One-round Secure Key Exchange Protocol With Strong Forward Secrecy

Xiaowei Li[1], Dengqi Yang[1], Benhui Chen[1], Yuqing Zhang[2]
[1] Department of Mathematics and Computer Science, Dali University
Yunnan, Dali – China
[2] National Computer Network Intrusion Protection Center, University of the Chinese Academy of Sciences
Beijing – China
[e-mail: lixiaowei_xidian@163.com, dqyang@dali.edu.cn, bhchen@dali.edu.cn, zhangyq@ucas.ac.cn]
*Corresponding author: Xiaowei Li

Received May 22, 2016; revised September 26, 2016; accepted October 19, 2016; published November 30, 2016

Abstract

Security models for key exchange protocols have been researched for years, however, lots of them only focus on what secret can be compromised but they do not differentiate the timing of secrets compromise, such as the extended Canetti-Krawczyk (eCK) model. In this paper, we propose a new security model for key exchange protocols which can not only consider what keys can be compromised as well as when they are compromised. The proposed security model is important to the security proof of the key exchange protocols with forward secrecy (either weak forward secrecy (wFS) or strong forward secrecy (sFS)). In addition, a new kind of key compromise impersonation (KCI) attacks which is called strong key compromise impersonation (sKCI) attack is proposed. Finally, we provide a new one-round key exchange protocol called mOT+ based on mOT protocol. The security of the mOT+ is given in the new model. It can provide the properties of sKCI-resilience and sFS and it is secure even if the ephemeral key reveal query is considered.

Keywords: one round, key exchange, forward secrecy, ephemeral key reveal

This research was supported by the National Natural Science Foundation of China (No. 61462003, No. 71462001, No. 61272481, No. 61572460), the National Key Research and Development Project (2016YFB0800703) and Education Foundation of Yunnan (No. 2016ZZX192). The authors would like to thank the editor and all the anonymous reviewers for their useful suggestion.

http://dx.doi.org/10.3837/tiis.2016.11.024 **ISSN : 1976-7277**

1. Introduction

Key exchange protocol can enable two or more participants to authenticate each other and establish a secure channel. It is one of the most widely used cryptographic protocols. Key exchange protocols have been studied for years and more and more attacks or security models have been strengthened or refined, such as attacks from passive to active and security models from Canetti-Krawczyk (CK) [1] to extended Canetti-Krawczyk (eCK) [2]. However, although the attacks and the security models were proposed, the attentions were always on what kind of secrets the adversary can get from the attacks but not on when the attacks can be launched. Considering the time of the secret compromise, *weak forward secrecy* (wFS) and *strong forward secrecy* (sFS) were present. Forward secrecy proposed by Diffie *et al*. [3] means compromise of long-term keys should not result in compromise of the session keys agreed before the long-term keys were compromised. It is a property which is difficult to be achieved since it allows the adversary to obtain the long-term keys of the parties. Meanwhile, key exchange protocols which provide forward secrecy are often computationally more expensive than those which do not provide forward secrecy since an extra Diffie-Hellman tuple is often needed in order to achieve this property. So some researchers may not pay much attention to it, especially in the case when the long-term keys of the parties are not assumed to be corrupted [4-5]. However, we do not know what will happen in the future. In some extent, the more robust the protocol is, the smaller risk we will meet. The classification of wFS and sFS was first appeared in [6] by Krawczyk. wFS means the adversary who is not active in the test session cannot compute the session key of a key exchange protocol even given both of the long-term keys of the parties after the test session is expired. sFS means the adversary who is active in the test session cannot compute the session key of a key exchange protocol even given both of the long-term keys of the parties after the test session is expired. Note “active” here means the adversary can send the message to an honest user A on behalf of another honest user B. Lots of efficient key exchange protocols can only provide wFS but not sFS [7-10].

Actually, sFS can be easily achieved by an explicitly authenticated key exchange protocols with wFS since the honest communication parties can send one more confirmation message which indicates that he has computed the session key. However, the key point is how can sFS be achieved in one-round key exchange protocols since explicit authentication cannot be achieved in one-round key exchange protocols. So considering the sFS property in one-round key exchange protocol is a meaningful work.

In Section 2, we review the previous work in one-round key exchange protocols. In Section 3, we give the motivation and contribution of the paper. In Section 4, we present a new kind of key compromise impersonate attack and a security model which includes this attack. In Section 5, we describe the mOT+ protocol we proposed. We then give the security analysis and the performance of our protocol in Section 6. Finally, in Section 7 we make a conclusion of this paper.

2. Related Work

In this section, we discuss one-round key exchange protocols and divide them into three categories: protocols without strong forward secrecy (sFS), protocols with sFS but without ephemeral key resistance and protocols with both of sFS and ephemeral key resistance.

Protocols without sFS. Diffie and Hellman [11] did a seminal work in designing key

exchange protocol and proposed the first one-round key exchange protocol called Diffie-Hellman (DH) key exchange protocol. Although the message of the Diffie-Hellman key exchange protocol is not authenticated, its idea is the foundation of many other key exchange protocols later. In order to give a secure key exchange protocol whose computation cost is similar to Diffie-Hellman key exchange protocol, Menezes *et al*. [12] proposed an efficient key exchange protocol which is called MQV protocol. However, Krawczyk [6] found MQV was vulnerable to key compromise impersonation attack and it can only provide weak forward secrecy (wFS). A variant of the MQV which is called HMQV was proposed by Krawczyk. However, as pointed by Krawczyk in [6], any 2-message key-exchange protocol (including HMQV) with no secure shared state previously established between the parties cannot provide sFS. For a few years, researches seemed to agree with Krawczyk's statement and did not consider one-round (two-flow) key exchange protocol with sFS. For example, "no 2-round AKE protocol can achieve full perfect forward secrecy (PFS)" stated by LaMacchia *et al*. [2] and "two-party protocols with only two-message flow and having no previous establishment of secure shared state cannot achieve perfect forward secrecy" stated by Chow and Choo [13].

