Secure Tripartite STS key Agreement Protocol in Random Oracle Model[1]

Chin-Feng Lee
Department of Information Management, ChaoYang University of Technology

Hung-Yu Chien*
Department of Information Management, National Chi Nan University

Chi-Sung Lai
Department of Electrical Engineering, National Cheng Kung University

Abstract

The Station-to-Station (STS) protocol is a well known two-party key agreement scheme that provides mutual entity authentication, key confirmation and forward secrecy. Al-Riyami and Paterson (2003) extended the STS protocol to the tripartite case, which is called TAKC-STS and is believed to be secure and pass-optimal for tripartite key confirmation protocols. However, in this paper, we will show that the TAKC-STS protocol cannot resist the man-in-the-middle attack and the insider attack. We then propose a secure tripartite STS protocol to conquer the weaknesses, and prove the security in the random oracle model.

Key words : security, key agreement, insider attack, STS, forward secrecy, man-in-the-middle attack

* Corresponding Author

[1] Acknowledgments: This work is partially supported by partially supported by Taiwan National Science Council with project number NSC97-2221-E-260-008-MY2, NSC 98-2221-E-324-020 and NSC 99-2219-E-006-011.

安全的三方式 STS 金鑰協訂

李金鳳
朝陽科技大學資訊管理系

簡宏宇*
國立暨南國際大學資訊管理學系

賴溪松
國立成功大學電機工程學系

摘要

著名的STS 通訊協訂可提供雙方認證、金鑰確認、及向前安全等功能。在 2003年Al-Riyami 及Paterson 學者將 STS 協訂擴充成三方式認證金鑰，並稱之為 TAKC-STS 協訂；迄今，學界認為TAKC-STS 協訂可提供足夠安全並可達成最好的回合數。此篇論文將指出此機制無法抵擋中間人攻擊及內部攻擊；我們也將提出一安全的三方式認證金鑰並證明其安全。

關鍵字：安全、金鑰協商協議、內部攻擊、STS、向前安全、中間人攻擊

* 通訊作者

1. Introduction

The Station-to-Station (STS) protocol is a well known two-party key agreement scheme based on classic Diffie-Hellman that provides mutual key and entity authentication. STS was originally presented in 1987 in the context of ISDN security [18], finalized in 1989 and generally presented in 1992 [13]. Important features of the STS protocol include no timestamps, key confirmation and perfect forward secrecy. Its explicit key confirmation makes it an authenticated key agreement with key confirmation (AKC) protocol. STS has inspired the following design of many two-party key agreement schemes like [10, 11, 12, 17, 23].

Due to the practical importance, the study of tripartite (three-party) key agreement schemes has attracted the attention of many researchers recently like [1, 2, 8, 9, 14, 15, 16, 20, 21]. A tripartite key agreement protocol allows three parties establish session keys. The three-party (or tripartite) case is of most practical importance not only because it is the most common size for electronic conferences but also because it can be used to provide a range of services for two parties communicating. For example, a third party can be added to chair, or referee a conversation for ad hoc auditing, data recovery or escrow purposes [1, 2, 15]. It can also facilitate the job of group communication.

However, many existing tripartite key protocols like [1, 2, 14, 15, 16, 20] suffer distinct degree of security weaknesses. Joux's scheme cannot resist the basic man-in-the-middle attack, Shim's scheme [20] was found to be vulnerable to the key compromise impersonation attack, and the Lin-Lin scheme [16] did not consider the insider attack. In the insider attack [8, 9], an insider *A* might be able to fool *B* into believing that they have participated in a protocol run with *C*, while in fact *C* has not been active. This kind of insider attack could result in serious fraud and loss, if the impersonated *C* acts as an on-line escrow agent or a referee. To resist the insider attack, Al-Riyma and Paterson [1, 2] extended the two-party STS protocol to the tripartite case, which is called the Tripartite Authenticated Key Confirmation STS (TAKC-STS) protocol and *TAKC-STS has been claimed and believed to be secure against insider attack, man-in-the-middle attack and providing key confirmation* (even though they did not formally proved this protocol in the original paper, and the previous models like [3, 4, 6, 19] did not cover insider attack). However, we find that the TAKC-STS protocol cannot resist the basic man-in-the-middle attack and the insider attack, and fails to commit key confirmation.

In this paper, we will show the weaknesses, and then propose a secure tripartite STS protocol that conquers the security weaknesses. The security of the proposed tripartite STS protocol is examined in the random oracle model. *However, we should note that, as Canetti et al. [7] had noted, a protocol proved secure in the random model does not imply there exist any secure implementations of the protocol in the real world, instead it serves as an engineering approach to rule out in-secure designs. Another tool used to verify the conformance and security of a protocol is Burrows et al,'s BAN logic [5]; however, in our opinions, BAN logic is strong in conformance testing but weak in security proof because there exist quite a few security protocol proved in BAN logic has been reported in-secure like [12]. Therefore, we still apply random oracle model to prove the security.*

2. Security weaknesses of TAKC-STS

Diffie, van Oorschot and Wiener [13] proposed a three-pass, two party key agreement protocol, the STS protocol, to defeat man-in-the-middle attacks. Al-Riyma and Paterson [1, 2] extended the STS protocol to three-parties and six-pass key confirmation protocol in the non-broadcast environment. This protocol is called the TAKC-STS protocol in this paper. TAKC-STS has been believed to be secure. However, this section will show the weaknesses.

2.1 Review of TAKC-STS protocol

In the protocol below, an appropriate prime p and a generator g for the multiplicative group Z_p^* are selected. Notation g^a denotes $g^a \bmod p$, and we omit modulo p operations in the rest of this paper for simplicity. $a, b, c \in Z_p^*$ are randomly ephemeral values selected by A, B and C respectively. I_A denotes the identity of A, $Cert_A$ denotes the public key certificate of A, $S_A()$ denotes A's signature, and $E_{K_{ABC}}()$ denotes symmetric encryption under the session key K_{ABC}. The TAKC-STS protocol is depicted as follows.

