Bilateral Unknown Key-Share Attacks in Key Agreement Protocols

Liqun Chen
Hewlett-Packard Laboratories
Filton Road, Bristol BS34 8QZ, UK
liqun.chen@hp.com
Qiang Tang*
Département d'Informatique, École Normale Supérieure
45 Rue d'Ulm, 75230 Paris Cedex 05, France
qiang.tang@rhul.ac.uk

June 1, 2007

Abstract

Unknown Key-Share (UKS) resilience is a basic security attribute in authenticated key agreement protocols, whereby two entities A and B should not be able to be coerced into sharing a key between them when in fact either A or B thinks that s/he is sharing the key with another entity C. In this paper we revisit some definitions of this attribute, the existing UKS attacks and the method of proving this attribute in the Bellare-Rogaway (BR) model in the literature.

We propose a new UKS attack, which coerces two entities A and B into sharing a key with each other but in fact A thinks that she is sharing the key with another entity C and B thinks that he is sharing the key with another entity D, where C and D might or might not be the same entity. We call this attack a Bilateral Unknown Key-Share (BUKS) attack and refer to the existing UKS attacks, which are against one entity only, as a Unilateral UKS (UUKS) attack. We demonstrate that a few well-known authenticated key agreement protocols, some of which have been proved holding the UUKS resilience property, are vulnerable to the BUKS attack. We then explore a gap between the traditional BR-type proof of UUKS resilience and a BUKS adversary's

*The work was partially done when the author was a full-time Ph.D student at Royal Holloway, University of London.

behavior, and extend the BR model to cover the BUKS resilience attribute. Finally we provide a simple countermeasure to prevent a key agreement protocol from BUKS attacks.

Keywords: authenticated key agreement, unknown key-share resilience, bilateral unknown key-share resilience, the Bellare-Rogaway model

1 Introduction

Generally speaking, in a key agreement protocol where two entities A and B establish a key between them, the key authentication property means at least one of the following four assurances is true: 1. *Implicit key authentication from A to B (or B to A)* is the assurance for entity B (or A) that A (or B) is the only other entity that can possibly be in possession of the key. 2. *Explicit key authentication from A to B (or B to A)* is the assurance for entity B (or A) that A (or B) is the only other entity that is in possession of the key. 3. *Implicit mutual key authentication* is the assurance for the two entities that only the other entity can possibly be in possession of the key. 4. *Explicit mutual key authentication* is the assurance for the two entities that only the other entity is in possession of the key.

The concept of Unknown Key-Share (UKS) attacks originated from the discussion about the key authentication property in authenticated key agreement protocols by Diffie, Oorschot and Wiener in 1992 [15]. They proposed the first UKS attack, where a dishonest entity C tempts two honest entities A and B to establish a key shared between them. At the end of the attack, A believes she shares the key with B, but B mistakenly believes the key is shared with C.

Let us recall the interesting hypothetical scenario described in [15], where the UKS attack can have damaging consequences. Assume that B were a bank and A and C were two account holders. A might make a deposit by running the key agreement protocol with B. Because B has been misled by C in the protocol, C could get credit for the deposit made by A; so eventually C might benefit and both A and B may be hurt. This scenario shows a fact that since B ends up the protocol by sharing a key with A but accepting C's identity, the protocol fails to provide (either implicit or explicit) key authentication from either A or C to B.

In the literature, a number of existing key agreement protocols have been shown vulnerable to UKS attacks, including MTI/A0 [22] attacked by Menezes *et al.* [23], the STS-MAC variant of the Station-to-Station (STS) protocol [15] attacked by Blake-Wilson and Menezes [7], the revised STS-

MAC protocol [7] and the KAP-HY98 protocol [17] attcked by Baek *et al.* [2, 19], the MQV protocol [21, 23] attacked by Kaliski [1], and the Harn-Lin's modified MQV protocol [16] attacked by Zhou *et al.* [26].

In the existing UKS attacks, the entity which is misled to accept a wrong identity of the partner entity is either the initiator or the responder but not both of them. Based on [7], the first case is called a UKS attack *against the initiator* and the second one is called a UKS attack *against the responder*. Since these UKS attacks are against one entity only, we refer them as Unilateral UKS (UUKS) attacks, and the corresponding attribute as UUKS resilience.

Some researchers have worked on how to give a formal proof of the UUKS resilience property for a number of authenticated key agreement protocols, for example [12, 14]. Their works are based on the Bellare-Rogaway (BR) model [3, 5].

Our major contributions in this paper are as follows. After revisiting some definitions of the UUKS attack, we propose a new UKS attack, which coerces two entities A and B into sharing a key with each other but in fact A thinks that she is sharing the key with another entity C and B thinks that he is sharing the key with another entity D, where C and D might or might not be the same entity.

We call this new attack a Bilateral Unknown Key-Share (BUKS) attack and the corresponding attribute BUKS resilience. We demonstrate that three well-known types of authenticated key agreement protocols, namely Shoup's DHKE protocols in [25], the modified STS protocols proposed in [8] and the modified Oakley protocol again in [8], are vulnerable to the BUKS attack. Considering some of these protocols have been proved holding the UUKS resilience property in the BR model, we explore what is the gap between the traditional BR-type proof of UUKS resilience and a BUKS adversary's behavior. Finally we extend the BR model to cover the BUKS resilience attribute.

The remainder of this paper is organised as follows. The concept of a BUKS attack is described in Section 2. Three examples as to how the BUKS attack affects three sets of well-known protocols are demonstrated in Section 3. In Section 4, we review the traditional formal proof of the UUKS resilience attribute and other well-studied security properties in the BR model. In Section 5 we explore the gap between the existing BR-type proof and a BUKS adversary's behavior, and then provide modifications to the BR model, in order to cover the BUKS resilience attribute. In Section 6, we suggest a simple countermeasure to protect an authenticated key agreement protocol from the BUKS attack. We finally conclude the paper in the last

section .

