TMQV: A Strongly eCK-secure Diffie-Hellman Protocol without Gap Assumption*

Jiaxin Pan and Libin Wang†
School of Computer, South China Normal University
Guangzhou 510631, China
csplator@gmail.com, lbwang@scnu.edu.cn

Abstract

In this paper, we propose an authenticated key exchange (AKE) protocol under the computational Diffie-Hellman (CDH) assumption with respect to the strengthened eCK-security (seCK-security) of Sarr *et al.*. To date, many AKE protocols either are provably secure under a rather strong and non-standard assumption named as the gap Diffie-Hellman (GDH) assumption, or fall to practical attacks on the intermediate result leakage which can be captured by the seCK model. In order to remove the gap assumption and to achieve stronger security requirements, we present the TMQV protocol using the twinning technique and the MQV key derivation method. With the help of trapdoor test theorem, TMQV is provably seCK-secure under the standard CDH assumption in the random oracle model. Compared with the related works, TMQV achieves not only stronger security but also higher implementation efficiency with weaker cryptographic assumptions.

Keywords: Strengthened eCK model, authenticated key exchange, Diffie-Hellman assumption, trapdoor test

1 Introduction

Motivation. Authenticated key exchange (AKE) protocols are cryptographic protocols by which two parties that communicate over a public network can generate a common session key. After the invention of the (extended) Canetti-Krawczyk (CK and eCK) models [4, 12] for AKE security, many authenticated Diffie-Hellman (DH) protocols have been formally proven secure. The security proofs for these protocols (e.g. HMQV [10], CMQV [21] and NAXOS [12]) requires an external decisional DH (DDH) oracle to verify the DH value, and then the gap assumption [16] seems to be essential for them. Moreover, many existing AKE protocols are shown to be insecure against the intermediate result leakage [6, 22, 19, 20]. Thus, it is interesting to design an authenticated DH key exchange protocol admits stronger security but without the gap assumption.

Trapdoor test method of Cash *et al.* [5] allows us to implement an effective decision oracle for the twin DH (TDH) problem without knowing any of the corresponding discrete logarithms. The TDH problem, given a random triple (X_1, X_2, Y) of finite cyclic group elements, outputs the DH results of (X_1, Y) and of (X_2, Y).

It is a non-trivial task to embed the trapdoor test method in the AKE security proof and do verification without a DDH oracle. On one hand, the gap DH (GDH) assumption is stronger than the TDH assumption, and, more important, the decision oracle for TDH problem is very different from that used in the AKE proof. The decision oracle involved in the AKE proof allows *fully* adversarial access to the DDH predicate which on the adversary chosen input $(\tilde{X}, \tilde{Y}, \tilde{Z})$ returns 1 if $\tilde{Z}$ is the DH value of $\tilde{X}$ and $\tilde{Y}$,

Journal of Internet Services and Information Security, volume: 1, number: 2/3, pp. 107-124

*An extended abstract of this paper has been published in ProvSec 2011, LNCS 6890, Springer, 2011.

†Corresponding author.

while the TDH decision oracle only provides *partially* adversarial access to the TDH predicate which, on input $(X_1, X_2, \tilde{Y}, \tilde{Z}_1, \tilde{Z}_2)$ where $\tilde{Y}, \tilde{Z}_1$ and $\tilde{Z}_2$ are adversary chosen and X_1 and X_2 are from the problem instance, returns 1 if both $\tilde{Z}_1$ is the DH value of $(X_1, \tilde{Y})$ and $\tilde{Z}_2$ is the DH value of $(X_2, \tilde{Y})$.

On the other hand, resilience to the leakage on intermediate results in computing session keys is a desired requirement for the authenticated DH protocol. In practice, an AKE protocol is always implemented using an untrusted host machine together with a computationally limited tamper-resistant device, which stores the static private keys. For many authenticated DH protocols, the computation of intermediate results is more costly than that of ephemeral public keys. To enhance the protocol implementation efficiency safely, we require the protocol is secure against the intermediate result leakage and compute the intermediate result in the host machine, while the ephemeral public key is pre-computed in the device's idle time.

Unfortunately, many existing AKE protocols without the gap assumption (e.g. NAXOS+ [14], HC [8] and KFU2 [9]) are insecure against the intermediate result leakage. That is to say the intermediate results of these protocols must be computed in the tamper-resistant device's non-idle time. What is worse, for these AKE protocols, the computation of intermediate results is much more costly than that of ephemeral public keys. For example, NAXOS+ requires 4 exponentiations for a session key, but only 1 exponentiation for an ephemeral public key. Thus, designing an AKE protocol resilient to the intermediate result leakage without the gap assumption is of not only theoretical interest but also practical values.

Contribution. According to the aforemention challenges, a twinning variant of MQV, named as TMQV, is proposed.

To capture the intermediate result leakage formally, we recall the strengthened eCK (seCK) model of Sarr *et al.* [20]. Three attempts in designing AKE protocols without the gap assumption are due to NAXOS+ [14] and to HC [8] and to KFU2 [9]. Unfortunately, these protocols does not consider the intermediate result leakage and we show the insecurity of these AKE protocols in the seCK model formally.

Motivated by the seCK-insecurity of these protocols, we enhance the MQV key derivation method by binding the session identifiers to the shared secrets, and apply the twinning technique to compute the static keys and the shared secrets in order to remove the gap assumption. With the help of the trapdoor test theorem, TMQV is provably secure under the standard CDH assumption in the seCK model.

Unlike the HMQV, CMQV and SMQV [20] proof, TMQV does not need the GDH assumption or KEA1 [2], and admits higher security than HMQV and CMQV. In addition, TMQV does not use NAXOS trick. As claimed in [22, Sect. 4.3] and [15, Sect. 3], the protocols based on NAXOS trick can not guarantee their security against the leakage on the discrete logarithm of the ephemeral public key. Finally, compared with the related works (especially for NAXOS+, HC and KFU2), TMQV achieves stronger security and higher implementation efficiency with weaker cryptographic assumptions.

Related works. Here we provide a brief overview on the AKE security models and the AKE protocols in the random oracle model.

SECURITY MODEL. The CK model [4] is a well known formal model for AKE protocols. It provides a security consideration on the session-state leakage. By specifying the session-state to be the ephemeral private key, an extended version (the eCK model) was defined by LaMacchia *et al.* [12]. Cremers [7] drew a separation result among some variants of the CK model (including the CK and eCK models) which stated none of them was stronger than the others.

More recently, Sarr *et al.* [20] proposed the strengthened eCK (seCK) model which can capture the intermediate result leakage resilience, in addition to the security attributes considered in the CK and eCK models. It is shown that the seCK model encompasses the eCK model and is practically stronger than the CK model.

AKE PROTOCOL. The MQV protocol [13] is an efficient and well studied AKE protocol. Its security

was formally proven by Kunz-Jacques *et al.* [11], but the proof was carried out in a weak model, which considers no session-state leakages, and relies on a non-standard DH assumption in the random oracle model. HMQV [10] is the widely accepted provably secure variant of MQV. After that, many provably secure MQV variants have been proposed: CMQV [21] was obtained from (H)MQV and NAXOS [12]; and SMQV [20] was obtained from the FXCR and FDCR signature schemes [19]. Besides the MQV method, several efficient AKE protocols have been proposed in the literature, such as NAXOS. The provable security of the above protocols is based on the non-standard assumptions, such as the GDH assumption and KEA1.

To remove the gap assumption, some solutions have been proposed (e.g. NAXOS+ [14] and HC [8] and KFU2 [9]). However, in the seCK model, we will show they fall to the key compromise impersonation attack and the secret replication attack in the presence of the intermediate result leakage. The insecurity limits their implementation efficiency. Thus, a seCK-secure AKE protocol without the gap assumption is in great need.

Organization. Section 2 describes the seCK model to capture the AKE security, and we will analyze the seCK-insecurity of some AKE protocols in Section 3. Section 4 proposes TMQV protocol combined with its design rationales and security result and protocols comparison. We draw a conclusion in Section 5.

2 Strengthened eCK Model

In this section, we recall the strengthened eCK (seCK) model [20] in order to capture the security of AKE protocols. For simplicity, we will only describe the model for two-pass DH key exchange protocols here.

Informal overview. For an AKE protocol, the seCK model considers two implementation approaches: Approach 1 allows only the ephemeral private leakage; Approach 2 allows the leakage on intermediate results in computing the session key. The seCK model allows two communication parties in a protocol execution follow different implementation approaches.