TAKC-STS

$$A \rightarrow B: \ g^a \| Cert_A; \tag{1}$$

$$B \rightarrow C: \ g^a \| Cert_A \| g^b \| Cert_B \| g^{ab} \tag{2}$$

$$C \rightarrow A: \ g^b \| Cert_B \| g^c \| Cert_C \| g^{bc} \| E_{K_{ABC}}(S_C(I_A \| I_B \| X)), \tag{3}$$

where $X = g^a \| g^b \| g^c$

$$A \to B:\ g^c \| Cert_C \| g^{ac} \| E_{K_{ABC}}(S_C(I_A \| I_B \| X)) \| E_{K_{ABC}}(S_A(I_B \| I_C \| X)) \quad (4)$$

$$B \to C:\ E_{K_{ABC}}(S_A(I_B \| I_C \| X)) \| E_{K_{ABC}}(S_B(I_A \| I_C \| X)) \quad (5)$$

$$B \to A:\ E_{K_{ABC}}(S_B(I_A \| I_C \| X)) \quad (6)$$

Entity *A* initiates the protocol by sending the message in (1). After that, *B* forwards the message, his ephemeral public key g^b and g^{ab} to *C*, where g^{ab} is computed using g^a and *b*. After receiving the messages in (2), *C* is able to compute the session key $K_{ABC} = (g^{ab})^c \bmod p$ and uses this key to encrypt his signature $S_C(I_A \| I_B \| X)$. Upon receiving the messages in (3), *A* can compute $K_{ABC} = (g^{bc})^a \bmod p$ and sends the messages in (4). Likewise, *B* can compute $K_{ABC} = (g^{ac})^b \bmod p$ and sends the messages in (5). The signatures in (3-6) are encrypted using the key K_{ABC}. So, if each receiver can use his key to decrypt the encrypted signature then he is assured of the session key. This provides key confirmation function.

2.2 Man-in-the-middle attack on TAKC-STS

Now we show the man-in-the-middle-attack on the TAKC-STS protocol. Here *E* denotes the adversary sitting between entities *A*, *B*, and *C*. The notation $A \not\to_E C$ means that the messages sent by *A* to *C* are intercepted by *E*, and the notation *A(E)*→*C* means that *E* impersonates *A* to send messages to *C*. $K_{AB\overline{C}}$ denotes the session key computed by *C* and to be used among *A*, *B* and *C*. $K_{AB\overline{C}E \to A}$ denotes the session key was originally computed by *C*, but is modified by the adversary *E* for *A*. $t, u, v \in Z_p^*$ are randomly ephemeral values selected by *E*.

The key idea behind the attack is that, given g^x, g^y and g^z, one cannot compute the value g^{xy} and cannot verify whether $g^z = g^{xy}$. Therefore, the adversary can modify the ephemeral D-H values g^{ab}, g^{bc} and g^{ac} without the receivers' notice. Applying this technique, the adversary can control the session key computed by each receiver respectively, and uses the fake session keys to re-encrypt each signer's signature. The attack scenario is as follows.

$$A \to B:\ g^a \| Cert_A; \quad (1)$$

$$B \not\to_E C:\ g^a \| Cert_A \| g^b \| Cert_B \| g^{ab} \quad (2)$$

$$B(E) \to C:\ g^a \| Cert_A \| g^b \| Cert_B \| g^t \quad (2')$$

$$C \twoheadrightarrow_E A:\ g^b \| Cert_B \| g^c \| Cert_C \| g^{bc} \| E_{K_{AB\bar{C}}}(S_C(I_A \| I_B \| X)), \quad (3)$$

where $X = g^a \| g^b \| g^c$ and $K_{AB\bar{C}} = g^{tc}$

$$C(E) \rightarrow A:\ g^b \| Cert_B \| g^c \| Cert_C \| g^u \| E_{K_{AB\bar{C}E\rightarrow A}}(S_C(I_A \| I_B \| X)), \quad (3')$$

where $K_{AB\bar{C}E\rightarrow A} = g^{ua}$

$$A \twoheadrightarrow_E B: g^c \| Cert_C \| g^{ac} \| E_{K_{AB\bar{C}E\rightarrow A}}(S_C(I_A \| I_B \| X)) \| E_{K_{\bar{A}BC}}(S_A(I_B \| I_C \| X)), \quad (4)$$

where $K_{AB\bar{C}E\rightarrow A} = K_{\bar{A}BC} = g^{ua}$

$$A(E) \rightarrow B: g^c \| Cert_C \| g^v \| E_{K_{AB\bar{C}E\rightarrow B}}(S_C(I_A \| I_B \| X)) \| E_{K_{\bar{A}BCE\rightarrow B}}(S_A(I_B \| I_C \| X)), \quad (4')$$

where $K_{AB\bar{C}E\rightarrow B} = K_{\bar{A}BCE\rightarrow B} = g^{vb}$

$$B \twoheadrightarrow_E C:\ E_{K_{\bar{A}BCE\rightarrow B}}(S_A(I_B \| I_C \| X)) \| E_{K_{A\bar{B}C}}(S_B(I_A \| I_C \| X)), \quad (5)$$

where $K_{\bar{A}BCE\rightarrow B} = K_{A\bar{B}C} = g^{vb}$

$$B(E) \rightarrow C:\ E_{K_{\bar{A}BCE\rightarrow C}}(S_A(I_B \| I_C \| X)) \| E_{K_{A\bar{B}CE\rightarrow C}}(S_B(I_A \| I_C \| X)), \quad (5')$$

where $K_{\bar{A}BCE\rightarrow C} = K_{A\bar{B}CE\rightarrow C} = K_{AB\bar{C}} = g^{tc}$

$$B \twoheadrightarrow_E A:\ E_{K_{A\bar{B}C}}(S_B(I_A \| I_C \| X)) \quad (6)$$

$$B(E) \rightarrow A:\ E_{K_{A\bar{B}CE\rightarrow A}}(S_B(I_A \| I_C \| X)), \quad (6')$$

where $K_{A\bar{B}CE\rightarrow A} = K_{AB\bar{C}E\rightarrow A} = K_{\bar{A}BC} = g^{ua}$

