed, and are unexpired and unexposed at the time. Let $k$ be the value of the corresponding session key. We toss a coin $b$, $b \xleftarrow{R} \{0,1\}$. If $b = 0$, we provide the attacker $I$ with the value $k$. Otherwise we provide the attacker $I$ with a value $r$ randomly chosen from the probability distribution of keys generated by the protocol $\Pi$. The attacker $I$ is now allowed to continue with the regular actions of a UM adversary but is not allowed to expose this test-session (namely, it is not allowed session-state reveals, session-key queries, or partner's corruption on this test-session or its matching session). At the end of its run, the attacker $I$ outputs a bit $b'$ (as its guess for $b$). We will refer to an attacker that is allowed test-session queries as a KE-adversary.

**Definition 6.2** (SK-Security[10]) A KE protocol $\Pi$ is called SK-secure if the following properties hold for any KE-adversary $I$ in the UM.

1) Protocol $\Pi$ satisfies the property that if two uncorrupted parties complete matching sessions then they both output the same key.

2) The probability that the adversary $I$ guesses correctly the bit $b$ (i.e., outputs $b' = b$) is no more than $1/2$ plus a negligible fraction in the security parameter.

The first condition is a "consistency" requirement for sessions completed by two uncorrupted parties. The second condition is the "core property"

for SK-security. Definition 6.2 is very powerful and can be shown to ensure many specific properties that are required from a good key-exchange protocol. However, a key establishment protocol where all key-sessions "hang" and never return also satisfies the definition. For simplicity, Canetti et al choose to leave the analysis of the termination properties of protocols out of the scope of the definition of security. This is also the case in this book.

## 6.3 Authentication based on trusted freshness

In this section, we will introduce the security definitions based on trusted freshness. Some notations and security definitions in [1, 3] are adopted in the trusted freshness security definition. Recall that the central ingredient in the trusted freshness security analysis approach is the freshness principles introduced in chapter 4 of this book. Let's review and introduce some notations related to trusted freshness security analysis.

— Principals, probabilistic polynomial time machines, which are interconnected by point-to-point links over which messages can be exchanged.
— Trusted Third Party (TTP), a principal that provides a centralized authentication service in an open system.
— Freshness identifier (or TVP), a unique freshness component generated for a particular protocol run, it can be a nonce, a timestamp, a session key or a shared part of a session key.
— Protocol, a communication procedure which is run between or among co-operative principals.
— Message-driven protocols, protocols are initially triggered at a party by an external "call" and later by the arrival of messages.
— Challenge-Response protocol, in a challenge-response mechanism, one participant can verify the lively correspondence of the intended opposite partner by inputting a freshness identifier (challenge) to a composition of a protocol message and the composition involves a cryptographic operation (response) performed by the intended opposite partner.
— Session, a copy of a protocol run at a party, several copies of any protocol may be simultaneously run by each party.

### 6.3.1 Trusted freshness

In the context of communication protocols, that a freshness identifier is fresh means that the identifier has not been used previously, and originated within an acceptably recent time. Formally, *fresh* typically means *recent*, and it is in the sense of having originated subsequent to the beginning of the current protocol instance. Note that such freshness alone does not rule out interleaving

attacks using parallel sessions[8, 9].

**Definition 6.3** (Freshness) Given a protocol $\Pi$ between partners $A$ and $B$. A component of the protocol $\Pi$ is fresh if the component is guaranteed to be new from the viewpoint of one party $A$ or $B$.

*Classification of freshness component*

Three freshness component categories are classified depending on the type of the component:

1) Timestamp Recording the time of creation, transmission, receipt or existence of information.

The party originating a message obtains a timestamp from its local (host) clock, and binds it to a message. Upon receiving a time-stamped message, the second party obtains the current time from its own (host) clock, and subtracts the timestamp received. The timestamp difference should be within the acceptance window, and each party should maintain a "loosely synchronized" clock which must be appropriate to accommodate the acceptance window used.

The time clock must be secure to prevent adversarial resetting of a clock backwards so as to restore the validity of old messages, or setting a clock forward to prepare a message for some future point in time.

2) Nonce A value originated subsequent to the beginning of the current protocol instance, and it is used no more than once for the same purpose.

In a challenge-response protocol, the term nonce is most often used to refer to a "random" number which is sampled from a sufficiently large space, but the required randomness properties vary. A key parameter or the shared parts of a key may be viewed as a nonce in some cases. If random numbers are chosen by $A$ and $B$, respectively, then the random numbers together with a signature may provide a guarantee of freshness and entity authentication.

3) Sequence number A value provided by a never-repeated sequential counter, and it serves as a unique number identifying a message and is typically used to detect message replay. A message is accepted only if the sequence number therein has not been used previously (or not used previously within a specified time period). Sequence number changes on every new protocol instance or new message depending on different purposes.

Each party should maintain the sequence number pairwise of the originator and the receiver, and be sufficient to determine previously used and/or still valid sequence numbers. Distinct sequences are customarily necessary for messages from $A$ to $B$ and from $B$ to $A$.

Sequence numbers may provide uniqueness, but not (real-time) timeliness, and thus are more appropriate to detect message replay than entity authentication. Sequence numbers may also be used to detect the deletion of entire messages; they thus allow data integrity to be checked over an ongoing sequence of messages, in addition to individual messages.

Note that a sequence number is not natively fresh even for the sequence

number generator.

The cost for a random number or a sequence number to provide a freshness guarantee is an additional message more than that for the timestamp in the one-pass technique.

**Definition 6.4** (Trusted freshness) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the freshness identifier $\gamma$ is fresh, or in other words, a trusted freshness, if for a participant $A$,

1) the freshness identifier $\gamma$ originates in the participant $A$ itself.

2) the freshness identifier $\gamma$ is a timestamp and the timestamp difference between the initiator and the receiver is within the acceptance window.

3) $A$ has corroborative evidence that $\gamma$ is fresh. Here the corroborative evidence may be a signature, a MAC or other one-way transformation including the freshness identifier.

The first sufficient condition of Definition 6.4 is based on the randomization of $A$'s nonce, which has been sampled at random from a sufficiently large space and so no one can predicate the value before sampling; the second sufficient condition of Definition 6.4 is based on a "loosely synchronized" clock, and the timestamp difference is within the acceptance window, hence the "recent" property could be checked by the opponent party; The third sufficient condition of Definition 6.4 is based on the security property of the cryptographic algorithms, and it is widely used and more useful in challenge-response protocols. Note that the freshness of a freshness identifier could be given via mathematical proofs.

**Theorem 6.1** (Generation Rule) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. If a freshness identifier $\gamma$ is a nonce or a timestamp, and it is generated by the participant $A$ itself, then $A$ believes that $\gamma$ is a trusted freshness.

**Proof** A freshness identifier $\gamma$ originating in the participant $A$ could be a nonce, a timestamp or a sequence number. Let's consider the freshness of $\gamma$ in these three cases:

(1) The freshness identifier $\gamma$ is a nonce.