The invention of the seCK model has not only the theoretical sense but also the practical value. In practice, an AKE protocol is always implemented using an untrusted host machine together with a (computationally limited) tamper-resistant device, which stores the static private key. The adversary can run a malware on the host machine at a party to learn all values computed in that party, except those stored in the tamper-resistant device. To reduce the tamper-resistant device's non-idle time computational effort, which is crucial for implementation efficiency, the ephemeral public key can be computed on the host machine and the other computation is done on the device (which is captured by Approach 1 in the seCK model), or the ephemeral public key can be pre-computed on the device in the idle-time while intermediate results in computing session keys are done in the host machine (which is captured by Approach 2 in the seCK model). For many authenticated DH protocols (e.g. HMQV [10], SMQV [20], NAXOS [12] and KFU [9]), the computation of the intermediate results is more costly than that of the ephemeral public key. Thus, the second approach in the seCK model achieves the practical value in the AKE implementation, which can not be achieved by many security models (e.g. the eCK model [12]).

Moreover, it is shown that the seCK-security encompasses the eCK-security, and is practically stronger than the CK-security [20].

Parties. In the seCK model there are $n(\lambda)$ parties each modeled by a probabilistic Turing machine and denoted by $\hat{A}$ ($\hat{B}$), where $\lambda \in \mathbb{N}$ is a security parameter and $n(\cdot)$ is polynomial. Each party has a static public-private key pair together with a certificate that binds the static public key to that party, and publishes its static public key before the protocol execution. We do not assume that the certifying authority (CA) requires parties to prove possession of their static private keys, but we require that the CA verifies that the static public key of a party belongs to the domain of public keys.

Session. A party $\hat{A}$ can be activated to execute an instance of the protocol called a *session*. $\hat{A}$ is activated by an incoming message with one of the following forms: (i) $(\hat{A},\hat{B})$ or (ii) $(\hat{A},\hat{B},Y)$ where Y is $\hat{B}$'s outgoing ephemeral public key. If $\hat{A}$ is activated by $(\hat{A},\hat{B})$ then $\hat{A}$ is the session *initiator* with peer $\hat{B}$, otherwise the session *responder* with peer $\hat{B}$.

The session is identified via a session identifier $sid = (role, ID_1, ID_2, out, in)$, where *role* is the role performed by the session (initiator or responder), ID_1 is the identifier of the participant executing *sid*, ID_2 is the identifier of the intended communication peer, and *out* and *in* denote the ephemeral public keys ID_1 sends to ID_2, or believes to be from ID_2 respectively.

Two sessions $sid = (role, ID_1, ID_2, out, in)$ and $sid^* = (role^*, ID_1^*, ID_2^*, out^*, in^*)$ are said to be **matching** if and only if $role \neq role^*$, $ID_1 = ID_2^*$, $ID_2 = ID_1^*$, and at completion $out = in^*$ and $in = out^*$. For example, sessions $(\mathcal{I},\hat{A},\hat{B},X,Y)$ and $(\mathcal{R},\hat{B},\hat{A},Y,X)$ are matching, where $\mathcal{I}$ denotes initiator and $\mathcal{R}$ denotes responder. Moreover, the session matching to $(\mathcal{R},\hat{B},\hat{A},Y,X)$ can be an incomplete session $(\mathcal{I},\hat{A},\hat{B},X,-)$, where $-$ is a symbol meaning the incoming public key is not received. A session can not have (except with negligible probability) more than one matching session.

Adversarial capability. The adversary $\mathcal{M}$ is modeled as a probabilistic Turing machine and makes oracle queries to parties. It means that the adversary can control all communications between the protocol participants. The adversary presents parties with incoming messages via $\mathsf{Send}(message)$, and thereby controls the activation of sessions. For different implementation approaches, we define different adversarial oracle queries to formalize possible leakage of private information. Note that the adversary $\mathcal{M}$ can only issue the queries corresponding to the implementation approach followed by a party.

ORACLE QUERIES FOR APPROACH 1:

- $\mathsf{EphemeralKeyReveal}(sid)$: The adversary obtains the ephemeral private key held by the session *sid*.
- $\mathsf{SessionKeyReveal}(sid)$: The adversary obtains the session key for a complete session *sid*.
- $\mathsf{StaticKeyReveal}(\hat{A})$: The adversary learns the static private key of party $\hat{A}$. Note that the $\mathsf{Corrupt}_{SC}$ query defined in the original paper [20] is the same as $\mathsf{StaticKeyReveal}$ query in the eCK model. For convenience, we rename $\mathsf{Corrupt}_{SC}$ as $\mathsf{StaticKeyReveal}$.
- $\mathsf{Establish}(\hat{A})$: This query allows the adversary to register a static public key on behalf of a party $\hat{A}$. In this way the adversary totally controls that party. Parties against whom the adversary did not issue this query are called *honest*.

ORACLE QUERIES FOR APPROACH 2: Definitions remain unchanged for queries belonging also to Approach 1.

- $\mathsf{InterReveal}(sid)$: The adversary $\mathcal{M}$ learns intermediate results in computing the session key of session *sid*. Take SMQV [20] for example: assume party $\hat{A}$ follows Approach 2; the ephemeral public key X and the secret exponent $s_A = xd + a$ are computed in the tamper-resistant device; s_A is sent to host machine through the card reader, and then $\sigma_{\hat{A}}$ is computed in the host machine. By issuing this query, $\mathcal{M}$ can get s_A and $\sigma_{\hat{A}}$.
- $\mathsf{SessionKeyReveal}(sid)$.
- $\mathsf{StaticKeyReveal}(\hat{A})$.
- $\mathsf{Establish}(\hat{A})$.

Adversarial goal. The aim of the adversary $\mathcal{M}$ against the AKE protocol is to distinguish the session key for the *fresh session* from a random key, which means $\mathcal{M}$ wins the seCK game.

The freshness notion is defined as follows in order to capture the intuitive fact that a session key can not be trivially known to the adversary.

Definition 1 (Fresh session)**.** *Let sid be the session identifier of a completed session, owned by an honest party $\hat{A}$ with peer $\hat{B}$, who is also honest. Let sid^* be the session identifier of the matching session of sid, if the corresponding matching session exists. sid is said to be **locally exposed** if one of the following holds:*

- *The adversary $\mathcal{M}$ issues* SessionKeyReveal(sid).
- *The implementation at $\hat{A}$ follows Approach 1 and $\mathcal{M}$ issues both* EphemeralKeyReveal(sid) *and* StaticKeyReveal$(\hat{A})$.
- *The implementation at $\hat{A}$ follows Approach 2 and $\mathcal{M}$ issues* InterReveal(sid).

*The session sid is said to be **exposed** if (a) it is locally exposed, or (b) its matching session sid^* exists and is locally exposed, or (c) sid^* does not exist and $\mathcal{M}$ issues* StaticKeyReveal$(\hat{B})$. *An **unexposed** session is said to be **fresh**.* □

We capture the security of AKE protocols by introducing a special query:

- Test(sid): This query tries to capture the adversary's ability to tell apart a real session key from a random one. This query can be asked by an adversary only once during its entire execution. If no session key K for the session sid is defined, or the session sid is not fresh, then an error symbol $\perp$ is returned. Otherwise, a private coin β is flipped ($\beta \xleftarrow{\$} \{0,1\}$), and K is returned if $\beta = 1$ or a random l_{sk}-bit string, where l_{sk} is the length of session key, is returned if $\beta = 0$.

Formally, the adversary $\mathcal{M}$ against the seCK-security wants to guess a bit β' such that $\beta' = \beta$ where β is the private coin involved in Test query. We define the advantage of the adversary $\mathcal{M}$ by $\mathbf{Adv}_{\mathcal{P}}^{seCK}(\mathcal{M}) = |\Pr[\beta' = \beta] - \frac{1}{2}|$.

Definition 2 (seCK-security)**.** *An AKE protocol $\mathcal{P}$ is secure if the following conditions hold:*

1. *If two honest parties complete matching sessions then, except with negligible probability, they both compute the same session key (or both output indication of protocol failure).*
2. *The polynomially bounded adversary $\mathcal{M}$ can distinguish a session key from a random string with a negligible advantage, namely, $\mathbf{Adv}_{\mathcal{P}}^{seCK}(\mathcal{M})$ is negligible in λ.*

□

3 seCK-Insecurity of Some AKE Protocols

In order to motivate the design of seCK-secure DH protocol without the gap assumption, we cryptanalyze NAXOS+ [14] and HC [8] and KFU2 [9] which are AKE protocols based on the standard CDH problem with respect to the ephemeral private key leakage.

Here we present a key compromise impersonation (KCI) attack and a secret replication (SR) attack on KFU2 [9] using oracle queries defined in Approach 2 of the seCK model. These attacks can be easily extended to the other two (i.e. NAXOS+ and HC). We do not describe these protocols here and recommend the original papers for the readers.