Protocols with sFS but without ephemeral key resistance. In order to address the problem of sFS and give a formal security proof of key exchange protocol, Canetti and Krawczyk proposed the CK model [1]. The definition of "session expiration" makes the property of sFS come true. A protocol SIG-DH with sFS was proposed in [1]. The trick used in SIG-DH is using a signature algorithm to authenticate the message in DH protocol. SIG-DH is a protocol with sFS, however, it is not efficeint since the signature algorithm is used. Gennaro *et al*. [14] proposed a one-round key exchange protocol based on Okamoto-Tanaka protocol [15] which is called mOT protocol. mOT is a remarkable one-round key exchange protocol since it provides sFS while achieving the minimal communication. However, the SIG-DH protocol in CK model and mOT are vulnerable to the ephemeral key reveal attack which is a strong attack proposed by LaMacchia *et al*. [2]. LaMacchia *et al*. [2] proposed a new security model including the ephemeral key reveal attack which is called eCK model. However, the limitation of eCK is that it does not differentiate the timing of secret compromise, which means that it cannot capture sFS.

Protocols with both of sFS and ephemeral key resistance. Boyd *et al*. [16] gave a comprehensive analysis of the one-round key exchange protocols with sFS and ephemeral key resistance. A compiler which can transform any secure key exchange protocol with wFS into a key exchange protocol with sFS was also proposed in [16]. However, Cremers and Feltz [17] showed Boyd and Nieto's construction [16] was not secure if one of the users's long-term key is revealed before the session begins. Cremers and Feltz [17] proposed a new compiler for one-round key exchange protocol which can transform any one-round key exchange protocol which is secure in eCK^{w} (a model which is slightly stronger than eCK) into a secure one-round key exchange protocol in eCK-PFS. The property of ephemeral key resistance is also considered in eCK-PFS. The work of Cremers and Feltz is very meaningful in designing the key exchange protocol with sFS. The trick used in [17] is using the deterministic signature which is strongly existentially unforgeable under an adaptive chosen-message attack (SUF-CMA) to sign the message in NAXOS. However, it is not easy to transform all the key exchange protocols into protocols with eCK-PFS. Meanwhile, Cremers and Feltz did not differentiate the weak key compromise impersonation (wKCI, definition is in 4.1) and the strong key compromise impersonation (sKCI, definition is in 4.1). Bergsma *et al*. [18] proposed the first generic construction for one-round key exchange protocol with sFS in the standard model. The limitation is that it is less efficient compared with the protocol in the random oracle model.

3. Motivation and contribution

Security models for key exchange protocols have been researched for years, however, lots of them only consider what secret can be compromised but they do not differentiate the timing of secret compromise. In this paper, considering the time of secret compromise, we first provide a new strong key compromise impersonation (sKCI) attack to key exchange protocols. Second, we propose a new security model for key exchange protocol which considers not only what keys can be compromised but also when they are compromised. Third, in order to design a one-round key exchange protocol which can be resistant to the sKCI attack and achieve the sFS property, we propose a one-round key exchange protocol based on the mOT protocol, called as mOT+. mOT+ is secure even if the ephemeral key is revealed. The security of the new protocol is given in the new model we proposed.

4. KCI Attack and The Model

4.1 KCI Attack

Key compromise impersonation (KCI) attack first appeared in [19] and was deeply analyzed by Strangio [20]. Generally speaking, KCI-resistance property is similar to the forward secrecy since both of the two secure properties allow the adversary to corrupt the long-term keys of the participants. The difference lies in the time of corrupting the long-term keys. One is before executing the test session while the other is after executing the test session. As forward secrecy aforementioned, KCI attack can also be divided into two levels:

Weak KCI attack (wKCI): Suppose an adversary with the knowledge of the long-term private key of a user A impersonates another honest user B to communicate with A, while A does not realize the corruption of his key. A then accepts the session and the adversary can compute the session key of this session although there is no such partner session at B.

Strong KCI attack (sKCI): Consider a deeper lever, once an adversary has corrupted a user A, he impersonates another user B to communicate with A. The adversary may not be able to compute the session key of this session at this point, however, some days later if the adversary has the ability to corrupt B then he can reveal the session key of A and B although the session has expired and there is no such partner session at B.

wKCI attack has been studied in some works in recent years [19, 20], however, few of researchers considers the sKCI attack [17]. Let's consider such a scenario in the real life: Alice wants to tell Bob a secret then she begins to establish a secure channel with Bob by using a one round key exchange protocol in order to reduce communication cost. However, Alice has no idea that her long term key has been revealed by Eve. When Alice establishes the key exchange protocol with Bob, Eve intercepts the message and impersonates Bob to communicate with Alice. Upon receiving the message from "Bob" (actually it is from Eve), Alice computes the session key and encrypts the message "I tell you a secret and do not tell anyone else. The secret is... " , then Alice sends the message to "Bob". If Eve can reveal the session key of this session and decrypts the message at this point without having additional ability, then it means Eve launches a successful wKCI attack. However, if Eve cannot reveal the session key of the target session at this point but several days later, when Eve has an additional ability to reveal Bob's long term key then Eve can recover the session key between Alice and "Bob". In such case, we say that Eve launches a successful sKCI attack. Cremers and Feltz [17] considered the sKCI attack as mentioned in this paper, however, they regarded the sKCI attack as the common KCI attack, i.e., the wKCI attack. Actually, it is meaningful to

distinguish the wKCI attack and the sKCI attack since they represent the attacks from the adversaries with different abilities respectively. It is just like the reason why we distinguish the wFS and the sFS. Note we can regard the property of sKCI-resilience as a special kind of sFS since both of these properties are against an adversary who can compromise the long-term keys of the honest user. The difference lies in when these long-term keys are compromised. Actually, sKC-resilience implies common sense sFS which we called before since the adversary who launches the sKCI attack has one more ability than the adversary who is against the sFS, i.e., corrupting one communication party before executing the test session.

4.2 Security Model

Before we give the formal security model, we need to explain the possibility of the attacks when the adversary has access to the secret information of a session. Actually, there are only two kinds of secrets in a key exchange protocol. One is the long-term key and the other is the ephemeral key. We have to guarantee that the session key is still secure if one kind of secrets is revealed by the adversary. If the ephemeral keys are revealed to the adversary in a session, it means he gets all the random values chosen in this session. Then the session key may be leaked in some protocols such as the SIG-DH protocol in CK model since the session key can be computed by the ephemeral keys without the long-term keys. So the leakage of the ephemeral keys means that the session key may be leaked and we have to defeat this attack. And if the long-term keys are revealed to the adversary, he can take advantage of these secrets to recover the session key agreed before the long-term keys are compromised (the adversary can be passive and active in the past session). The reason lies in that the session key can be computed by the long-term keys without the ephemeral keys such as the NAXOS protocol in eCK model (the attack to the property of sFS or sKCI resistance). In a word, if the adversary has access to secret information (ephemeral keys or long-term keys) of a key exchange protocol, then the session key may be revealed. So we have to give a security model which includes both of the ephemeral key reveal attack and the long-term key reveal attack.