Let's recall the supposition: the term nonce is most often used to refer to a "random" number which is sampled from a sufficiently large space.

The "random" numbers are in fact pseudo-random numbers, they are generated by a pseudorandom number generator and they have a distribution totally determined (i.e., in a deterministic fashion) by a seed. Yet, a good pseudo-random number generator yields pseudorandom numbers which are polynomially indistinguishable from truly random numbers.

Recall that the adversary $I$ is a probabilistic polynomial-time machine which has full control of the communication links. Hence, the adversary $I$

couldn't distinguish the distribution of the random variables output from a pseudorandom number generator from the uniform distribution of strings (truly random numbers) which are of the same length as those of the pseudorandom variables.

Hence, if a freshness identifier $\gamma$ is generated by the participant $A$ itself, then $A$ believes that $\gamma$ is recent and it is a "random" number. Since $\gamma$ is a "random" number, it couldn't be guessed by a probabilistic polynomial-time attacker $I$, and we can guarantee that the freshness identifier $\gamma$ is used no more than once for the same purpose. That is, the freshness identifier $\gamma$ is a trusted freshness.

(2) The freshness identifier $\gamma$ is a timestamp.

Since the freshness identifier $\gamma$ is generated by the participant $A$ itself, then $A$ believes that $\gamma$ is recent. Recall the supposition that the timestamp difference between the initiator and the receiver is within the acceptance window, so there is no time gap for the freshness identifier $\gamma$ to be used for other purpose. That is, the freshness identifier $\gamma$ is used no more than once for the same purpose, hence the timestamp $\gamma$ is a trusted freshness.

(3) The freshness identifier $\gamma$ is a sequence number.

The sequence number is a value provided by a never repeated sequential counter, and it is typically used to detect message replay. Since the freshness identifier $\gamma$ is generated by the participant $A$ itself, $A$ believes that $\gamma$ is recent. However, the sequence number $\gamma$ may be guessed even by a probabilistic polynomial-time attacker $I$, so $I$ could obtain the intending response messages from sending request messages to the victim oracle, and the attacker $I$ may replay the achieved messages including $\gamma$ for other purpose. Hence, we do not regard a sequence number as a trusted freshness.

**Theorem 6.2** (Timestamp Rule) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. If a freshness identifier $\gamma$ is a timestamp, and it is received by $A$ from the opponent $B$, then $A$ believes that $\gamma$ is a trusted freshness.

**Proof** Since the freshness identifier $\gamma$ is a timestamp, and the timestamp difference between the initiator and the receiver is within the acceptance window, so $A$ believes that $\gamma$ is recent and there is no time gap for the freshness identifier $\gamma$ to be used for other purpose. That is, the freshness identifier $\gamma$ is used no more than once for the same purpose, hence $\gamma$ is a trusted freshness.

Recall the term, maximal term, signed term and similar term notions in Chapter 4, the follows is a example.

**Example 6.1** Here is another illustration of terms. Suppose there exists a message $B \rightarrow A : \{B, A, N_A, \{N_A, N_B, A, B\}_K\}$, the principal $B$ believes that $N_B$ is a trusted freshness identifier, and the principal $A$ believes that $N_A$ is a trusted freshness identifier too.

Upon sending this message, $B$ has the terms:

| | | |
|---|---|---|
| 1 | $+\{...N_B...\}$ | Definition 4.10 (1) |
| 2 | $+\{...N_B...\}_K$ | Term 1 and Definition 4.10 (3) |
| 3 | $+\{...N_A, N_B...\}_K$ | Term 2 and Definition 4.10 (2),(3) |
| 4 | $+\{...N_B, A...\}_K$ | Term 2 and Definition 4.10 (2),(3) |
| 5 | $+\{...N_B, B...\}_K$ | Term 2 and Definition 4.10 (2),(3) |
| 6 | $+\{...N_A, N_B...\}_K$ | Term 2 and Definition 4.10 (2),(3) |
| 7 | $+\{...N_A, N_B, A...\}_K$ | Term 6 and Definition 4.10 (2),(3) |
| 8 | $+\{...N_A, N_B, B...\}_K$ | Term 6 and Definition 4.10 (2),(3) |
| 9 | $+\{...N_A, N_B, A, B...\}_K$ | Term 8 and Definition 4.10 (2),(3) |

Upon sending this message, $A$ has the terms:

| | | |
|---|---|---|
| 1 | $-\{...N_A...\}$ | Definition 4.10 (1) |
| 2 | $-\{...N_A...\}_K$ | Term 1 and Definition 4.10 (3) |
| 3 | $-\{...N_A, N_B...\}_K$ | Term 2 and Definition 4.10 (2),(3) |
| 4 | $-\{...N_A, A...\}_K$ | Term 2 and Definition 4.10 (2),(3) |
| 5 | $-\{...N_A, B...\}_K$ | Term 2 and Definition 4.10 (2),(3) |
| 6 | $-\{...N_A, N_B...\}_K$ | Term 2 and Definition 4.10 (2),(3) |
| 7 | $-\{...N_A, N_B, A...\}_K$ | Term 6 and Definition 4.10 (2),(3) |
| 8 | $-\{...N_A, N_B, B...\}_K$ | Term 6 and Definition 4.10 (2),(3) |

$\{...N_A, N_B, A, B...\}_K$ is the maximal term of the message $\{N_A, N_B, A, B\}_K$ sent from $B$ to $A$.

To detect parallel attack, the maximal term of each message in the same protocol should not be the same.

### 6.3.2 Liveness of principal

The notation of the principal's liveness originates from [8], which means the presence of the intended partner. We try to make the liveness property more specific by giving the principal's liveness definition and a liveness rule.

**Definition 6.5** (Liveness of a principal) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. The liveness of $B$ means that $B$ is the intended partner of the communication from the point of view of $A$, and $B$ is responsive to the communications in the current protocol run.

**Theorem 6.3** (Liveness Rule) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the liveness of $B$ has been

authenticated by $A$ if $A$ has corroborative evidence that $B$ has participated in this protocol run.

Suppose $\gamma$ is a trusted freshness from the point of view of $A$. $A$ may have the liveness of $B$ from receiving the following "loose" one-way transformation,

1) A signature including the trusted freshness $\gamma$, which is signed by $B$.

2) An encryption including the trusted freshness $\gamma$, which is encrypted with MAC by $B$ using the shared long-term key between $A$ and $B$.

3) Other one-way transformation of a message including the trusted freshness identifier $\gamma$, and this cryptographic operation can provide evidence of $B$'s generation of this one-way transformation.

**Proof** Let's consider the liveness of $B$ from the point of view of $A$ in the following three scenarios:

(1) Signature including the trusted freshness identifier $\gamma$ by $B$.