The adversary *E* starts to intercept the messages from Pass (2), where the messages sent by *B* to *C* are intercepted by *E*. *E* forwards the intercepted data except that g^{ab} is replaced with g^t in (2'). Upon receiving the modified data, *C* wrongly believes that the Diffie-Hellman (D-H) key between *A* and *B* is g^t and computes his session key $K_{AB\bar{C}} = g^{tc}$. *C* then sends the data in (3) that include the encryption of *C*'s signature using the key $K_{AB\bar{C}}$, but the messages are intercepted by *E*. Since *E* can compute the key $K_{AB\bar{C}}$, he can decrypt the data in (3) to derive *C*'s signature $S_C(I_A \| I_B \| X)$. *E* replaces the D-H key between *B* and *C* with g^u and encrypts *C*'s signature $S_C(I_A \| I_B \| X)$ using the key $E_{K_{AB\bar{C}E\rightarrow A}} = g^{ua}$ in Pass (3') so that *A* will

wrongly accept the key confirmation data $E_{K_{AB\overline{C}E\to A}}(S_C(I_A \| I_B \| X))$, because A will compute his session key $K_{\overline{A}BC} = g^{ua} = K_{AB\overline{C}E\to A}$ and can use this key to derive C's signature from $E_{K_{AB\overline{C}E\to A}}(S_C(I_A \| I_B \| X))$. So A will send the messages in Pass (4), where the messages are intercepted by E. E replaces the D-H key g^{ac} between A and C with the key g^{v}, and encrypts A's and C's signatures using the key g^{vb} in (4') so that B will wrongly compute the key as g^{vb} and will use this key to derive A's and C's signatures. B also uses this wrong key to encrypt his signature in (5), where the messages are intercepted by E, too. E re-encrypts the signatures in (5') such that C can use his computed key $K_{AB\overline{C}} = g^{tc}$ to derive the signatures. This makes C wrongly believe that A and B uses the same key $K_{AB\overline{C}}$, but actually they do not. B sends his key confirmation to A in (6), but the message is modified by E in (6') such that A can use his wrong key g^{ua} to decrypt the encrypted signature.

The key factor of this attack is that a receiver cannot verify the received D.-H. key even if he has the two ephemeral values g^{a} and g^{b}. So, the attacker can replace the D.-H. key with a random value chosen by him without the receiver's notice. Therefore, he can control the session keys computed by A, B and C respectively. The attacker also use these fake keys to re-encrypt A's, B's and C's signatures such that the designated receivers can use the wrong key to decrypt the encrypted signatures. These factors make the attack successful. Finally, the adversary E shares the key $K_{A\overline{B}CE\to A} = K_{AB\overline{C}E\to A} = K_{\overline{A}BC} = g^{ua}$ with A, the key $K_{AB\overline{C}E\to B} = K_{\overline{A}BCE\to B} = K_{A\overline{B}C} = g^{vb}$ with B, and the key $K_{\overline{A}BCE\to C} = K_{A\overline{B}CE\to C} = K_{AB\overline{C}} = g^{tc}$ with C respectively. Unfortunately, A, B and C wrongly believe that they share the same key with their designated receivers. The main-in-the-middle attack succeeds.

2.3 Insider attack

In addition to the above attack, we will show that an insider (say B) can easily fool one party (say C) into accepting a wrong key such that C will be excluded from the communications. This might result in serious risk, for example if C acts as an on-line escrow agent, an auditor or a referee. If B could impersonate C to A, then B can communicate with or performing transactions with A; whereas A would do the transactions or communications only if C (the referee) is monitoring the contents on-line. With no referee involved, this might cause serious risk for A. We demonstrate one example attack as follows. In the following, the notation $K_{AB\overline{C}(B)\to A}$ denotes that the key was originally computed by C for A but is modified by B.

$$A \rightarrow B:\ g^a \| Cert_A; \quad (1)$$

$$B \rightarrow C:\ g^a \| Cert_A \| g^b \| Cert_B \| g^t \quad (2)$$

$$C \not\rightarrow_B A:\ g^b \| Cert_B \| g^c \| Cert_C \| g^{bc} \| E_{K_{AB\overline{C}}}(S_C(I_A \| I_B \| X)), \quad (3)$$

where $X = g^a \| g^b \| g^c$ and $K_{AB\overline{C}} = g^{tc}$

$$C(B) \rightarrow A:\ g^b \| Cert_B \| g^c \| Cert_C \| g^{bc} \| E_{K_{AB\overline{C}(B)\rightarrow A}}(S_C(I_A \| I_B \| X)), \quad (3')$$

where $K_{AB\overline{C}(B)\rightarrow A} = g^{abc}$

$$A \rightarrow B: g^c \| Cert_C \| g^{ac} \| E_{K_{AB\overline{C}(B)\rightarrow A}}(S_C(I_A \| I_B \| X)) \| E_{K_{\overline{A}BC}}(S_A(I_B \| I_C \| X)), \quad (4)$$

where $K_{AB\overline{C}(B)\rightarrow A} = K_{\overline{A}BC} = g^{abc}$

$$B \rightarrow C:\ E_{K_{\overline{A}BC(B)\rightarrow C}}(S_A(I_B \| I_C \| X)) \| E_{K_{A\overline{B}C\rightarrow C}}(S_B(I_A \| I_C \| X)), \quad (5)$$

where $K_{A\overline{B}C\rightarrow C} = K_{\overline{A}BC(B)\rightarrow C} = K_{AB\overline{C}} = g^{tc}$

$$B \rightarrow A:\ E_{K_{A\overline{B}C}}(S_B(I_A \| I_C \| X)) \quad (6)$$

where $K_{A\overline{B}C} = g^{abc}$

The goal of the inside attacker, *B*, is to let *C* accept a wrong key $K_{AB\overline{C}} = g^{tc}$, whereas *B* and *A* share the same key $K_{\overline{A}BC} = K_{A\overline{B}C} = g^{abc}$ so that *C* will be excluded from the communications. In Pass (2), *B* honestly sends the two ephemeral values g^a and g^b, but, instead of a correct D-H key g^{ab}, sends a wrong value g^t such that C will wrongly compute $K_{AB\overline{C}} = g^{tc}$. In order to successfully cheat *C* and *A*, *B* should intercept the messages in Pass (3), and replaces $E_{K_{AB\overline{C}}}(S_C(I_A \| I_B \| X))$ with $E_{K_{AB\overline{C}E\rightarrow A}}(S_C(I_A \| I_B \| X))$ in pass (3') such that *A* can decrypt the message without noticing the cheating. *A* can generate the correct messages in Pass (4), and *B* uses the wrong key g^{tc} to encrypt the signature such that *C* will not notice the cheating in Pass (5). Finally, *A* and *B* share the same key $K_{\overline{A}BC} = K_{A\overline{B}C} = g^{abc}$, but C owns a wrong key g^{tc}.