The model proposed here is based on the CK model [1] and eCK model [2]. Let U be one of the users in $\{U_1, U_2, ..., U_n\}$. The protocol P is running among these users. During the execution of P, each user may have multiple *instances*. One execution of the protocol is called an instance or a session. Let $\prod_U^i$ be the i-th instance at U. Every session has a unique session identifier. The session identifier of a completed session can be regarded as the concatenation of the messages a user sent and received, *i.e.*, $\{s_{sent}, s_{recv}\}$. Two sessions are said to be matching sessions if they have the same session identifier. Note the matching session may not exist when a session completes. We call the state as *accepte*d when an instance computes the session key. An instance may terminate before arriving an accepted state. Here we consider the communication networks to be fully controlled by an adversary $\mathcal{A}$. Meanwhile, $\mathcal{A}$ is allowed to ask the following queries to the protocol instances polynomial times.

Execute $(\prod_U^i, \prod_{U'}^j)$. The adversary $\mathcal{A}$ gets access to ith honest executions between U and U' by eavesdropping.

Send $(\prod_U^i, m)$. Upon receiving the message m, $\prod_U^i$ executes the protocol and responds with an outgoing message or a decision to indicate accepting or rejecting the session. If $\prod_U^i$ does not exist, it will be created as an initiator, or as a responder otherwise.

Session key reveal $(\prod_U^i)$. This query takes as input a session identifier and returns the session key if the session is complete.

Ephemeral key reveal ($\prod_U^i$). This query takes as input a session identifier. If the session is incomplete then the ephemeral keys are returned.

Long – term key reveal$_b$ (*U*). The long-term secret key of *U* is returned to $\mathcal{A}$ before *U* executes a session.

Long – term key reveal$_a$ (*U*). The long-term secret key of *U* is returned to $\mathcal{A}$ after *U* completes a session.

Test ($\prod_U^i$). After several interaction with the users' instance, $\mathcal{A}$ will choose a session as the test session. In such case, a random bit b is secretly chosen. If b = 1, $\mathcal{A}$ is given the session key of the test session. Otherwise, a random value chosen from the session key space is given. Note that a Test query is allowed only to an accepted and *fresh* instance.

Fresh: A protocol instance $\prod_U^i$ is said to be *fresh* if the following conditions holds:

1. The instance $\prod_U^i$ or its matching session (if it exists) has not been asked a *Session key reveal* query;

2. If the instance $\prod_U^i$ has been asked an *Ephemeral key reveal* query, then either *Long – term key reveal* query is not allowed to *U* and vice versa.

3. Let $\prod_U^i = \{s_{sent}, s_{recv}\}$ and $\prod_U^j = \{s_{sent}', s_{recv}'\}$ be two sessions of *U*, if $s_{sent} = s_{sent}'$ and $\prod_U^i$ has been asked an *Ephemeral key reveal* query, then both of the *Long – term key reveal* queries are not allowed to *U* and vice versa.

4. If $\prod_U^i$ is the matching session of $\prod_{U'}^j$ and $\mathcal{A}$ has asked a *Long – term key reveal*$_b$ query to *U*, then any message that $\mathcal{A}$ sends to $\prod_{U'}^j$ on behalf of *U* must come from $\prod_U^i$ itself.

5. If $\prod_U^i$ is the matching session of $\prod_{U'}^j$, then $\mathcal{A}$ is only allowed to ask *Long – term key reveal*$_b$ query to one of *U* and *U*'.

Security Definition. We use the security definition of CKsecure [1]. A key-exchange protocol is said to be secure in the proposed model if for all Probabilistic Polynomial-Time (PPT) adversaries $\mathcal{A}$:

1. if two uncorrupted parties complete matching sessions in a run of the protocol then, except for a negligible probability, the session key output in these sessions is the same;

2. $\mathcal{A}$ succeeds in guessing the bit b in the test session with probability not more than 1/2 plus a negligible fraction.

Differences from the traditional security models. Lots of security models for key exchange protocols only consider what secret can be compromised but they do not differentiate the timing of secret compromise, such as eCK model. The new security model proposed in this paper considers not only in terms of which keys can be compromised but when they are compromised. Our model is similar to the eCK-PFS model proposed in [17] and the difference of the two models is that we differentiate the timing of secret compromise as *Long – term key reveal*$_b$ and *Long – term key reveal*$_a$ which is more intuitive to define the sFS as well as the sKCI attack. eCK-PFS model does not refine the different attacks. So our model can be seemed as a supplement of eCK-PFS. Meanwhile, eCK-PFS uses a new definition origin-session to define the fresh session while our model does not use new

definition, so it is easy for the readers to have a transition from the old models (such as CK and eCK) to the new model.

5. mOT+ Protocol

In this section, we propose a new one-round strongly secure key exchange protocol with sKCI-resistance and sFS based on mOT protocol which we call the mOT$^+$ protocol. The differences of mOT and mOT$^+$ are: (1) mOT is not secure if the ephemeral keys of the session are leaked while mOT$^+$ is secure even if the adversary is allowed to obtain both of the parties' ephemeral keys. (2) The assumptions used in mOT are RSA and Knowledge of Exponent while the assumption used in mOT$^+$ is only RSA since there is a verification process, so the additional assumption is not needed. (3) The sKCI attack is not considered in the security proof of mOT while a formal proof that mOT$^+$ is secure against sKCI attack is given in this paper. The details of mOT$^+$ are shown in **Fig. 1**.