KCI attack. Resistance to the KCI attack guarantees the disclosure of an honest user $\hat{A}$'s static private key does not enable an adversary to impersonate other uncorrupted entities to $\hat{A}$. KFU2 falls to the KCI attack:

1. The adversary $\mathcal{M}$ issues $\mathsf{StaticKeyReveal}(\hat{A})$ to learn the static private key (a_1, a_2) of party $\hat{A}$; $\mathcal{M}$ chooses $(a_1', a_2') \xleftarrow{\$} \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ ($a_1' \neq a_1$ and $a_2' \neq a_2$) and computes $(A_1', A_2') = (g^{a_1'}, g^{a_2'})$ and registers a party $\hat{A}'$ ($\hat{A}' \neq \hat{A}$) and the corresponding static public key (A_1', A_2') through $\mathsf{Establish}$ query.

2. $\mathcal{M}$ issues $\mathsf{Send}(\hat{A}, \hat{B})$ to initiate $\hat{A}$ who creates a session *sid* as an initiator with the honest peer $\hat{B}$ and returns $(\hat{B}, \hat{A}, X)$. Then $\mathcal{M}$ records the ephemeral public key X.

3. $\mathcal{M}$ issues $\mathsf{Send}(\hat{B}, \hat{A}', X)$ to initiate $\hat{B}$ who creates a session *sid*$'$ as a responder with the peer $\hat{A}'$. $\hat{B}$ computes the ephemeral public key $Y = g^y$ ($y \xleftarrow{\$} \mathbb{Z}_q^*$) and the shared secrets $Z_1' = (XA_1')^{y+b_1}$, $Z_2' = (XA_1')^{y+b_2}$, $Z_3' = (XA_2')^{y+b_1}$ and $Z_4' = (XA_2')^{y+b_2}$. During this computation $\mathcal{M}$ issues $\mathsf{InterReveal}(sid')$ to get (Z_1', Z_2', Z_3', Z_4'). $\hat{B}$ returns $(\hat{A}', \hat{B}, X, Y)$ and completes the session $sid' = (\mathcal{R}, \hat{B}, \hat{A}', Y, X)$ with session key $SK' = H(Z_1', Z_2', Z_3', Z_4', X, Y, \hat{A}', \hat{B})$.

4. $\mathcal{M}$ issues $\mathsf{Send}(sid, (\hat{A}, \hat{B}, X, Y))$ to $\hat{A}$ and $\hat{A}$ computes $Z_1 = (YB_1)^{x+a_1}$, $Z_2 = (YB_2)^{x+a_1}$, $Z_3 = (YB_1)^{x+a_2}$ and $Z_4 = (YB_2)^{x+a_2}$ and completes the session $sid = (\mathcal{I}, \hat{A}, \hat{B}, X, Y)$ with session key $SK = H(Z_1, Z_2, Z_3, Z_4, X, Y, \hat{A}, \hat{B})$.

5. $\mathcal{M}$ can compute (Z_1, Z_2, Z_3, Z_4) correctly according to the relations $Z_1 = Z_1' \cdot (YB_1)^{a_1 - a_1'}$, $Z_2 = Z_2' \cdot (YB_2)^{a_1 - a_1'}$, $Z_3 = Z_3' \cdot (YB_1)^{a_2 - a_2'}$ and $Z_4 = Z_4' \cdot (YB_2)^{a_2 - a_2'}$. Session *sid* is fresh, since *sid* and *sid*$'$ are non-matching and *sid* is locally unexposed and no $\mathsf{StaticKeyReveal}(\hat{B})$ is issued. $\mathcal{M}$ chooses *sid* as the test session and breaks the seCK-security of *sid* easily. That is to say $\mathcal{M}$ successfully impersonates any uncorrupted party $\hat{B}$ to party $\hat{A}$ to compute a common session key after revealing $\hat{A}$'s static private key.

The attack shown as above can be easily extended to NAXOS+ and HC: $\mathcal{M}$ executes Step 1 to 4 faithfully according to the definitions of NAXOS+ and HC except for the relation between the secrets of two non-matching sessions *sid* and *sid*$'$ in Step 5: for NAXOS+, $Z_1 = B^a$, $Z_2 = Y^a$, $Z_3 = Z_3'$ and $Z_4 = Z_4'$ (where $Z_1 = \mathsf{CDH}(A, B)$, $Z_2 = \mathsf{CDH}(A, Y)$, $Z_3 = \mathsf{CDH}(X, B)$ and $Z_4 = \mathsf{CDH}(X, Y)$), and for HC, $Z_1 = Z_1'$, $Z_2 = Z_2'$, $Z_3 = Z_3' \cdot Y^{a_1 - a_1'}$ and $Z_4 = Z_4' \cdot Y^{a_2 - a_2'}$. In addition, we recognize a trivial attack on HC using $\mathsf{InterReveal}$ query: by issuing $\mathsf{InterReveal}$ query on any session in $\hat{B}$, $\mathcal{M}$ gets y, $y + b_1$ and $y + b_2$, and then $\mathcal{M}$ figures out (b_1, b_2) and impersonates $\hat{B}$ without issuing $\mathsf{StaticKeyReveal}(\hat{B})$.

The KCI attack is possible since the shared secrets of these protocols are not related to the identifiers of the participants in the protocol execution. As a result of that, the adversary $\mathcal{M}$ can act as a dishonest party $\hat{A}'$ in the middle and forward the messages between $\hat{A}$ and $\hat{B}$. At the end of the execution, $\hat{A}$ and $\hat{B}$ compute the similar secrets with non-matching peers' static public keys. Since $\mathcal{M}$ knows the static private keys of $\hat{A}$ and $\hat{A}'$, according to the property of the CDH problem, $\mathcal{M}$ can extract $\hat{A}$'s secret from $\hat{B}$'s secret successfully.

SR attack. The SR attack was proposed firstly by Cremers [6]. In the attack, the adversary forces two non-matching sessions with the same communication parties to establish similar shared secrets. Then $\mathcal{M}$ can compute the shared secret of the test session from that of the non-matching session. SR attacks differ from KCI attacks shown as above in the sense that $\mathcal{M}$ does not learn the static private key of any party. The SR attack on KFU2 is shown as follows:

1. $\mathcal{M}$ issues $\mathsf{Send}(\hat{A}, \hat{B})$ to initiate $\hat{A}$ who creates a session *sid* as an initiator and returns $(\hat{B}, \hat{A}, X)$. Then $\mathcal{M}$ records $\hat{A}$'s ephemeral public key X.

2. $\mathcal{M}$ issues $\mathsf{Send}(\hat{B},\hat{A})$ to initiate $\hat{B}$ who creates a session sid' as an initiator and returns $(\hat{A},\hat{B},Y)$. Then $\mathcal{M}$ records $\hat{B}$'s ephemeral public key Y.

3. $\mathcal{M}$ issues $\mathsf{Send}(sid',(\hat{B},\hat{A},Y,X))$ to $\hat{B}$ and $\mathsf{Send}(sid,(\hat{A},\hat{B},X,Y))$ to $\hat{A}$. Then, $\hat{B}$ computes the shared secret as $Z'_1=(XA_1)^{y+b_1}$, $Z'_2=(XA_2)^{y+b_1}$, $Z'_3=(XA_1)^{y+b_2}$ and $Z'_4=(XA_2)^{y+b_2}$. $\hat{A}$ computes the shared secret as $Z_1=(YB_1)^{x+a_1}$, $Z_2=(YB_2)^{x+a_1}$, $Z_3=(YB_1)^{x+a_2}$ and $Z_4=(YB_2)^{x+a_2}$. We can see $Z_1=Z'_1$, $Z_2=Z'_3$, $Z_3=Z'_2$ and $Z_4=Z'_4$, and $sid=(\mathcal{I},\hat{A},\hat{B},X,Y)$ and $sid'=(\mathcal{I},\hat{B},\hat{A},Y,X)$.

4. $\mathcal{M}$ issues $\mathsf{InterReveal}(sid')$ to get (Z'_1,Z'_2,Z'_3,Z'_4) and computes the secret (Z_1,Z_2,Z_3,Z_4) of sid according to the relation described in Step 3.

5. Session sid is fresh, since sid and sid' are non-matching and sid is locally unexposed and no $\mathsf{StaticKeyReveal}(\hat{B})$ is issued. $\mathcal{M}$ issues $\mathsf{Test}(sid)$ and guesses the private bit involved in Test query correctly.