3. Secure Tripartite STS (Tri-STS) protocol

To conquer the security weaknesses of TAKC-STS, we, based on the CDHP problem and cryptographic one-way hash function, shall propose a new tripartite STS protocol. The security of the proposed scheme can be proved in modified Bellare-Rogaway model [8, 10].

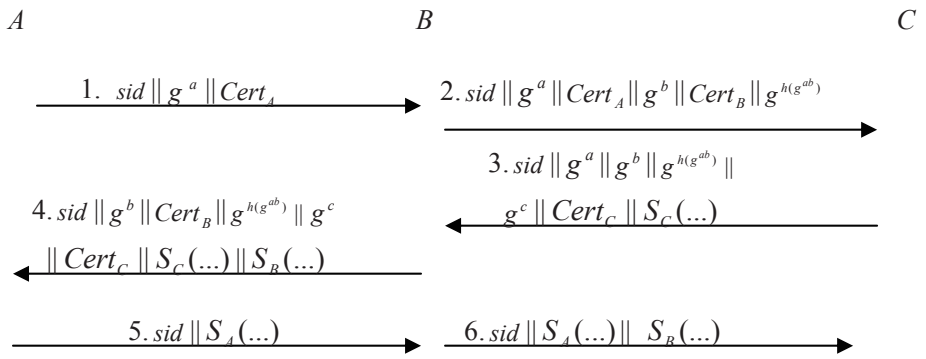

Figure 1. Tripartite STS protocol

Definition 1. Computational Diffie-Hellman problem (CDHP): Given $g^a \bmod p$ and $g^b \bmod p$, where a and b are random numbers form Z_{q-1}, compute $g^{ab} \bmod p$.

Now we are ready to introduce our Tri-STS protocol. In the protocol below, an appropriate prime p and a generator g for the multiplicative group Z_p^* are selected. sid denotes the session identifier that can uniquely identify one session from others, $S_A()$ denotes the signature of entity A and the underlying signature scheme (for example, [22]) is secure against adaptively chosen message attack. $h()$ denotes a cryptographic one-way function, and can be used as a key derivation function. Notation g^a denotes $g^a \bmod p$, and we omit modulo p operations in the rest of this paper for simplicity. $a, b, c \in Z_p^*$ are randomly ephemeral values selected by A, B and C respectively. K_{ABC} denotes the final session key, I_A denotes the identity of A, and $Cert_A$ denotes the public key certificate of A. We describe the protocol in a non-broadcast environment, and we can easily modify it with less message runs if broadcast environment is available. The protocol is depicted in Figure 1, and is described in the following.

(1) $A \rightarrow B$: $sid \| g^a \| Cert_A$

(2) $B \rightarrow C$: $sid \| g^a \| Cert_A \| g^b \| Cert_B \| g^{h(g^{ab})}$

(3) $C \rightarrow B$: $sid \| g^a \| g^b \| g^{h(g^{ab})} \| g^c \| Cert_C \| S_C(X)$

, where C computes $K_{ABC} = h(sid \| I_A \| I_B \| I_C \| g^a \| g^b \| g^c \| g^{h(g^{ab})} \| g^{c \cdot h(g^{ab})})$ and generates its signature on $X = h(sid \| I_A \| I_B \| I_C \| g^a \| g^b \| g^{h(g^{ab})} \| g^c)$.

(4) $B \rightarrow A$: $sid \| g^b \| Cert_B \| g^{h(g^{ab})} \| g^c \| Cert_C \| S_C(X) \| S_B(X)$,

, where B computes $K_{ABC} = h(sid \| I_A \| I_B \| I_C \| g^a \| g^b \| g^c \| g^{h(g^{ab})} \| g^{c \cdot h(g^{ab})})$ and generates its signature on $X = h(sid \| I_A \| I_B \| I_C \| g^a \| g^b \| g^{h(g^{ab})} \| g^c)$.

(5) $A \rightarrow B$: $sid \| S_A(X)$

, where A computes $K_{ABC} = h(sid \| I_A \| I_B \| I_C \| g^a \| g^b \| g^c \| g^{h(g^{ab})} \| g^{c \cdot h(g^{ab})})$ and generates its signature on $X = h(sid \| I_A \| I_B \| I_C \| g^a \| g^b \| g^{h(g^{ab})} \| g^c)$.

(6) $B \rightarrow C$: $sid \| S_A(X) \| S_B(X)$

Step 1 is the same as that of the TAKC-STS protocol, but we requires entity B to convey the value $g^{h(g^{ab})}$ to entity C in step 2 such that C can compute the session key $K_{ABC} = h(g^{c \cdot h(g^{ab})})$ using its random integer *c*. C also generates its signature on the ephemeral public values $X = h(sid \| I_A \| I_B \| I_C \| g^a \| g^b \| g^{h(g^{ab})} \| g^c)$. The final session key is $K_{ABC} = h(sid \| I_A \| I_B \| I_C \| g^a \| g^b \| g^c \| g^{h(g^{ab})} \| g^{c \cdot h(g^{ab})})$. Since both A and B can compute g^{ab}, $h(g^{ab})$ and $g^{h(g^{ab})}$, they can also compute the value $g^{c \cdot h(g^{ab})}$ and the session key K_{ABC}. In steps 4~6, A and B respectively generate the signatures on the ephemeral public values. Only when the signatures from the communicating parties are successfully verified, A, B and C will accept the session key.

It is easy to extend to key confirmation function by appending the hash value of the session key in the message to be signed. Since each session key depends on the ephemeral values *X* and the signatures is generated on the related data- identities and the ephemeral values, the proposed scheme achieve the forward secrecy, key confirmation, and explicit authentication. In the next section, we will prove the security in the random oracle model.

4. Security Notations and Proof

The security of the proposed schemes concerns the privacy of the authenticated session key. To capture the security of the tripartite key agreement scheme, we should consider the in-distinguishability property [3, 4, 6, 10], and the resistance to key-compromise impersonation and the insider attack. In all the models of BR95 [3] and BPR2000 [4], a session with corrupted entities is not considered as fresh; therefore, it cannot model the key-compromise impersonation and the insider attack. We, therefore, prove the in-distinguishability in a modified model [8, 10], and gain the advantage of insider attack and key-compromise impersonation attack related to the advantage of forging advantage of underlying signature scheme. Regarding the in-distinguishability, we adopt the BPR2000 model with some modifications- (1) extension to the tripartite case, and (2) extension for the Corrupt query.