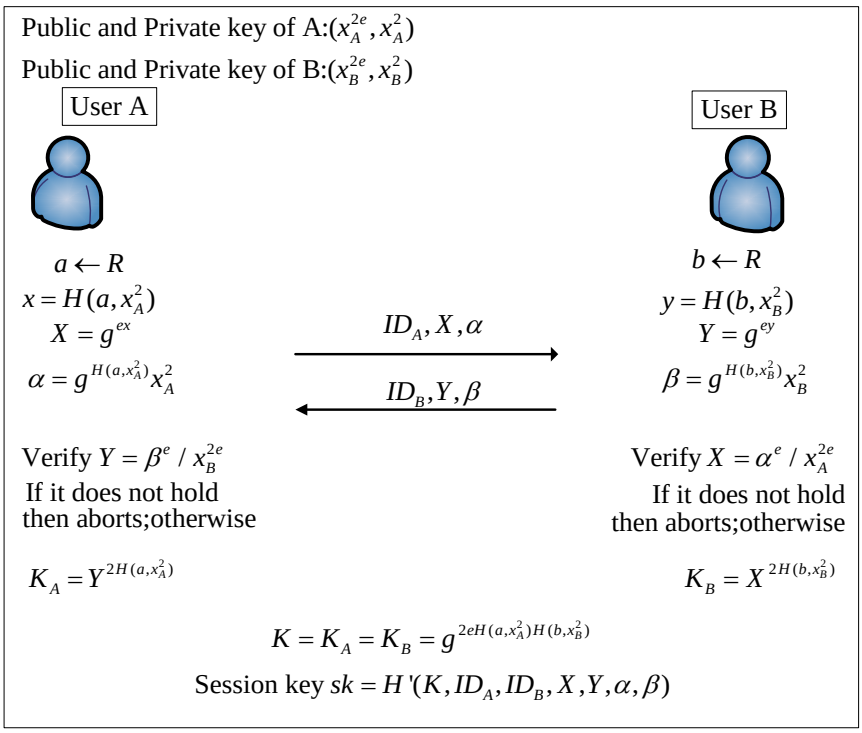

Fig. 1. mOT$^+$ protocol

System Setup: The system chooses a common modulus $N = pq$ as well as a public exponent e for all users, where p and q are two secure large primes and e is prime to $\varphi(N) = (p-1)(q-1)$ mod N. g is a generator of QR_N, where QR_N is a subgroup of quadratic residues mod N. The system publishes the parameters (N, e, g) and erases the factors of N. There are two hash functions: $H: (0,1)^* \rightarrow Z^*_{\lfloor \sqrt{N}/4 \rfloor}$ and $H': (0,1)^* \rightarrow (0,1)^\lambda$, where λ is secure parameter. Each user in the system chooses a random value x_U and sets the

public and private key pair of U as x_U^{2e} mod N and x_U^{2} mod N. All the pubic keys of the users must be certified by the Certificate Authority (CA). We omit mod N in the rest of the paper if there is no ambiguity.

Key exchange: If a user A with identity ID_A and a user B with identity ID_B wants to establish a key exchange protocol with each other, they first pick ephemeral secret keys a and b at random from R where $R = Z_N^*$. Then A computes $x = H(a, x_A^2)$, $X = g^{ex}$ and $\alpha = g^{H(a,x_A^2)} x_A^2$ and sends $\{ID_A, X, \alpha\}$ to B. On the other side B computes $y = H(b, x_b^2)$, $Y = g^{ey}$ and $\beta = g^{H(b,x_B^2)} x_B^2$ and sends $\{ID_B, Y, \beta\}$ to A. On receiving the messages, both of the users check if the messages they received are in Z_N^* and if the equations $Y = \beta^e / x_B^{2e}$ and $X = \alpha^e / x_A^{2e}$ hold or not. If the checks succeed, A computes $K_A = Y^{2H(a,x_A^2)}$ while B computes $K_B = X^{2H(b,x_B^2)}$. The session key is $H'(K, ID_A, ID_B, X, Y, \alpha, \beta)$ where $K = K_A = K_B = g^{2eH(a,x_A^2)H(b,x_B^2)}$. Otherwise, they refuse the messages.

6. Security and Performance Analysis of mOT$^+$ Protocol

6.1 Security Analysis

Before we give the security analysis of mOT$^+$ we give two assumptions and a lemma first.

RSA assumption. Let N be a positive integer and p, q be two odd primes. Let e be a randomly chosen positive integer relatively prime to $\varphi(N) = (p-1)(q-1)$. Given (N, e) and a random value $y \in Z_N^*$, for any probabilistic polynomial time adversary $\mathcal{A}$, it is infeasible to find the value x such that $x^e = y \bmod N$.

Computational Diffie-Hellman (CDH) assumption. For QR_N, let g be a random generator of QR_N. Given two random values $X = g^x \bmod N$ and $Y = g^y \bmod N$, for any probabilistic polynomial time adversary $\mathcal{A}$, it is infeasible to find the value Z such that $Z = g^{xy} \bmod N$.

Lemma 1 [14]. Let $\{N, e, d\}$ be RSA parameters and f be an integer relatively prime to e. There is an efficient procedure that given $\{N, e, f\}$ and a value $(x^f)^d \bmod N$, for $x \in Z_N^*$, computes $x^d \bmod N$.

Theorem 1. *Under the RSA assumption, if we model* H and H' *as random oracles,* mOT$^+$ *is a secure key exchange protocol in the model we proposed above.*

Proof. It is easy to see mOT$^+$ meets the first condition of the **Security Definition** and we do not explain it here. Now we show mOT$^+$ meets the second condition. Let's define a simulator $\mathcal{S}$ who simulates mOT$^+$ to the adversary $\mathcal{A}$. The goal of $\mathcal{A}$ is to guess the bit b in the test session, i.e., distinguishing the session key of the test session from a random value. Let's divide the event that $\mathcal{A}$ successes in the game between $\mathcal{A}$ and $\mathcal{S}$ into two case: (1) The test session $\mathcal{A}$ chooses has a matching session. (2) The test session $\mathcal{A}$ chooses has no matching session.

Matching session exists. Assume that the test session $\mathcal{A}$ selects has a matching session. Then if $\mathcal{A}$ can break the security definition aforementioned, $\mathcal{S}$ can solve the CDH problem