Suppose $B$ is not alive and is not responsive to the communications in the current protocol run. Since the signature (it is believed to be signed by $B$'s private key) includes the trusted freshness identifier $\gamma$, then this signature could not be a replay of an old recorded message. So, it must be the adversary $I$ who has recently constructed the signature including the trusted freshness identifier $\gamma$. That is to say, the adversary $I$ has the ability to construct the signature without knowing the corresponding private key. This has translated an advantage for an alleged attack on the protocol to a similar (up to polynomial difference) advantage for inverting the signature used in the scheme. This contradicts the wide belief that there exists no efficient algorithm for inverting a signature.

Hence, if $A$ has received the signature including the trusted freshness identifier $\gamma$ from $B$, then $A$ is assured that $B$ is alive and responsive to the communications in the current protocol run.

(2) Encryption including the trusted freshness identifier $\gamma$ under the shared long-term key between $A$and $B$.

Recall that the maximal term of each message in the same protocol should not be the same, so this one-way transformation could not be a replay of encryption by $A$ itself. Other proof procedures are similar to the case (1).

(3) One-way transformation including the trusted freshness identifier $\gamma$, and this cryptographic operation can provide evidence of $B$'s generation of this one-way transformation.

The proof procedures are similar to the case (1).

The second condition of Theorem 6.3 also includes the case – keyed hash where the encryption algorithm is a hash function but not a traditional block cipher, and the key is the shared long-term key between $A$ and $B$.

**Definition 6.6** (Liveness of a principal with origin) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. The liveness of $B$ means that $B$ is the intended partner of the communication from the point of view of $A$, and $B$ is specially responsive to the communication with $A$ in

the current protocol run.

**Theorem 6.4** (Origin Liveness Rule) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the Origin liveness of $B$ has been authenticated by $A$ if $A$ has specially corroborative evidence that $B$ has participated in this protocol run with this origin participant $A$.

Suppose $\gamma$ is a trusted freshness from the point of view of $A$. $A$ may have the origin liveness of $B$ from receiving the following "loose" one-way transformation.

1) A signature including the trusted freshness $\gamma$ and the origin identity $A$, which is signed by $B$.

2) An encryption including the trusted freshness $\gamma$ and the origin identity $A$, which is encrypted with MAC by $B$ using the shared long-term key between $A$ and $B$.

3) Other one-way transformation of a message including the trusted freshness identifier $\gamma$ and the origin identity $A$, and this cryptographic operation can provide evidence of $B$'s generation of this one-way transformation.

### 6.3.3 Confidentiality of freshness identifier

**Definition 6.7** (Confidentiality of freshness identifier) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the freshness identifier is confidential if the adversary $I$ could not know $\gamma$ from the protocol run.

**Theorem 6.5** (Confidentiality Rule) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. Suppose all freshness identifiers are confidential at the beginning of a protocol run. The confidential property may change in the following two scenarios:

1) The freshness identifier is transmitted in plain text.
2) The freshness identifier is transmitted in an encryption whose decryption key is known by the adversary $I$.

**Proof** Omitted for its obviousness.

### 6.3.4 Freshness of freshness identifier

**Definition 6.8** (Freshness of a Freshness Identifier) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the

freshness identifier $\gamma$ is fresh if $\gamma$ is new for this protocol run.

**Theorem 6.6** (Freshness Rule) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the freshness of $\gamma$ has been authenticated by $A$ if $A$ has corroborative evidence that $\gamma$ is new for this protocol run. The corroborative evidence includes the following three scenarios:

1) The freshness identifier $\gamma$ originates in the participant $A$ itself for this protocol run.

2) The freshness identifier $\gamma$ is a timestamp and the timestamp difference between the initiator and the receiver is within the acceptance window.

3) $A$ has corroborative evidence that $\gamma$ is fresh. Here the corroborative evidence may be a signature, a MAC or other one-way transformation to assure the freshness of the freshness identifier.

**Proof** Refer to proofs in Theorem 6.4 and Theorem 6.1.

### 6.3.5 Association of freshness identifier

**Definition 6.9** (Association of Freshness Identifier) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the freshness identifier $\gamma$ is associated with $A$ (and/or $B$) if $\gamma$ is generated for the protocol run related with $A$ (and/or $B$).

**Theorem 6.7** (Association Rule) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the freshness identifier $\gamma$ is associated with $A$ (and/or $B$) from the point of view of $A$ if $A$ has corroborative evidence that $\gamma$ is generated for this protocol run related with $A$ (and/or $B$). The corroborative evidence includes the following scenarios:

1) $A$ sends or receives a message which is encrypted under the shared long-term key between two parties, then the trusted freshness identifier $\gamma$ in this message is related to these two parties.

2) $A$ sends or receives a message including the identity of $B$, which is encrypted under the shared long-term key between $A$ and the trusted third party $S$, then the trusted freshness identifier $\gamma$ in this message is related to $B$.

3) $A$ receives an encryption under the public-key of $A$, then the trusted freshness identifier $\gamma$ in this encryption is related to $A$.

4) $A$ sends an encryption including the identity of $A$ using the public-key of the opponent $B$, then the trusted freshness identifier $\gamma$ in this encryption is related to $A$.

5) $A$ receives a signature of $B$, then the trusted freshness identifier $\gamma$ in this signature is related to $B$.

6) $A$ receives a signature of $B$ including the identity of $A$, then the trusted freshness identifier $\gamma$ in this signature is related to $A$.

7) Other one-way transformation including the trusted freshness identifier $\gamma$, and this cryptographic operation can provide evidence of $B$'s (and/or $A$'s) association with $\gamma$, then the trusted freshness identifier $\gamma$ in this one-way transformation is related to $B$ (and/or $A$).

**Proof** We show that 1) of the theorem is satisfied. Suppose the two parties in case 1) are $A$ and $B$. If $A$ receives a message including the trusted freshness identifier which is encrypted under the shared long-term key between $A$ and $B$, then $A$ knows that it must be a message originating from $B$ and it could not be a replay one. Since the received message could only be decrypted by both $A$ and $B$, and the adversary could do nothing but replay this one, hence $A$ believes that the trusted freshness identifier $\gamma$ in this message is related to these two parties $A$ and $B$.

The proofs for 2) to 7) of the theorem are similar, hence omitted for concision.

The encryption algorithm in 1) and 2) of Theorem 6.7 may be block cipher or keyed hash; In case 3) of Theorem 6.7, if $A$ receives an encryption including the identity of $B$ under the public-key of $A$, then the freshness identifier $\gamma$ in this encryption needn't be related to $B$, since this message could even be generated by the adversary $I$.

### 6.3.6 Security analysis based on trusted freshness

We mainly discuss Challenge-Response authentication protocols.

#### 6.3.6.1 Notion

— Unilateral entity authentication: the identity of one protocol participant is authenticated.
— Mutual entity authentication: the identities of both protocol participants are authenticated to each other.
— Unilateral authenticated key transport: the identity of one protocol participant is authenticated, and the opposite unauthenticated party believes that the session key generated by the authenticated participant or a TTP can provide a secure channel over an insecure network.
— Mutual authenticated key transport: the identities of both protocol participants are authenticated to each other, and both protocol participants believe that the new session key generated by one of the participants or a TTP can provide a secure channel over an insecure network.