2 The Concept of a BUKS Attack

For a two-party authenticated key agreement protocol, the following three conditions may be expected to be held:

- *Condition 1.* There are two honest players, say A and B. For an honest player, we mean that he or she knows a valid long-term authentication key and follows the protocol specification properly.
- *Condition 2.* At the end of the protocol A and B share the same key.
- *Condition 3.* Both A and B correctly accept each other's identity as their key sharing partner, that implies the property of (either implicit or explicit) mutual key authentication between A and B.

Blake-Wilson and Menezes in 1999 [7] proposed the first formal definition of the UKS attack, which has been adopted by many researchers after that. The definition was stated as follows.

Definition 1. *An unknown key-share attack on an authenticated key agreement protocol is an attack whereby an entity A ends up believing it shares a key with another entity B and although this is in fact the case that B mistakenly believes the key is instead shared with an entity $C \neq A$.*

In this definition, Conditions 1 and 2 have not been clearly addressed, but we can expect that they are held. As to Condition 3, it is restricted that A accepts the correct identity of her key sharing partner B, but B accepts a wrong identity of his key sharing partner. So a protocol, which suffers from this type of UKS attacks, will not hold key authentication from A to B, but will hold key authentication from B to A. Obviously, the attack defined in Definition 1 is a UUKS attack. Note that for simplicity, hereinafter we do not distinguish that the key authentication property is implicit or explicit.

To make the definition more general, in other places (e.g. [12, 14]), the UKS attack is defined as follows.

Definition 2. *An unknown key-share attack on an authenticated key agreement protocol is an attack whereby an entity A is coerced into sharing a key with an entity B when in fact A thinks that she is sharing the key with another entity C.*

In this definition, Condition 1 has not been clearly addressed, but again we can expect that it is held. It is explicitly addressed that Condition 2 is held. As to Condition 3, it is explicitly addressed that A accepts a wrong identity of her key sharing partner. But there is no restriction whether B accepts the correct identity of his key sharing partner A or not. So a protocol, which suffers from this type of UKS attacks, will not hold key authentication from B to A, but it may or may not hold key authentication from A to B.

In both of the above definitions, one of the two honest players is definitely a victim. The difference between these two definitions is that in Definition 1, the other player is not a victim, but in Definition 2, the other player may or may not be a victim. We can then think about that Definition 1 is a special case of Definition 2. In the following discussion, we propose another special case of Definition 2, which is opposite to Definition 1. We assume that the other player is also a victim. We call the new case a Bilateral Unknown Key-Share (BUKS) attack and define it as follows.

Definition 3. *A bilateral unknown key-share attack on an authenticated key agreement protocol is an attack whereby two honest entities A and B ends up sharing a key between them but A believes it shares the key with another entity C, and B believes it shares the key with another entity D, where C is not equal to B and D is not equal to A.*

In this definition, it is clearly addressed that both the entities A and B are honest and victims. The entities C and D may or may not be the same entity, and they may or may not be honest. A protocol, which suffers from the BUKS attacks, will hold neither key authentication from B to A nor from A to B.

Following the hypothetical scenario for the UUKS attack originally given in [15] and specified in Section 1, we can see a similar hypothetical scenario where a BUKS attack can have damaging consequences as follows. Suppose that B is an honest service provider selling e-goods over the Internet, A is an honest customer buying an e-good over the Internet, C is a dishonest entity pretending an honest service provider and D is a dishonest entity pretending an honest customer. Every entity has a universally verifiable certificate, which is issued by a trusted third party and within each certificate is the public key and the e-post address of the holder. When A searches an e-good over the Internet, C hijacks this request and responds to A's request. But meanwhile, C colludes with D and forwards A's request to B by using D's identity instead of A's. After that the two protocols, respectively between A and C and between D and B, are concurrently running. At the end of

these two protocols, A and B share a key, which is then used to protect the negotiation on what e-good A wants and what the price B offers. After a successful bargain, A pays an e-cash that is encrypted under C's certified public key and B sends the e-good to the certified e-post address of D.

3 Examples of BUKS Attacks

In this section, we will demonstrate BUKS attacks against three types of protocols: (1) the four DHKE protocols (called DHKE, DHKE-1, DHKE-2 and DHKE-3) proposed by Shoup in [25], (2) the two modified STS protocols proposed in [8] and (3) the alternative Oakley protocol in [8].

A general solution used to protect many earlier key agreement protocols against UKS attacks was inclusion of the principal identities either in signed messages or in encrypted messages. We will show that this general solution may be strong enough to prevent from UUKS attacks, but certainly is not strong enough to prevent from the BUKS attack.

3.1 Analysis of the DHKE protocols

In [25], Shoup proposed a set of four DHKE protocols (called DHKE, DHKE-1, DHKE-2 and DHKE-3), and proved them secure, where security of DHKE was proved in the static corruption mode and security of the other three was proved in both the adaptive corruption mode and the strong adaptive corruption mode. In this subsection, we will take DHKE-1 as an example to show how it suffers from the BUKS attack. Note that the same attack also applies to DHKE, DHKE-2 and DHLE-3 in the same way.

3.1.1 Description of the scheme

Let the users be denoted as U_i ($i \geq 1$) with unique identity ID_i. The system generates the following parameters: a digital signature scheme (KeyGen, Sign, Verify), a group $\mathbb{G}$ of prime order q, a generator g of $\mathbb{G}$, a family of pair-wise independent hash functions $\mathsf{H_k}$ indexed by a bit string k, and a pseudorandom function BitGen. Every user U_i chooses a public/private key pair (pk_i, sk_i) for the digital signature scheme. It is assumed that U_i's public key consists of the public key of the signature scheme and a description of $\mathbb{G}$ and g, while the private key consists of the private key of the signature scheme. Let $Cert_i$ be the certificate that binds U_i's public key with its identity.

