Journal of Universal Computer Science, vol. 14, no. 3 (2008), 416-440
submitted: 2/6/07, accepted: 1/11/07, appeared: 1/2/08 © J.UCS

Bilateral Unknown Key-Share Attacks in Key Agreement Protocols

Liqun Chen
(Hewlett-Packard Laboratories, UK
liqun.chen@hp.com)

Qiang Tang[1]
(Ecole Normale Supérieure, France
qiang.tang@ens.fr)

Abstract: Unknown Key-Share (UKS) resilience is a basic security attribute in authenticated key agreement protocols. In this paper we revisit the definitions of this attribute and the method of proving this attribute under the Bellare-Rogaway (BR) model in the literature. We propose a new type of UKS attack, which coerces two entities A and B into sharing a key with each other but in fact A thinks that he 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. We demonstrate that a few well-known authenticated key agreement protocols are vulnerable to this attack. We then explore a gap between the conventional BR-type proof and a BUKS adversary's behavior, and extend the BR model to cover the BUKS resilience attribute. At the end of the paper, we provide a general countermeasure and its security proof under the extended model and the assumption that a collision-resistance function exists.

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

Categories: E.3, H.1.1, H.4.3

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.

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

The concept of UKS resilience was originated by Diffie, Oorschot and Wiener in [Diffie et al. 1992] based on the discussion about the key authentication property in key agreement protocols. 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.

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 the definitions in [Blake-Wilson et al. 1999], 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.

To show the potential vulnerability caused by a UUKS attack, let us recall the interesting hypothetical scenario described in [Diffie et al. 1992]. 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, it has been shown that a number of key agreement protocols are vulnerable to the UUKS attack; for example, the MTI/A0 protocol [Matsumoto et al. 2000] was attacked by Menezes *et al.* [Menezes et al. 1995], the STS-MAC variant of the Station-to-Station (STS) protocol [Diffie et al. 1992] was attacked by Blake-Wilson and Menezes [Blake-Wilson et al. 1999], the revised STS-MAC protocol [Blake-Wilson et al. 1999] and the KAP-HY98 protocol [Hirose et al. 1998] were both attacked by Baek *et al.* [Baek et al. 2000, Baek and Kim 2000], the MQV protocol [Law et al. 2003, Menezes et al. 1995] was attacked by Kaliski [Kaliski 2001], and the Harn-Lin's modified MQV protocol [Harn and Lin 2001] was attacked by Zhou *et al.* [Zhou et al. 2003].

Recently, some researchers have worked on modelling the UUKS resilience for authenticated key agreement protocols (e.g. those in [Chen and Kudla 2003, Cheng et al. 2006]). Their works are based on the Bellare-Rogaway (BR) model in [Bellare and Rogaway 1993, Blake-Wilson et al. 1997]. It is worth mentioning that proving the property of UUKS resilience in a key agreement protocol is not easy. Some protocols, which had flawed security proof under the BR model, have subsequently been found not to hold the UUKS resilience. Examples include the conference key agreement protocol of Boyd and González Nieto in [Boyd et al. 2003] that was broken by Choo et al. in [Choo et al. 2005-2].

1.1 Our Contributions

Our major contributions in this paper are as follows. After revisiting some definitions of the UUKS attack, we propose a new type of 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 [Shoup 1998], the modified STS protocols [Boyd and Mathuria 2004] and the modified Oakley protocol [Boyd and Mathuria 2004], are vulnerable to the BUKS attack. Considering some of these protocols have been proved holding the UUKS resilience in the BR model, we explore the gap between the traditional Bellare-Rogaway-type proof of UUKS resilience and a BUKS adversary's behavior. Finally, we extend the BR model to cover the BUKS resilience attribute.

1.2 Organization

The remainder of this paper is organized as follows. The concept of a BUKS attack is described in Section 2. In Section 3, we show that three sets of well-known protocols do not achieve BUKS resilience. 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 Bellare-Rogaway-type proof and a BUKS adversary's behavior, and then provide an extension 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 and prove it works under the extended BR model. We finally conclude the paper with comments to how the Canetti and Krawczyk model [Canetti and Krawczyk 2001] could cover the BUKS resilience and some further work 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 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 [Blake-Wilson et al. 1999] proposed the first formal definition of the UKS attack, which has been adopted by many researchers after that. Their definition is as follows.

Definition 1. An unknown key-share attack against an authenticated key agreement protocol is an attack, whereby an entity A ends up believing it shares a key with another entity B which 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 attack, 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.

In other places (e.g. [Chen and Kudla 2003, Cheng et al. 2006]), to make the definition more general, the UKS attack is defined as follows.

Definition 2. An unknown key-share attack against 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 attack, 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 consider Definition 1 to be 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 against 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 key agreement protocol, which suffers from the BUKS attacks, will achieve key authentication for neither direction.

Following the hypothetical scenario for the UUKS attack 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

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

In this section, we demonstrate how the following three sets of protocols are vulnerable to the BUKS attack: (1) the DHKE protocols [Shoup 1998], (2) the two modified STS protocols [Boyd and Mathuria 2004], and (3) the alternative Oakley protocol [Boyd and Mathuria 2004].