— Mutual authenticated key exchange (or key agreement): the identities of both protocol participants are authenticated to each other, and both protocol participants believe that the new session key which is the output of a function of all protocol participants' random input can provide a secure channel over an insecure network.

#### 6.3.6.2 Hypothesis

Suppose we have a probabilistic polynomial-time (PPT) attacker $I$ that has full control of the communication links as described in Dolev-Yao threat model[11]. Besides this, we suppose that the Dolev-Yao attacker $I$ in this book is allowed to perform a kind of cryptanalysis training course, and $I$ can also launch the adaptive chosen ciphertext attacks (CCA2) without limitations.

Suppose we have cryptographic primitives with security against indistinguishable adaptive chosen ciphertext attack (IND-CCA2). That is, in IND-CCA2 security strength, the failures in cryptographic protocols are not in any way related to the strength or weakness of the particular cryptographic primitives used, but related to the protocol logic flaws, which permits the attacker to break the security goals of cryptographic protocols without necessarily breaking the particular cryptographic primitives used. And we also suppose that a legitimate party is either totally corrupted or totally secure.

Suppose that each participant has his own private key and all other parties' public-keys (respectively, the shared long-term key between co-operative principals or the trusted third party) in public-key case (respectively, in shared key case), which are deployed safely before the cryptographic protocol run via authenticated channel or even traditional communication means. Furthermore, private keys and shared keys are commonly assumed to be too long to guess in a computationally feasible way.

Suppose that all freshness identifiers are confidential at the beginning of the protocol run.

In general, an authentication protocol is considered flawed if a principal concludes a normal run of the protocol with its intended communication partners while the intended partner would have a different conclusion.

#### 6.3.6.3 Notation

— $\rho$, arbitrary principal, ranges over the participants of the protocol run.
— $P_i$ or $P_j$, a principal indexed by subscript in a protocol run.
— $S$, trusted third party.
— $t$, arbitrary time, a moment, not a period of time.
— $t_0, t_1, t_2, ..., t_\$$, various time points. $t_0$ means time before the start of a protocol run, $t_i$ means time at message $i$ ($i = 1, 2, \dots$) exchange, and $t_\$$ means time at the termination of a protocol run respectively.
— $N$ or $N'$, arbitrary freshness identifier, it can be a nonce, a timestamp, a session key or the shared part of a session key.
— $N_{P_i}$, a freshness identifier invented by subscript principal $P_i$.

— $k$, a cryptographic key; $k^{-1}$, the inverse of $k$. In shared key case, $k$ and $k^{-1}$ are equal.
— $K$, a long-term key, it may be a secret key in shared key schemes, or a public-key and a private key in public-key schemes.
— $K_{P_iP_j}$, the long-term key shared by principal $P_i$ and $P_j$ in shared key case.
— $K_{P_i}$ and $K_{P_i}^{-1}$, the public-key and private key subscripted by the principal identity respectively in public-key case.
— $\varphi, \psi$ or $\Gamma$, a fact which has the value of *True* or *False* respectively.
— $\neg$, it is the same as negation used in logic.
— $Key(P_i, k)$, the principal $P_i$ has the knowledge of key $k$.
— $Belief(P_i, \varphi)$, the principal $P_i$ asserts that the fact $\varphi$ is *True*. For example, $Belief(P_i, Key(P_j, k))$ means that the principal $P_i$ asserts that the principal $P_j$ knows the key $k$.
— $Existing(P_j)$, the intended partner $P_j$ is in lively correspondence in this protocol run; $Belief(P_i, Existing(P_j))$, the principal $P_i$ asserts that the intended partner $P_j$ is in lively correspondence in this protocol run.
— $Originexisting(P_j)$, the intended partner $P_j$ is specially in lively correspondence in this protocol run with the origin participant $P_i$; $Belief(P_i, Originexisting(P_j))$, the principal $P_i$ asserts that the intended partner $P_j$ is specially in lively correspondence in this protocol run with the origin participant $P_i$.
— $Secret(N)$, the freshness identifier $N$ is confidential in this protocol run; $Belief(P_i, Secret(N))$, the principal $P_i$ asserts that the freshness identifier $N$ is confidential in this protocol run.
— $Fresh(N)$, the freshness identifier $N$ is fresh in this protocol run; $Belief(P_i, Fresh(N))$, the principal $P_i$ asserts that the freshness identifier $N$ is fresh in this protocol run.
— $Associate(N, P_1, P_2, P_3, ...)$, the freshness identifier $N$ is associated with a participant $P_1$, (or with both $P_1$ and $P_2$, or with $P_1, P_2$ and $P_3$...) in this protocol run; $Belief(P_i, Associate(N, P_1, P_2, P_3, ...))$, the principal $P_i$ asserts that the freshness identifier $N$ is associated with a participant $P_1$, (or with both $P_1$ and $P_2$, or with $P_1, P_2$ and $P_3$...) in this protocol run. For example, $Belief(P_i, Associate(N_A, A, B))$ means the principal $P_i$ asserts that the freshness identifier $N_A$ is associated with the principals $A$ and $B$.

### 6.3.7 Definition of security

The security definition under computational model provides a high confidence of the security of a cryptosystem[1, 3, 8, 13]. Recall the notations "conversation", "matching conversations", and we make tiny changes: only one-way transformation that includes a trusted freshness identifier is considered as an

*efficient message* of a conversation in our security analysis of cryptographic protocols based on trusted freshness. That is to say, we only concern with the fresh messages but omit the message parts that do not contribute to our protocol security properties to be proved in the trusted freshness approach.

Recall that at the end of a protocol run, each participant makes a decision about the authentication of the intended partner: "accept", "reject", or "undetermined"[8]. Given a protocol $\Pi$ between the principal $A$ and the principal $B$, if a principal like $A$ with a conversation *conv* believes that $B$ always has a conversation *conv*′ which matches *conv* whenever they are allowed to complete a protocol run, then this authentication protocol is secure from the point of view of $A$. Here the attacker wins if the principal $A$ has reached "accept" decision while $B$ does not have a matching conversation in $B$.

Semantic security is widely accepted in the cryptographic area, and we follow the probabilistic indistinguishability definitional approach in [3] to define confidentiality security. In this book, that the attacker has broken the scheme means that without breaking any cryptographic algorithm and knowing the corresponding key, the attacker can still learn something about the established new session key under the run of a cryptographic protocol. Here we define "learn" as distinguishing the value of a key generated by the cryptographic protocol from an independent randomly chosen key.

Based on the security definition of authenticity, we have presented the Unilateral entity Authentication Secure definition (UA-Secure) (Definition 4.3) and Mutual entity Authentication Secure definition (MA-Secure) (Definition 4.4); based on the security definition of authenticity and confidentiality, we have presented the Unilateral authenticated Key Secure (UK-Secure) (Definition 4.5) and Mutual authenticated Key Secure (MK-Secure[10]) (Definition 4.6).