If U_i and U_j want to establish a session key, they perform as follows.

1. U_i randomly selects s_i from $\mathbb{Z}_q$, and sends $(g^{s_i}, \sigma_i, Cert_i)$ to U_j, where

$$\sigma_i = \mathsf{Sign}(g^{s_i}||ID_j; sk_i).$$

2. U_j randomly selects s_j from $\mathbb{Z}_q$, and sends $(g^{s_j}, k, \sigma_j, Cert_j)$ to U_i, where k is a random hash function index and

$$\sigma_j = \mathsf{Sign}(g^{s_i}||g^{s_j}||k||ID_i; sk_j).$$

3. U_i computes $(k_1, k_2) = \mathsf{BitGen}(\mathsf{H_k}(g^{s_i s_j}))$, where BitGen is a bit generation function, sends k_1 to U_j and keeps k_2 as an established session key. U_i believes that U_j is the only other entity that can possibly be in possession of k_2.

4. U_j computes $(k_1, k_2) = \mathsf{BitGen}(\mathsf{H_k}(g^{s_i s_j}))$, verifies whether k_1 matches to the received k_1 value, and accepts k_2 as an established session key if the verification succeeds. U_j then believes that U_i is the only other entity that is in possession of k_2.

3.1.2 Description of the attack

Suppose that if there are two sessions, one is for $\{U_1, U_2\}$ and the other one is for $\{U_3, U_4\}$. Suppose also that U_2, U_3 are malicious, then they can mount a BUKS attack, which is depicted in **Fig.** 1.

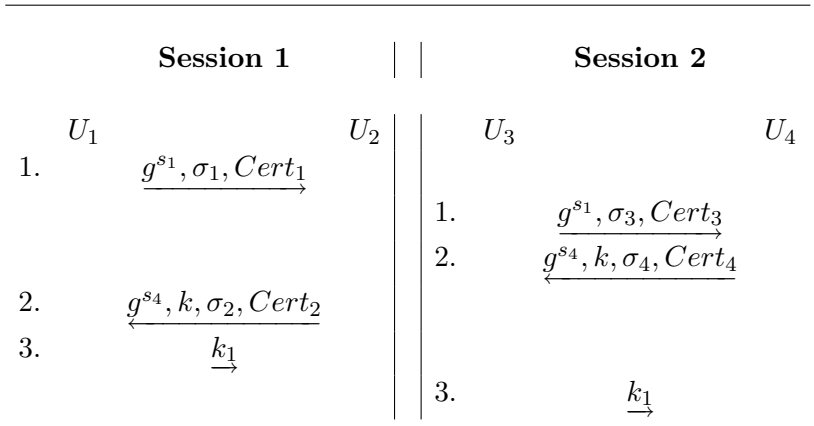

Figure 1: The BUKS attack to DHKE-1

Note that U_3 sends its first message in the second session after U_2 receives U_1's first message in the first session, and U_2 sends its first message in the first session after U_3 receives U_4's first message in the second session, and U_3 sends its second message in the second session after U_2 receives U_1's second message in the first session.

It is easy to see that, both sessions end successfully, U_1 and U_4 compute the same session key k_2, and U_1 and U_4 share the same key although they accept the identities of U_2 and U_3 respectively as their key sharing partner.

3.2 Analysis of two Modified STS protocols

Bellare, Canetti, and Krawczyk [4] proposed a modular approach to construct authenticated key agreement protocols. Using this approach, given a key agreement protocol which is secure in an authenticated communication network, that a authenticated protocol which is secure in an unauthenticated communication network can be obtained by employing a message authenticator. The protocols described in this section is generated using this modular approach based on a typical Diffie-Hellman key agreement protocol, where a signature-based message authenticator is employed.

There are two modified STS protocols specified in [8], Protocol 5.16 and Protocol 5.17. Both of them suffer from the BUKS attack, although Protocol 5.17 makes use of a Message Authentication Code (MAC) algorithm to enhance security.

3.2.1 Description of the protocol

Let the users be denoted as U_i ($i \geq 1$) with unique identity ID_i. The system generates the following parameters: a digital signature scheme (KeyGen, Sign, Verify), a group $\mathbb{G}$ of prime order q, a generator g of $\mathbb{G}$. Every user U_i generates a public/private key pair (pk_i, sk_i) for the digital signature scheme. If U_i and U_j want to establish a session key, they perform as follows by following Protocol 5.16.

1. U_i randomly selects s_i from $\mathbb{Z}_q$, and sends g^{s_i} to U_j.
2. U_j randomly selects s_j from $\mathbb{Z}_q$, and sends (g^{s_j}, σ_j) to U_i, where σ_j is computed as follows:

$$\sigma_j = \mathsf{Sign}(g^{s_j} || g^{s_i} || ID_i; sk_j).$$

3. U_i verifies σ_j, and sends σ_i to U_j, if the verification passes, otherwise aborts the protocol execution.

$$\sigma_i = \mathsf{Sign}(g^{s_i}||g^{s_j}||ID_j; sk_i).$$

At the end of the protocol execution, the session key is computed as $K = g^{s_i s_j}$.

In [8], another modified STS protocol, Protocol 5.17, makes use of Message Authentication Codes (MACs). The only difference from the above protocol is computation of the following two values:

$$\sigma_j = (\mathsf{Sign}(g^{s_j}||g^{s_i}; sk_j), \mathsf{MAC}_{K_{ij}}(g^{s_j}||g^{s_i}))$$

and the third message is

$$\sigma_i = (\mathsf{Sign}(g^{s_i}||g^{s_j}; sk_i), \mathsf{MAC}_{K_{ij}}(g^{s_i}||g^{s_j})),$$

where K_{ij} is derived from the value K. In the BUKS attack described in the next subsection, we include these two protocols without distinguishing between them.