Please note that our BUKS attack to these protocols does not mean that the design of these protocols have been failed. As was stated by Boyd and Mathuria in [Boyd and Mathuria 2004], "any attack on a protocol is only valid if it violates some property that the protocol was intended to achieve". Neither of these protocols originally has the target of preventing from the BUKS attack. However, since applications of authenticated key agreement protocols have been developing continuously, many new applications require new and more sensitive security attributes. As we discussed in Section 2, the application of on-line purchase would require a robust authenticated key agreement protocol holding the BUKS resilience property. From demonstrating the attack we hope to build the awareness of that a number of protocols do not hold the BUKS resilience property, so it should be careful when using them in the applications that do require this property.

3.1 Analysis of the DHKE protocols

Shoup [Shoup 1998] proposed a set of four DHKE protocols (called DHKE, DHKE-1, DHKE-2 and DHKE-3), and proved they are secure, where the security of DHKE was proved in the static corruption mode and the security of others 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 BUKS 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 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 there are two sessions, where one is for $\{U_1, U_2\}$ and the other is for $\{U_3, U_4\}$. If U_2 and U_3 are malicious, then they can mount a BUKS attack which is depicted in Figure 1.

	Session 1				**Session 2**	
	U_1		U_2		U_3	
1.	$\xrightarrow{g^{s_1}, \sigma_1, Cert_1}$					
				1*.	$\xrightarrow{g^{s_1}, \sigma_3, Cert_3}$	
				2*.	$\xleftarrow{g^{s_4}, k, \sigma_4, Cert_4}$	
2.	$\xleftarrow{g^{s_4}, k, \sigma_2, Cert_2}$					
3.	$\xrightarrow{k_1}$					
				3*.	$\xrightarrow{k_1}$	

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, when both sessions end successfully, U_1 and U_4 share the same session 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 [Bellare et al. 1998] 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 an 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 [Boyd and Mathuria 2004], namely Protocol 5.16 and Protocol 5.17. We show that both protocols suffer from the BUKS attack, although Protocol 5.17 makes use of an additional Message Authentication Code (MAC) algorithm to enhance its 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 in 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 $K = g^{s_i s_j}$.

Compared to Protocol 5.16, the only difference in Protocol 5.17 is in the computation of σ_j and σ_i, where

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

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

and 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 there are two sessions, where one is for $\{U_1, U_2\}$ and the other is for $\{U_3, U_4\}$. If U_2 and U_3 are malicious, then they can mount a BUKS attack which is depicted in Figure 2.

	Session 1		**Session 2**
	$U_1 \qquad\qquad U_2$		$U_3 \qquad\qquad U_4$
1.	$\xrightarrow{g^{s_1}}$		
		1^*.	$\xrightarrow{g^{s_1}}$
		2^*.	$\xleftarrow{g^{s_4}, \sigma_4}$
2.	$\xleftarrow{g^{s_4}, \sigma_2}$		
3.	$\xrightarrow{\sigma_1}$		
		3^*.	$\xrightarrow{\sigma_3}$

Figure 2: The BUKS attack to the modified STS protocol

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, when both sessions end successfully, 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 [Boyd and Mathuria 2004]. The original Oakley protocol was given in Internet RFC 2412 [Orman 1998].

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_2 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 the MAC value in σ_2. 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 there are two sessions, where one is for $\{U_1, U_2\}$ and the other is for $\{U_3, U_4\}$. If U_2 and U_3 are malicious, then they can mount a BUKS attack which is depicted in Figure 3.

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, when both sessions end successfully, U_1 and U_4 at the end share the same key between them although they both accept a wrong entity's identifier.

	Session 1				**Session 2**	
U_1		U_2	U_3			U_4
1	$\xrightarrow{CK_1, t_1, list, ID_2, \sigma_1}$					
				1*	$\xrightarrow{CK_3, t_1, list, ID_4, \sigma_1^*}$	
				2*	$\xleftarrow{CK_4, CK_3, t_4, algo, \sigma_2^*}$	
2	$\xleftarrow{CK_2, CK_1, t_4, algo, \sigma_2}$					
3	$\xrightarrow{CK_1, CK_2, \sigma_3}$					
				3*	$\xrightarrow{CK_3, CK_4, \sigma_3^*}$	

Figure 3: The BUKS attack to the alternative Oakley protocol

4 Bellare-Rogaway (BR) Model and its Extensions

The pioneering work in investigating complexity-theoretic security models for authenticated key agreement protocols originates from the work of Bellare and Rogaway [Bellare and Rogaway 1993, Bellare and Rogaway 1995]. The security notions in [Bellare and Rogaway 1993, Bellare and Rogaway 1995] 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.* [Blake-Wilson et al. 1997, Blake-Wilson and Menezes 1997] adapted this model into public-key setting for two-party key agreement protocols. Bresson *et al.* [Bresson et al. 2001] adapted this model for authenticated group key agreement protocols.