In Chapter 4, we have also presented the security properties to clarify whether a cryptographic protocol is adequate for the security goals or not. Here, four formal security specifications will be given based on Theorem 4.1, Theorem 4.2, Theorem 4.3 and Theorem 4.4.

**Theorem 6.8** (UA-Secure) Given a protocol $\Pi$ between partners $P_i$ and $P_j$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. The authentication protocol $\Pi$ is UA-Secure if and only if the following property holds for one of the participants, say $P_i$: $P_i$ has Existing$(P_i, P_j)$, that is, $P_i$ believes that the intended opposite participant $P_j$ is in lively correspondence with $P_i$ in this protocol run.

**Proof** We show that the authentication protocol $\Pi$ is UA-Secure if and only if $P_i$ has *Existing*$(P_j)$.

(1) Sufficiency proof. We show that if $P_i$ has *Existing*$(P_j)$, then the protocol $\Pi$ meets Definition 4.3, that is UA-Secure. Suppose $P_i$ wants to authenticate the identity of the opponent partner $P_j$. This try could be made by $P_i$ or be a reply to the message sent by the opponent partner $P_j$. Recall

that only one-way transformation that includes a trusted freshness identifier is considered as an efficient message of a conversation in our trusted freshness. If $P_i$ has $Existing(P_j)$, that is $P_i$ believes the liveness of the intended opposite principal $P_j$, according to the liveness rule (Theorem 6.3), $P_i$ must have invented a challenge $N_{P_i}$ for this particular protocol run, and received a one-way transformation that includes $N_{P_i}$, where the one-way transformation can only be accomplished by the principal $P_j$. Recall that we have a cryptographic algorithm under IND-CCA2, if $P_i$ sees the conversation $conv_{P_i} = (\tau_0, \cdots, N_{P_i}), (\tau_2, \{N_{P_i}\}_{K_{P_iP_j}}, \cdots)$ in shared key case or $conv_{P_i} = (\tau_0, \cdots, N_{P_i}), (\tau_2, \{N_{P_i}\}_{K_{P_j}^{-1}}, \cdots)$ in public-key case, then $P_i$ sees that the uniformly random string $\{N_{P_i}\}_{K_{P_iP_j}}$ or $\{N_{P_i}\}_{K_{P_j}^{-1}}$ is computed using $N_{P_i}$ invented by $P_i$ itself; it can therefore conclude that the probability for this bit string not having been computed by its intended partner (in other words, having been computed by the attacker) is at the level of $2^{-k}$. Consequently, $P_i$ can conclude that its intended partner has a conversation which is prefixed by $conv_{P_j} = (\tau_1, N_{P_i}, \{N_{P_i}\}_{K_{P_iP_j}})$ or $conv_{P_j} = (\tau_1, N_{P_i}, \{N_{P_i}\}_{K_{P_j}^{-1}})$. This essentially shows that there exists a conversation $conv_{P_j}$ matching $conv_{P_i}$, and the conversation $conv_{P_j}$ has been computed by the intended partner $P_j$ in an overwhelming probability (in the security parameter $K_{P_iP_j}$ or $K_{P_j}^{-1}$). According to the security definition of authentication, hence $P_i$ believes that $P_j$ is in lively correspondence with $P_i$ in the matching session.

(2) Necessary proof. We show that if the protocol $\Pi$ is UA-Secure, then $P_i$ has $Existing(P_j)$. Suppose the UA-Secure protocol $\Pi$ doesn't hold the listed security property $Existing(P_j)$, that is, $P_i$ does not believe the liveness of the intended opposite principal $P_j$, then, according to the liveness rules (Theorem 6.3), $P_i$ has either sent a compromised or an old challenge $N_{P_i}$ to the intended opposite partner $P_j$ (namely, the attacker can replay a recorded stale message to $P_i$ by impersonating $P_j$), or $P_i$ does not require a response to $P_i$'s challenge (namely, the attacker can launch an attack directly). So the protocol $\Pi$ cannot be UA-secure, hence there exists a contradiction with the initial assumption that the protocol $\Pi$ is UA-Secure. Typical examples include Otway-Rees protocol[13], revised Woo-Lam protocol[14].

Therefore, we can conclude that the listed UA-secure security property is not only sufficient but also necessary for the protocol $\Pi$ to be UA-secure.

**Theorem 6.9** (MA-Secure) Given a protocol $\Pi$ between partners $P_i$ and $P_j$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. The authentication protocol $\Pi$ is called MA-Secure if the following properties hold for the participants $P_i$ and $P_j$:

1) $P_i$ has $Existing(P_j)$, that is, $P_i$ believes that the intended opposite participant $P_j$ is in lively correspondence with $P_i$ in this protocol run;

2) $P_j$ has $Existing(P_i)$, that is, $P_j$ believes that the intended opposite participant $P_i$ is in lively correspondence with $P_j$ in this protocol run;

**Proof** Similar to Theorem 6.8, omitted.

**Theorem 6.10** (Origin UA-secure) Given a protocol $\Pi$ between partners $P_i$ and $P_j$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. The authentication protocol $\Pi$ is Origin UA-Secure if and only if the following property holds for one of the participants, say $P_i$: $P_i$ has Originexisting($P_j$), that is, $P_i$ believes that the intended opposite participant $P_j$ is specially in lively correspondence with this origin participant $P_i$ in this protocol run.

**Theorem 6.11** (Origin MA-secure) Given a protocol $\Pi$ between partners $P_i$ and $P_j$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. The authentication protocol $\Pi$ is called Origin MA-Secure if the following properties hold for the participants $P_i$ and $P_j$:

1) $P_i$ has Originexisting($P_j$), that is, $P_i$ believes that the intended opposite participant $P_j$ is specially in lively correspondence with this origin participant $P_i$ in this protocol run;

2) $P_j$ has Originexisting($P_i$), that is, $P_j$ believes that the intended opposite participant $P_i$ is specially in lively correspondence with this origin participant $P_j$ in this protocol run;

Unilateral entity authentication secure (UA-secure) and Mutual entity authentication secure (MA-secure) are the most common cases in real world applications for identity authentication and access control services, which may suffer the replay attacks and at last may provide false identity authentication and access control. In deed, Origin Unilateral entity authentication secure (Origin UA-secure) and Origin Mutual entity authentication secure (Origin MA-secure) could meet these real world application requirements.

**Theorem 6.12** (UK-secure) Given an authentication protocol $\Pi$ between partners $P_i$ and $P_j$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. $k$ is the new session or the shared part of the new session key for this protocol run. The authentication protocol $\Pi$ is called UK-Secure if the following properties hold for one of the participants, say $P_i$:

1) $P_i$ has $Existing(P_j)$. That is, $P_i$ believes that the intended opposite participant $P_j$ is in lively correspondence with $P_i$ in this protocol.