3.2.2 Description of the attack

Suppose that if there are two sessions, one is for $\{U_1, U_2\}$ and the other one is for $\{U_3, U_4\}$. Suppose also that U_2, U_3 are malicious, then they can mount a BUKS attack, which is depicted in **Fig.** 2.

Note that U_3 sends its message in the second session after U_2 receives U_1's message in the first session, and U_2 sends its message in the first session after U_3 receives U_4's message in the second session.

It is easy to see that, both sessions end successfully, U_1 and U_4 compute the same session keys, and U_1 and U_4 share the same key although they accept the identities of U_2 and U_3 respectively as their key sharing partner.

3.3 Analysis of an alternative Oakley protocol

We now take a look at another type of authenticated key agreement protocol, which is based on encryption instead of signatures. We choose a modified Oakley protocol, which is introduced in [8]. The original Oakley protocol was given in Internet RFC 2412 [24].

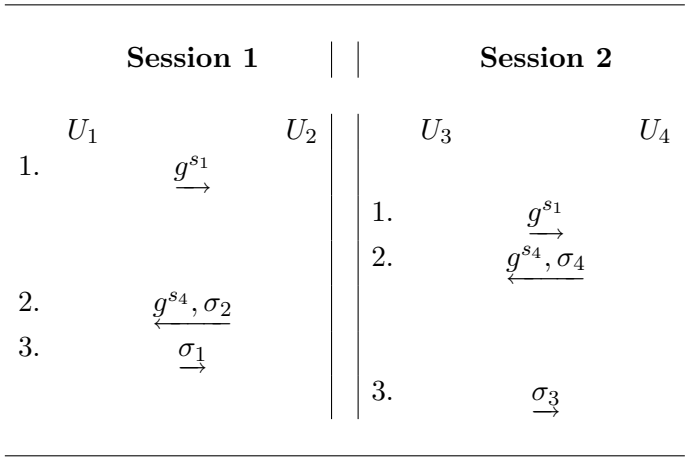

Figure 2: The BUKS attack to the modified STS protocol

3.3.1 Description of the protocol

Let the users be denoted as U_i ($i \geq 1$) with unique identity ID_i. The system generates the following parameters: an encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, a group $\mathbb{G}$ of prime order q, a generator g of $\mathbb{G}$. Every user U_i generates a public/private key pair (pk_i, sk_i) for the encryption scheme. If U_i and U_j want to establish a session key, they perform as follows.

1. U_i first takes a cookie CK_i, which was pre-agreed with U_j, arranges an indication of the set of used algorithms $list$ and collects U_j's domain identity ID_j and the domain public key pk_j. U_i then randomly selects a nonce n_i and s_i, and sends σ_1 together with $(CK_i, t_i, list, ID_j)$ to U_j, where t_i and σ_1 are computed as follows:

$$t_i = g^{s_i} \text{ and } \sigma_1 = \mathsf{Enc}(ID_i \| ID_j \| \mathsf{Enc}(n_i; pk_j); pk_j).$$

2. Upon the receipt of the first message from U_i, U_j first decrypts n_i, takes a cookie CK_j, which again was pre-agreed with U_i, and arranges a responded indication of the particular algorithm set $algo$. U_j then randomly selects a nonce n_j and s_j , and sends σ_2 together with $(CK_j, CK_i, t_j, algo)$ to U_i, where t_j and σ_j are computed as follows:

$$t_j = g^{s_j} \text{ and } k = \mathsf{Hash}(n_i, n_j) \text{ and}$$

$$\sigma_2 = (\mathsf{Enc}(ID_j \| ID_i \| n_j; sk_j), \mathsf{MAC}(ID_j \| ID_i \| t_j \| t_i \| algo; k)).$$

3. Upon the receipt of the first message from U_j, U_i first decrypts the nonce n_j, computes k and verifies MAC in σ_j. If the verification fails, U_i aborts the protocol execution. Otherwise U_i sends σ_3 together with (CK_i, CK_j) to U_j, where σ_3 is computed as follows:

$$\sigma_3 = \mathsf{MAC}(ID_i \| ID_j \| t_i \| t_j \| algo; k).$$

At the end of the protocol execution, the session key is computed as $K = g^{s_i s_j}$.

3.3.2 Description of the attack

Suppose that if there are two sessions, one is for $\{U_1, U_2\}$ and the other one is for $\{U_3, U_4\}$. Suppose also that U_2, U_3 is malicious, then they can mount a BUKS attack, which is depicted in **Fig.** 3.

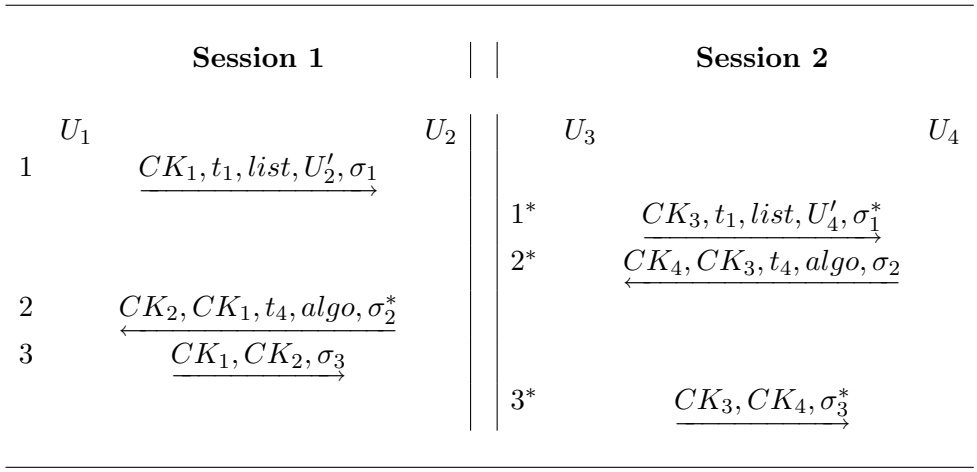

Figure 3: The BUKS attack to the alternative Oakley protocol

Note that messages 1^*, 2^* and 3^* are sent after messages 1, 2 and 3 respectively, and that the values σ_1^*, σ_2^* and σ_3^* are computed by changing to the appropriate keys, identities and cookies from the values σ_1, σ_2 and σ_3 respectively.