4.1 Security notations

The in-distinguishability

In the model, the adversary *Adver* is a probabilistic machine that controls all the communications that take place between parties by interacting with a set of $\Pi^{i}_{U_1,U_2,U_3}$ oracles ($\Pi^{i}_{U_1,U_2,U_3}$ is defined to be the *ith* instantiation of an entity U_1 in a specific run, and U_2 and U_3 are the entities with whom U_1 wishes to establish a session key). The pre-defined oracle queries are described informally as follows.

- **Send**(U_1, U_2, U_3, *i*, *m*) allows *Adver* to send some message *m* of his choice to $\Pi^{i}_{U_1,U_2,U_3}$ at will. $\Pi^{i}_{U_1,U_2,U_3}$, upon receiving the query, will compute what the protocol specification demands and return to *Adver* the response message and/or decision. If $\Pi^{i}_{U_1,U_2,U_3}$ has either accepted with some session key or terminated, this will be made known to *Adver*.
- **Reveal**(U_1, U_2, U_3, *i*) query allows *Adver* to expose an old session key that has been previously accepted. $\Pi^{i}_{U_1,U_2,U_3}$, upon receiving the query and if it has accepted and holds some session key, will send this session key back to *Adver*.
- **Corrupt(**U_1, K_E**)** query allows *Adver* to corrupt the entity U_1 at will, and thereby learns the complete internal state of the entity. The corrupt query also allows *Adver* to overwrite the long-term key of the corrupted entity to the value of his choice (i.e., K_E).

This query can be used to model the real world scenarios of an insider co-operating with the adversary or an insider who has been completely compromised by the adversary.

- **Test(U_1, U_2, U_3, i)** query**:** If $\Pi^{i}_{U_1,U_2,U_3}$ has accepted with some session key and is being asked a Test(U_1, U_2, U_3, i **)**, then depending on a random bit b, *Adver* is given either the actual session key or a session key drawn randomly from the session key distribution.

The definition of security depends on the notations of partnership of oracles and in-distinguishability [3, 4]. In the BPR2000 model, partnership of oracles is defined using *SID*s (session identifiers). The definition of partnership is used in the definition of security to restrict the adversary's Reveal and Corrupt queries to oracles that are not partners of the oracles whose key the adversary is trying to guess [3, 4].

Partnership [3]: The oracles $\Pi^{i}_{U_1,U_2,U_3}$, $\Pi^{j}_{U_2,U_1,U_3}$ and $\Pi^{n}_{U_3,U_2,U_1}$ are partners if, and only if, the three oracles have accepted the same session key with the same *SID*, have agreed on the same set of entities, and no other oracles besides $\Pi^{i}_{U_1,U_2,U_3}$, $\Pi^{j}_{U_2,U_1,U_3}$ and $\Pi^{n}_{U_3,U_2,U_1}$ have accepted with the same *SID*.

Definition of security in both BR95 and BPR2000 also depend on the notation of freshness of the oracle to whom the *Test* query is sent [3, 4, 19]. For $\Pi^{i}_{U_1,U_2,U_3}$ to be fresh, the adversary in the BR95 model is not restricted from sending Corrupt queries to entities apart from the entities of oracles $\Pi^{i}_{U_1,U_2,U_3}$ and its partner oracles $\Pi^{j}_{U_2,U_1,U_3}$ and $\Pi^{n}_{U_3,U_2,U_1}$ (if such partners exist). We, therefore, adopt the definition of freshness of BR95 model.

Definition 3. Freshness [3]: $\Pi^{i}_{U_1,U_2,U_3}$ is fresh (or it holds a fresh session key) at the end of execution, if, and only if, oracle $\Pi^{i}_{U_1,U_2,U_3}$ has accepted with or without a partner oracles $\Pi^{j}_{U_2,U_1,U_3}$ and $\Pi^{n}_{U_3,U_2,U_1}$, all the oracles $\Pi^{i}_{U_1,U_2,U_3}$, $\Pi^{j}_{U_2,U_1,U_3}$ and $\Pi^{n}_{U_3,U_2,U_1}$ (if such an partner oracles exist) have not been sent a *Reveal* query, and the entities U_1, U_2 and U_3 of oracles $\Pi^{i}_{U_1,U_2,U_3}$, $\Pi^{j}_{U_2,U_1,U_3}$ and $\Pi^{n}_{U_3,U_2,U_1}$ (if such partners exist) have not been sent a *Corrupt* query.

Security is defined using the game *G*, played between the adversary *Adver* and a collections of $\Pi^{i}_{U_x,U_y,U_z}$ oracles for players U_x, U_y and $U_z \in \{U_1, U_2, ..., U_{N_P}\}$ and instances $i \in \{1, ..., N_S\}$. The adversary *Adver* runs the game simulation *G* with setting as follows.

- **Stage 1**: *Adver* is able to send *Send*, *Reveal* and *Corrupt* queries in the simulation.
- **Stage 2**: At some point during *G*, *Adver* will choose a fresh session and send a *Test* query to the fresh oracle associated with the test session. Depending on the randomly chosen bit *b*, *Adver* is given either the actual session key or a session key drawn from the session key distribution.
- **Stage 3**: *Adver* continues making any Send, Reveal and Corrupt oracle queries to its choice.
- **Stage 4**: Eventually, *Adver* terminates the game simulation and output its guess bit *b'*.

Success of *Adver* in *G* is measured in terms of *Adver*'s advantage in distinguishing whether *Adver* receives the real key or a random value. Let the advantage function of *Adver* be denoted by $Adv^{Adver}(k)$, where *k* is the security parameter and $Adv^{Adver}(k)$ =2Pr[*b*=*b'*]-1.

Key-compromise impersonation

The participating entities (except the adversary) are always considered honest in all of the BR95 model, the BPR2000 model and the Canetti-Krawczyk model [6], and a session with any corrupted entity is not considered as fresh for testing. It, therefore, cannot capture the key-compromise impersonation attack. However, we can gain the advantage of key-compromise impersonation to that of forging a signature with one private key belonging to one of the three communicating parties. In our tripartite scheme, the adversary who has compromised U_1's private key should try to impersonate U_2 to both U_1 and U_3.Therefore, the adversary should generate U_2's signature on the fresh *sid* and ephemeral public keys. Therefore, his advantage of impersonation is directly related to the advantage of forging U_2's signature.