on QR_N. More precisely, upon receiving a CDH challenge (X^*, Y^*) where $X^* = g^x$, $Y^* = g^y \in QR_N$ and g is a generator of QR_N, $\mathcal{S}$ run **System Setup** algorithm and initializes all the information needed including $(N, \bar{g}, e)$ and all the long-term keys of the parties, where e is prime to $\varphi(N)$ and $\bar{g} = g^{2e}$. Then $\mathcal{S}$ chooses two matching sessions between two honest parties A and B. When either of the two sessions is activated, $\mathcal{S}$ follows the mOT$^+$ protocol and chooses two ephemeral keys a and b. Instead of sending $(X = g^{eH(a,x_A^2)}, \alpha = g^{H(a,x_A^2)} x_A^2)$ and $(Y = g^{eH(b,x_B^2)}, \alpha = g^{eH(b,x_B^2)} x_B^2)$, $\mathcal{S}$ sets $\alpha = \bar{g}^{\bar{x}} x_A^2 = g^{2e\bar{x}} x_A^2 = g^x x_A^2 = X^* x_A^2$ and $\beta = \bar{g}^{\bar{y}} x_B^2 = g^{2e\bar{y}} x_B^2 = g^y x_B^2 = Y^* x_B^2$ and also sets $X = (X^*)^e$ and $Y = (Y^*)^e$. With probability $1/q_s^2$ that $\mathcal{A}$ chooses one of the sessions selected by $\mathcal{S}$ as the test session and another as its matching session, where q_s is the total sessions that $\mathcal{A}$ can interact with. Since the protocol is running in the random oracle model, if $\mathcal{A}$ can correctly guess the bit b in the test session, then $\mathcal{A}$ must have asked a hash query to H' by $(\bar{g}^{2e\bar{x}\bar{y}}, ID_A, ID_B, \alpha, \beta)$. Then $\mathcal{S}$ can check the hash list of H' and get the value $\bar{g}^{2e\bar{x}\bar{y}}$ from list of H'. $\mathcal{S}$ can further compute $CDH(X,Y)$ since $\bar{g}^{2e\bar{x}\bar{y}} = g^{2e\bar{x}^2e\bar{y}} = g^{xy} = CDH(X,Y)$. Note $\mathcal{A}$ can ask the ephemeral key reveal query and long-term key reveal query and $\mathcal{S}$ can answer these queries since $\mathcal{S}$ knows all the secrets of the users. However, $\mathcal{A}$ cannot reveal both of the ephemeral key and long-term key of the test session and its matching session. So if $\mathcal{A}$ wins the game only by revealing the ephemeral keys of the communication users, it means $\mathcal{A}$ must have computed x_A^2 or x_B^2 from x_A^{2e} or x_B^{2e}. However, we know this is computationally intractable provided the RSA assumption holds. Since the RSA assumption implies the CDH assumption [14], **Theorem 1** holds when the test session has a matching session.

No matching session. The test session $\mathcal{A}$ chooses has no matching session means that $\mathcal{A}$ impersonates the honest user to participate the test session so there is no matching session at the honest user who is impersonated. In order to give a clear proof here, we list three cases based on whether or when the corruption occurs:

Case 1: $\mathcal{A}$ does not ask any *long-term key reveal* queries to the user in the test session.

Case 2: $\mathcal{A}$ only asks a $long-term\ key\ reveal_b$ query to one of the users in the test session and launches the wKCI attack.

Case 3: $\mathcal{A}$ asks a $long-term\ key\ reveal_b$ query to one of the users in the test session and asks $long-term\ key\ reveal_a$ query to the partner of the user when the test session completes, i.e., launching the sKCI attack.

Case 1. In this case, if $\mathcal{A}$ can break the security definition aforementioned, then we show the simulator $\mathcal{S}$ can solve RSA problem by calling $\mathcal{A}$ as a subroutine.

Set up. $\mathcal{S}$ chooses the pubic keys and long-term private keys for all of the participants except B. More precisely, for all the users except B, $\mathcal{S}$ chooses a random value s_U and sets the public and private key pair of U as (s_U^{2e}, s_U^2). For B, when receiving the RSA tuple (N, e, P), $\mathcal{S}$ sets the public and private key pair of B as $(Z = P^2, null)$. In order to simulate the protocol, $\mathcal{S}$ chooses a random value $\bar{r} \in QR_N$ and sets $r = \bar{r}^e$. The aim of $\mathcal{S}$ is to generate a generator of QR_N. So $\mathcal{S}$ lets $g = (rZ)^e$. If g is not a generator of QR_N, then $\mathcal{S}$

will choose a new random value $\bar{r}$ untill g is a generator of QR_N.

Simulating the non-Test sessions. Since $\mathcal{A}$ can activate mOT$^+$ protocol between any two users and he also can tamper the messages and impersonate honest user in the key exchange process, $\mathcal{S}$ must maintain the consistency of the simulation. Simulating all the participants other than B is trivial since $\mathcal{S}$ knows all of their private keys. When B is activated to send messages in a instance between C and B, $\mathcal{S}$ cannot generate the correct messages since $\mathcal{S}$ does not know the private key of B, i.e., Z^d where $ed = 1 \bmod \phi(N)$. In such case, $\mathcal{S}$ chooses a random value $b \in Z_N^*$ as the ephemeral key, picks *h* at random from $b \in Z_{\lfloor \sqrt{N}/4 \rfloor}^*$ and sets $\beta = g^h / \bar{r}$ and $Y = \beta^e / Z$ instead of $\beta = g^{H(b,Z^d)} Z^d$ and $Y = g^{eH(b,Z^d)}$. Y can pass the verification since $Y = \beta^e / Z$. If $\mathcal{A}$ does not ask a query to H' by $(K, C, B, X, Y, \gamma, \beta)$ in this session where $\gamma / x_C^{2e} = \bar{X}$ and $K = CDH(\gamma / x_C^2, Y)^2 = CDH(\bar{X}, Y)$, $\mathcal{S}$ can choose a random value to answer the session key query. However, if $\mathcal{A}$ can compute *K* and asks a query to H' by $(K, C, B, X, Y, \gamma, \beta)$ then $\mathcal{S}$ has to answer this query consistently since $\mathcal{S}$ cannot answer the session key query directly without knowing the private key of B. In such case, when receiving this query, $\mathcal{S}$ first checks whether there is a record $(K, C, B, X, Y, \gamma, \beta)$ in the hash list which $\mathcal{S}$ keeps. If there is already a record in the hash list, $\mathcal{S}$ answers the query by this record. Otherwise, $\mathcal{S}$ will check whether the following equation holds by $Z = g^d / r$ and $r = \bar{r}^e$:

$$\begin{aligned} K &= CDH(\bar{X}, Y)^2 = CDH(\bar{X}, \beta^e / Z)^2 \\ &= CDH(\bar{X}, (g^h / \bar{r}^e) / Z)^2 = CDH(\bar{X}, (g^h / \bar{r}^e) / (g^d r^{-1}))^2 \\ &= CDH(\bar{X}, g^{eh-d})^2 = \bar{X}^{2(eh-d)} \end{aligned} \tag{1}$$

Although $\mathcal{S}$ does not know *d*, he can verify equation (1) by the following method: $\mathcal{S}$ verifies whether $K^e = \bar{X}^{2e(eh-d)} = \bar{X}^{2(e^2h-1)}$ holds or not. $\mathcal{S}$ can verify this since he knows value *h*. If the verification holds, then $\mathcal{S}$ chooses a random value and marks this value as the session key of this session. Then $\mathcal{S}$ answers the hash query $H'(K, C, B, X, Y, \gamma, \beta)$ by this value and adds this record in hash list. Otherwise, $\mathcal{S}$ just answers it with a random value and also adds the record in hash list. Note the verification is correct since the mapping from $\bar{X}$ to $\bar{X}^e$ is a permutation over Z_N^*. When simulating the protocol, if $\mathcal{A}$ asks the ephemeral key reveal query to B's instance, then $\mathcal{S}$ answers the query with *b*. $\mathcal{A}$ cannot find any abnormal unless he solves the RSA problem and asks a hash query to *H* by (b, Z^d). However, this is impossible if the RSA assumption holds.