There are a number of other adapted variants of the BR model (e.g. those in [Chen and Kudla 2003, Cheng et al. 2006]), 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 [Bellare et al. 1998], and later Shoup further developed this concept [Shoup 1998]. There are also a number adapted variants of these security models (e.g. that of Canetti and Krawczyk [Canetti and Krawczyk 2001]), 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. So we need a further extension to the model in order to cover this property.

4.1 Overview of the BR model

In the BR 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 with), a session identifier sid, i.e. the transcripts of the session, 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 simulates the protocol executions. In each attack game, the adversary can interact with oracles by issuing some specified queries, as follows, 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[2].

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 partner oracle and the other is fresh oracle. In the original BR model, 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

[2] 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.

session identifier[3]. It is worth mentioning an extended definition of partner oracle in [Kudla and Paterson 2005] as follows. In the remaining part of this paper, we make use of this definition for the partner oracles.

Definition 4. Let sk stand for a session key, sid stand for a session identifier, and pid stand for a partner's identity. $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$, which hold $(sk; sid; pid)$ and $(sk'; sid'; pid')$ respectively, are partner oracles to each other if 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.

Definition 5. 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.

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 $\Pi^s_{i,j}$ nor 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'.

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

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

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

Definition 6. 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. $Adv^{\mathcal{A}}(k)$ is negligible.

Note that in Definition 6, 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 Existing Extensions to the BR model

Besides the UKS resilience, the following security properties are also commonly required by an authenticated key agreement protocol:

– Known-key security: the compromise of one session key should not compromise other session keys.

– Forward secrecy: 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. 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: compromising a 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.

We now take a look at whether the definition of a AK-secure key agreement protocol in 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.

First, the property of known-key security is implied by the definition. As addressed in [Chen and Kudla 2003], this follows because of two features in the model: (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 string is still negligible. Therefore, the knowledge of any other session keys does not help $\mathcal{A}$ to deduce any information about K_{ij}.

Secondly, UUKS resilience is implied by the definition. To show this, we recall a sketch, given in [Chen and Kudla 2003], of a proof by contradiction as follows: Suppose $\mathit{\Pi}$ is a AK-secure protocol and suppose that $\mathit{\Pi}$ is susceptible to the unknown key-share attack. Then $\mathcal{A}$ has a non-negligible probability of making an oracle $\mathit{\Pi}^s_{i,j}$ accept holding a key K where U_i believes that there has been a matching conversation with $\mathit{\Pi}^t_{j,i}$ and K is shared with U_j, but K is in fact shared with some other oracle $\mathit{\Pi}^v_{x,y}$ (usually $y = i$ here). By the definitions of the security model, $\mathcal{A}$ can make a Reveal query to $\mathit{\Pi}^v_{x,y}$ to obtain K because it is neither $\mathit{\Pi}^s_{i,j}$ nor $\mathit{\Pi}^t_{j,i}$. $\mathcal{A}$ can then choose oracle $\mathit{\Pi}^s_{i,j}$ to answer the Test query. $\mathit{\Pi}^s_{i,j}$ will answer the test query (because both $\mathit{\Pi}^s_{i,j}$ and $\mathit{\Pi}^t_{j,i}$ are unopened and both U_i and U_j are uncorrupted) and $\mathcal{A}$ will win the game. $Adv^{\mathcal{A}}(k)$ would therefore be non-negligible, contradicting the definitions.

Thirdly, key-compromise impersonation resilience is not implied by the definition, because the model does not allow the adversary to corrupt U_i or U_j. A simple extension has been used in the literature (e.g. [Chen and Kudla 2003, Kudla and Paterson 2005]), where the adversary is allowed to make a Test query to any oracle $\mathit{\Pi}^s_{i,j}$ 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).

Fourthly, forward secrecy is not implied by the definition, because the model does not allow the adversary to corrupted either U_i or U_j. Again, a simple extension has been used (e.g. [Chen et al. 2006, Kudla and Paterson 2005]) in the literature, where an adversary is allowed to make a Test query to any oracle $\mathit{\Pi}^s_{i,j}$ where either 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 oracle which is generated before the corruption (otherwise the adversary can trivially win the game).

Note that we have not included the property of key control, i.e. neither entity should be able to force the session key to be a preselected value, because most of the well-known authenticated key agreement schemes including these discussed in this paper hold this property at the same level, as shown in [Mitchell et al. 1998]. Another security property, namely resetting compromised long-term keys in a key establishment protocol, also has not been considered in our discussion. Boyd et al. [Boyd et al. 2006] extended the BR model by adding a Reset query to cover this property.

Based on the definition of the BUKS attacks described in Section 3, it is not difficult to see that the BR model and its existing extensions are unable to cover the BUKS resilience. An obvious fact is that in any of them, 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 (i.e. U_j is corrupted by the adversary). Therefore, in order to cover the BUKS resilience, we should increase the adversary's privilege by allowing it to corrupt U_j. This modification is trivial, since the same trick has been used in some of the above extensions. However, this extension is not complete in order to cover the BUKS resilience for the following reason.