Insider attack

For the tripartite case involving entities U_1, U_2 and U_3, we consider the following two scenarios are non-sense: (1) U_1 and U_2 co-operatively impersonate U_3 to themselves, and (2) U_1 impersonates U_2 and U_3 simultaneously to himself. So, the only meaningful attack scenarios are like that U_1 impersonates U_2 to U_3 such that U_3 wrongly believes that itself, U_1 and U_2 will share the same key. In our protocol, U_3 will complete the protocol and compute the session key if only if U_3 has validated the signatures from U_1 and U_2. Of course, U_1 (the inside attacker) can generate his own signature. But, to generate valid signatures on the session-bound data $h(sid \| I_{U_1} \| I_{U_2} \| I_{U_3} \| g^a \| g^b \| g^c \| g^{h(g^{ab})})$ on behalf of U_2, U_1 should access U_2's private key. So, the inside attacker's (U_1's) advantage in impersonating U_2 is the same as that advantage of forging U_2's signature. Since the underlying signature

scheme is secure against adaptively chosen message attack and U_2's ephemeral public key $g^b \bmod p$ is random and fresh, the advantage is negligible. Now we are ready to define the security.

Definition 4 (Secure tripartite key agreement protocol): A tripartite key agreement protocol is secure in our model if the following thee requirements are satisfied:

1. Validity: When the protocol is run among three oracles in the absence of a malicious adversary, the three oracles accept the same key.

2. Indistinguishability: For all probabilistic, polynomial-time adversaries $Adver$, $Adv^{Adver}(k)$ is negligible.

3. Security against insider impersonation and key-compromise impersonation: Even an insider (and a key-compromise impersonator) cannot impersonate another entity to the third entity and complete the session run with the third one.

4.2 Security proof

Theorem 1. The proposed tripartite STS key confirmation protocol is secure in the sense of Definition 4 if the underlying digital signature scheme is secure against the adaptively chosen message attack and the CDHP is hard.

Proof: the proof is given in the appendix.

5. Conclusions

This paper has shown the man-in-the-middle attack and the insider attack on the TAKC-STS protocol. To conquer the security weaknesses, we have proposed a new tripartite STS protocol, which preserves the practical merits of the STS protocol. The security of the proposed protocol is proved in the random oracle model.

References

1. Al-Riyami, S. S., and K. G. Paterson. “Tripartite Authenticated Key Agreement Protocols from Pairings.”*IMA Conference on Cryptography and Coding*, LNCS 2898, Springer-Verlag, 2003, pp. 332-359.
2. Al-Riyami, S. S., and K. G. Paterson.“Authenticated three party key agreement protocols from pairings. ”*Cryptology ePrint Archive*, Report 2002/035.
3. Bellare, M., and P. Rogaway.“Provably secure session key distribution: The three party case. ”*27th ACM Symposium on the Theory of Computing*, ACM press, 1995, pp. 57-66.
4. Bellare, M., D. Pointcheval, and P. Rogaway.“Authenticated key exchange secure against dictionary attacks.” *Eurocrypt* 2000, LNCS 1807, Springer, pp. 139-155.
5. Burrows, M., Abadi, M., Needham, R. “A logic of authentication.” *ACM TRANSACTIONS ON COMPUTER SYSTEMS*, (8:1), 1990, 18-36.
6. Canetti, R., and H. Krawczyk.“Analysis of key-exchange protocols and their use for building secure channels.”*Eurocrypt*, LNCS 2045, Springer, 2001, pp. 451-472.
7. Canetti, R., Goldreich, O., and Halevi, S. “The random oracle methodology, revisited.” *Journal of the ACM* , (51:4), 2004, 557-591.
8. Chien, H. Y.“ID-based Tripartite Multiple Key Agreement Protocol facilitating Computer Auditing and Transaction Refereeing.” *Journal of Information Management* , (13:4) , 2006, 185~204.
9. Chien, H. Y. “Comments: Insider attack on Cheng et al.'s pairing-based tripartite key agreement protocols.” *Cryptology ePrint Archive*: Report 2005/013.
10. Chien, H. Y., and R. Y. Lin. Improved ID-based security framework for ad hoc network.” *Ad Hoc Networks,* (6:1), Jan 2008, 47-60.
11. Chien, H. Y., Wang, R. C., and Yang, C. C. “Note on Robust and Simple Authentication Protocol.”*The Computer Journal,* (48:1), 2005, 27- 29.
12. Chien, H. Y., Wu, T. C., Jan, J. K. and Tseng, Y. M. ”Cryptanalysis of Chang-Wu’s Group-oriented Authentication and Key Exchange Protocols”, *Information Processing Letters*, Amsterdam, (80:2), Oct. 2001, pp. 113-117.
13. Diffie, W., P.C. van Oorshot, and M. Wiener. “Authentication and authenticated key exchanges.”*Designs, codes, and Cryptography* (2:2), 1992, 107-125.

14. Horng, G., C.-L. Liu, and H.-Y. Liu. “Security Analysis of a Tripartite Authenticated Key Agreement Protocol Based on Weil Pairing.” *ICS 2004 - International Computer Symposium*, Taiwan.
15. Joux, A., A “One round protocol for tripartite Diffie-Hellman.” *ANTS IV*, LNCS1838, Spring-Verlag, 2000, pp. 385-394.
16. Lin, C. H., and H. H. Lin. “Secure One-Round Tripartite Authenticated Key Agreement Protocol from Weil Pairing.” *Proceedings of International Conference on Advanced Information Networking and Applications* (AINA 2005), (2), March 25-30, 2005 , pp. 135-138.
17. Lu, R., Cao, Z., Wang, S. B., Bao, H. Y.. “A New ID-Based Deniable Authentication Protocol.” *Informatica* , (18 :1), 2007, 67-78.
18. O'Higgins, B., W. Diffie, L. Strawczynski, and R. Hoog. “Encryption and ISDN - A Natural Fit." *International Switching Symposium (ISS87),* 1987.
19. Raymond, C. K. K., Colin, B., and Yvonne, H. “Examining indistinguishability-based proof models for key establishment protocols.” *ASIACRYPT* 2005, LNCS 3788, Springer, pp. 585-604.
20. Shim, K. “Efficient one round tripartite authenticated key agreement protocol from Weil pairing.” *Electron.* Lett. (39:2), 2003, 208-209.
21. Shim, K. “Cryptanalysis of Al-Riyami-Paterson's Authenticated Three Party Key Agreement Protocols.”, *Cryptology ePrint Archive*, Report 2003/122.