Simulating the Test sessions. Suppose the test session is owned by user A and its matching session is owned by user B. When executing the test session between A and B, $\mathcal{S}$ selects a random value *a* as the ephemeral key of A and sets $\alpha = (rZ)^l x_A^2$, $X = \alpha^e / x_A^{2e}$ and $\bar{X} = \alpha / x_A^2$, where $l = te + 1$ and $t \in [1 ... \lfloor N/4 \rfloor]$. Suppose $\bar{X} = \alpha / x_A^2 = g^{\bar{x}}$, then we have $K = Y^{2\bar{x}} = (\beta^e / Z)^{2\bar{x}}$ where β and *Y* are the messages sent from $\mathcal{A}$ who impersonates B to communicate with A. Since $\alpha = (rZ)^l x_A^2$, we can obtain that $\bar{X} = \alpha / x_A^2 = (rZ)^l = (g^d)^l = g^{\bar{x}}$ (recall that we let $g = (rZ)^e$ in the *Setup*). Thus, we have:

$$K = Y^{2\bar{x}} = (\beta^e / Z)^{2\bar{x}} \tag{2}$$

The goal of $\mathcal{S}$ is to compute P^d, i.e., $Z^{d/2}$. From the description above we can see, the simulation is perfect to $\mathcal{A}$. So if $\mathcal{A}$ can distinguish between the session key of the test session and a random value, then $\mathcal{S}$ can solve the RSA problem by calling $\mathcal{A}$ as a subroutine. The reason is that all the queries are answered in the random oracle model in the simulation, so successfully distinguishing the session key of the test session from a random value means $\mathcal{A}$ has asked the hash query to H' with $(K=(\beta^e/B)^{2df}, IDA, IDB, \alpha, \beta)$. Then $\mathcal{S}$ can obtain $(\beta^e/Z)^{2dl}$ by searching the hash list it keeps. It is easy to see that $(Z^{2l})^d=\beta^{2l}/K$. Recall that $\mathcal{S}$ lets $Z=P^2$ so $\mathcal{S}$ can compute $(Z^{2l})^d=(P^{4l})^d=\beta^{2l}/K$. Because $4f$ is prime to e so using the **Lemma 1** we can obtain $P^d=(P^{4l})^d$.

Case 2. When $\mathcal{A}$ asks a *long term key reveal*$_b$ query to A in the test session and launches the wKCI attack, $\mathcal{S}$ can answer all the queries $\mathcal{A}$ asks as in the **Case 1** since $\mathcal{S}$ knows all the participants' long-term keys except B. Note as mentioned above, if $\mathcal{A}$ asks a *long term key reveal*$_b$ query to A in the test session then he cannot impersonate A to communicate with B. The simulation fails if $\mathcal{A}$ asks a *long term key reveal*$_a$ query to B, however, this also means $\mathcal{A}$ cannot guess the session key of the test session. So we can get the same conclusion in the **Case 1**.

Case 3. In this case $\mathcal{A}$ can also ask a *long term key reveal*$_a$ query to B on the basis of **Case 2**. $\mathcal{S}$ cannot simulate the protocol for $\mathcal{A}$ as in **Case 2** since when $\mathcal{A}$ asks *long-term key reveal query* to B, $\mathcal{S}$ cannot answer it. In such case we divide the event that $\mathcal{A}$ generates the correct messages in the test session and guesses the session key of this session into two cases:

Case 3.1: $\mathcal{A}$ chooses Y by himself and generates the corresponding value β and impersonates B to send (Y,β) to other users. Then he computes the session key by the secret value he chose in Y.

Case 3.2: $\mathcal{A}$ generates a new message pair (Y',β') by using the message pairs sent from B and impersonates B to send (Y',β') to other user. Then he computes the session key by some tricks.

Case 3.1. From **Fig.1** we can see, there is a verification before the participants compute the session key in mOT$^+$. If the messages do not pass the verification, then the protocol halts. So if the adversary $\mathcal{A}$ wants to break the security of mOT$^+$, then he must generate the correct messages to pass the verification first. Without loss of generation, we suppose $\mathcal{A}$ chooses $Y=g^y$ in **Case 3.1**. Now if he can generate a correct value β where $Y=\beta^e/x_B^{2e}$, then he can further compute the session key of this session since he knows the value y of Y. However, if this happens then $\mathcal{S}$ can solve RSA problem by calling $\mathcal{A}$ as a subroutine. Because if the equation $Y=\beta^e/x_B^{2e}$ holds where β is generated by $\mathcal{A}$, it means $\mathcal{A}$ can compute $x_B^2=\beta/g^{y/e}$. Here we have to note that $\mathcal{S}$ can check the internal status of $\mathcal{A}$ all the time, so when $\mathcal{A}$ generates a correct messages pair and pass the verification in Case 3.1 then $\mathcal{A}$ stops running the protocol since $\mathcal{S}$ can solve the RSA problem in this point and $\mathcal{A}$ does not need to simulate the protocol any longer. It means if $\mathcal{S}$ asks the *long term key reveal*$_a$ query to B after this point, $\mathcal{S}$ does not answer it. So if the RSA assumption holds, then $\mathcal{A}$ cannot generate the correct messages to pass the verification in **Case 3.1**, i.e., the sKCI attack cannot work in **Case 3.1**.