The attack shown as above can be easily extended to NAXOS+ and HC: $\mathcal{M}$ executes Step 1, 2, 4 and 5 according to the definition of NAXOS+ and HC except the relation between the secrets of two non-matching sessions in Step 3; for NAXOS+, $Z_1=Z'_1$, $Z_2=Z'_3$, $Z_3=Z'_2$ and $Z_4=Z'_4$, and for HC, $Z_1=Z'_3$, $Z_2=Z'_4$, $Z_3=Z'_1$ and $Z_4=Z'_2$.

The SR attack is possible since the shared secrets of these protocols are not bounded to the corresponding sessions, and then the adversary can easily force two non-matching sessions with similar secrets.

These attacks conclude, for NAXOS+ and HC and KFU2, the shared secrets should be computed in the computationally limited tamper-resistant device (by their insecurity against the intermediate result leakage), while the ephemeral public keys can be computed in the untrusted host machine (by their eCK-security). In order to reduce the tamper-resistant device's computational effort and achieve higher AKE security, it is interesting to design an AKE protocol based on CDH with security against the leakages on the intermediate result and the ephemeral private key respectively.

4 Proposed AKE Protocol: TMQV

We present several design rationales of TMQV before its definition.

Prevent KCI and SR attacks. Hashing the session identifiers into the exponents of the corresponding secrets (i.e. d and e in TMQV) can make the secrets of two non-matching sessions independent, due to the properties of non-matching session definition and the hash function (which is modeled as a random oracle). The independence makes KCI and SR attacks impossible. Moreover, as shown formally in Appendix B, TMQV is provably seCK-secure.

Remove the gap assumption. In the TMQV initialization procedure, we apply the twinning technique to compute the static keys appropriately such that the trapdoor test method can be used in the security reduction, and thus the simulator can answer the DDH oracle faithfully (except with negligible probability) and solve the CDH problem easily. That is equivalent to say the security of TMQV is provable without the GDH assumption and admits weaker cryptographic assumptions. Compared with the related methods to remove the gap assumption (e.g. NAXOS+, HC and KFU2), TMQV is more effective (see Section 4.3).

Avoid the NAXOS trick. We avoid the use of the controversial NAXOS trick [12] in the ephemeral public key derivation, which is recommended by Ustaoglu [22, Sect. 4.3] and Moriyama *et al.* [15, Sect. 3]. In the design of TMQV, we derive the ephemeral public key for a party following the original Diffie-Hellman protocol, and improve the secret derivation method of MQV by the twin static key: the secret

for party $\hat{A}$ is computed as $(\sigma_1,\sigma_2)=(\mathsf{CDH}(XA_1^d,YB_2^e),\mathsf{CDH}(XA_2^d,YB_1^e))$. Without the knowledge of x and (a_1,a_2), an entity can not learn $(\mathsf{DLOG}(XA_1^d),\mathsf{DLOG}(XA_2^d))$ and figure out (σ_1,σ_2).

4.1 Protocol Description

TMQV is a two-pass AKE protocol. Let $\lambda\in\mathbb{N}$ be the security parameter and $\mathbb{G}=\langle g\rangle$ be a cyclic group of prime order q with $|q|=\lambda$ and H_1 and H_2 be two hash functions modeled as random oracles. H_1 outputs integers of length $|q|/2$. This output length provides the best trad-off between security and performance [10, Remark 4.2]

We compute the twin static key of a party: the static private key of party $\hat{A}$ is $(a_1,a_2)\xleftarrow{\$}\mathbb{Z}_q^*\times\mathbb{Z}_q^*$ (where $\mathbb{Z}_q^*=[1,q-1]$) and the static public key is $(A_1,A_2)=(g^{a_1},g^{a_2})$; similarly, the static private key of $\hat{B}$ is (b_1,b_2) and the static public key is $(B_1,B_2)=(g^{b_1},g^{b_2})$. As usual, we require a certificate binds the static public key to the corresponding party. TMQV proceeds as follows:

1. Upon activation $(\hat{A},\hat{B})$, party $\hat{A}$ (the initiator) performs the steps: (a) Select an ephemeral private key $x\xleftarrow{\$}\mathbb{Z}_q^*$; (b) Compute the ephemeral public key $X=g^x$; (c) Initiate session $sid=(\mathcal{I},\hat{A},\hat{B},X,-)$ and send $(\hat{B},\hat{A},X)$ to $\hat{B}$.

2. Upon receiving $(\hat{B},\hat{A},X)$, party $\hat{B}$ verifies that $X\in\mathbb{G}^*=\mathbb{G}\backslash\{1\}$. If so, $\hat{B}$ performs the following steps: (a) Select an ephemeral private key $y\xleftarrow{\$}\mathbb{Z}_q^*$; (b) Compute the ephemeral public key $Y=g^y$. (c) Compute $d=H_1(\mathcal{I},\hat{A},\hat{B},X,Y)$ and $e=H_1(\mathcal{R},\hat{B},\hat{A},Y,X)$; (d) Compute the secret exponents $s_{1b}=y+eb_2$ and $s_{2b}=y+eb_1$ and the secret $\sigma_1=(XA_1^d)^{s_{1b}}$ and $\sigma_2=(XA_2^d)^{s_{2b}}$; (e) Compute the session key $K_{\hat{B}}=H_2(\sigma_1,\sigma_2,\hat{A},\hat{B},X,Y)$; (f) Complete session $sid^*=(\mathcal{R},\hat{B},\hat{A},Y,X)$ with the session key $K_{\hat{B}}$ and send $(\hat{A},\hat{B},X,Y)$ to $\hat{A}$.

3. Upon receiving $(\hat{A},\hat{B},X,Y)$, party $\hat{A}$ checks if he owns a session with identifier $(\mathcal{I},\hat{A},\hat{B},X,-)$. If so, $\hat{A}$ verifies that $Y\in\mathbb{G}^*$ and performs the following steps: (a) Compute $d=H_1(\mathcal{I},\hat{A},\hat{B},X,Y)$ and $e=H_1(\mathcal{R},\hat{B},\hat{A},Y,X)$; (b) Compute the secret exponents $s_{1a}=x+da_1$ and $s_{2a}=x+da_2$ and the shared secret $\sigma_1=(YB_2^e)^{s_{1a}}$ and $\sigma_2=(YB_1^e)^{s_{2a}}$; (c) Compute the session key $K_{\hat{A}}=H_2(\sigma_1,\sigma_2,\hat{A},\hat{B},X,Y)$; (d) Complete session $sid=(\mathcal{I},\hat{A},\hat{B},X,Y)$ with the session key $K_{\hat{A}}$.

If any verification fails the party erases all session specific information from its memory and aborts the session. It is easy to see two honest users execute TMQV directly can agree on the same session key. A graphical description of TMQV is shown in Figure 1.

4.2 Security

Informally the design rationales suggest that TMQV is secure. Formally, the following theorem shows TMQV is seCK-secure under the standard CDH assumption and the random oracle assumption.

Theorem 1 (seCK-security of TMQV)**.** *For any probabilistic polynomial-time (PPT) adversary $\mathcal{M}$ against the seCK-security of TMQV that involves at most $n(\lambda)$ honest parties and activates at most $s(\lambda)$ sessions, and makes $h_1(\lambda)$ and $h_2(\lambda)$ random oracle queries to H_1 and H_2 respectively, where $\lambda\in\mathbb{N}$ is a security parameter and $s(\cdot)$, $n(\cdot)$, $h_1(\cdot)$ and $h_2(\cdot)$ are polynomial, we can construct a PPT attacker $\mathcal{S}$ who can solve the CDH problem with probability*

$$\mathbf{Succ}_{\mathbb{G}}^{cdh}(\mathcal{S})\geq\max\{\frac{1}{n(\lambda)s(\lambda)^2},\frac{1}{s(\lambda)^2h_2(\lambda)},\frac{C}{n(\lambda)^2s(\lambda)h_1(\lambda)},\frac{C}{n(\lambda)s(\lambda)h_1(\lambda)}\}\mathbf{Adv}_{\mathrm{TMQV}}^{seCK}(\mathcal{M})$$

where $\mathbf{Adv}_{\mathrm{TMQV}}^{seCK}(\mathcal{M})$ is the advantage of $\mathcal{M}$ to attack the seCK-security of TMQV and C is a constant arising from the forking lemma [17]. □

