A New Way to Prevent UKS Attacks Using Hardware Security Chips

Qianying Zhang[1,2], Zhiping Shi[1,3]
(Corresponding author: Qianying Zhang)
College of Information Engineering[1]
Beijing Advanced Innovation Center for Imaging Technology[2]
Beijing Key Laboratory of Electronic System Reliability Technology[3]
Capital Normal University, Beijing 100048, China
(Email: qyzhang@cnu.edu.cn)
(Received Nov. 30, 2016; revised and accepted Feb. 13 & Mar. 11, 2017)

Abstract

UKS (unknown key-share) attacks are common attacks on AKE (Authenticated Key Exchange) protocols. We summarize two common countermeasures against UKS attacks on a kind of AKE protocols whose message flows are basic Diffie-Hellman exchanges. The first countermeasure forces the CA to check the possession of private key during registration, which is impractical for the CA. The second countermeasure adds identities in the derivation of the session key, which leads to modification of the protocols which might already be standardized and widely used in practice. By using protection of cryptographic keys provided by hardware security chips, such as TPM or TCM, we propose a new way that requires no check of possession of private key and no addition of identity during the derivation of the session key to prevent UKS attacks. We modify the CK model to adapt protocols using hardware security chip. We then implement a protocol once used in NSA, called KEA and subject to UKS attacks, using TCM chips. Our implementation, called tKEA, without forcing the CA to check during registration and modifying the original KEA, is proven to be secure. To show the generality of our way, we also show that it can prevent UKS attacks on the MQV protocol.

Keywords: Authenticated Key Exchange; CK Model; KEA; Trusted Cryptography Module; UKS Attacks

1 Introduction

The key exchange protocol, first proposed by Diffie and Hellman [11], allows two entities to establish a shared secret key via public communication. In order to authenticate identities of the two entities involved in the protocol, authenticated key exchange (AKE) is proposed. AKE not only allows two entities to compute a shared secret key but also ensures the authenticity of entities.

To date, a great number of AKE protocols have been proposed [1, 2, 3, 8, 12, 13, 14, 15, 19, 25, 27, 31] and many of them are subsequently broken, such as KEA [25] and MQV [22, 23]. KEA was designed by NSA (National Security Agency) in 1994 and kept secret until 1998. Microsoft researchers K.Lauter and A.Mityagin find that the original KEA protocol is susceptible to UKS attacks [21]. Then they present a modified version of KEA protocol, called KEA+ [21], which is resistant to UKS attacks, and give a formal proof. The MQV protocol is a famous and efficient AKE protocol designed by Law, Menezes, Qu, Solinas and Vanstone. This protocol was found to be susceptible to UKS attacks by Kaliski [17], then Krawczyk found that MQV hold none of its stated security goals, such as resistance to KCI attacks and the security property of perfect forward secrecy (PFS). To achieve the security goals of MQV, Krawczyk proposes a hashed variant of MQV, HMQV [19].

1.1 Related Work

In order to formally prove that AKE protocols are secure, Bellare and Rogaway in 1993 provided the first formal definition for an AKE model [4], which we refer to as the BR model. After that, a lot of variants of BR model were represented and many authenticated key exchange protocols were proposed. For more details, we refer the readers to [10] for a comparison and discussion of variants models for authenticated key exchange. Based on BR model, Canetti and Krwaczyk proposed the CK model [7], based on which the HMQV protocol was proved. LaMacchia, Lauter, and Mityagin defined a new model called eCK [20], which is much stronger than BR and CK models. They also introduced a new AKE protocol called NAXOS and proved its security in eCK model. However the NAXOS protocol is less efficient in that it requires 4 exponentiations per entity compared to 3 exponentiations for KEA.

Katz [18] first gives the idea of using secure hardware to achieve stronger security properties, and proves that tamper-proof hardware suffices to circumvent the impossibility result of secure computation of general functionalities without an honest majority. Recently, some works extend this idea to the improvement and security analysis of AKE protocols on modern AKE security models. [30] analyzes the SM2 key exchange protocol in TPM 2.0 security chip [29], and shows that protection provided by the TPM security chip indeed helps the protocol to resist two kinds of UKS attacks. [33] leverages the tamper-proof hardware to protect cryptographic keys and designs a set of APIs for HMQV, and formally proves that the HMQV protocol achieves full PFS property with the help of tamper-proof hardware in the CK model. [31] proposes an efficient key exchange protocol called sHMQV, which is a variant of HMQV. sHMQV eliminates the validation of public ephemeral key by protecting the ephemeral private key in trusted hardware devices, and enjoys the best efficiency in current one-round key exchange protocols. [32] models the protection provided by TPM 2.0 security chip as an oracle, and formally proves that under the protection of TPM 2.0 the key exchange primitive in TPM 2.0 is secure in modern AKE model.

1.2 UKS Attacks

A UKS attack on an AKE protocol is that an entity A ends up believing that he shares a key with an entity B, and although this is in fact the case, B mistakenly believes that the key is shared with an entity E $\neq$ A. Since the adversary E does not obtain the shared secret key, he cannot modify or decrypt the messages between A and B. However, E can take advantage of the entities' false assumptions about the identity who shares the key. Take a scenario described in [12] for example: B is a bank, and A sends him a digital coin, encrypted with the shared secret key, for deposit into her account. Believing that the key is shared with E, B assumes the coin is from E and deposits it into E's account instead. Several UKS attacks have been proposed in the literature, such as attacks on STS [5], KEA [21], and MQV [17].