In the BR model and its existing extensions, there are three types of oracles: the tested oracle, its partner oracle and a set of other oracles. Any oracle except the first two types can be corrupted and revealed. A non-trivial observation is that in the BUKS attack there is a special oracle, which cannot be allocated in either of the three types: It is not the tested oracle; it is not the partner oracle, because it does not have a matching conversation with the tested oracle; it is not the ordinary other oracle, because it is not corrupted. So, this is the gap between the traditional Bellare-Rogaway-type security formalization and the BUKS adversary's behavior.

In the next section, we will take this type of oracles into account along with the above simple modification to describe a further extension to the BR model, which covers the BUKS resilience.

5 New Extension to the BR Model

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

5.1 The proposed extension

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

Definition 7. Two accepted oracles $\Pi^s_{i,j}$ and $\Pi^w_{u,v}$, which possess $(sk; sid; pid)$ and $(sk'; sid'; pid')$ respectively, are semi-partner oracles to each other if the condition $sk = sk'$ holds but the condition $pid = U_u$ and $pid' = U_i$ does not hold.

With respect to the definition of partner oracles in Definition 4, if $\Pi^s_{i,j}$ and $\Pi^w_{u,v}$ are partner oracles then they are not semi-partner oracles. Based on the definitions of the semi-partner oracles and the partner oracles, We now define a new type of fresh oracles.

Definition 8. An oracle $\Pi^s_{i,j}$ is fresh if it satisfies the following requirements:

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

2. Regarding the existence of partner oracles and semi-partner oracles, one of the following four conditions must hold:
   (a) If both a partner oracle $\mathit{\Pi}_{j,i}^{t}$ and a semi-partner oracle $\mathit{\Pi}_{u,v}^{w}$ exist, U_u has not been corrupted. [4]
   (b) If no partner oracle exists but a semi-partner oracle $\mathit{\Pi}_{u,v}^{w}$ exists, U_u has not been corrupted. [5]
   (c) If a partner oracle $\mathit{\Pi}_{j,i}^{t}$ exists but no semi-partner oracle exists, then $\mathit{\Pi}_{j,i}^{t}$ has not been issued any Reveal query; U_j might have been corrupted, but in that case the adversary is not allowed to control over the ephemeral secret which forms the input to $\mathit{\Pi}_{i,j}^{s}$; alternatively, the ephemeral secret which forms the input to $\mathit{\Pi}_{i,j}^{s}$ might have been controlled by the adversary, but in that case, U_j has not been corrupted.
   (d) Neither a partner oracle nor a semi-partner oracle exists.

We now describe an attack game for modelling *the BUKS resilience*, which is as follows:

1. In the first phase, the adversary $\mathcal{A}$ issues any type of oracle queries including Send, Reveal, and Corrupt, as specified in Section 4.1.
2. At some point, the adversary chooses a fresh oracle $\mathit{\Pi}_{i,j}^{s}$ as defined in Definition 8 and issues a Test query.
3. 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.
4. The adversary $\mathcal{A}$ can continue querying the oracles as in the first phase, but cannot make any query, which will make $\mathit{\Pi}_{i,j}^{s}$ no longer satisfying the definition of a fresh oracle as defined in Definition 8. The adversary terminates by outputting a guess b'.

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

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

[4] In this case, the adversary is allowed to corrupt U_j and to control the ephemeral secret of $\mathit{\Pi}_{j,i}^{t}$; the adversary is also allowed to issue a Reveal query to $\mathit{\Pi}_{u,v}^{w}$. So if this case happens, the adversary can trivially win the game.

[5] Again, if this case happens, the adversary can trivially win the game, since the adversary is allowed to issue a Reveal query to $\mathit{\Pi}_{u,v}^{w}$.

In contrast to the original game, this new attack game has the following two additional features which bridge the gap identified in the previous section.

- Compared with the definition of a fresh oracle in Definition 5, the oracle $\Pi_{i,j}^s$ above might have corrupted partner oracles and uncorrupted semi-partner oracles. More specifically, the oracle $\Pi_{i,j}^s$ might have an uncorrupted semi-partner oracle $\Pi_{u,v}^w$, which implies that the situation of unknown key share may exist; the oracle U_j and U_v might have been corrupted, which implies that malicious insider adversary may exist.
- As the same as in the original BR model, the adversary is not allowed to issue a Reveal query to the partner oracle of $\Pi_{i,j}^s$; otherwise the adversary can trivially win the game. However, in this new attack game, the adversary is allowed to make a Reveal query to a semi-partner oracle. As a result, if a two-party authenticated key agreement protocol fails to hold the BUKS resilience then the adversary will win the new attack game.

Consequently, we have the following enhanced security definition for authenticated key agreement protocols.

Definition 9. An authenticated key agreement protocol is said to AK-secure with BUKS resilience, if it is AK-secure and any polynomial adversary has only negligible advantage in the above game.

We have the following security result.

Theorem 1 *The property of BUKS resilience is implied by the definition of the above extended BR model.*

Proof. Suppose the following two facts exist with a single protocol Π: (I) Π is a AK-secure protocol with BUKS resilience (i.e. for any polynomial adversary $\mathcal{A}$ in the above attack game, $Adv^{\mathcal{A}}(k)$ is negligible) and (II) Π is susceptible to the BUKS attack. We now show contradiction between (I) and (II), which is equivalent to Theorem 1.