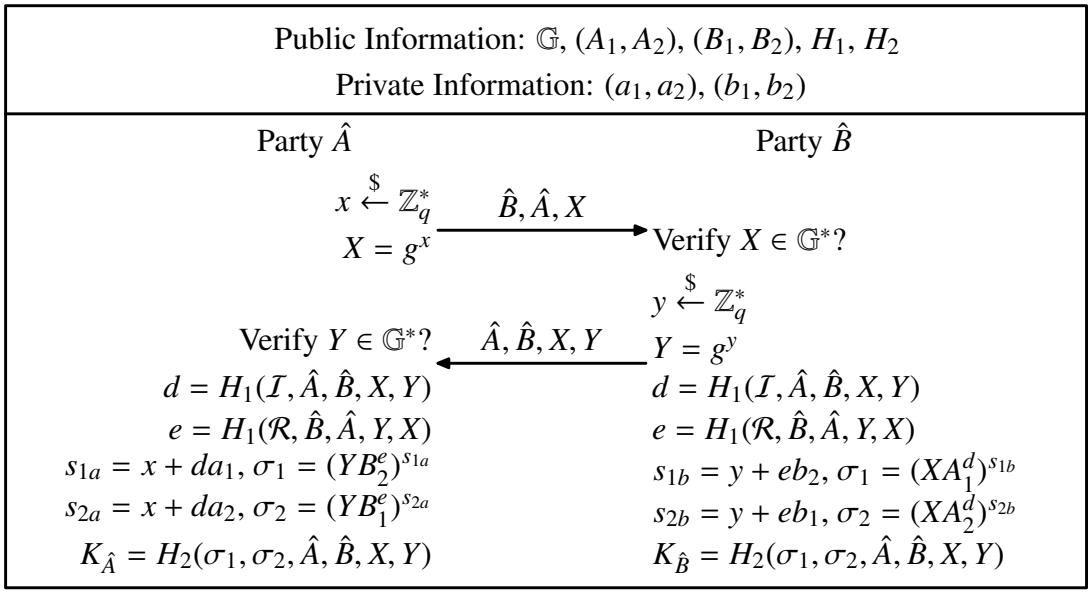

Figure 1: An honest execution of TMQV

A detailed reductionist proof is presented in Appendix B to show the seCK-security of TMQV without the GDH assumption. In the proof, we embed the CDH instance in the static public key or the ephemeral public key appropriately according to the freshness of the test session, and use the trapdoor test theorem [5] to remove the gap assumption due to the twin static keys; and we also have a patch in the proof to treat a special kind of attackers using the derandomization technique.

4.3 Protocols Comparison

Table 1 provides a comparison between TMQV and some well accepted AKE protocols in the random oracle model. The comparison focuses on the efficiency (numbers of exponentiations and hash computation per party) and the security (security model, cryptographic assumption and tightness of security reduction). "#Exp-s" and "#Exp-e" and "#Exp-ss" denote the numbers of exponentiations used to compute the static public keys and ephemeral public keys and shared secrets respectively. Note that, for TMQV, #Exp-ss is counted as 1.5 exponentiations for σ_1 (0.5 for e-exponent and 1 for s_{1a}-exponent) and 1.5 exponentiations for σ_2. All protocols are assumed to perform public key validation, so it is left out efficiency. The number of exponentiations is counted in the naive way without accounting for possible improvements. "#Hash" denotes the number of hash computation. We say a security reduction *is* tight if the forking lemma [17] is not used in the reduction, otherwise the security reduction is *not* tight. The advantages of TMQV are shown:

Higher security level and more practical value. TMQV not only achieves higher security level but also is more practical than NAXOS+ and HC and KFU2 which are also based on CDH. TMQV is provably seCK-secure while the other three are shown to be seCK-insecure. The seCK-security achieves both theoretical and practical advantages and encompass the eCK-security.

TMQV requires less exponentiations in computing the shared secret. #Exp-ss of TMQV is 3, since computing σ_1 requires 1.5 exponentiations (0.5 for e-exponentiation and 1 for s_{1a}-exponentiation) and computing σ_2 requires 1.5 exponentiations. More hash computation is required for TMQV. As it is known to us, the hashing is much more efficient than the exponentiation. Although TMQV requires one more exponentiation in computing the static public key than NAXOS+, but the computation of the shared

Protocol	#Exp-s	#Exp-e	#Exp-ss	#Hash	Model	Assumption	Tight
HMQV [10]	1	1	1.5	3	CK (not seCK)	GDH, KEA1 [2]	no
CMQV [21]	1	1	2	4	eCK (not seCK)	GDH	no
SMQV [20]	1	1	1.5	3	seCK	GDH	no
NAXOS [12]	1	1	3	2	eCK (not seCK)	GDH	yes
NAXOS+ [14]	1	1	4	2	eCK (not seCK)	CDH	yes
HC [8]	2	1	4	2	eCK (not seCK)	CDH	yes
KFU2 [9]	2	1	4	1	eCK (not seCK)	CDH	yes
TMQV	2	1	3	3	seCK	CDH	no

Table 1: Protocols Comparison.

secret is more costly than that of the static public key. For $s(\lambda) > 1$ sessions, NAXOS+ requires $s(\lambda) - 1$ more exponentiations than TMQV.

TMQV can follow different implementation approaches to improve its efficiency, while NAXOS+ and HC and KFU2 can not. For these four protocols, the computation of shared secrets is more costly than that of ephemeral public keys. Implementation efficiency of TMQV can be significantly enhanced following Approach 2 in the seCK model, but that is not the case for the other three due to their insecurity against the intermediate result leakage.

Weaker cryptographic assumptions. Amongst the provably secure MQV variants, TMQV admits weaker assumptions with comparable security level. As shown in [19, Sect. 2] and [20, Sect. 4], HMQV and CMQV are not ephemeral secret exponent leakage resilient. That is to say HMQV and CMQV are both seCK-insecure. For SMQV, TMQV achieves the same security level but weaker assumptions.

NAXOS-trick-free design. TMQV is a NAXOS-trick-free design, while NAXOS, NAXOS+ and HC compute the ephemeral public keys using the NAXOS trick. Ustaoglu [22] and Moriyama *et al.* [15] claimed that security proofs of the protocol using NAXOS trick can not guarantee the security against the leakage on the discrete logarithm of the ephemeral public key. Moreover, it is not difficult to see NAXOS can not meet the seCK-security.

5 Conclusion

We have proposed an authenticated Diffie-Hellman protocol, TMQV, by utilizing the twinning technique and the key derivation method of MQV. TMQV is provably secure in the strengthened eCK model under the standard computational Diffie-Hellman (CDH) assumption. The implementation efficiency of TMQV can be significantly reduced, while the related protocols can not. Compared with the related works, TMQV admits the strengthened security and enhanced efficiency with weaker and more standard cryptographic assumptions. On the negative side, the security reduction of TMQV is not tight, since the forking lemma is essential. It is interesting to consider an appropriate modification to the shared secret derivation of MQV in order to construct a tight reduction under the CDH assumption.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments, and Augustin P. Sarr for helpful discussions on improving the security proof.

References

[1] F. Bao, R. H. Deng, and H. Zhu. Variations of Diffie-Hellman problem. In *Proc. of the 5th International Conference on Information and Communications Security (ICICS'03), Huhehaote, China, LNCS*, volume 2836, pages 301–312. Springer-Verlag, October 2003.

[2] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero knowledge protocols. In *Proc. of the 24th Annual International Cryptology Conference (CRYPTO'04), Santa Barbara, California, USA, LNCS*, volume 3152, pages 273–289. Springer-Verlag, August 2004.

[3] E. R. Berlekamp. Factoring polynomials over large finite fields. *Mathmatics of Computation*, 24(111):713–735, July 1970.

[4] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Proc. of the 20th International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'01), Innsbruck, Austria, LNCS*, volume 2045, pages 453–474. Springer-Verlag, May 2001.

[5] D. Cash, E. Kiltz, and V. Shoup. The twin Diffie-Hellman problem and applications. *Journal of Cryptology*, 22(4):470–504, October 2009.

[6] C. J. Cremers. Session-state reveal is stronger than Ephemeral Key Reveal: Attacking the NAXOS authenticated key exchange protocol. In *Proc. of the 7th International Conference on Applied Cryptography and Network Security (ACNS'09), Paris-Rocquencourt, France, LNCS*, volume 5536, pages 20–33. Springer-Verlag, June 2009.

[7] C. J. Cremers. Formally and Practically Relating the CK, CK-HMQV, and eCK Security Models for Authenticated Key Exchange. `http://eprint.iacr.org/2009/253.pdf`, July 2010.

[8] H. Huang and Z. Cao. Strongly Secure Authenticated Key Exchange Protocol Based on Computational Diffie-Hellman Problem. `http://eprint.iacr.org/2008/500.pdf`, November 2008.

[9] M. Kim, A. Fujioka, and B. Ustaoglu. Strongly secure authenticated key exchange without NAXOS' approach. In *Proc. of the 4th International Workshop on Information and Computer Security Security (IWSEC'09), Toyama, Japan, LNCS*, volume 5824, pages 174–191. Springer-Verlag, October 2009.