1.3 Contributions

We give our contributions as follows:

1) We summarize UKS attacks on AKE protocols and existing countermeasures in the literature, and identify two kinds of attacks, called *public key substitution UKS attack* and *public key registration UKS attack* respectively. The details of the two attack are described in Section 2. The usual way to resist the two kinds of UKS attacks are: 1) force the CA to check the possession of the private key, 2) add the identity during the derivation of the session key. We illustrate these countermeasures by overview of existing works on preventing UKS attacks on KEA and MQV.

2) We present a new way to prevent the two kinds of UKS attacks using the protection capability of hardware security chip, such as Trusted Platform Module (TPM) [29] and Trusted Cryptography Module (TCM) [26]. The key idea is to make use of the security chip to generate the long-term secret key, and register it to a CA who does not check the possession of the private key and only makes sure that the key comes from a real hardware security chip. The protection capability of the security chip prevents the adversary from getting the plaintext of the private key even he corrupts and controls the security chip. In our security analysis we will show that the protection capability is crucial for the KEA protocol to resist UKS attacks. Our new way of preventing UKS attacks has advantages of not requiring the CA to check the possession of the private key nor modifying the original protocol. The former advantage makes the protocol can be deployed in practical CAs who usually do not check the possession of the private key. The later advantage improves the security of such protocols that have already been standardized and deployed in many fields. Upgrading the standards might require quite a long time, and in some fields system upgrades are rigorously controlled, such as the industrial control field. To show the generality of our way to resist UKS attacks, we also demonstrate that our proposed way can prevent UKS attacks on MQV protocol.

3) We give a variant of CK model to adapt protocols implemented by hardware security chip. Then we implement the KEA protocol (subject to UKS attacks) using TCM chips and prove that our implementation prevents UKS attacks. We make a comparison among typical protocols with the ability of resisting UKS attacks in terms of key registration (whether adversary-controlled entities can register arbitrary public keys), modification (whether the original protocol is modified in order to be proven secure formally), efficiency (whether extra computation is added), security properties and assumptions in Table 1. The Modif column shows that both KEA+ and HMQV modify the original protocols (KEA and MQV respectively) in order to be proven secure formally. The Effic column shows that the HMQV adds 25% extra computation to MQV while tKEA adds no extra computation to KEA. Compared to KEA+ and HMQV, tKEA obtains same security properties while making no modification of the original protocol and adding no extra computation.

1.4 Organization

We summarize the two kinds of UKS attacks and corresponding countermeasures in Section 2. In Section 3, we give a detailed description of protection of cryptographic keys provided by one kind of hardware security

Table 1: Comparison of HMQV, KEA+ and tKEA

	Key Reg.	Modif.	Effic.	Security	Assumptions
tKEA	Arbitrary	No	No	CK, KCI, wPFS	GDP+RO
KEA+	Arbitrary	Yes	No	CK, KCI, wPFS	GDP+RO
HMQV	Arbitrary	Yes	25%	CK, KCI, wPFS	GDP+KEA1+RO

chip, TCM, show that how it can be used on AKE protocols, and give our implementation, which we call tKEA (the 't' stands for trusted). Section 4 describes the security model on which the formal security analysis of tKEA is based. Section 5 proves the security of tKEA. We also show how the protection provided by TCM prevents the UKS attack on MQV protocol described in [19] in this section. We end the paper with concluding and our future work in Section 6.

2 UKS Attacks and Their Countermeasures

AKE protocols can be categorized as the explicitly authenticated or the implicitly authenticated by the way they are authenticated. Both of the two kinds of AKE protocols are vulnerable to UKS attacks. Baek and Kim have summarized UKS attacks on the explicitly authenticated key exchange protocol [21]. In this paper we give an overview of UKS attacks on the implicitly authenticated key exchange protocol.

In this section, we first introduce the explicitly and implicitly authenticated key exchange protocols, and then summarize two kinds of UKS attacks on the implicitly authenticated key exchange protocol and the usual countermeasures to prevent these two kinds of attacks.

2.1 Explicitly Authenticated Key Exchange Protocol

The explicitly authenticated key exchange protocol is such a kind of protocol that first executes a basic Diffie-Hellman key exchange and then uses digital signatures or additional authenticating message flows to provide authentication explicitly. ISO-DH [16], STS [12], SIG-DH [28], SIGMA [8] are such typical protocols. In the following, we take the ISO-DH protocol as an example to illustrate such kind of protocol.

Let G be a group of primer order and denote by g a generator of G. Assume that entities have secret/public keys for some digital signature scheme SIG and that entities know each other's registered public keys. The hat notation, such as $\hat{A}$, denotes the identities of entities in the protocol. Denote the signature of a message $\mathcal{M}$ under the secrete key of an entity $\hat{A}$ by $SIG_{\hat{A}}(\mathcal{M})$. We depict the protocol in Figure 1. First, an entity $\hat{A}$ as an initiator randomly generates an ephemeral private key x and sends a tuple $\{g^x, SIG_{\hat{A}}(g^x, \hat{B})\}$ to $\hat{B}$, the responder. The responder $\hat{B}$ generates an ephemeral private key y and replies with a tuple $\{g^y, SIG_{\hat{B}}(g^y, g^x\hat{A})\}$. Both $\hat{A}$ and $\hat{B}$ then verify each other's signatures, and compute a shared session key $K = g^{xy}$ if the verification successes.