Based on the fact (II), $\mathcal{A}$ has a non-negligible probability of making an oracle $\Pi_{i,j}^s$ accept holding a key K where U_i believes that there has been a matching conversation with another oracle $\Pi_{j,i}^t$ (i.e. $\Pi_{j,i}^t$ is its partner oracle and K is shared with U_j). But K is in fact shared with some other uncorrupted oracle $\Pi_{u,v}^w$ (where $u \neq j$ here). Following Definition 7, $\Pi_{u,v}^w$ is its semi-partner oracle. By the definition of the extended BR security model, $\mathcal{A}$ can then choose $\Pi_{i,j}^s$ to answer the Test query. The oracle $\Pi_{i,j}^s$ is fresh because the following conditions hold:

- $\Pi_{i,j}^s$ has accepted and has not been issued any Reveal query; U_i has not been corrupted.

– $\Pi^s_{i,j}$ has a semi-partner oracle $\Pi^w_{u,v}$, and U_u has not been corrupted.

By the definition of the extended BR security model, $\mathcal{A}$ can make a Reveal query to $\Pi^w_{u,v}$ to obtain K. As a result, $\mathcal{A}$ will win the game and $Adv^{\mathcal{A}}(k)$ would therefore be non-negligible. This result contradicts the fact (I). So the theorem follows. □

5.2 Further analysis of the modified STS protocol

We now show why the modified STS protocol is insecure in this extended Bellare-Rogaway 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_4 is the ephemeral secret.
3. $\mathsf{Corrupt}(U_2)$, where $\mathcal{A}$ is responded with U_2's private signing key.
4. $\mathsf{Send}(\Pi^s_{12}, m_2)$, where
$$m_2 = (g^{s_4}, \sigma_2),\ \sigma_2 = \mathsf{Sign}(g^{s_4}||g^{s_1}||ID_1; sk_2),$$
and $\mathcal{A}$ is then 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. $\mathsf{Send}(\Pi^u_{43}, \sigma_3)$, where
$$\sigma_3 = \mathsf{Sign}(g^{s_1}||g^{s_4}||ID_4; sk_3),$$
7. $\mathsf{Test}(\Pi^s_{12})$, where $\mathcal{A}$ is responded with a value k'.
8. $\mathsf{Reveal}(\Pi^u_{43})$, where $\mathcal{A}$ is responded with a session key $k = g^{s_1 s_4}$, because obviously Π^u_{43} is not the partner oracle but a semi-partner oracle of the test oracle Π^s_{12}. $\mathcal{A}$ terminates the game by outputting a guess $b' = 0$ if $k = k'$, otherwise $b' = 1$.

Since the session key accepted by Π^u_{43} is identical to the session key accepted by Π^s_{12}, $\mathcal{A}$ 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 corrupt U_1 and U_4 and does not issue any Reveal query to $\Pi^s_{i,j}$.

6 A Simple Countermeasure

Let Π be a two party authenticated key agreement protocol. By the definition of the extended BR security model, if Π has any pair of semi-partner oracles, Π does not hold the property of BUKS resilience; otherwise, Π holds the property of BUKS resilience. Our goal is to provide a general countermeasure to prevent Π from the BUKS attack, i.e., to avoid such a pair of semi-partner oracles when running Π with overwhelming probability.

Let ID_i denote the identifier of participant U_i and $A\|B$ denote the concatenation between the data strings A and B. Let $f(\Pi^s_{i,j})$ denote a function, which takes $\Pi^s_{i,j}$ as input and outputs the value $pid_{ij} = ID_i\|ID_j$, if U_i believes that $\Pi^s_{i,j}$ is an initiator in Π and it has a partner oracle $\Pi^t_{j,i}$ as a responder; or outputs the value $pid_{ij} = ID_j\|ID_i$, if U_i believes that $\Pi^s_{i,j}$ is a responder in Π and it has a partner oracle $\Pi^t_{j,i}$ as an initiator. Suppose each participant has a unique identifier. In practice, it is possible to include the certificates (or certified public keys) in the content of the identifier, since, due to various reasons, it might be hard to guarantee that an ordinary identity, such as a name, will never be used by two different users.

Let $\Pi^s_{i,j}$ and $\Pi^w_{u,v}$ be an arbitrary pair of oracles running Π and they hold (sk_i, sid_i, pid_{ij}) and (sk_u, sid_u, pid_{uv}) respectively. Our proposed solution is to add the identifiers of the parties and the transcripts of the protocol along with the original session secret as input to a Key Derivation Function KDF to create a new session key.