[10] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *Proc. of the 25th Annual International Cryptology Conference (CRYPTO'05), Santa Barbara, California, USA, LNCS*, volume 3621, pages 546–566. Springer-Verlag, August 2005.

[11] S. Kunz-Jacques and D. Pointcheval. About the security of MTI/C0 and MQV. In *Proc. of the 5th International Conference on Security and Cryptography for Networks (SCN'06), Maiori, Italy, LNCS*, volume 4116, pages 156–172. Springer-Verlag, September 2006.

[12] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *Proc. of the 1st International Conference on Provable Security (ProvSec'07), Wollongong, Australia, LNCS*, volume 4784, pages 1–16. Springer-Verlag, November 2007.

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

[14] J. Lee and J. H. Park. Authenticated Key Exchange Secure under the Computational Diffie-Hellman Assumption. `http://eprint.iacr.org/2008/344`, August 2008.

[15] D. Moriyama and T. Okamoto. An eCK-secure authenticated key exchange protocol without random oracles. In *Proc. of the 3rd International Conference on Provable Security (ProvSec'09), Guangzhou, China, LNCS*, volume 5848, pages 154–167. Springer-Verlag, November 2009.

[16] T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *Proc. of the 4th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC'01), Cheju Island, Korea, LNCS*, volume 1992, pages 104–118. Springer-Verlag, February 2001.

[17] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, December 2000.

[18] M. Rabin. Probabilistic algorithms in finite fields. *SIAM J. on Computing*, 9(2):273–280, May 1980.

[19] A. P. Sarr, P. Elbaz-Vincent, and J.-C. Bajard. A secure and efficient authenticated Diffie-Hellman protocol. In *Proc. of the 6th European Workshop on Public Key Infrastructures (EuroPKI'09), Pisa, Italy, LNCS*, volume 6391, pages 83–98. Springer-Verlag, September 2009.

[20] A. P. Sarr, P. Elbaz-Vincent, and J.-C. Bajard. A new security model for authenticated key agreement. In *Proc. of the 7th International Conference on Security and Cryptography for Networks (SCN'10), Amalfi, Italy, LNCS*, volume 6280, pages 219–234. Springer-Verlag, september 2010.

[21] B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, 46(3):329–342, March 2008.

[22] B. Ustaoglu. Comparing SessionStateReveal and EphemeralKeyReveal for Diffe-Hellman protocol. In *Proc. of the 3rd International Conference on Provable Security (ProvSec'09), Guangzhou, China, LNCS*, volume 5848, pages 183–197. Springer-Verlag, November 2009.

[23] K. Yoneyama and Y. Zhao. Taxonomical security consideration of authenticated key exchange resilient to intermediate computation leakage. In *Proc. of the 5th International Conference on Provable Security (ProvSec'11), Xi'an, China, LNCS*, volume 6980, pages 348–365. Springer-Verlag, October 2011.

A Diffie-Hellman Assumptions

Let $\lambda \in \mathbb{N}$ be a security parameter and $\mathbb{G} = \langle g \rangle$ denote a multiplicative cyclic group of known prime order q with $|q| = \lambda$, generated by $g \in \mathbb{G}$.

The discrete logarithm (DLOG) function $\mathsf{DLOG}(\cdot)$ in $\mathbb{G}$ takes input a random element $U \in \mathbb{G}$ and returns $u \in \mathbb{Z}_q$ such that $U = g^u$. The computational Diffie-Hellman (CDH) function $\mathsf{CDH}(\cdot,\cdot)$ takes as input a pair of random elements $(U,V) \in \mathbb{G}^2$ and returns $g^{\mathsf{DLOG}(U)\cdot\mathsf{DLOG}(V)}$. The decisional Diffie-Hellman (DDH) function $\mathsf{DDH}(\cdot,\cdot,\cdot)$ takes as input a triple of random elements $(U,V,W) \in \mathbb{G}^3$ and returns 1 if $W = \mathsf{CDH}(U,V)$ and 0 otherwise. An algorithm $\mathcal{S}$ in solving the Gap Diffie-Hellman (GDH) problem is given as input $(U,V) \xleftarrow{\$} \mathbb{G}^2$ and oracle access to $\mathsf{DDH}(\cdot,\cdot,\cdot)$ outputs $\mathsf{CDH}(U,V)$. The twin Diffie-Hellman (TDH) function $\mathsf{TDH}(\cdot,\cdot,\cdot)$ takes as input a triple of random elements $(U,V,W) \in \mathbb{G}^3$ and returns $(g^{\mathsf{DLOG}(U)\mathsf{DLOG}(W)}, g^{\mathsf{DLOG}(V)\mathsf{DLOG}(W)})$.

The DLOG/CDH/DDH/GDH/TDH assumption says the probability that any polynomial time (in λ) algorithm can solve the DLOG/CDH/DDH/GDH/TDH problem is negligible (in λ).

B Security Proof of TMQV

Assume the adversary $\mathcal{M}$ can distinguish the session key from a random key and break the seCK-security with probability greater than $\frac{1}{2} + p(\lambda)$, where $p(\lambda)$ is non-negligible. Then we use $\mathcal{M}$ to construct another attacker $\mathcal{S}$ to solve the CDH problem. Let (U,V) be a random CDH instance. We define the following events:

- Succ, which occurs if $\mathcal{M}$ correctly guesses the private bit β involved in Test query. That is equivalent to say $\mathcal{M}$ wins the distinguishing game.
- $\mathsf{AskH2}$, which occurs if $\mathcal{M}$ queries H_2 with $(\sigma_1, \sigma_2, \hat{A}, \hat{B}, X, Y)$.

Without querying H_2, $\mathcal{M}$ can succeed no better than flipping a coin. Hence $\Pr[\mathsf{Succ} \wedge \overline{\mathsf{AskH2}}] \leq \frac{1}{2}$ and

$$\Pr[\mathsf{Succ}] = \Pr[\mathsf{Succ} \wedge \mathsf{AskH2}] + \Pr[\mathsf{Succ} \wedge \overline{\mathsf{AskH2}}] \leq \Pr[\mathsf{Succ} \wedge \mathsf{AskH2}] + \frac{1}{2}$$

whence $\Pr[\mathsf{Succ} \wedge \mathsf{AskH2}] \geq p(\lambda)$. We denote $\mathsf{Succ} \wedge \mathsf{AskH2}$ by SA.

For the different implementation approaches in the seCK model, we define the events:

- I_1, which occurs if both $\hat{A}$ and $\hat{B}$ follow Approach 1;
- I_2, which occurs if both $\hat{A}$ and $\hat{B}$ follow Approach 2;

- I_{1+2}, which occurs if $\hat{A}$ and $\hat{B}$ follow different implementation approaches.

Define $\mathsf{SA}_1 \triangleq \mathsf{SA} \wedge I_1$, $\mathsf{SA}_2 \triangleq \mathsf{SA} \wedge I_2$ and $\mathsf{SA}_{1+2} \triangleq \mathsf{SA} \wedge I_{1+2}$. If event SA occurs with non-negligible probability, then at least one event from the set $\{\mathsf{SA}_1, \mathsf{SA}_2, \mathsf{SA}_{1+2}\}$ occurs with non-negligible probability. In the remainder of the security proof, we will analyze these complementary events and have a patch concerning on a special kind of attackers in [23].

Assume *sid* is the test session with owner $\hat{A}$ and communication peer $\hat{B}$. We assume $\hat{A}$ and $\hat{B}$ are distinct. If $\hat{A}$ and $\hat{B}$ are the same (for the reflection attack), one can modify the following reduction to solve the square CDH problem [1] without the DDH oracle. The square CDH problem is equivalent to the CDH problem in the prime order cyclic group.

The following convention will be used in the security argument. $\xi : \mathbb{G}^* \times \mathbb{G}^* \to \mathbb{G}^*$ is a random function known only to the simulator $\mathcal{S}$, such that $\xi(X,Y) = \xi(Y,X)$ for all $X, Y \in \mathbb{G}^*$. $\mathcal{S}$, which simulates $\mathcal{M}$'s environment, will use $\xi(X,Y)$ to "represent" $\mathsf{CDH}(X,Y)$ in the situation where $\mathcal{S}$ may not know $\mathsf{DLOG}(X)$ and $\mathsf{DLOG}(Y)$. Except with negligible probability, $\mathcal{M}$ will not detect that $\xi(X,Y)$ is being used instead of $\mathsf{CDH}(X,Y)$. A naive way to instantiate ξ is to choose a random element Z in $\mathbb{G}^*$ as the output of $\xi(X,Y)$.