2.2 Implicitly Authenticated Key Exchange Protocol

The implicitly authenticated key exchange protocol only needs basic Diffie-Hellman exchanges, and provides authentication by combining ephemeral keys and long-term keys during the derivation of the session key. KEA and MQV are typical protocols of this kind of AKE. Figure 2 gives an illustration of KEA and its variant KEA+. KEA involves two entities, $\hat{A}$ and $\hat{B}$, with respective secret keys a and b and public keys g^a and g^b. KEA assumes that entities know each other's registered public keys. The protocol first runs a Diffie-Hellman key exchange: $\hat{A}$ and $\hat{B}$ each generates its ephemeral private key, x and y respectively, and exchanges the ephemeral public keys g^x and g^y. Then each entity computes g^{ay} and g^{bx} and computes a session key by applying a hash function H to (g^{ay}, g^{bx}). The KEA+ protocol differs from KEA when computing the session key, it applies the hash function to a tuple $(g^{ay}, g^{bx}, \hat{A}, \hat{B})$, adding the identities to the tuple of KEA.

2.3 UKS Attacks on Implicitly Authenticated Key Exchange Protocol

As Baek and Kim have given a conclusion of UKS attacks on the explicitly authenticated key exchange protocol [21], here we only summarize UKS attacks on the implicitly key exchange protocol. We categorize these attacks as *public key substitution UKS attack* and *public key registration UKS attack*. We also summarize existing countermeasures on the two kinds of UKS attacks.

2.3.1 Public Key Substitution UKS Attack

This kind of attack happens to some protocols when the CA does not check the possession of the private key. In the following we illustrate this attack on the KEA protocol. Consider two entities $\hat{A}$ and $\hat{B}$ preparing to start a session. An adversary $\mathcal{M}$ registers a public key g^a of $\hat{A}$ as his own public key. Then $\mathcal{M}$ intercepts the session between $\hat{A}$ and $\hat{B}$. $\mathcal{M}$ forwards the ephemeral public key g^x from $\hat{A}$ to $\hat{B}$ and ephemeral public key g^y from $\hat{B}$ to $\hat{A}$. Since $\mathcal{M}$ has the same public key as $\hat{A}$, both $\hat{A}$ and $\hat{B}$ will compute identical session keys. However, $\hat{A}$ completes a session with $\hat{B}$ and $\hat{B}$ completes a session with $\mathcal{M}$.

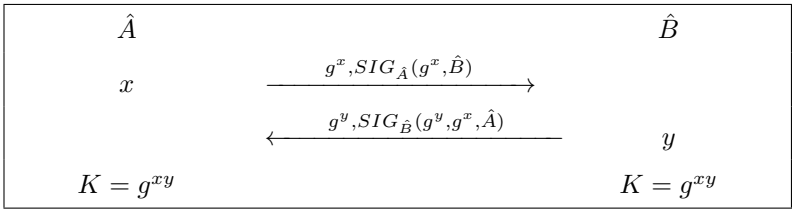

Figure 1: Explicitly authenticated key exchange protocol: ISO-DH

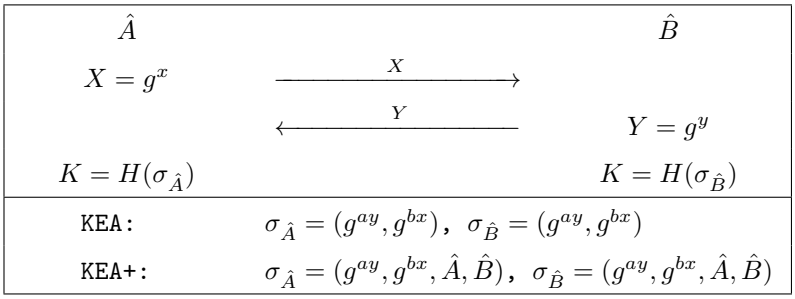

Figure 2: Implicitly authenticated key exchange protocol: KEA and KEA+

The usual way to solve this kind of UKS attack is to force the CA to check the possession of the private key. If the CA does, $\mathcal{M}$ cannot register the public key of $\hat{A}$, then $\hat{A}$ and $\hat{B}$ will compute non-identical session keys. However, as the proof of knowledge check are rarely done by CA in practice, this way to prevent UKS attacks are impractical.

2.3.2 Public Key Registration UKS Attack

The typical attack example is a UKS attack on MQV found by Kaliski [17]. Let me introduce MQV first. MQV is a famous implicitly authenticated key exchange protocol, which was stated to have a lot of security properties, such as resistance to UKS attacks and KCI attacks. We depict MQV and HMQV in Figure 3. Entities $\hat{A}$ and $\hat{B}$ have their private/public key pairs (a, g^a) and (b, g^b) respectively. The ephemeral public keys in their exchange messages are g^x and g^y. The computation of the session key by $\hat{A}$ ($\hat{B}$) is a hash value to $(YB^e)^{x+da}$ $((XA^d)^{y+be})$. The only difference between MQV and HMQV is the computation of d and e. The former only uses the ephemeral public key, while the later adds the identity information and uses a hash function in the computation. However, we will show below that this slight modification is crucial for the security of HMQV.