Appendix

Theorem 1. The proposed tripartite STS key confirmation protocol is secure in the sense of Definition 4 if the underlying digital signature scheme is secure against the adaptively chosen message attack and the CDHP is hard.

Proof:

1. The validity is straightforward due to our protocol specification.
2. The security against insider impersonation (and the key-compromise impersonation) is equivalent to the security of the underlying signature scheme. This has been discussed in Section 4.1.
3. So, we concentrate on the in-distinguishability. The general notation of this in-distinguishable proof is to assume an adversary *Adver* who can gain a non-negligible advantage in distinguishing the test key in the game, and uses *Adver* to construct a breaker ***B*** that solves the CDHP with non-negligible probability. The hash function $h()$ is modeled as a random oracle here.

The proof can be divided into two cases since the adversary *Adver* can either gain its advantage against the protocol by forging a participating entity's signature or gain its advantage against the protocol without forging a participating entity's signature.

Case 1. *Adver* gains its advantage by forging a participating entity's signature.

We denote by $\Pr[Succ^{Sig}(k)]$ the probability of a successful signature forgery under adaptively chosen message attack, and define an event *SigForgery* to be an event that at some point in the game *Adver* asks a Send(U_1, U_2, U_3, i, $sid \parallel S_{U_1}(...)$) query to some partner oracles $\Pi^{j}_{U_2,U_1,U_3}$ or $\Pi^{n}_{U_3,U_2,U_1}$ such that the oracles accept, but the signature value ($S_{U_1}(...)$) used in the query was not previously output by a fresh oracle. We construct an adaptive *Signature* forger *F* against the message authentication scheme using *Adver* in the following game G_F.

Stage 1: *F* is provided permanent access to the *Signature* oracle O_U associated with its private key of U throughout the game G_F.

- *F* randomly chooses an entity $\overline{U} \in \{U_1,..., U_{N_P}\}$. $\overline{U}$ is *F*'s guess at which *Adver* will choose for the *SigForgery*.
- *F* generates a list of public key/private key pairs for the entities $\{U_1,..., U_{N_P}\} \setminus \{\overline{U}\}$.

Stage 2: *F* runs *Adver* and answers all queries from *Adver*. This can be easily done since *F*

can respond to all oracle queries as required using the keys chosen in Stage 1 and O_U. F also records all the signatures it receives from O_U. If, during the execution, $Adver$ make an oracle query that includes a forged signature for $\overline{U}$, then F outputs the signature forgery as its own and halts. Otherwise, F halts as $Adver$ halts.

Since $\overline{U}$ is randomly chosen from the N_P entities, the probability that $\overline{U}$ is the entity for whom $Adver$ generates a forgery is at least $1/N_P$. Therefore, the success probability of F is $\Pr[Succ^F(k)] \geq \Pr[SigForgery(k)]/N_P$. Hence,

$$N_P \cdot \Pr[Succ^F(k)] \geq \Pr[SigForgery(k)]. \quad \text{(A.1)}$$

Since the underlying signature scheme is assumed to be secure against adaptively chosen message attack and the N_P is polynomial in k, the $\Pr[SigForgery(k)]$ is negligible.

Case 2. $Adver$ gains its advantage without forging a participating entity's signature.

This part assumes that $Adver$ gains its advantage without forging a participating entity's signature.

Denote $Adver$'s advantage in differentiating the real session key from a random generated key without forging a signature as $Adv^{Adver}_{no\ forgery}(k)$ = |Pr[$Adver$ succeeds in correctly guessing]-1/2|.

Now suppose, by the way of contradiction, the $Adv^{Adver}_{no\ forgery}(k)$ is non-negligible. Suppose that there exists an oracle $\Pi^i_{A,B,C}$ has accepted the session key of the form $h(sid \| I_A \| I_B \| I_C \| g^a \| g^b \| g^c \| g^{h(g^{ab})} \| g^{c \cdot h(g^{ab})})$ and has the partnership with fresh oracles $\Pi^j_{B,A,C}$ and $\Pi^n_{C,A,B}$. We say that $Adver$ succeeds if at the end of $Adver$'s experiment, $Adver$ picks $\Pi^i_{A,B,C}$ to ask a *Test* query and outputs the correct bit guess. Thus, $Pr[\ \Pi^i_{A,B,C}$ succeeds]=1/2+$\eta(k)$, where $\eta(k)$ is non-negligible. Now define Q_h be the event that $h()$ has been queried on $... \| g^{c \cdot h(g^{ab})}$ by $Adver$ or some oracle other than $\Pi^i_{A,B,C}$, $\Pi^j_{B,A,C}$ and $\Pi^n_{C,A,B}$. Then

$Pr[$ $Adver$ succeeds]= $Pr[$ $Adver$ succeeds$|\ Q_h\ ]$* $Pr[\ Q_h\ ]$+ $Pr[$ $Adver$ succeeds$|\ \overline{Q}_h\ ]$* $\Pr[\ \overline{Q}_h\ ]$. Since $h()$ is a random oracle and $\Pi^i_{A,B,C}$, $\Pi^j_{B,A,C}$ and $\Pi^n_{C,A,B}$ are fresh oracles, $Pr[$ $Adver$ succeeds $|\ \overline{Q}_h\ ]$=1/2. Thus 1/2+$\eta(k)$ ≤ $Pr[$ $Adver$ succeeds $|\ Q_h\ ]$* $Pr[\ Q_h\ ]$+1/2, so that $Pr[\ Q_h\ ] \geq \eta(k)$. That is, given $Adver$ picks some fresh oracle $\Pi^i_{A,B,C}$ that has accepted the session, then the probability that $h()$ has been queried on $... \| g^{c \cdot h(g^{ab})}$ by $Adver$ or some oracle other than $\Pi^i_{A,B,C}$, $\Pi^j_{B,A,C}$ and $\Pi^n_{C,A,B}$ is non-negligible. We can, therefore, use $Adver$ to construct a breaker D which solves the CDHP with non-negligible probability.

D's task: Given g, p, $g^a \bmod p$ **and** $g^b \bmod p$, where a, b are random numbers from

Z_{p-1}, compute $g^{ab} \bmod p$.