Let ℓ_{KDF} be a security parameter and $\mathsf{KDF} : \{0,1\}^* \rightarrow \{0,1\}^{\ell_{\mathsf{KDF}}}$ denote a collision-resistant function[6], where the output of KDF is distributed from $\{0,1\}^{\ell_{\mathsf{KDF}}}$ uniformly at random. In the proposed countermeasure, the new session keys of $\Pi^s_{i,j}$ and $\Pi^w_{u,v}$ are computed respectively by

$$sk'_i = \mathsf{KDF}(pid_{ij}, sid_i, sk_i), pid_{ij} = f(\Pi^s_{i,j});$$
$$sk'_u = \mathsf{KDF}(pid_{uv}, sid_u, sk_u), pid_{uv} = f(\Pi^s_{u,v}).$$

We have the following security result for the above countermeasure based on the collision-resistance property of KDF.

Theorem 2 *Let Π be a key agreement protocol and $\tilde{\Pi}$ be the modification of Π by replacing a session key sk_i of oracle $\Pi^s_{i,j}$ running Π with $sk'_i = \mathsf{KDF}(pid_{ij}$, sid_i, $sk_i)$. If KDF is a collision-resistant function, then $\tilde{\Pi}$ holds the BUKS resilience under the extended BR model defined in Section 5.1.*

Proof. Suppose $\tilde{\Pi}$ is susceptible to the BUKS attack, i.e., there exists an adversary $\mathcal{A}$ in the attack game defined in Section 5.1 such that $Adv^{\mathcal{A}}(k)$ is non-negligible. We will show how such an adversary $\mathcal{A}$ may be used to construct a

[6] It is computationally infeasible to find two different inputs that result the same output of KDF.

simulator $\mathcal{S}$ that breaks the collision-resistance property of KDF, i.e. find a pair of different inputs mapping to the same output.

We now describe the construction of the simulator $\mathcal{S}$. The simulator runs $\mathcal{A}$ and creates algorithms to respond to queries made by $\mathcal{A}$ during its attack. To maintain consistency between the queries, $\mathcal{S}$ keeps a list L_{KDF} for query/response pairs to KDF, in which each item includes $((pid, sid, sk), sk')$, and another list $L_{\Pi^s_{i,j}}$ for query/response records to Send and Reveal, in which each item includes $(\Pi^s_{i,j}, pid_{ij}, sid_i, sk_i, sk'_i)$. We suppose that the function KDF is under the control of $\mathcal{S}$.

At the beginning of the game, $\mathcal{S}$ runs the Setup algorithm of Π by following the protocol specification to get the system parameters and the public and private keys for participants, and then gives the system parameters and public keys to $\mathcal{A}$. $\mathcal{S}$ answers the following queries, which are asked by $\mathcal{A}$ in an arbitrary order.

- $\mathsf{KDF}(pid, sid, sk)$: If $((pid, sid, sk), sk') \in L_{\mathsf{KDF}}$, for some sk', return sk'. Else compute $sk' = \mathsf{KDF}(pid, sid, sk)$; add $((pid, sid, sk), sk')$ to L_{KDF} and return sk'.
- $\mathsf{Send}(\Pi^s_{i,j}, x)$: $\mathcal{S}$ executes the protocol Π and responds with an outgoing message m by following the protocol specification. If the oracle $\Pi^s_{i,j}$ does not exist, it will be created. If the oracle has not accepted, the values of sk and sk' in $L_{\Pi^s_{i,j}}$ are $*$ (a specifical symbol). Otherwise, $\mathcal{S}$ computes $pid_{ij} = f(\Pi^s_{i,j})$; if the input message $x = \lambda$ (another specifical symbol) the oracle is an initiator, otherwise it is a responder; we assume that $i \neq j$. $\mathcal{S}$ then asks the $\mathsf{KDG}(pid_{ij},\ sid,\ sk)$ query to get sk'. $\mathcal{S}$ updates the list of $L_{\Pi^s_{i,j}}$.
- $\mathsf{Reveal}(\Pi^s_{i,j})$. If the oracle has not accepted, i.e. either no record in $L_{\Pi^s_{i,j}}$ or sk and sk' are both $*$, $\mathcal{S}$ returns $\perp$; otherwise, it returns sk'.
- $\mathsf{Corrupt}(i)$. $\mathcal{S}$ responds with U_i's long-term private key.
- $\mathsf{Test}(\Pi^{\tilde{s}}_{\tilde{i},\tilde{j}})$. $\mathcal{S}$ randomly chooses $b \in \{0, 1\}$ and responds with the session key $sk'_{\tilde{i}}$ of $\Pi^{\tilde{s}}_{\tilde{i},\tilde{j}}$ from the list $L_{\Pi^s_{i,j}}$, if $b = 0$, or a random sample from the distribution of the session key otherwise.