We describe the *public key registration UKS attack* on MQV in Figure 4. An adversary $\mathcal{M}$ intercepts the ephemeral public key $X = g^x$ sent from $\hat{A}$ to $\hat{B}$. Based on X, $\mathcal{M}$ computes a private/public key pair (c, g^c), and sends an ephemeral public key Z. After receiving Z, $\hat{B}$ generates a random ephemeral key $Y = g^y$ and sends it to $\mathcal{M}$. $\mathcal{M}$ transmits Y to $\hat{A}$. We denote the session between $\hat{A}$ to $\hat{B}$ by s, and the session between $\hat{B}$ to $\mathcal{M}$ by s'. We can see that the key pair (c, g^c) and the ephemeral key Z are computed so cleverly that s and s' have the identical shared secret key.

From the attack described above, we can see that check proof of knowledge of private key cannot prevent this attack as the the adversary holds the private key c that he registers. The usual way to prevent this kind of attack is to add the identities in the derivation of the session key. Krawczyk and Menezes respectively present HMQV and a variant of MQV [24] which both resist this kind of UKS attack. HMQV adds the identity and uses a hash function when computing d and e, while [24] adds the identities in the derivation of the session key. From their solutions we can see that adding identities in the derivation of the session key is an effective way to prevent the *public key registration UKS attack*. Although it might be easy to modify the protocol to achieve a higher security level, for protocols that have been standardized it might take a long time for them to be upgraded. So we need to consider how to protect systems adopting non-secure protocols while upgrades of protocols are still unavailable. And for some fields, such as industrial control field, upgrades are rigorously controlled as the system deals with very crucial tasks involving electricity and other infrastructures and any modification must be tested rigorously. So research on improving the security of AKE protocols without modifying the original protocol is meaningful.

3 Protection Provided by TCM

Trusted Cryptography Module (TCM), a hardware security chip similar to Trusted Platform Module (TPM), is a small tamper-resistant cryptographic chip embedded in computer platforms (e.g. on a PC motherboard). TCM provides a set of cryptography capabilities that allow some cryptographic functions to be executed in TCM, such as public-key decryption/encryption (SM2-1), hash (SM3), random number generating, key exchange protocol (SM2-2) and so on. TCM stores the secret data, such

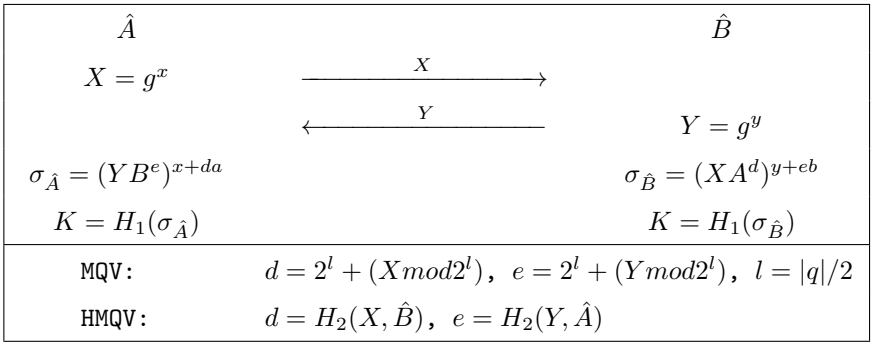

Figure 3: The MQV and HMQV protocols

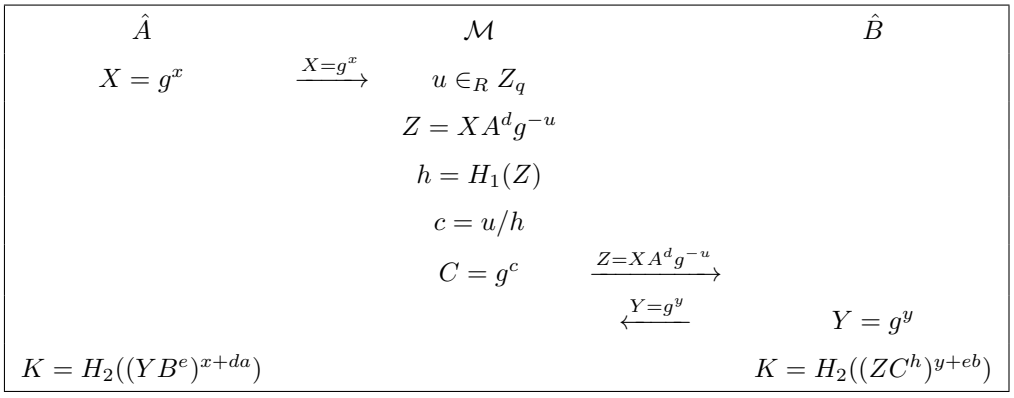

Figure 4: A UKS attack on MQV

as keys and crucial user data, in a shielded location where data is protected against interference and prying.

To operate the secret data in the shielded location, TCM provides a set of cryptographic APIs for users. Take the SM2-2 key exchange for example, TCM provides *TCM_CreateKeyExchange* and *TCM_GetKeyExchange* to generate a private/public key pair and generate a session key respectively:

- *TCM_CreateKeyExchange*: TCM generates a private/public SM2 key pair, which we denote by (a, g^a), in the TCM's shield location, and returns the public part of the SM2 key pair.

- *TCM_GetKeyExchange*: Input a public key of SM2, e.g, g^b, and return a session key g^{ab}.