2) $P_i$ has $Secret(k)$, $Fresh(k)$ and $Associate(k, P_i, P_j)$. That is, $P_i$ believes that the adversary $I$ could not know the new session key $k$ and $k$ is new for this protocol run between $P_i$ and $P_j$.

**Proof** Similar to Theorem 6.13, omitted.

**Theorem 6.13** (MK-secure) Given an authentication protocol $\Pi$ between partners $P_i$ and $P_j$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. $k$ is the new session key or the shared part of the new session key for this protocol run. The au-

thentication protocol $\Pi$ is called MK-Secure if the following properties hold for the participants $P_i$ and $P_j$:

1) $P_i$ has $Existing(P_j)$. That is, $P_i$ believes that the intended opposite participant $P_j$ is in lively correspondence with $P_i$ in this protocol run.

2) $P_j$ has $Existing(P_i)$. That is, $P_j$ believes that the intended opposite participant $P_i$ is in lively correspondence with $P_j$ in this protocol run.

3) $P_i$ has $Secret(k)$, $Fresh(k)$ and $Associate(k, P_i, P_j)$. That is, $P_i$ believes that the adversary $I$ could not know the new session key $k$ and $k$ is new for this protocol run between $P_i$ and $P_j$.

4) $P_j$ has $Secret(k)$, $Fresh(k)$ and $Associate(k, P_i, P_j)$. That is, $P_j$ believes that the adversary $I$ could not know the new session key $k$ and $k$ is new for this protocol run between $P_i$ and $P_j$.

**Proof** Sufficiency proof. We show that 1) of the Definition 4.6 is satisfied by protocol $\Pi$. Since the principal $P_i$ believes the liveness of the intended opposite principal $P_j$ and only one-way transformation that includes a trusted freshness identifier is considered as an efficient message of a conversation in our approach, according to liveness rule (Theorem 6.3), $P_i$ must have generated a challenge $N_{P_i}$ for this particular protocol run, and received a one-way transformation that includes a trusted freshness identifier $N_{P_i}$, where the one-way transformation can only be accomplished by the principal $P_j$. Recall that we have an ideal cryptographic algorithm with security against IND-CCA2, if $P_i$ sees the conversation $conv_{P_i} = (\tau_0, \cdots, N_{P_i}), (\tau_2, \{N_{P_i}\}_{K_{P_iP_j}}, \cdots)$ in shared key case or $conv_{P_j} = (\tau_1, N_{P_i}, \{N_{P_i}\}_{K_{P_j}^{-1}})$ in public-key case, then $P_i$ sees that the uniformly random string $\{N_{P_i}\}_{K_{P_iP_j}}$ or $\{N_{P_i}\}_{K_{P_j}^{-1}}$ is computed using $N_{P_i}$ invented by $P_i$ itself; it can therefore conclude that the probability for this bit string not having been computed by its intended partner (in other words, having been computed by the attacker) is at the level of $2^{-k}$. Consequently, $P_i$ can conclude that its intended partner has a conversation which is prefixed by $conv_{P_j} = (\tau_1, N_{P_i}, \{N_{P_i}\}_{K_{P_iP_j}})$ or $conv_{P_j} = (\tau_1, N_{P_i}, \{N_{P_i}\}_{K_{P_j}^{-1}})$. This essentially shows that there exists a conversation $conv_{P_j}$ matching $conv_{P_i}$ and the conversation $conv_{P_j}$ has been computed by the intended partner in an overwhelming probability (in the security parameter $K_{P_iP_j}$ or $K_{P_j}^{-1}$). According to the security definition of authentication, $P_i$ believes that $P_j$ is in lively correspondence with $P_i$ in this session. Similarly, $P_j$ believes that $P_i$ is in lively correspondence with $P_j$ in this session.

Since principal $P_i$ believes the freshness of the new session key $k$, according to the Freshness Rule (Theorem 6.6), $k$ must be a new generated session key for this run. Since principal $P_i$ believes the association of the new session key $k$ with the principals $P_i$ and $P_j$, according to the Association Rule (Theorem 6.7), $k$ must be a session key for a particular protocol run between $P_i$ and $P_j$. Up to now, $k$ must be a new generated session key for a particular protocol run between $P_i$ and $P_j$, hence $k$ is the same key for both $P_i$ and $P_j$, and it is different from other generated keys in any other sessions. That is to say, if

two uncorrupted parties complete matching sessions then they both output the same key.

We show that 2) of the Definition 4.6 is also satisfied by protocol $\Pi$. Recall that the attacker $I$ is a PPT machine who has full control of the communication links and $I$ is allowed to perform a kind of cryptanalysis training course. We can specify that $l$ is an upper bound of the session number for training invoked by $I$ in any interactions. Let $k$ be the value of the corresponding session key selected randomly in this protocol $\Pi$. Let $P_i$ play the game with the attacker $I$ as in Definition 4.6. Let $Bad$ be the events that the information of $k$ may leak during the cryptanalysis training courses. Let $I_{wins}$ denote the event that $I$ makes a correct guess of the challenge bit $b$. It is clear that in absence of the event $Bad$, due to the uniform randomness of the selected session key $k$, the challenge bit $b$ is independent of the challenge ciphertext $b$'. Thus we have

$$\text{Prob}[I_{wins}|\overline{Bad}] = \frac{1}{2}.$$

Since

$$\text{Prob}[I_{wins}|\overline{Bad}] = \frac{\text{Prob}[I_{wins} \cap \overline{Bad}]}{\text{Prob}[\overline{Bad}]},$$

we have

$$\text{Prob}[I_{wins} \cap \overline{Bad}] = \frac{1}{2}\text{Prob}[\overline{Bad}] = \frac{1}{2}(1 - \text{Prob}[Bad]).$$

While

$$\text{Prob}[I_{wins}] = \text{Prob}[I_{wins} \cap Bad] + \text{Prob}[I_{wins} \cap \overline{Bad}],$$

therefore

$$\begin{aligned}\text{Prob}[I_{wins}] &\leqslant \text{Prob}[Bad] + \text{Prob}[I_{wins} \cap \overline{Bad}] \\ &= \text{Prob}[Bad] + \frac{1}{2}(1 - \text{Prob}[Bad]) = \frac{1}{2}(1 + \text{Prob}[Bad]).\end{aligned}$$

Since we have an ideal cryptosystem, even the attacker $I$ has invoked $l$ times cryptanalysis training course, the probability that the information of $k$ may be leaked to $I$ by the underlying cryptosystem (say $Bad1$) is negligible in the security parameter, that is $\text{Prob}[Bad1] \leqslant l * Adv$ where $Adv$ is a negligible fraction. Since principal $P_i$ believes the liveness of the intended partner $P_j$, the probability that the information of $k$ might be leaked by $P_j$ to $I$ (say $Bad2$) is 0. Since principal $P_i$ believes the association of the new session key $k$ with the principals $P_i$ and $P_j$, the probability that the attacker $I$ could persuade $P_j$ to believe a key between $I$ and $P_j$ (or $I$ and $P_i$) to be the key $k$ between $P_i$ and $P_j$ (say $Bad3$) is 0. Then we have

$$\text{Prob}[Bad] = \text{Prob}[Bad1] + Prob[Bad2] + Prob[Bad3] \leqslant l * Adv,$$

therefore

$$\text{Prob}[I_{wins}] \leqslant \frac{1}{2}(1 + \text{Prob}[Bad]) \leqslant \frac{1}{2}(1 + l * Adv) = \frac{1}{2} + \frac{l * Adv}{2}.$$

Since attacker $I$ is a PPT attacker, then $(l * Adv)/2$ is negligible. Therefore, the probability that $I$ guesses correctly the bit $b$ is no more than $1/2$ plus a negligible fraction in the security parameter.