Case 3.2. In this case, $\mathcal{S}$ chooses all the participants's (including B's) private and public keys by himself and simulates the mOT$^+$ protocol to $\mathcal{A}$. If $\mathcal{A}$ can break the session key security of mOT$^+$ in this case, then $\mathcal{S}$ can solve CDH problem over QR_N. Suppose $\mathcal{A}$ has eavesdropped a message pair (Y,β) from B's instance, he cannot generate a correct message pair (Y',β') which he knows the discrete logarithm of Y' and β' under the RSA assumption (analysis is in **Case 3.1**). So if $\mathcal{A}$ wants to pass the verification in mOT$^+$, he has to do the following steps: $\mathcal{A}$ first chooses a value $t \in [1...\lfloor N/4 \rfloor]$ and lets $Y' = Y * f(t)^e$ and $\beta' = \beta * f(t)$ where $f(t)$ is a function of t and it is chosen by $\mathcal{A}$. We can see the equation $Y' = \beta'^e / x_B^{2e}$ holds since $Y = \beta^e / x_B^{2e}$ holds. Then, $\mathcal{A}$ sends (Y',β') to a participant A who is not aware of being asked by a long-term key reveal query. A follows the protocol honestly and replies $\mathcal{A}$ with (X,α). The aim of $\mathcal{A}$ is to find the session key of this session, i.e., computing $H'(K', A, B, X, Y', \alpha, \beta')$ where $K' = CDH(X, \beta'/x_B^2) = CDH(Y', \alpha / x_A^2)$. On one side, since $\beta' = \beta * f(t)$, we can obtain $CDH(X, \beta'/x_B^2) = CDH(X, \beta / x_B^2) * CDH(X, f(t))$. Now $\mathcal{A}$ can compute $CDH(X, f(t))$ with the knowledge of $f(t)$. After the test session is expired, $\mathcal{A}$ can also ask the *long term key reveal*$_a$ query to B and obtains x_B, however, $\mathcal{A}$ cannot compute $CDH(X, \beta / x_B^2)$ without knowing the exponent of X and β provided the CDH assumption holds. Note $\mathcal{A}$ cannot ask the ephemeral key reveal query to the session $\prod_U^i$ if the long-term key reveal attack has been asked to U. On the other side, $Y' = Y * f(t)^e$, then we can obtain $CDH(Y', \alpha / x_A^2) = CDH(Y, \alpha / x_A^2) * CDH(f(t)^e, \alpha / x_A^2)$. $\mathcal{A}$ can compute $CDH(f(t)^e, \alpha / x_A^2)$ since he knows the structure of $f(t)^e$ and x_A^2 ($\mathcal{A}$ has asked a *long term key reveal*$_b$ query to A). However, $\mathcal{A}$ cannot compute $CDH(Y, \alpha / x_A^2)$ without knowing the exponent of Y and α provided the CDH assumption holds. So the sKCI attack cannot work in **Case 3.2**.

Sum up all the analysis above, if $\mathcal{A}$ cannot break the RSA assumption (RSA assumption implies the CDH assumption for QR [11]) then the simulation of the mOT$^+$ is perfect and $\mathcal{A}$ cannot get any advantage in guessing the session key of the test session in the random oracle model, i.e., the probability of $\mathcal{A}$ succeeds in guessing the session key of the test session is not more than:

$$1/2 + poly(q_s) * Adv_N^{RSA}(t) \tag{3}$$

Where q_s denotes the total sessions that A can interact with, $poly(q_s)$ denotes a polynomial of q_s and $Adv_N^{RSA}(t)$ denotes the probability that one breaks the RSA assumption in time t. It is easy to see that $poly(q_s) * Adv_N^{RSA}(t)$ is a negligible fraction based on RSA assumption. So we can draw the conclusion of **Theorem 1**.

6.2 Performance

We show the performance analysis of mOT$^+$ protocol by comparing with some one-round key exchange protocols in term of the security model, underlying assumption, whether or not it is sKCI-resistance and the computation cost. The comparisons are shown in **Table 1**. In **Table 1**, we list several security models for key exchange protocols. CK model captures the forward secrecy, however, it does not capture the ephemeral key reveal attack. So when an adversary

who has the ability of revealing the ephemeral key, then he will recover the session key. eCK model captures the ephemeral key reveal resistance property, however, it does not capture sFS. So both of CK and eCK are not perfect. sFS model means a model who captures the strong forward secrecy.

Table 1. Comparisons between one-round key exchange protocols

Protocols	Security model	Assumption	sKCI-resistance	ephemeral key reveal-resistance	Computation	Computation cost after precomputaion
HMQV	CK_{HMQV}	GDH,KEA1	No	Yes	3exp	2exp
NAXOS	eCK	GDH	No	Yes	4exp	3exp
Boyd and Nieto's	eCK,sFS	GDH,MAC	No	Yes	5exp	3exp
SIG-DH	CK, sFS	DDH	Yes	No	2exp+1sig+ 1verif	1exp+1verif
SIG-DH	CK, sFS	DDH	Yes	No	5exp	3exp
mOT	CK,sFS	RSA,KEA1	Yes	No	3exp	2exp
SIG(NAXOS)	eCK^w,sFS	GDH	Yes	Yes	4exp+1sig+ 1verif	3exp+1verif
SIG(NAXOS)	eCK^w,sFS	GDH	Yes	Yes	7exp	5exp
mOT^+	Ours	RSA	Yes	Yes	4exp	2exp

exp: exponent operation; sig: signature operation; verif: signature verification operation.
The concrete signature algorithm we use in Table 1 is the DSA signature. The computation cost of DSA signature is 1exp and the verification cost of DSA is 2exp.

From the comparison we can see, there are two advantages of mOT^+. On one hand, compared with HMQV, NAXOS and Boyd and Nieto's protocol, mOT^+ can provide sFS as well as the sKCI-resistance property. On the other hand, compared with SIG-DH [1] and mOT [14], mOT^+ is secure against ephemeral key reveal attack and a weak underlying assumption is used. As we know, the weaker the underlying assumption that the key exchange protocol bases on, the stronger the security it can provide. Cramers and Feltz's protocol [17] has the same security property with mOT^+, however, it is not easy to transform all the key exchane protocols into protocols with eCK-PFS using the compiler in [17]. So the practical application needs to be tested. Meanwhile, the computation cost of mOT^+ as shown in **Table 1** is also more efficient than the protocol in [17]. Therefore, based on an overall consideration of these security properties and efficiency, mOT^+ has better performance compared with related one-round key exchange protocols.

7. Conclusion

In this paper, we propose a new kind of attack to the key exchange protocol which is called the strong key compromise impersonation (sKCI) attack. Then, we propose a strong security model for the key exchange protocol which considers not only what keys can be compromised but also when they are compromised. In order to give a key exchange protocol which is secure in the propose model, we propose a strongly secure one-round key exchange protocol called mOT^+ based on the mOT protocol. The new attack and the new security model may provide a

new topic in analyzing the security of the key exchange protocol. Our further work will be on designing a secure one-round key exchange protocol whose computation cost is similar to that of the Diffie-Hellman key exchange protocol.