TCM provides protection for cryptographic keys in the following two aspects. First, a user who controls an SM2-2 key pair generated by TCM cannot get plaintext of the private key, and the only way he can use the SM2-2 key is through TCM APIs. Second, as the key is randomly generated by TCM and the user has no control of the generation of a specific keys, a user cannot make TCM chips generate a specified key pair. The second protection feature constrains the adversary $\mathcal{M}$ from using TCM to register a specified key.

3.1 Implementation of tKEA

Here we show how to implement KEA protocol using TCM. Our implementation consists of two phases: registration phase and key exchange phase.

The registration phase involves a security TCM chip $\mathcal{T}$, a Host $\mathcal{H}$, and a CA $\mathcal{C}$. $\mathcal{T}$ and its host $\mathcal{H}$ compose a whole entity. Before the registration phase, $\mathcal{T}$ generates an attestation identity key (AIK) pair (sk_T, pk_T) (AIK is used to identify the platform in trusted computing, here we use it to certify the long-term key of an entity) and then registers the public key pk_T to a CA (note that this CA issues certificates to platforms, and is not the CA in the registration phase, which issues certificates to long-term keys) through protocols such as Privacy-CA [9], which is out of the scope of this paper. If higher anonymity is required, please refer to DAA [6] solution. After getting the AIK certificate, the registration proceeds as follows:

1) $\mathcal{H}$ calls *TCM_CreateKeyExchange* command of $\mathcal{T}$, and $\mathcal{T}$ generates an SM2-2 key pair (a, g^a) representing the long-term key of this entity.

2) $\mathcal{H}$ then calls *TCM_CerifyKey* command of $\mathcal{T}$, and $\mathcal{T}$ makes a **statement** about (a, g^a) using the AIK: "this key is held in a TCM-shielded location, and it will never be revealed", and returns the **statement** to $\mathcal{H}$. The **statement** is actually a signature of the SM2-2 key by AIK. The AIK has a feature that it only signs the key generated within the TCM. This

feature assures the CA that the SM2-2 key is a real TCM-generated key if it has a legal signature.

3) $\mathcal{H}$ transmits the **statement** to $\mathcal{C}$. $\mathcal{C}$ verifies the **statement** to make sure that the public key g^a is generated by a real TCM chip. If the verification passes, $\mathcal{C}$ issues a Cert about g^a and gives it to $\mathcal{H}$.

The key exchange phase is shown in Figure 5, and actually is the procedure of running the KEA protocol between two entities, e.g., $\hat{A}$ and $\hat{B}$. $\hat{A}$ consists of a TCM $\mathcal{T}_1$ and its host $\mathcal{H}_1$, and $\hat{B}$ consists of a TCM $\mathcal{T}_2$ and its host $\mathcal{H}_2$. $\hat{A}$'s long-term public key is $A = g^a$, and $\hat{B}$'s long-term public key is $B = g^b$.

4 Security Model for tKEA

In this section we introduce a variant of CK model on which the security analysis of tKEA is based. For further details of CK model, please consult [7] for complete details. We modify the CK model by 1) modifying the *corruption(entity)* in the CK model, 2) adding an *establish(entity)* query to the queries of an adversary in the AKE experiment. The modified *corruption(entity)* query can simulate the protection of cryptographic keys provided by TCM, and the *establish(entity)* query allows an adversary to register public keys of adversary-controlled entities at any time in the experiment, that is, the adversary is allowed to mount the UKS attack.

4.1 Sessions

tKEA runs in a network of interconnected entities where each entity can be activated to run an instance of the protocol called a session. Within a session an entity can be activated to initiate the session or to respond to an incoming message. As a result of these activations, the entity creates and maintains a session state, generates outgoing messages, and eventually completes the session by outputting a session key and erasing the session state. There are two roles during a session, the entity that sends the first message in a session is called the **initiator** and the other the **responder**. We let $\mathcal{I}$ denote initiator and $\mathcal{R}$ denote responder. We identify an AKE session by a 5-tuple$(role, \hat{A}, \hat{B}, X, Y)$ where $role$ denotes the role, X is the outgoing DH value and Y is the incoming DH value to the session. The session $(\mathcal{R}, \hat{B}, \hat{A}, Y, X)$ (if it exists) is said to be **matching** to session $(\mathcal{I}, \hat{A}, \hat{B}, X, Y)$.

4.2 Adversary

The AKE experiment involves multiple honest entities and an adversary $\mathcal{M}$ connected via an unauthenticated network. The adversary $\mathcal{M}$ is modeled as a probabilistic Turing machine and controls all communications. $\mathcal{M}$ can intercept and modify messages sent over the network. $\mathcal{M}$ also schedules all session activations and session-message delivery. In addition, in order to model potential disclosure of secret information, the adversary is allowed to access secret information via the following queries:

- *session-state(s)*: $\mathcal{M}$ queries directly at session s which is still incomplete and learns the session state for s. The session state may include, for example, the secret exponent of an ephemeral DH value but not the long-term secret key.
- *session-key(s)*: $\mathcal{M}$ obtains the session key for a session s, provided that the session holds a key.
- *corruption(entity)*: For the information not stored in the TCM's shield location, such as the session states and session keys, $\mathcal{M}$ learns all of them. For the long-term key stored in the TCM's shield location, $\mathcal{M}$ has the ability to use it, such as computing $CDH(A, X)$ (A stands for the long-term public key, X stands for an element in G whose exponent is unknown) but cannot get the plaintext of the private key.
- *establish(entity)*: This query allows $\mathcal{M}$ to register a public key generated in TCM, and $\mathcal{M}$ has the ability to use the private key of the registered key. If $\mathcal{M}$ registers a public key not generated in TCM, the CA will deny this registration after checking the AIK signature of the public key. $\mathcal{M}$ can use this query to control an entity.