B.1 Event SA_1

By I_1 and the freshness of *sid*, we consider the following complementary cases:

1. There exists session sid^* matching to the test session *sid* and the adversary $\mathcal{M}$ does not issue $\mathsf{EphemeralKeyReveal}(sid^*)$; and either of the following:
   (a) $\mathcal{M}$ does not issue $\mathsf{StaticKeyReveal}(\hat{A})$ - Event E_{1a}.
   (b) $\mathcal{M}$ does not issue $\mathsf{EphemeralKeyReveal}(sid)$ - Event E_{1b}.
2. $\mathcal{M}$ issues $\mathsf{EphemeralKeyReveal}(sid^*)$ (if the matching session sid^* exists), but does not issue $\mathsf{StaticKeyReveal}(\hat{B})$; and either of the following:
   (a) $\mathcal{M}$ does not issue $\mathsf{StaticKeyReveal}(\hat{A})$ - Event E_{2a}.
   (b) $\mathcal{M}$ does not issue $\mathsf{EphemeralKeyReveal}(sid)$ - Event E_{2b}.

Event $E_{1a} \wedge \mathsf{SA}_1$. Assume that $\mathcal{M}$ always selects a test session *sid* with the owner $\hat{A}$ for which the matching session sid^* exists. In this case, $\mathcal{S}$ chooses $r, s \xleftarrow{\$} \mathbb{Z}_q^*$ uniformly at random, and assigns the static public key $A_1 = U$, $A_2 = g^s/U^r$ for party $\hat{A}$, and selects random static key pairs for the remaining $n(\lambda) - 1$ parties. The random oracles H_1 and H_2 are simulated as usual. The protocol execution and all the oracle queries are simulated as the definition of TMQV. For the simulation of *sid* and sid^*, by modifying the simulation of Send queries, $\mathcal{S}$ embeds V into the ephemeral public key as $Y = V$, and computes $\sigma_1 = \xi(XA_1^d, YB_2^e)$ and $\sigma_2 = \xi(XA_2^d, YB_1^e)$. The remainder of sessions *sid* and sid^* is simulated as the protocol definition.

This simulation is perfect except with negligible probability. The adversary $\mathcal{M}$ can learn $(A_1, A_2) \neq (g^{a_1}, g^{a_2})$ or $Y \neq g^y$ or $(\sigma_1, \sigma_2) \neq (\mathsf{CDH}(XA_1^d, YB_2^e), \mathsf{CDH}(XA_2^d, YB_1^e))$ with negligible probability, since the definitions of E_{1a} and ξ.

With probability at least $1/s(\lambda)^2$ $\mathcal{M}$ picks *sid* and sid^* as the test session and its matching session, and with probability $1/n(\lambda)$ $\mathcal{M}$ selects $\hat{A}$ as the owner of *sid*. If $\mathcal{M}$ wins the distinguishing game, $\mathcal{S}$ can solve the CDH problem as follows: for each item $(\sigma_1, \sigma_2, \hat{A}, \hat{B}, X, Y)$ in the random oracle list of H_2, $\mathcal{S}$ computes $Z_1 = (\sigma_1 \cdot Y^{-x} \cdot g^{-eb_2x} \cdot A_1^{-eb_2d})^{d^{-1}}$ and $Z_2 = (\sigma_2 \cdot Y^{-x} \cdot g^{-eb_1x} \cdot A_2^{-eb_1d})^{d^{-1}}$ (where it is feasible

to compute N-th residues in $\mathbb{Z}_q^*$ [3, 18]); and if $Z_1^r Z_2 = V^s$ then $\mathsf{CDH}(U,V) = Z_1$. With probability $1/q$, $\mathcal{S}$ will fail, since the trapdoor test theorem [5] states that $Z_1^r Z_2$ is equal to V^s even if $Z_1 \neq V^{\mathsf{DLOG}(A_1)}$ and $Z_2 \neq V^{\mathsf{DLOG}(A_2)}$ with probability $1/q$. Thus, $\mathcal{S}$ can solve the CDH problem with probability

$$\Pr[\mathcal{S}] \geq \frac{1}{n(\lambda)s(\lambda)^2} p(\lambda) - \frac{1}{q} \tag{1}$$

In the following parts of our proof, we exclude the negligible error probability $1/q$ and rewrite (1) as

$$\Pr[\mathcal{S}] \geq \frac{1}{n(\lambda)s(\lambda)^2} p(\lambda) \tag{2}$$

Event $E_{1b} \wedge \mathsf{SA}_1$. Assume that $\mathcal{M}$ always selects a test session *sid* for which the matching session *sid** exists. In this case, $\mathcal{S}$ simulates the protocol execution and oracle queries as the description of TMQV except for sessions *sid* and *sid**. For the simulation of *sid* and *sid**, $\mathcal{S}$ embeds (U,V) into the ephemeral keys as $X = U$ and $Y = V$, and computes $\sigma_1 = \xi(XA_1^d, YB_2^e)$ and $\sigma_2 = \xi(XA_2^d, YB_1^e)$. This simulation is perfect except with negligible probability by E_{1b} and ξ.

With probability at least $1/s(\lambda)^2$ $\mathcal{M}$ picks *sid* and *sid** as the test session and its matching session. During the simulation, if $\mathcal{M}$ wins the distinguishing game, $\mathcal{S}$ can solve the CDH problem as follows: for item $(\sigma_1, \sigma_2, \hat{A}, \hat{B}, X, Y)$ (such item must be in the list), $\mathcal{S}$ computes $\mathsf{CDH}(U,V) = \sigma_1 \cdot Y^{-a_1 d} \cdot X^{-eb_2} \cdot g^{-eb_2 d a_1}$. Thus, $\mathcal{S}$ can solve the CDH problem with probability

$$\Pr[\mathcal{S}] \geq \frac{1}{s(\lambda)^2 h_2(\lambda)} \cdot p(\lambda) \tag{3}$$

Event $E_{2a} \wedge \mathsf{SA}_1$. Assume that $\mathcal{M}$ always selects a test session *sid* with the owner $\hat{A}$ whose communication peer is $\hat{B}$. In this case, $\mathcal{S}$ chooses $\theta, \phi \xleftarrow{\$} \mathbb{Z}_q^*$ uniformly at random, and assigns the static public key $A_1 = U^\theta$, $A_2 = U^\phi$ for party $\hat{A}$, and chooses $r, s \xleftarrow{\$} \mathbb{Z}_q^*$ uniformly at random, and assigns static public key $B_1 = V$, $B_2 = g^s / V^r$ for $\hat{B}$, and selects random static key pairs for the remaining $n(\lambda) - 2$ parties. For *sid* and *sid**, $\mathcal{S}$ computes $\sigma_1 = \xi(XA_1^d, YB_2^e)$ and $\sigma_2 = \xi(XA_2^d, YB_1^e)$ where Y may not be computed by $\mathcal{S}$ and thus $\mathsf{DLOG}(Y)$ is not known to $\mathcal{S}$. This simulation is perfect except with negligible probability by E_{2a} and ξ.

With probability at least $1/s(\lambda)$ $\mathcal{M}$ picks *sid* as the test session, and with probability at least $1/n(\lambda)^2$ $\mathcal{M}$ chooses $\hat{A}$ as the owner of *sid* and $\hat{B}$ as the communication peer of $\hat{A}$. If $\mathcal{M}$ wins the distinguishing game, $\mathcal{S}$ can solve the CDH problem as follows: for each item $(\sigma_1, \sigma_2, \hat{A}, \hat{B}, X, Y)$, $\mathcal{S}$ computes

$$\Pi_1 = \sigma_2 \cdot Y^{-x} \cdot B_1^{-ex}, \text{ and } \Pi_2 = \sigma_1 \cdot Y^{-x} \cdot B_2^{-ex}.$$

Without the knowledge of $\mathsf{DLOG}(Y)$, $\mathcal{S}$ is unable to compute $Z_1 = \mathsf{CDH}(U, B_1)$ and $Z_2 = \mathsf{CDH}(U, B_2)$ from Π_1 and Π_2 and present further analysis using the trapdoor test theorem. Following the forking technique [17], $\mathcal{S}$ runs $\mathcal{M}$ twice on the same input and coin flips, but only modifies the responses to the $H_1(\mathcal{R}, \hat{B}, \hat{A}, Y, X)$ queries. In $\mathcal{M}$'s first run, he must have queried $H_1(\mathcal{R}, \hat{B}, \hat{A}, Y, X)$ (where the response is a random value e), otherwise it is infeasible for $\mathcal{M}$ to compute (σ_1, σ_2); for $\mathcal{M}$'s second run, $\mathcal{S}$ selects a random value $e' \neq e$ uniformly and answers $H_1(\mathcal{R}, \hat{B}, \hat{A}, Y, X)$ with e' and the response to $H_1(\mathcal{I}, \hat{A}, \hat{B}, X, Y)$ is the same as the first run. If $\mathcal{M}$ succeeds in the second run, $\mathcal{S}$ computes

$$\Pi_1' = \sigma_2' \cdot Y^{-x} \cdot B_1^{-e'x}, \text{ and } \Pi_2' = \sigma_1' \cdot Y^{-x} \cdot B_2^{-e'x}$$

and figures out

$$Z_1 = \left(\frac{\Pi_1}{\Pi_1'}\right)^{((e-e')d\phi)^{-1}}, \text{ and } Z_2 = \left(\frac{\Pi_2}{\Pi_2'}\right)^{((e-e')d\theta)^{-1}}$$

By the trapdoor test theorem, if $Z_1^r \cdot Z_2 = U^s$ then $\mathsf{CDH}(U,V) = Z_1$. Thus, $\mathcal{S}$ can solve the CDH problem with probability

$$\Pr[\mathcal{S}] \geq \frac{1}{n(\lambda)^2 s(\lambda)} \cdot \frac{C}{h_1(\lambda)} p(\lambda) \tag{4}$$

where C is a constant arising from the use of the forking lemma.