Necessary proof. Suppose that a MK-secure protocol $\Pi$ doesn't hold the listed security properties.

If a participant like $P_i$ (or $P_j$) does not believe the liveness of the intended opposite principal $P_j$ (or $P_i$), then, according to Theorem 6.3, $P_i$ (or $P_j$) has sent either a compromised or an old challenge to the intended opposite partner $P_j$ (or $P_i$) (That is to say, the attacker can replay a recorded stale message to $P_i$ (or $P_j$) by impersonating $P_j$ (or $P_i$), or does not require a response to $P_i$'s (or $P_j$'s) challenge (That is to say, the attacker can launch an attack directly). This contradicts the initial assumption. The typical examples are Otway-Rees protocol[13], revised Woo-Lam protocol[14]. Hence the protocol $\Pi$ cannot be MK-secure.

If a participant like $P_i$ (or $P_j$) believes that the confidentiality of the new session key $k$ is open, then, according to Lemma 4.2, the attacker wins with the probability of 1 when playing the game in Definition 4.6. Hence the protocol $\Pi$ cannot be MK-secure.

If a participant like $P_i$ (or $P_j$) does not believe the freshness of the session key $k$, then, according to Lemma 4.3, the attacker can replay a recorded message including a compromised key $k'$ as response to $P_i$ (or $P_j$). A typical example is the Needham-Schroeder shared key protocol[15]. Hence the protocol $\Pi$ cannot be MK-secure.

If a participant like $P_i$ (or $P_j$) does not believe the association of the session key $k$ with the co-operative participants, then, according to Lemma 4.4, the attacker may cheat a legitimate participant by confusing a key between the attacker and another to be the key between two legitimate participants. A typical example is the Needham-Schroeder public-key protocol[8]. Hence the protocol $\Pi$ cannot be MK-secure.

Therefore, we can conclude that the listed MK-secure security properties are not only substantial but also necessary for the protocol $\Pi$ to be MK-secure.

### 6.3.8 Non-repudiation based on trusted freshness

Non-repudiation means a principal could not deny his sending or receiving a message. These non-repudiation services (protection against false denials) relate to the transfer of a message from an originator to a recipient. Non-repudiation is an important security service for many applications and it is

a necessary security requirement in electronic commerce. Non-repudiation mechanisms are specified for non-repudiation of origin (denial of being the originator of a message), non-repudiation of delivery (denial of having received a message). Commonly used *fairness* security in electronic commerce protocols can also be derived from non-repudiation.

When disputes arise due to a principal denying that certain actions (having sent or received a message) were taken, a trusted third party can be called upon to make an arbitration: give the proof of message origin, or the proof of message delivery. For example, a principal may authorize a purchase of a car and later he may deny such an authorization granted. A procedure involving a third party is needed to resolve the dispute. What evidence would be submitted to the third party, and what precise process the third party is to follow to render judgement on disputes should be specified in detail for a non-repudiation service.

Digital signatures can provide the non-repudiation service because a signature of a message is verifiable universally. The non-repudiation service provided by a digital signature means a proof of knowledge that a signer owns exclusively a private key (knowledge) which has enabled him (her) to issue the signature. The non-repudiation aspect of digital signatures is a primary advantage of public-key cryptography.

Symmetric techniques (including encipherment and keyed one-way functions) typically cannot provide non-repudiation service effectively, since the data integrity techniques based on a shared secret key (e.g., MACs) typically involve mutual trust and do not address true (single-source) data origin authentication, that is, either party sharing the secret key can equally originate a message using the shared key.

If the resolution of subsequent disputes is a potential requirement, then either an on-line trusted third party is in a notary role, or asymmetric techniques should be used.

**Definition 6.10** (Non-repudiation) A crypto protocol $\Pi$ can provide non-repudia-tion service if any attacker $I$ cannot win with a non-negligible probability in Dolev-Yao threat model. Here the attacker wins if a principal $A$ has reached the conclusion of message origin of $B$, or message delivery of $B$ while $B$ has not sent or received the message.

**Rule 6.1** (Non-repudiation rule) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the protocol $\Pi$ can provide non-repudiation service if $A$ has corroborative evidence that the dispute message $m$ originates from $B$, or the dispute message $m$ has been received by $B$. The corroborative evidence includes these following scenarios:

1) The dispute message $m$ is a signature of its originator $B$.

2) The dispute message $m$ is guaranteed known by the dispute opponent $B$ via the message composition involving a cryptographic operation (response) that could only be performed by the intended opposite partner $B$.

Note: Suppose the trusted third party $S$ is secure, then the corroborative evidence could be an encryption sent by $B$ to the trusted third party encrypted under the shared key between $B$ and $S$.

**Definition 6.11** (Real time non-repudiation) A cryptographic protocol $\Pi$ can provide realtime non-repudiation service if any attacker $I$ cannot win with a non-negligible probability for a particular protocol run in Dolev-Yao threat model. Here the attacker wins if a principal $A$ has reached the conclusion of message origin of $B$, or message delivery of $B$ for this particular protocol run while $B$ has not sent or received the message.

**Rule 6.2** (Real time non-repudiation rule) Given a protocol $\Pi$ between partners $A$ and $B$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. We say that the protocol $\Pi$ can provide non-repudiation service if $A$ has corroborative evidence that the dispute message $m$ originates from $B$ for this particular protocol run, or the dispute message $m$ has been received by $B$ for this particular protocol run. The corroborative evidence includes these following scenarios:

1) The dispute message $m$ including a timestamp is a signature of its originator $B$.

2) The dispute message $m$ is guaranteed known by the dispute opponent $B$ via $B$'s cryptographic operation (response), which could only be performed by $B$, on the message composition including a timestamp.

3) The dispute message $m$ is guaranteed known by the dispute opponent $B$ via the message composition involving a cryptographic operation (response) that could only be performed by $B$, and the cryptographic operation includes a timestamp.

Here are some notations supplemented in trusted frehsness approach:

— $Originate(\rho, m, P_j)$, every principal $\rho$ or at least a dispute judger could assert that the dispute message $m$ originates from $P_j$.
— $Respond(\rho, m, P_j)$, every principal $\rho$ or at least a dispute judger could assert that the dispute message $m$ has been received by $P_j$.
— $RealTimeOriginate(\rho, m, P_j)$, every principal $\rho$ or at least a dispute judger could assert that the dispute message $m$ originates from $P_j$ for a particular protocol run.
— $RealTimeRespond(\rho, m, P_j)$, every principal $\rho$ or at least a dispute judger could assert that the dispute message $m$ has been received by $P_j$ for a particular protocol run.