The adversary can make queries above to gain local information. We say that a completed session is "clean" if this session as well as its matching session (if it exists) is not subject to any of session-state, session-key, corruption queries.

Eventually $\mathcal{M}$ should select a clean completed session, which is called a test session, and make query **Test**(s) and is given a challenge value C.

- *Test(s)*: Pick $b \xleftarrow{R} 0, 1$. If $b = 1$, provide $\mathcal{M}$ with $C \leftarrow$ *sesssion-key(s)*; otherwise provide $\mathcal{M}$ with C, which is a value r randomly chosen from the probability distribution of session keys.

Now $\mathcal{M}$ can continue to make session-state, session-key, corruption and establish queries but is not allowed to expose the test nor any of the entities involved in the test session. At the end of its run, $\mathcal{M}$ outputs a bit b'. We will refer to an adversary that is allowed the Test query as a **KE-adversary**.

Definition 1. *An AKE protocol* Π *is called SK-secure if the following properties hold for any KE-adversary* $\mathcal{M}$ *defined above:*

1) when two uncorrupted entities complete matching sessions, they output the same key, and

2) the probability that $\mathcal{M}$ *correctly guesses the bit* b *(i.e., outputs* $b' = b$*) from the Test query is no more than 1/2 plus a negligible fraction.*

[1] X and Y are transmitted to $\mathcal{T}$ by *TCM_GetKeyExchange*.

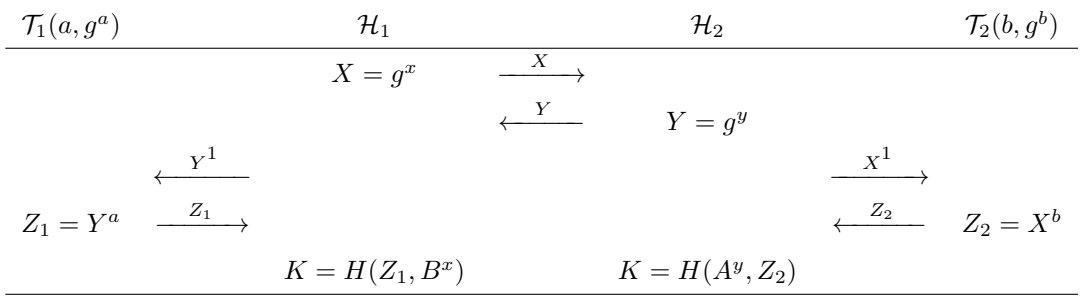

Figure 5: Implementation of KEA: tKEA

The advantage of **KE-adversary** participating in above AKE experiment against a protocol Π is defined as

$$\mathbf{Adv}_{\Pi}^{AKE}(\mathcal{M}) = \Pr[\mathcal{M} \text{ wins}] - \tfrac{1}{2}.$$

5 Security of tKEA and MQV

5.1 Security Proof of tKEA

Under the GDH assumption in a group G and the protection provided by TCM chips, with the hash function $H()$ modeled as a random oracle, we show that tKEA satisfies AKE security against a **KE-adversary** defined in Section 4. The GDH assumption is that the CDH problem in G cannot be solved in polynomial time with non-negligible success probability even when a DDH oracle for G is available.

Let $\mathcal{M}$ be any AKE adversary against tKEA. We start by observing that since the session key of the test session is computed as $K = H(\sigma)$ for some 2-tuple σ, the adversary $\mathcal{M}$ has only two ways to distinguish K from a random value:

1) Forging attack. At some point $\mathcal{M}$ queries H on the same 2-tuple σ as that of the test session.

2) Key-replication attack. $\mathcal{M}$ succeeds in forcing the establishment of another session that has the same session key as the test session.

Let us first show that the key-replication attack is impossible if random oracles produce no collisions. If $\mathcal{M}$ finds some session with the same 4-tuples as that of the test session, then this session must be executed by the same two entities, A and B. Let the ephemeral public keys of this session be X' and Y'. Since the session has the same signature as the test session, $CDH(A, Y')$ must be equal to $CDH(A, Y)$ and $CDH(B, X')$ - equal to $CDH(B, X)$. This implies that $X = X'$ and $Y = Y'$, and thus the session must be identical to the test session, which conflicts with the fact that the session is different from the test session.

However, the key-replication attack can happen to KEA. Lauter and Mityagin describe this attack in [21]. We here review this attack. An adversary $\mathcal{M}$ registers a public key g^a of some honest entity $\hat{A}$ as $\mathcal{M}$'s own public key. Then $\mathcal{M}$ intercepts a key-exchange session between $\hat{A}$ and $\hat{B}$, and at the same time starts a session between $\mathcal{M}$ and $\hat{B}$. $\mathcal{M}$ forwards ephemeral public key g^x from $\hat{A}$ to $\hat{B}$ and ephemeral public key g^y from $\hat{B}$ to $\hat{A}$. Since $\mathcal{M}$ has the same public key as $\hat{A}$, both $\hat{A}$ and $\hat{B}$ will complete identical session keys, however they participate in two different sessions. $\hat{B}$ participates in a session with $\mathcal{M}$ while $\hat{A}$ participates in a session with $\hat{B}$. Then $\mathcal{M}$ can announces one of the two sessions as a test session and reveals the session key of the other session. To avoid UKS attacks, KEA+ adds the identities of the participating entities to the tuples, see Figure 2. This slight modification prevents adversaries to activate a session with the same tuple, thereby preventing $\mathcal{M}$ from performing a key-replication attack. We show below that the protection provided by TCM can also prevent UKS attacks.