Once $\mathcal{A}$ finishes queries and returns its guess, $\mathcal{S}$ checks the list of L_{KDF}. Based on the definition of BUKS resilience in Definition 3 and the extended BR model in Section 5.1, if $\mathcal{A}$ wins the attack game, it should have made the tested oracle $\Pi^{\tilde{s}}_{\tilde{i},\tilde{j}}$ have a semi-partner oracle $\Pi^{\tilde{w}}_{\tilde{u},\tilde{v}}$, which holds $(pid_{\tilde{u}\tilde{v}}, sid_{\tilde{u}}, sk'_{\tilde{u}})$ satisfying $sk'_{\tilde{u}} = sk'_{\tilde{i}}$ and $pid_{\tilde{u}\tilde{v}} \neq pid_{\tilde{i}\tilde{j}}$. Since both the two sk' values are computed from KDF, which was controlled by $\mathcal{S}$, therefore, $\mathcal{S}$ should be able to find the following two entries of the list L_{KDF}: $((pid_{\tilde{i}\tilde{j}}, sid_{\tilde{i}}, sk_{\tilde{i}}), sk'_{\tilde{i}})$ and $((pid_{\tilde{u}\tilde{v}}, sid_{\tilde{u}}, sk_{\tilde{u}}), sk'_{\tilde{u}})$. Because $sk'_{\tilde{i}}$ and $sk'_{\tilde{u}}$ are identical and their inputs are different from each other, so $\mathcal{S}$ finds a collision of KDF. this result contradicts the assumption that KDF is a collision-resistent function. The theorem follows. $\square$

We can make use of the above general countermeasure to enhance the security of these protocols, which were analyzed in Section 3. 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

$$\mathsf{KDF}(k||U_i||U_j||g^{s_i}||\sigma_i||Cert_i||g^{s_j}||\sigma_j||Cert_j||g^{s_i s_j}),$$

supposing that U_i is the initiator and U_j is the responder.

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

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

again, supposing that U_i is the initiator and U_j is the responder. A similar change can be made in the alternative Oakley protocol of Section 3.3.1.

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 Hitchcock in [Choo et al. 2005-1] and by Cheng and Chen in [Cheng and Chen 2007] 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 Bellare-Rogaway-type proof of UUKS resilience and a BUKS adversary's behavior, and provided a modification of the BR model to cover the BUKS resilience attribute. We have also provided a general countermeasure for these vulnerable protocols, and proved that the countermeasure helps these protocols to hold the BUKS resilience property.

In [Canetti and Krawczyk 2001], Canetti and Krawczyk studied security of a key agreement protocol by including an Authenticated-link Model (AM) and a Unauthenticated-link Model (UM), where the adversary is passive in the AM while is active in the UM. They also presented a modular approach to transform a secure protocol in AM into a secure protocol in UM using a simulatibility-based approach. However, the security analysis in both the AM and the UM is defined in the same way as in the BR model (using an indistinguishability-based approach), though it is assumed that a session identifier is distributed to the target users before any protocol execution. Considering these facts, it is natural to expect that we can extend the security models of [Canetti and Krawczyk 2001] in a similar way as we have done in the BR model.

As we mentioned in section 4, the security model of Bellare, Canetti and Krawczyk in [Bellare et al. 1998] and the Shoup model in [Shoup 1998] adopt a simulatability-based approach to model the security of a key agreement protocol. Therefore, the extension technique proposed in this paper for the BR model cannot straightforwardly be applied here. Nonetheless, how to extend these models to cover the BUKS resilience remains a future research topic.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable comments, which are very helpful to improve the paper.

References

[Baek and Kim 2000] Baek, J., Kim, K.: “Remarks on the unknown key share attacks”; IEICE TRANSACTIONS E83-A, 12 (2000), 2766-2769.

[Baek et al. 2000] Baek, J., Kim, K., Matsumoto, T.: “On the significance of unknown key-share attacks: how to cope with them?”; Proc. Symposium on Cryptography and Information Security (2000).

[Bellare and Rogaway 1993] Bellare, M., Rogaway, P.: “Entity authentication and key distribution”; Advances in Cryptology – Crypto’93, Lect. Notes Comp. Sci. 773, Springer, Berlin, 110-125.

[Bellare and Rogaway 1995] Bellare, M., Rogaway, P.: “Provably secure session key distribution: the three party case”; Proc. 27^{th} ACM Symposium on Theory of Computing, ACM Press (1995), 57-66.

[Bellare et al. 1998] Bellare, M., Canetti, R., Krawczyk, H.: “A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract)”; Proc. 30^{th} ACM symposium on Theory of computing, ACM Press (1998), 419-428.

[Blake-Wilson et al. 1997] Blake-Wilson, S., Johnson, D., Menezes, A.: “Key agreement protocols and their security analysis’; Proc. 6^{th} IMA Conf. Cryptography and Coding, Lect. Notes Comp. Sci. 1355, Springer, Berlin (1997), 30-45.

[Blake-Wilson and Menezes 1997] Blake-Wilson, S., Menezes, A.: “Entity authentication and authenticated key transport protocols employing asymmetric techniques”; Proc. 5^{th} Int. Workshop Security Protocols, Lect. Notes Comp. Sci. 1361, Springer, Berlin (1997), 137-158.

[Blake-Wilson et al. 1999] Blake-Wilson, S., Menezes, A.: “Unknown key-share attacks on the station-to-atation (STS) protocol”; Proc. 2^{nd} Int. Workshop Practice and Theory in Public Key Cryptography, Lect. Notes Comp. Sci. 1560, Springer, Berlin (1999), 154-170.