References

[1] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," *Advances in Cryptology-EUROCRYPT*, pp. 451-472, 2001. Article (CrossRef Link)

[2] B. LaMacchia, K. Lauter and A. Mityagin, "Stronger security of authenticated key exchange," in *Proc. of Provable Security*, pp. 1-16, 2007. Article (CrossRef Link)

[3] W. Diffie, P.V. Oorschot and M. Wiener, "Authentication and authenticated key exchange," *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107-125, 1992. Article (CrossRef Link)

[4] N.T.T. Huyen, M. Jo, T.D. Nguyen, and E.N. Huh, "A beneficial analysis of deployment knowledge for key distribution in wireless sensor networks," *Security and Communication Network*, vol.5, no.5, pp.485-495, 2011. Article (CrossRef Link)

[5] H.T.T. Nguyen, M. Guizani, M. Jo, and E.N. Huh, "An efficient signal-range-based probabilistic key pre-distribution scheme in a wireless sensor network," *IEEE Transactions on Vehicular Technology*, vol. 58, no.5, pp.2482-2497, 2009. Article (CrossRef Link)

[6] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," *Advances in Cryptology -CRYPTO*, pp. 546-566, 2005. Article (CrossRef Link)

[7] D. Mishra, A.K. Das, A. Chaturvedi, and S. Mukhopadhyay, "A secure password-based authentication and key agreement scheme using smart cards," *Journal of Information Security and Applications*, vol. 23, pp. 28-43, 2015. Article (CrossRef Link)

[8] D. Mishra and S. Mukhopadhyay, "Cryptanalysis of pairing-free identity-based authenticated key agreement protocols," in *Proc. of International Conference on Information Systems Security*, pp.247-254, 2013. Article (CrossRef Link)

[9] Z. Yang, W. Yang, L. Zhu, and D. Zhang, "Towards modelling perfect forward secrecy in two-message authenticated key exchange under ephemeral-key revelation," *Security and Communication Networks*, vol. 8, no. 18, pp. 3356–3371, 2015. Article (CrossRef Link)

[10] G. Barthe, J.M. Crespo, Y. Lakhnech, and B. Schmidt, "Mind the gap: modular machine-checked proofs of one-round key exchange protocols," *Advances in Cryptology-EUROCRYPT*, pp. 689-718, 2015. Article (CrossRef Link)

[11] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, 1976. Article (CrossRef Link)

[12] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An efficient protocol for authenticated key agreement," *Designs, Codes and Cryptography*, vol. 28, no. 2, pp. 119-134, 2008. Article (CrossRef Link)

[13] S. Chow and K. Choo, "Strongly-secure identity-based key agreement and anonymous extension," in *Proc. of Information Security*, pp.203-220, 2007. Article (CrossRef Link)

[14] R. Gennaro, H. Krawczyk and T. Rabin, "Okamoto-Tanaka revisited: fully authenticated Diffie-Hellman with minimal overhead," in *Proc. of Applied Cryptography and Network Security*, pp. 309-328, 2010. Article (CrossRef Link)

[15] E. Okamoto and K. Tanaka, "Key distribution system based on identification information," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 4, pp. 481-485, 1989. Article (CrossRef Link)

[16] C. Boyd and J. Nieto, "On forward secrecy in one-round key exchange," in *Proc. of Cryptography and Coding*, pp. 451-468, 2011. Article (CrossRef Link)

[17] C. Cremers and M. Feltz, "Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal," *Designs, Codes and Cryptography*, vol. 74, no. 1, pp.183-218, 2015. Article (CrossRef Link)

[18] F. Bergsma, T. Jager and J. Schwenk, "One-round key exchange with strong security: an efficient and generic construction in the standard model," in *Proc. of PKC* 2015, pp. 477-494, 2015. Article (CrossRef Link)
[19] M. Just and S. Vaudenay, "Authenticated multi-party key agreement," *Advances in Cryptology-ASIACRYPT*, pp. 36-49, 1996. Article (CrossRef Link)
[20] M.A. Strangio, "On the resilience of key agreement protocols to key compromise impersonation," in *Proc. of European PKI Workshop on Public Key Infrastructure*, pp. 233-247, 2006. Article (CrossRef Link)

Xiaowei Li He received his B.S. degree in Mathematics and Applied Mathematics from Xidian University, China, in 2008. He received his doctor's degree in Information Security from Xidian University, China, in 2013. He is currently a lecturer in Department of Mathematics and Computer Science, Dali University, China. He has published many papers in International Journals and Conferences including Security and Communication Networks, KSII Transactions On Internet and Information Systems and Globecom. His research interests include network protocol security, wireless network security and cloud computing security.

Dengqi Yang received the B.E. degree in Information and Computing Science from Yunnan University, China in 2003, the M.Sc. degree in Computational Mathematics from Yunnan University, China in 2006, and the Ph.D. degree in Computer Science from Sichuan University, China in 2012. He worked, as an Associate Professor, at Dali University, China. His research interests mainly focus on network architecture and information security.

Benhui Chen received the B.E. degree in Computer Science from Yunnan University, China in 1999, the M.Sc. degree in Fundamental Mathematics from Yunnan Normal University, China in 2005, and the Ph.D degree in Computer Science from Waseda University, Japan in 2011. From April 2008 to Mar. 2011, he had been a doctor candidate at Graduate School of Information, Production and Systems of Waseda University, Japan. From 1999 to 2011, he worked as a Research Associate, Lecturer and Associate Professor, and since August 2011, he has been a Professor at Dali University, China. His current research interests include Computational Intelligence such as Support Vector Machine and Evolutionary Algorithms, and their applications to bioinformatics, computer networks, and so on.

Yuqing Zhang is a professor and supervisor of Ph.D. candidates of Graduate University of Chinese Academy of Sciences. He received his B.S. and M.S. degree in computer science from Xidian University, China, in 1987 and 1990 respectively. He received his Ph.D. degree in Cryptography from Xidian University in 2000. He is a member of IEEE Communications Society and IEICE Transactions on Communications. He has published lots of papers in International Journals and conferences including IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Wireless Communications, IEEE Communications Letters, Globecom, RAID and so on. His research interests include cryptography and network security.