In the tKEA, we demonstrate that if an adversary $\mathcal{M}$ plays a key-replication attack, he can break the protection provided by TCM. We denote the test session by s and the corresponding 2-tuple by $(CDH(A, Y), CDH(B, X))$. Correspondingly, we denote another session by s' which has the same session key with s, and the corresponding 2-tuple on which $\mathcal{M}$ queries H to get the session key of s by $(CDH(A', Y'), CDH(B', X'))$. A' and B' are public keys $\mathcal{M}$ registers to the CA through the *establish(entity)*, and $\mathcal{M}$ can do the computation of $CDH(A' or B', T)$ for any T whose exponent is unknown. Since s and s' has the same session key, $CDH(A', Y')$ must be equal to $Z_1 = CDH(A, Y)$ and $CDH(B', X')$ must be equal to $Z_2 = CDH(B, X)$. Since $CDH(A', Y') = Z_1$, we can get $Y' = Z_1^{\frac{1}{a'}}$ and $A' = Z_1^{\frac{1}{y'}}$. The only two ways for $\mathcal{M}$ to get a pair (A', Y') meeting equation $CDH(A', Y') = Z_1$ are:

1) Register a controlled key A' to the CA, and compute the ephemeral public key $Y' = Z_1^{\frac{1}{a'}}$ where a' denotes the private key of A'.

2) Generate an ephemeral key pair $(y', Y' = g^{y'})$, and register $A' = Z_1^{\frac{1}{y'}}$ to the CA.

We can see that the first way requires $\mathcal{M}$ to get the plaittext of the public key A', and the second way requires $\mathcal{M}$ to register a specified key. However, both of the two ways violate the protection provided by TCM which is described in Section 3.

We are left to show the impossibility of a forging attack. The proof of tKEA is similar to KEA+ [21]. It can be directly obtained by placing the 4-tuple of KEA+ on which is used to query H with tKEA's 2-tuple. So we omit the proof.

To summarize the proof, for any AKE adversary $\mathcal{M}$ running in time t we can construct a GDH solver $\mathcal{S}$ which runs in time $O(t^2)$ such that

$$\mathbf{Adv}^{GDH}(\mathcal{S}) \geq \frac{1}{nk}\mathbf{Adv}^{AKE}_{tKEA}(\mathcal{M})$$

As for the wPFS and KCI security property of tKEA, they can be proved directly following the proof in [21].

5.2 Securing MQV

To prove the generality of our way, we show that our way of using the protection capability provided by TCM/TPM to prevent UKS attacks can prevent the UKS attack [17] on MQV protocol. Figure 4 shows this attack. To attack MQV, the adversary $\mathcal{M}$ registers an public key $C = g^c$ to the CA. As $\mathcal{M}$ knows the private key of C, the CA cannot deny the registration of C even it require proof of knowledge of the private key. However, if the CA requires that the key must come from a security chip, such as TPM or TCM, this UKS attack can be prevented. That's because if the key is generated in a security chip, $\mathcal{M}$ cannot generate a key whose private key is specified to be c. That's to say, $\mathcal{M}$ cannot register $C = g^c$ to the CA.

6 Conclusion and Future Work

This paper summarizes two kinds of UKS attacks on the implicitly authenticated key exchange protocol and corresponding countermeasures to the two kinds of attacks. One of the countermeasure requires the CA to check the possession of the private key, which is unpractical, and the other countermeasure is to add the identity during the derivation of the session key, which modifies the original protocol. Motivated by the protection capability provided by security chips, we present a new way of preventing UKS attacks on AKE protocol.

We introduce the protection capability provided by hardware security chips and give a variant of CK model which covers UKS attacks. Through the security proof of tKEA in our variant model, we show that our new way of preventing UKS attacks is effective and have some advantages compared to existing countermeasures. We also show the generality of our new way by preventing the UKS attack on MQV protocol.

In Section 5, we show that our new way can prevent the UKS attack on MQV without a formal proof. In the future, we hope to implement a 'tMQV' using a hardware security chip like TCM, and give it a formal proof. We also hope to check whether the protection capability of hardware security chips can provide other advantages to AKE protocols.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61602325, 61170304, 61472468, 61572331), the International Cooperation Program on Science and Technology (2011DFG13000), the Project of Beijing Municipal Science & Technology Commission (Z141100002014001), the Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges Under Beijing Municipality (No.IDHT20150507), and the Scientific Research Base Development Program of the Beijing Municipal Commission of Education (TJSHG201310028014).

References

[1] M. Abdalla, F. Benhamouda, and P. Mackenzie, "Security of the j-pake password-authenticated key exchange protocol," in *Security and Privacy*, pp. 571–587, 2015.

[2] R. Amin and G. P. Biswas, "Cryptanalysis and design of a three-party authenticated key exchange protocol using smart card," *Arabian Journal for Science and Engineering*, vol. 40, no. 11, pp. 3135–3149, 2015.