It is easy to see that, both sessions end "successfully", and U_1 and U_4 at the end share the same key between them, although they both accept a wrong entity's identifier.

4 The BR Model and its Existing Extensions

The pioneering work in investigating complexity-theoretic security models for key agreement protocols originates from the work of Bellare and Rogaway [3, 4]. The security notions in [3, 4] were originally proposed for key distribution protocols in the symmetric-key setting (of two-party and three party cases), but they are widely adopted for building security models for key agreement protocols. Blake-Wilson *et al.* [5, 6] adapted this model into public-key setting for two-party key agreement protocols. Bresson *et al.* [9] adapted this model for authenticated group key agreement protocols.

There are a number of other adapted variants of the Bellare-Rogaway model (e.g. those in [12, 14]), but we omit a full enumeration of them. In the literature, these models are said to be *indistinguishability-based*, which simply comes from the fact that the session key security of a protocol is evaluated by the (computational) indistinguishability between the session key and a random string.

The other type of complexity-theoretic model is those based on the *simulatability* techniques. In such models, the session key security of a protocol is evaluated by the (computational) simulatability between the ideal-world and the real-world protocol executions. The first model of this type is proposed by Bellare, Canetti, and Krawczyk [4], and later Shoup further developed this concept [25]. There are also a number adapted variants of these security models (e.g. Canetti and Krawczyk in [10]), but we also omit a full enumeration of them.

In this section we first review the BR model, and discuss how the model was extended in a number of different ways to cover some well-known security properties. We then argue that the model and its existing extension do not cover the BUKS resilience property. So we need a further extension to the model in order to cover this property.

4.1 Overview of the BR model

In the Bellare-Rogaway model, each party involved in a session is treated as an oracle. An oracle $\Pi^s_{i,j}$ denotes the s-th instance of party U_i involved with a partner party U_j in a session. The oracle $\Pi^s_{i,j}$ may accept at any time, and once accepts it should hold a partner identifier $pid = U_j$ (the identifier of the oracle with which it assumes it is communicating), a session identifier sid, and a session key sk.

The security of a key agreement protocol is evaluated by an attack game played between an adversary $\mathcal{A}$ and a hypothetical challenger $\mathcal{C}$ which sim-

ulates the protocol executions. In each attack game, the adversary can interact with oracles by issuing some specified queries which are answered by the challenger.

1. $\mathsf{Send}(\Pi^s_{i,j}, x)$. Upon receiving the message x, oracle $\Pi^s_{i,j}$ executes the protocol and responds with an outgoing message m or a decision to indicate accepting or rejecting the session. If the oracle $\Pi^s_{i,j}$ does not exist, it will be created; if $x = \lambda$ (a specifical symbol) the oracle is an initiator, otherwise it is a responder. In many existing papers, it is required that $i \neq j$, i.e., a party will not run a session with itself.

2. $\mathsf{Reveal}(\Pi^s_{i,j})$. If the oracle has not accepted, the challenger returns $\perp$; otherwise, it reveals the session key.

3. $\mathsf{Corrupt}(i)$. The challenger responds with U_i's long-term private key. Note that this is normally said to be the weak corruption model. In the case of a strong corruption model, the challenger returns all the ephemeral states of the unaccepted and unaborted oracles, besides the long-term private key.

4. $\mathsf{Test}(\Pi^s_{i,j})$. The challenger $\mathcal{C}$ acts on the input of the fresh oracle $\Pi^s_{i,j}$, randomly chooses $b \in \{0, 1\}$ and responds with the session key, if $b = 0$, or a random sample from the distribution of the session key otherwise.

For the security analysis, two types of oracles are defined: one is a partner oracle and the other is a fresh oracle.

Definition 4. *Given any oracle, let its session identifier be the concatenation of the exchanged messages in the session, then two oracles* $\Pi^s_{i,j}$ *and* $\Pi^t_{j,i}$ *are partner oracles if they have the same session identifier*[1]*. An oracle* $\Pi^s_{i,j}$ *is fresh if it satisfies the following requirements:*

1. $\Pi^s_{i,j}$ *has accepted;*

2. $\Pi^s_{i,j}$ *has not been issued any* Reveal *query;*

3. *If a partner oracle* $\Pi^t_{j,i}$ *exists,* $\Pi^t_{j,i}$ *has not been issued any* Reveal *query;*

4. *Neither* U_i *nor* U_j *has been issued any* Corrupt *query.*

[1] If two oracles hold the same session identifier, they are said to have matching conversions.

It is worth mentioning an alternative definition of partner oracles in [20], where $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ are partner oracles to each other if the following conditions hold. Let sk stand for a session key, sid stand for a session identifier, i.e. the transcripts of the session, and pid stand for a partner's identity. When running the protocol, if oracles $\Pi^s_{i,j}$ holding (sk; sid; pid) and $\Pi^t_{j,i}$ holding (sk'; sid'; pid') have both accepted and the following conditions hold :

1. $sid = sid'$, $sk = sk'$, $pid = U_j$ and $pid' = U_i$;
2. U_i is an initiator and U_j is a responder or vice versa;
3. No oracle in the game besides $\Pi^s_{i,j}$ or $\Pi^t_{j,i}$ accepts with a session identifier equal to sid.