**Theorem 6.14** (Non-repudiation secure) Given an authentication protocol $\Pi$ between partners $P_i$ and $P_j$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. $m$ is a dispute message between partners $P_i$ and $P_j$. We say that the protocol $\Pi$ can provide non-repudiation service if either of the following properties holds for one of the participants, say $P_i$:

1) $P_i$ has Originate$(P_i, m, P_j)$. That is, $P_i$ has corroborative evidence that

the dispute message $m$ is generated by $P_j$.

2) $P_i$ has Respond$(P_i, m, P_j)$. That is, $P_i$ has corroborative evidence that the dispute message $m$ has been received by $P_j$.

**Proof** We show that 1) of the Theorem 6.14 is satisfied. Suppose that an adversary $I$ can win with a non-negligible probability to make the principal $P_i$ believe that a forged message $m$ originates from the opponent partner $P_j$. Recall that $P_i$ has $Originate(P_i, m, P_j)$. That is, $P_i$ has corroborative evidence that the dispute message $m$ is generated by $P_j$. Suppose the corroborative evidence is that the dispute message $m$ is a signature of its originator $P_j$. Hence, the forged message $m$ is a successfully constructed signature of its originator $P_j$. The adversary's ability for signature forgery can be fully translated to one for inverting the hard function (i.e., the underlying one-way trapdoor function). This contradicts the assumed well-known intractability problem (i.e., factoring of RSA moduli).

Therefore, we can conclude that $Originate(P_i, m, P_j)$ is substantial for the protocol $\Pi$ to provide non-repudiation service.

We show that 2) of the Theorem 6.14 is satisfied. Suppose that an adversary $I$ can win with a non-negligible probability to make the principal $P_i$ believe that a forged message $m$ has been received by the opponent partner $P_j$. Recall that $P_i$ has $Respond(P_i, m, P_j)$. That is, $P_i$ has corroborative evidence that the dispute message $m$ has been received by $P_j$. The corroborative evidence is that the dispute message $m$ is known by the dispute opponent $P_j$ via a cryptographic operation that could only be performed by the intended opposite partner (i.e., a decryption using $P_j$'s private key of receiving an encryption including $m$). Hence, if the adversary wins, then the adversary could decrypt a public-key encryption without knowing the corresponding private key in public-key case. The adversary's ability for decryption without corresponding key can also be fully translated to one for inverting the hard function. This also contradicts the assumed well-known intractability problem.

Therefore, we can conclude that $Respond(P_i, m, P_j)$ is also substantial for the protocol $\Pi$ to provide non-repudiation service.

**Theorem 6.15** (Realtime non-repudiation secure) Given an authentication protocol $\Pi$ between partners $P_i$ and $P_j$ in the presence of a probabilistic polynomial-time adversary $I$ that has full control of the communication links. $m$ is the dispute message for this protocol run. We say that the protocol $\Pi$ can provide non-repudiation service if either of the following properties holds for one of the participants, say $P_i$:

1) $P_i$ has $RealTimeOriginate(P_i, m, P_j)$. That is, $P_i$ has corroborative evidence that the dispute message $m$ is generated by $P_j$.

2) $P_i$ has $RealTimeRespond(P_i, m, P_j)$. That is, $P_i$ has corroborative evidence that the dispute message $m$ has been received by $P_j$.

**Proof** We show that 1) of the Theorem 6.15 is satisfied. The proof that

the Non-repudiation property, the dispute message $m$, originates from $P_j$, in this theorem is similar to that proof in Theorem 6.14, hence omitted.

We show that the real time property in this theorem is also satisfied. Recall that $P_i$ has $RealTimeOriginate(P_i, m, P_j)$. That is, $P_i$ (respectively, $P_j$) has corroborative evidence that the dispute message $m$ has been sent by $P_j$ in a "loosely synchronized" clock time (the timestamp records the time of sending). In deed, this timestamp is obtained by the party $P_j$ from its local (host) clock, and is bound to the dispute message $m$. Since each party has maintained a "loosely synchronized" clock which is appropriate to accommodate the acceptance window used, if the timestamp difference is within the acceptance window, then $P_i$ believes that this dispute message $m$ is a real time message.

Therefore, we can conclude that $RealTimeOriginate(P_i, m, P_j)$ is sufficient for the protocol $\Pi$ to provide real time non-repudiation service.

From the proof of the real time property in 1) of Theorem 6.15 and also the proof of delivery property in 2) of Theorem 6.14, we can obtain the proof of 2) real time non-repudiation in Theorem 6.15 similarly.

## References

[1] Bellare M, Rogaway P (1993) Entity Authentication and Key Distribution. In: CRYPTO'93 Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara, 22 – 26 Aug 1993. Lecture Notes in Computer Science, vol 773, pp 232 – 249, Springer

[2] Bellare M, Canetti R, Krawczyk H (1998) A Modular Approach to the Design and Analysis of Authentication and Key-exchange Protocols. In: Proceedings of the 30th STOC, Dallas, 23 – 26 May 1998

[3] Goldwasser S, Micali S (1984) Probabilistic Encryption. Journal of Computer and System Sciences 28(2): 270 – 299

[4] Goldwasser S, Micali S, Rivest R (1988) A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks. SIAM Journal of Computing 17(2): 281 – 308

[5] Blum M, Micali S (1984) How to Generate Cryptographically Strong Sequences of Pseudorandom Bits. SIAM Journal on Computing 13(4): 850 – 864

[6] Yao AC (1982) Theory and Applications of Trapdoor Functions. In: Proceedings of the IEEE 23rd Annual Symposium on the Foundations of Computer Science, Chicago, 3 – 5 Nov 1982

[7] Bellare M, Rogaway P (1995) Provably Secure Session Key Distribution – the Three Party Case. In: Proceedings of the 27th ACM Symposium on the Theory of Computing, Las Vegas, 29 May – 1 June 1995

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

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

[10] Canetti R, Krawczy H (2001) Analysis of Key-exchange Protocols and Their Use for Building Secure Channels. In: EUROCRYPT'01 Proceedings of the

International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology, Innsbruck, 6 – 10 May 2001. Lecture Notes in Computer Science, vol 2045, pp 453 – 474, Springer

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

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

[13] Otway D, Rees O (1987) Efficient and Timely Mutual Authentication. Operating Systems Review 21(1): 8 – 10

[14] Abadi M, Needham R (1996) Prudent Engineering Practice for Cryptographic Protocols. IEEE Transactions on Software Engineering 22(1): 6 – 15

[15] Lowe G (1995) An Attack on the Needham-Schroeder Public Key Authentication Protocol. Information Processing Letters 56(3): 131 – 133