[3] R. Amin, S. K. H. Islam, G. P. Biswas, M. K. Khan, L. Lu, and N. Kumar, "Design of anonymity preserving three-factor authenticated key exchange protocol for wireless sensor network," *Computer Networks*, vol. 101, pp. 42–62, 2016.

[4] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Annual International Cryptology Conference*, pp. 232–249, 1993.

[5] S. Blake-Wilson and A. Menezes, "Unknown key-share attacks on the station-to-station (STS) protocol," in *International Workshop on Public Key Cryptography*, pp. 154–170, 1999.

[6] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proceedings of the 11th ACM conference on Computer and Communications Security*, pp. 132–145, 2004.

[7] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 453–474, 2001.

[8] R. Canetti and H. Krawczyk, "Security analysis of ikes signature-based key-exchange protocol," in *Annual International Cryptology Conference*, pp. 143–161, 2002.

[9] L. Chen and B. Warinschi, "Security of the tcg privacy-ca solution," in *IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC'10)*, pp. 609–616, 2010.

[10] K.-K. R. Choo, C. Boyd, and Y. Hitchcock, "Examining indistinguishability-based proof models for

key establishment protocols," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 585–604, 2005.
[11] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.
[12] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.
[13] M. S. Farash, S. H. Islam, and M. S. Obaidat, "A provably secure and efficient two-party password-based explicit authenticated key exchange protocol resistance to password guessing attacks," *Concurrency & Computation Practice & Experience*, vol. 27, no. 17, pp. 4897–4913, 2015.
[14] A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama, "Strongly secure authenticated key exchange from factoring, codes, and lattices," in *International Conference on Practice and Theory in Public Key Cryptography*, pp. 467–484, 2015.
[15] S. H. Islam, "Design and analysis of a three party password-based authenticated key exchange protocol using extended chaotic maps," *Information Sciences*, vol. 312(C), pp. 104–130, 2015.
[16] ISO/IEC, *Entity Authentication Mechanisms - Part 3: Entity Authentication Using Asymmetric Techniques*, ISO/IEC IS 9798-3, 1993.
[17] B. S. Kaliski Jr, "An unknown key-share attack on the mqv key agreement protocol," *ACM Transactions on Information and System Security*, vol. 4, no. 3, pp. 275–288, 2001.
[18] J. Katz, "Universally composable multi-party computation using tamper-proof hardware," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 115–128, 2007.
[19] H. Krawczyk, "HMQV: A high-performance secure diffie-hellman protocol," in *Annual International Cryptology Conference*, pp. 546–566, 2005.
[20] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," in *International Conference on Provable Security*, pp. 1–16, 2007.
[21] K. Lauter and A. Mityagin, "Security analysis of kea authenticated key exchange protocol," in *International Workshop on Public Key Cryptography*, pp. 378–394, 2006.
[22] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An efficient protocol for authenticated key agreement," *Designs, Codes and Cryptography*, vol. 28, no. 2, pp. 119–134, 2003.
[23] A. Menezes, M. Qu, and S. Vanstone, "Some new key agreement protocols providing mutual implicit authentication," in *Second Workshop on Selected Areas in Cryptography (SAC'95)*, 1995.
[24] A. Menezes and B. Ustaoglu, "On the importance of public-key validation in the mqv and hmqv key agreement protocols," in *International Conference on Cryptology in India*, pp. 133–147, 2006.
[25] National Institute of Standards and Technology, *Skipjack and KEA Algorithm Specifications, Ver. 2.0*, May 29, 1998.
[26] Official of State Commercial Cryptography Administration, *Functionality and Interface Specification of Cryptographic Support Platform for Trusted Computing*, 2007.
[27] R. Pecori and L. Veltri, "3AKEP: Triple-authenticated key exchange protocol for peer-to-peer voip applications," *Computer Communications*, vol. 85, pp. 28–40, 2016.
[28] V. Shoup, *On Formal Models for Secure Key Exchange*, Technical Report RZ 3120 (#93166), IBM, Apr. 19, 1999 .
[29] Trusted Computing Group, *Trusted Platform Module Library Part 3: Architecture Family 2.0*, Jan. 7, 2014.
[30] S. Zhao, L. Xi, Q. Zhang, Y. Qin, and D. Feng, "Security analysis of sm2 key exchange protocol in TPM2.0," *Security and Communication Networks*, vol. 8, no. 3, pp. 383–395, 2015.
[31] S. Zhao and Q. Zhang, "SHMQV: An efficient key exchange protocol for power-limited devices," in *Information Security Practice and Experience*, pp. 154–167, 2015.
[32] S. Zhao and Q. Zhang, "A unified security analysis of two-phase key exchange protocols in TPM 2.0," in *International Conference on Trust and Trustworthy Computing*, pp. 40–57, 2015.
[33] Q. Zhang, S. Zhao, Y. Qin, and D. Feng, "Improving the security of the hmqv protocol using tamper-proof hardware," in *International Conference on Security and Privacy in Communication Systems*, pp. 343–361, 2014.

Biography

Qianying Zhang received her Ph.D degree from Institute of Software, Chinese Academy of Sciences in 2015. She is currently a lecturer in Capital Normal University. Her research interests include information security, operating system security, and formal verification.

Zhiping Shi received his Ph.D degree from Institute of Computing Technology, Chinese Academy of Sciences in 2005. He is currently an associate researcher in Capital Normal University. His research interests include formal verification, and artificial intelligence.