The attack game for modelling session key security, played between an adversary $\mathcal{A}$ and a hypothetical challenger $\mathcal{C}$, is defined as follows:

1. The adversary $\mathcal{A}$ issues any of the following types of oracle queries: Send, Reveal, and Corrupt. At some point, the adversary chooses a fresh oracle $\Pi^s_{i,j}$ and issues a Test($\Pi^s_{i,j}$) query.
2. The challenger $\mathcal{C}$ randomly chooses $b \in \{0, 1\}$ and responds with the session key, if $b = 0$, or a random sample from the distribution of the session key otherwise.
3. The adversary $\mathcal{A}$ can continue querying the oracles as in the first phase, but neither Reveal query to the tested oracle $\Pi^s_{i,j}$ or its partner $\Pi^t_{j,i}$ (if it exists) nor Corrupt query to U_i or U_j. The adversary terminates by outputting a guess b' for b.

In this attack game, the adversary wins if $b' = b$, and its advantage is defined to be

$$Adv^E(k) = |\Pr[b' = b] - \frac{1}{2}|.$$

Definition 5. *A key agreement protocol is defined to* AK*-secure, if it satisfies the following requirements:*

1. *In the presence of a benign adversary, which faithfully conveys messages, on $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$, both oracles always accept holding the same session key, and this key is distributed uniformly on $\{0,1\}^k$;*
2. *$Adv^E(k)$ is negligible.*

Note that in Definition 5, we may require that even one oracle acts maliciously (for example, randomness is not sampled from a uniform distribution), the session key of its partner oracle is still distributed uniformly on $\{0,1\}^k$.

4.2 The existing extensions of the BR model

Besides the property of unknown-key-share (UKS), the following security properties are also commonly required by an authenticated key agreement protocol:

- Known session key security, i.e., that the compromise of one session key should not compromise other session keys.
- Forward secrecy, i.e., that if long-term private keys of one or more of the entities are compromised, the secrecy of previously established session keys should not be affected. In the literature there are three cases for different levels of this property: (1) the property holds if an adversary gets either one of the two player's long-term private key; (2) the property holds if an adversary gets both of the two player's long-term private keys; (3) In the identity-based key agreement protocols, the property holds if an adversary gets the master private key of the Key Generation Centre (KGC).
- Key-compromise impersonation resilience, i.e., that compromising an player's long-term private key will allow an adversary to impersonate this player, but it should not enable the adversary to impersonate other player to this player.
- Key control, i.e., that neither entity should be able to force the session key to be a preselected value.

We now take a look at whether the definition of a AK-secure key agreement protocol under the BR model in Section 4.1 implies the above properties naturally and what kinds of extension have been proposed in order to cover them. But we do not take the property of key control into account, because most of the well-known authenticated key agreement schemes including these discussed in this paper hold this property at the same level, as discussed in [24].

The property of known-key security is implied by the definitions. It follows by the following two properties of the model, as addressed in [12]: (i)

the adversary $\mathcal{A}$ is allowed to make Reveal queries to any oracle except for $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ to obtain any session keys except for the key shared between $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$, called K_{ij}, and (ii) after knowing all the other keys, her ability to distinguish between K_{ij} and a random number is still negligible. Therefore, the knowledge of any other session keys does not help $\mathcal{A}$ to deduce any information about K_{ij}.

The definitions cover UUKS resilience property. To show this we recall a small sketch, given in [12], of a proof by contradiction as follows: Suppose Π is a AK-secure protocol and suppose that Π is susceptible to the unknown key-share attack. Then $\mathcal{A}$ has a non-negligible probability of making an oracle $\Pi^s_{i,j}$ accept holding a key K where U_i believes that there has been a matching conversation with $\Pi^t_{j,i}$ and K is shared with U_j, but K is in fact shared with some other oracle $\Pi^v_{x,y}$ (usually $y = i$ here). By the definitions of the security model, $\mathcal{A}$ can make a Reveal query to $\Pi^v_{x,y}$ to obtain K because it is neither $\Pi^s_{i,j}$ nor $\Pi^t_{j,i}$. $\mathcal{A}$ can then choose oracle $\Pi^s_{i,j}$ to answer the test query. $\Pi^s_{i,j}$ will answer the test query (because both $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ are unopened and both U_i and U_j are uncorrupted) and $\mathcal{A}$ will win the game. $Adv^E(k)$ would therefore be non-negligible, contradicting the definitions.

The definitions do not imply the key-compromise impersonation property, because the model does not allow the adversary to corrupted either U_i oracle or U_j. A simple extension has been used in the literature, e.g. [12, 20], where the adversary is allowed to make a Test query to any oracle $\Pi^s_{i,j}$ where U_i (but not U_j) has been corrupted in the first step of the attack game. However, it is assumed that, although the adversary may know the long term key of the tested oracle, the adversary is not allowed to control over the the ephemeral secret of this oracle (otherwise the adversary can trivially win the game).

The definitions do not imply the forward secrecy property, again, because the model does not allow the adversary to corrupted either U_i oracle or U_j. A simple extension has also been used in the literature, e.g. [11, 20], where an adversary is allowed to make a Test query to any oracle $\Pi^s_{i,j}$ where either U_i or U_j or both U_i and U_j might be corrupted in the third step. However, it is assumed that, although the adversary may know the long term key of these two oracles, the adversary is not allowed to control over the the ephemeral secret of any of two oracles if they are corrupted (otherwise the adversary can trivially win the game).

It is not difficult to see that these existing extensions are still not able to cover the BUKS resilience property. An obvious fact is that in the BR model and its existing extensions, the adversary is not allowed to corrupt U_j in the first step of the attack game, which implies that the attack game cannot

cover the situation where U_j acts maliciously in the protocol execution or U_j is compromised by the adversary. However this is not the major reason why the BUKS resilience property cannot be analyzed under the BR model and its existing extensions.