D 's operation: D randomly picks A, B and $C \in \{U_1, U_2, ..., U_{N_P}\}$, instance $i, j, n \in \{1, ..., N_S\}$ and $u \in \{1, ..., N_h\}$, where N_P, N_S, and N_h respectively denote the number of entities, the number of session per entity, and the number of distinct queries to $h()$, and all the three functions are polynomial function on the security parameter *k*. D guesses that *Adver* will select $\Pi^i_{A,B,C}$ to ask its *Test* query after $\Pi^i_{A,B,C}$ has accepted the session, and also guesses that *u*th distinct $h()$ query made during the experiment will be on $... \| g^{c \cdot h(g^{ab})}$.

Given the challenge (g, p , $g^a \bmod p$, $g^b \bmod p$), D sets U_i 's public key as $Y_i = g^{r_i} \bmod p$ and its private key as r_i for all $U_i \in \{U_1, ... U_{N_P}\}$.

During the experiment, D answers *Adver* 's queries as follows.

1. ***Hash* query**. D answers all $h()$ queries at random, just like a real random oracle does, and records the (*query*, *response*) pair in its L_h list to keep consistent answers.
2. **Corrupt(U , *K*) query**. If $U \in \{A, B, C\}$, then D gives up; otherwise, D hands in all internal of U to *Adver*, and updates U 's key pair as *K*.
3. **Reveal(U_1, U_2, U_3, l) query.** D answers all *reveal* queries in normal cases (reveals the session keys), except that if *Adver* asks $\Pi^i_{A,B,C}$, $\Pi^j_{B,A,C}$ and $\Pi^n_{C,A,B}$ a *Reveal* query, then D gives up.
4. **Send (U_1, U_2, U_3, l, m)query.** D answers all *Send* queries as specified by a normal oracle, except that if *Adver* asks $\Pi^i_{A,B,C}$, $\Pi^j_{B,A,C}$ and $\Pi^n_{C,A,B}$ *Send* query. The queries are processed, according to the following rules:

4.1 If ($\prod^l_{U_1,U_2,U_3} \notin \{ \Pi^i_{A,B,C}$, $\Pi^j_{B,A,C}$ and $\Pi^n_{C,A,B} \}$), then follows the protocol specification to generate its outputs. Depending on which step (1~6) of this query, we have the following situations:

- For Step 1, randomly chooses an integer $w \in Z_{p-1}$ and outputs $sid \| g^w \| Cert_{U_1}$. D also records the data (i, U_1, U_2, U_3, *sid*, (w, $g^w \bmod p$)) in his Send-list.
- For Step 2, if the input $m = sid \| x \| Cert_U$ conforms to the format of Step 1, then D randomly chooses an integer $w \in Z_{p-1}$ and computes $x^w \bmod p$. D consults its L_h list to check whether an entry of the form ($x^w \bmod p$, α) exits. If so, it takes $\alpha = h(x^w \bmod p)$; otherwise, it randomly chooses an integer α and stores ($x^w \bmod p$, α) in L_h list. It outputs $sid \| x \| Cert_U \| g^w \| Cert_{U_1} \| g^\alpha \bmod p$. D also records the data (i, U_1, U_2, U_3, *sid*, (x , w, $g^w \bmod p$, $x^w \bmod p$, α)) in its Send-list.

- For Step 3~6, *D* follows the protocol specification and responds similarly as above, except that it will consult in its Send-list to compute the corresponding session key and use the corresponding private key to generate the signature. Upon receiving the input, it also follows the protocol specification to verify the signature and the message to decide whether accept the session key.

4.2 If ($\prod^{l}_{U_1,U_2,U_3} \in \{ \Pi^{i}_{A,B,C}$, $\Pi^{j}_{B,A,C}$ and $\Pi^{n}_{C,A,B} \}$), then *D* generates the output as follows.

- If $\prod^{l}_{U_1,U_2,U_3} = \Pi^{i}_{A,B,C}$ and it corresponds to step 1, *D* randomly chooses an integer x and outputs the outgoing message as (*i*, U_1, U_2, U_3, *sid*, g^x, $Cert_A$). *D* also records the data in its Send-list.
- If $U_1 = B$ and it corresponds to step 2, *D* randomly chooses an integer y, sets $g^{h(g^{xy})}$ to be g^a (the challenge from the CDHP problem) and outputs the outgoing message as (*i*, U_1, U_2, $U_3, g^x, Cert_A$, $g^y, Cert_B, g^a$). If there is already an entry of the form (g^{xy}, α) for some α in its L_h list, it re-selects the value y. Since *D* does not know the actual value of a, it records ($g^{xy}, a?$) in the L_h list. It also records (*i*, U_1, U_2, U_3, *sid*, $g^x, Cert_A$, $g^y, Cert_B, g^a$) in its Send-list.
- If $U_1 = C$ and it corresponds to step 3, *D* sets its ephemeral value as g^b (another challenge from CDHP) and computes the corresponding signature as specified by the protocol. Since *D* can access the private key, it can generate the signature. It outputs the outgoing message as (*i*, U_1, U_2, U_3, $sid \| g^x \| g^y \| g^a \| g^b \| Cert_C \| S_C(...)$, and records the data in the Send-list.
- For other steps, *D* accesses the corresponding private key/public key to generate the signature and to verify the received signature as specified in the protocol. It also records the data in the Send-list.

There are the following possible results for the above experiment:

1. *Adver* does not make its queries in such a way that $\Pi^{i}_{A,B,C}$ has accepted the session, then *D* gives up.
2. *Adver* and its oracles do not make *u* distinct hash oracle calls before *Adver* asks its *Test* query, then *D* gives up.
3. *Adver* does make its queries in this way, then $\Pi^{i}_{A,B,C}$ will accept the session and

hold the key formed $h(...\| g^{ab})$.

4. If case 3 does happens and the *u* th distinct query to hash is made on value $...\| g^{ab}$, then D stops and outputs g^{ab}.

If the *u*th distinct $h()$ query made by $Adver$ and its oracles is on $...\| g^{ab}$, then D certainly wins its experiment. Therefore, the probability that D outputs the correct value g^{ab} is: $Pr[\ Q_h\ ]/\ (\ N_p{}^3 N_S N_h\ ) \geq \eta(k)/(N_P{}^3 N_S N_h)$, which is non-negligible. This contradicts the CDHP assumption. We, therefore, conclude that $\eta(k)$ must be negligible and so is $Adv^{\ Adver}(k)$.