[Boyd et al. 2006] Boyd, C., Choo, K., Mathuria, A.: “An extension to Bellare and Rogaway (1993) model: resetting compromised long-term keys”; Proc. 11^{th} Australasian Conf. Information Security and Privacy, Lect. Notes Comp. Sci. 4058, Springer, Berlin (2006), 371-382.

[Boyd et al. 2003] Boyd, C., J. González Nieto: “Round-optimal contributory conference key agreement”; Proc. 6^{th} Int. Workshop Theory and Practice in Public Key Cryptography, Lect. Notes Comp. Sci. 2567, Springer, Berlin (2002), 161-174.

[Boyd and Mathuria 2004] Boyd, C., Mathuria, A.: "Protocols for authentication and key establishment"; Springer-Verlag (2004).

[Bresson et al. 2001] Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: "Provably authenticated group Diffie-Hellman key exchange"; Proc. 8^{th} ACM Conf. Computer and Communications Security, ACM Press (2001), 255-264.

[Canetti and Krawczyk 2001] Canetti, R., Krawczyk, H.: "Analysis of key-exchange protocols and their use for building secure channels"; Advances in Cryptology – Eurocrypt 2001, Lect. Notes Comp. Sci. 2045, Springer, Berlin (2001), 453-474.

[Chen et al. 2006] Chen, L., Cheng, Z., Smart, N.: "Identity-based key agreement protocols from pairings"; International Journal of Information Security 6, 4, Springer, Berlin (2006) 213-241.

[Chen and Kudla 2003] Chen, L., Kudla, C.: "Identity based authenticated key agreement protocols from pairings"; Proc. 16^{th} IEEE Computer Security Foundations Workshop, IEEE Computer Society Press (2003), 219-233.

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

[Cheng et al. 2006] Cheng, Z., Chen, L., Comley, R., Tang, Q.: "Identity-based key agreement with unilateral identity privacy using pairings"; Proc. 2^{nd} Int. Conf. Information Security Practice and Experience, Lect. Notes Comp. Sci. 3903, Springer, Berlin (2006), 202-213.

[Choo et al. 2005-1] Choo, K., Boyd, C., Hitchcock Y.: "On session key construction in provably-secure key establishment protocols: revisiting Chen & Kudla (2003) and McCullagh & Barreto (2005) ID-based protocols"; Proc. *Mycrypt 2005*, Lect. Notes Comp. Sci. 3715, Springer, Berlin (2005), 116-131.

[Choo et al. 2005-2] Choo, K., Boyd, C., Hitchcock Y.: "Errors in computational complexity proofs for protocols"; Proc. *Asiacrypt 2005*, Lect. Notes Comp. Sci. 3788, Springer, Berlin (2005), 624-643.

[Diffie et al. 1992] Diffie, W., Oorschot, P., Wiener, M.: "Authentication and authenticated key exchanges"; Des. Codes Cryptography 2, 2 (1992), 107-125.

[Harn and Lin 2001] Harn, L., Lin, H.: "Authenticated key agreement without using oneway hash functions"; Electronics Letters 37, 10, IEE (2001), 1429-1431.

[Hirose et al. 1998] Hirose, S., Yoshida, S.: "An authenticated Diffie-Hellman key agreement protocol secure against active attacks"; Proc. 1^{st} Int. Workshop Practice and Theory in Public Key Cryptography, Lect. Notes Comp. Sci. 1431, Springer, Berlin (1998), 135-148.

[Kaliski 2001] Kaliski, B.: "An unknown key-share attack on the MQV key agreement protocol"; ACM Trans. Inf. Sys. Security 4, 3, ACM Press (2001), 275-288.

[Kudla and Paterson 2005] Kudla, C., Paterson, K.: "Modular security proofs for key agreement protocols"; Advances in Cryptology – ASIACRYPT 2005, Lect. Notes Comp. Sci. 3788, Springer, Berlin (2005), 549-565.

[Law et al. 2003] Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: "An efficient protocol for authenticated key agreement"; Des. Codes Cryptography 28, 2, Kluwer Academic Publishers (2003), 119-134.

[Matsumoto et al. 2000] Matsumoto, T., Takashima, Y., Imai, H.: "On seeking smart public-key-distribution systems"; IEICE TRANSACTIONS E69, 2, IEICE (2000), 99-106.

[Menezes et al. 1995] Menezes, A., Vanstone, S.: "Some new key agreement protocols providing mutual implicit authentication"; Proc. 2^{nd} Workshop Selected Areas in Cryptography (1995), 22-32.

[Mitchell et al. 1998] Mitchell, C., Ward, M., Wilson, P.: "On key control in key agreement protocols"; Electronics Letters 34 (1998), 980-981.

[Orman 1998] Orman, H.: "The OAKLEY key determination protocol"; The Internet Society RFC 2412 (1998).

[Shoup 1998] Shoup, V.: “On formal models for secure key exchange”; IBM Research Report RZ 3120 (1998).

[Zhou et al. 2003] Zhou, H., Fan, L., Li, J.: “Remarks on unknown key-share attack on authenticated multiple-key agreement protocol”; Electronics Letters 39, 17, IEE (2003), 1248-1249.