A non-trivial point is that the BR model and its existing extensions do not distinguish a special oracle, which is not a partner oracle of the test oracle because it does not have a matching conversation with the test oracle. However it does have a distinguishable attribute since it shares the established session key with the test oracle. In the BR model and its existing extensions, any oracle except the test oracle and its partner oracles can be corrupted and revealed. The speciality of this oracle in the BUKS attack is that it is not compromised by a BUKS adversary. Therefore, there is a gap between the traditional BR-type security proof and the BUKS adversary's behavior. In the next session, we take this type of oracles into account when making a further extension of the BR model. As a result, the extended BR model implies the BUKS resilience property.

5 New BR Model Extension to Cover BUKS

In this section, we present a new extension to the BR model in order to cover the BUKS resilience property. We then take the modified STS protocol as an example to demonstrate that this protocol is insecure under the extended model.

5.1 The proposed extension

We first define a new type of partner oracles, which are called *semi-partner* oracles.

Definition 6. *$\Pi^s_{i,j}$ and $\Pi^t_{u,v}$ are semi-partner oracles to each other if when running the protocol, $\Pi^s_{i,j}$ holding (sk; sid; pid) and $\Pi^t_{u,v}$ holding (sk′; sid′; pid′) have both accepted and the condition $sk = sk'$ holds.*

Compared with the definition of partner oracles in [20], a pair of semi-partner oracles might not hold the conditions of $sid = sid'$, $pid = U_u$ and $pid' = U_i$.

We now describe an attack game for modelling the BUKS resilience property. The game is identical to the attack game for session key security, except that an adversary $\mathcal{A}$ is allowed to issue the Test query to any oracle $\Pi^s_{i,j}$, which satisfies the following conditions:

1. $\Pi^s_{i,j}$ has accepted and has not been issued any Reveal query; U_i has not been corrupted.

2. If a partner oracle $\Pi^t_{j,i}$ exists, then $\Pi^t_{j,i}$ has not been issued any Reveal query; U_j might have been corrupted, but the adversary is not allowed to control over the ephemeral secret which forms the input to $\Pi^s_{i,j}$ [2];

3. If a semi-partner orale $\Pi^w_{u,v}$ exists, U_u has not been corrupted.

Compared with the definition of a fresh oracle in Definition 4, the oracle $\Pi^s_{i,j}$ above might have corrupted partner oracles and incorrupted semi-partner oracles. Note that in this game the adversary is allowed to make a Reveal query to a semi-partner oracle if it is not a partner oracle. So if the test oracle has a semi-partner oracle, the adversary can easily win the game.

In a general speak, any two-party authenticated key agreement protocol fails to hold the BUKS resilience property, under this extended BR model, as long as there are a pair of semi-partner oracles. In the other words, in the authenticated key agreement protocol, if no such a pair of semi-partner oracles exist, we can argue that the protocol holds the property of BUKS resilience.

Inherently, we have the following definition for a key agreement protocol to be AK-secure with BUKS resilience.

Definition 7. *A key agreement protocol is said to* AK*-secure with BUKS resilience, if it is* AK*-secure and any adversary has only negligible advantage in the above game.*

5.2 Further analysis of the modified STS protocol

We now show why the modified STS protocol is insecure in this extended BR model. We reuse the attack specified in Section 3.2.2, and show that the adversary could win the game for modelling BUKS resilience with a non-negligible advantage. During the attack, the adversary $\mathcal{A}$ asks the following queries:

1. $\mathsf{Send}(\Pi^s_{12}, \lambda)$, where $\mathcal{A}$ is responded with $m_1 = g^{s_1}$ and s_1 is the ephemeral secret.

2. $\mathsf{Send}(\Pi^u_{43}, m_3)$, where $m_3 = m_1$ and $\mathcal{A}$ is responded with

$$m_4 = (g^{s_4}, \sigma_4),\ \sigma_4 = \mathsf{Sign}(g^{s_4}||g^{s_1}||ID_3; sk_4),$$

and s_2 is the ephemeral secret.

[2] Otherwise the adversary can trivially win the game.

3. $\mathsf{Corrupt}(U_2)$, where $\mathcal{A}$ is responded with U_2's private signing key.

4. $\mathsf{Send}(\Pi_{12}^{s}, m_2)$, where

$$m_2 = (g^{s_4}, \sigma_2),\ \sigma_2 = \mathsf{Sign}(g^{s_4}||g^{s_1}||ID_1; sk_2),$$

and is responded with σ_1 where

$$\sigma_1 = \mathsf{Sign}(g^{s_1}||g^{s_4}||ID_2; sk_1),$$

5. $\mathsf{Corrupt}(U_3)$, where $\mathcal{A}$ is responded with U_3's private signing key.

6. $\mathsf{Send}(\Pi_{43}^{u}, \sigma_3)$, where

$$\sigma_3 = \mathsf{Sign}(g^{s_1}||g^{s_4}||ID_4; sk_3),$$

7. $\mathsf{Test}(\Pi_{12}^{s})$, where $\mathcal{A}$ is responded with a value k'.

8. $\mathsf{Reveal}(\Pi_{43}^{u})$, where $\mathcal{A}$ is responded with a session key $k = g^{s_1 s_2}$, because obviously Π_{43}^{u} is not the partner oracle but a semi-partner oracle of the test oracle Π_{12}^{s}. $\mathcal{A}$ terminates the game by outputting a guess $b' = 0$ if $k = k'$, otherwise $b' = 1$.

Since the session key accepted by Π_{43}^{u} is identical to the session key accepted by Π_{12}^{s}, $\mathcal{A}$ wins the game with the probability 1. The game is valid according to our extended BR model in Section 5.1, because the adversary does not control over the ephemeral secret of U_2 but only its long-term private signing key, and the adversary does not corrupt U_4.

6 A Simple Countermeasure

One possible countermeasure to prevent an authenticated key agreement protocol from the BUKS attack is to add the identifiers of the parties and the transcripts of the protocol as part of input to a key derivation function.

For example, in the DHKE-1 protocol of Section 3.1.1, the value $\mathsf{H_k}(g^{s_i s_j})$ can be replaced with

$$\mathsf{H_k}(U_i||U_j||g^{s_i}||\sigma_i||Cert_i||g^{s_j}||\sigma_j||Cert_j||g^{s_i s_j}).$$

In the modified STS protocol of Section 3.2.1, the established session key $K = g^{s_i s_j}$ can be replaced with

$$K = \mathsf{H}(ID_i||ID_j||g^{s_i}||\sigma_i||pk_i||g^{s_j}||\sigma_j||pk_j||g^{s_i s_j}),$$

where H is a key derivation function. A similar change can be made in the alternative Oakley protocol of Section 3.3.1.

In practice, it is very important to include the certificates (or certified public keys) in the computation of the session key, since, due to various reasons, it might be hard to guarantee that an identifier will never be used by two different users.

By taking the modified STS protocol as an example again, we can show that this simple countermeasure works. After adding the player identifiers and the protocol transcripts into the session key derivation, the key computed by Π_{12}^{s} is not equal to the key computed by Π_{43}^{u} with a reasonably large probability. So the two oracles are no longer a pair of semi-partners. Therefore the BUKS resilience attribute holds.

Note that adding the identifiers and transcripts into the key derivation function is not a new solution. The same idea has been used in many papers for different purposes; for example, it was suggested by Choo, Boyd and Hitchcockin in [18] and by Cheng and Chen in [13] to help security proof.

7 Conclusions

In this paper, we have proposed a new UKS attack, namely the BUKS attack, and demonstrated that some well-known authenticated key agreement protocols are vulnerable to it. We have also investigated the gap between the traditional BR-type proof of UUKS resilience and a BUKS adversary's behavior, and provided a modification of the proof and these protocols to cover the BUKS resilience attribute.

References

[1] Jr. B. S. Kaliski. An unknown key-share attack on the MQV key agreement protocol. *ACM Transactions on Information and System Security*, 4(3):275–288, 2001.

[2] J. Baek, K. Kim, and T. Matsumoto. On the significance of unknown key-share attacks: How to cope with them? In xx, editor, *Proc. of Symposium on Cryptography and Information Security*, page xx, 2000.

[3] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *Advances in Cryptology – Crypto 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 110–125. Springer-Verlag, 1993.

[4] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *STOC '95: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 57–66. ACM Press, 1995.

[5] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *Proceedings of Cryptography and Coding, 6th IMA International Conference*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 1997.

[6] S. Blake-Wilson and A. Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. In B. Christianson, B. Crispo, T. Lomas, and M. Roe, editors, *Proceedings of Security Protocols, 5th International Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 137–158. Springer, 1997.

[7] S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the station-to-station (sts) protocol. In H. Imai and Y. Zheng, editors, *Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 154–170. Springer, 1999.

[8] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, 2004.

[9] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 255–264. ACM Press, 2001.

[10] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology — Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer-Verlag, 2001.

[11] L. Chen, Z. Cheng, and N. Smart. Identity-based key agreement protocols from pairings. Cryptology ePrint Archive: Report 2006/199, 2006.

[12] L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairings. In *Proc. of the 16th IEEE Computer Security Foundations Workshop — CSFW 2003*, pages 219–233. IEEE Computer Society Press, 2003.

[13] Z. Cheng and L. Chen. On security proof of Mccullagh-Barreto's key agreement protocol and its variants. *International Journal of Security and Networks*, 2(3/4):251–259, 2007.

[14] Z. Cheng, L. Chen, R. Comley, and Q. Tang. Identity-based key agreement with unilateral identity privacy using pairings. In K. Chen, R. H. Deng, X. Lai, and J. Zhou, editors, *Proceedings of the Second International Conference on Information Security Practice and Experience*, volume 3903 of *Lecture Notes in Computer Science*, pages 202–213. Springer, 2006.

[15] W. Diffie, P. Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2(2):107–125, 1992.

[16] L. Harn and H. Y. Lin. Authenticated key agreement without using oneway hash functions. *Electronics Letters*, 37(10):1429–1431, 2001.

[17] S. Hirose and S. Yoshida. An authenticated diffie-hellman key agreement protocol secure against active attacks. In H. Imai and Y. Zheng, editors, *Proceedings of the first International Workshop on Practice and Theory in Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 135–148. Springer, 1998.

[18] C. Boyd K. Choo and Y. Hitchcock. On session key construction in provably-secure key establishment protocols: revisiting chen & kudla (2003) and mccullagh & barreto (2005) id-based protocols. In *Proceedings of* Mycrypt 2005, volume 3715 of *Lecture Notes in Computer Science*, pages 116–131. Springer-Verlag, 2005.

[19] J. Baek K. Kim. Remarks on the unknown key share attacks. *IEICE TRANSACTIONS*, E83-A(12):2766–2769, 2000.

[20] C. Kudla and K. G. Paterson. Modular security proofs for key agreement protocols. In B. K. Roy, editor, *Advances in Cryptology — ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 549–565. Springer-Verlag, 2005.

[21] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography*, 28(2):119–134, 2003.

[22] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key-distribution systems. *IEICE TRANSACTIONS*, E69(2):99–106, 2000.

[23] A. Menezes and S. Vanstone. Some new key agreement protocols providing mutual implicit authentication. In *Proceedings of the Second Workshop on Selected Areas in Cryptography*, pages 22–32, 1995.

[24] H. Orman. The oakley key determination protocol. The Internet Society. RFC 2412, 1998.

[25] V. Shoup. On formal models for secure key exchange. Technical report, IBM Research Report RZ 3120, 1998.

[26] H. Zhou, L. Fan, and J. Li. Remarks on unknown key-share attack on authenticated multiple-key agreement protocol. *Electronics Letters*, 39(17):1248–1249, 2003.