Event $E_{2b} \wedge \mathsf{SA}_1$. Assume that $\mathcal{M}$ always selects a test session *sid* with the peer $\hat{B}$. In this case, $\mathcal{S}$ selects $r,s \xleftarrow{\$} \mathbb{Z}_q^*$, and assigns the static public key $B_1 = V$, $B_2 = g^s/V^r$ for party $\hat{B}$, and selects random static key pairs for the remaining $n(\lambda)-1$ parties. For *sid* and *sid**, $\mathcal{S}$ embeds U into the ephemeral public key as $X = U$, and computes $\sigma_1 = \xi(XA_1^d, YB_2^e)$ and $\sigma_2 = \xi(XA_2^d, YB_1^e)$. This simulation is perfect except with negligible probability by E_{2b} and ξ.

With probability at least $1/s(\lambda)$ $\mathcal{M}$ picks *sid* as the test session, and with probability at least $1/n(\lambda)$ $\mathcal{M}$ chooses $\hat{B}$ as the peer of the test session. If $\mathcal{M}$ wins the distinguishing game, $\mathcal{S}$ can solve the CDH problem as follows: for each item $(\sigma_1, \sigma_2, \hat{A}, \hat{B}, X, Y)$, $\mathcal{S}$ computes

$$\Pi_1 = \sigma_2 \cdot Y^{-da_2} \cdot B_1^{-eda_2}, \text{ and } \Pi_2 = \sigma_1 \cdot Y^{-da_1} \cdot B_2^{-eda_1}.$$

Similar to Event $E_{2a} \wedge \mathsf{SA}$, following the forking technique, $\mathcal{S}$ runs $\mathcal{M}$ twice on the same input and coin flips, but only modifies the responses to the $H_1(\mathcal{R}, \hat{B}, \hat{A}, Y, X)$ queries. In $\mathcal{M}$'s first run, the response to $H_1(\mathcal{R}, \hat{B}, \hat{A}, Y, X)$ is e; and in $\mathcal{M}$'s second run, $\mathcal{S}$ answers $H_1(\mathcal{R}, \hat{B}, \hat{A}, Y, X)$ with another $e' \neq e$, and the response to $H_1(\mathcal{I}, \hat{A}, \hat{B}, X, Y)$ is the same as the first run. If $\mathcal{M}$ succeeds in the second run, $\mathcal{S}$ computes

$$\Pi_1' = \sigma_2' \cdot Y^{-da_2} \cdot B_1^{-e'da_2}, \text{ and } \Pi_2' = \sigma_1' \cdot Y^{-da_1} \cdot B_2^{-e'da_1}$$

and figures out

$$Z_1 = (\frac{\Pi_1}{\Pi_1'})^{(e-e')^{-1}}, \text{ and } Z_2 = (\frac{\Pi_2}{\Pi_2'})^{(e-e')^{-1}}.$$

By the trapdoor test theorem, if $Z_1^r Z_2 = X^s$ then $\mathsf{CDH}(U,V) = Z_1$. Thus, $\mathcal{S}$ can solve the CDH problem with probability

$$\Pr[\mathcal{S}] \geq \frac{1}{n(\lambda) s(\lambda)} \cdot \frac{C}{h_1(\lambda)} p(\lambda) \tag{5}$$

B.2 Event SA_2

For this event, we only need to simulate two complementary cases: *sid* has the matching session *sid** (denoted by E_m) or not (denoted by E_{nm}).

Event $E_m \wedge \mathsf{SA}_2$. The simulation is the same as $E_{1b} \wedge \mathsf{SA}_1$ where $X = U$ and $Y = V$. By I_2 and the freshness of *sid*, the adversary $\mathcal{M}$ can not learn the ephemeral private key of a session, and $\mathcal{M}$ can not issue the $\mathsf{InterReveal}$ query to learn the shared secrets and their exponents of *sid* and *sid**. Thus, the simulation is perfect, and we have the success probability of $\mathcal{S}$ is the same as that of $E_{1b} \wedge \mathsf{SA}_1$.

Event $E_{nm} \wedge \mathsf{SA}_2$. $\mathcal{M}$ can not issue $\mathsf{StaticKeyReveal}(\hat{B})$ by the freshness of *sid*, and can not learn the ephemeral private key of any session by I_2. Thus, the simulation and the success probability of $\mathcal{S}$ are the same as those of $E_{2b} \wedge \mathsf{SA}_1$.

B.3 Event SA_{1+2}

We are also interested in whether *sid* has a matching session *sid** or not.

1. Issue $\mathsf{Send}(\hat{B},\hat{P}_i)$ to active $\hat{B}$ as an initiator to communicate with $\hat{P}_i$ ($\hat{P}_i \neq \hat{A}$) and obtain Y_i;
2. Issue an arbitrary number of H_1 oracle queries on $(\mathcal{I},\hat{B},\hat{P}_i,Y_i,Z_j)$, and record the answers. Denote $\{Z_j\}$ the set of queried Z_j;
3. Choose $Z'_j \xleftarrow{\$} \{Z_j\}$ and issue $\mathsf{Send}(sid,\hat{B},\hat{P}_i,Y_i,Z'_j)$ where session sid is owned by $\hat{B}$;
4. Issue $\mathsf{InterReveal}(sid)$.

Figure 2: Attacking Queries in [23].

Event $E_m \wedge \mathsf{SA}_{1+2}$. We suppose that $\hat{A}$ follows Approach 1. As sid^* exists, from any polynomial time algorithm which succeeds in $E_m \wedge \mathsf{SA}_{1+2}$ when $\hat{A}$ follows Approach 1, one can derive a polynomial time machine which succeeds with the same probability when $\hat{A}$ follows Approach 2.

For Approach 2, $\mathcal{M}$ can not learn the ephemeral private key of any session in $\hat{B}$. We embed $Y = V$. By the freshness of sid, we have $\mathcal{M}$ (i) does not issue $\mathsf{StaticKeyReveal}(\hat{A})$ or (ii) does not issue $\mathsf{EphemeralKeyReveal}(sid)$.

The simulation and the success probability of (i) are the same as those of $E_{1a} \wedge \mathsf{SA}_1$; and the simulation and the success probability of (ii) are the same as those of $E_{1b} \wedge \mathsf{SA}_1$.

Event $E_{nm} \wedge \mathsf{SA}_{1+2}$. By the freshness of sid,

- If $\hat{A}$ follows Approach 1 and $\mathcal{M}$ does not issue $\mathsf{StaticKeyReveal}(\hat{A})$, then $\mathcal{M}$ can not issue $\mathsf{StaticKeyReveal}(\hat{B})$ and the simulation and the success probability are the same as those of $E_{2a} \wedge \mathsf{SA}_1$.
- If $\hat{A}$ follows Approach 1 and $\mathcal{M}$ does not issue $\mathsf{EphemeralKeyReveal}(sid)$, then the simulation and the success probability are the same as those of $E_{2b} \wedge \mathsf{SA}_1$.
- If $\hat{A}$ follows Approach 2, $\mathcal{M}$ can not learn the ephemeral private keys and the shared secrets of the sessions in $\hat{A}$. Then $\mathcal{S}$ embeds $X = U$. $\mathcal{M}$ does not issue $\mathsf{StaticKeyReveal}(\hat{B})$. Thus, the simulation and the success probability are the same as those of $E_{2b} \wedge \mathsf{SA}_1$.

B.4 A Patch on SA_2 and SA_{1+2}

We are aware of the proof attackers shown in [23] will make the simulation for SA_2 and SA_{1+2} imperfect, where the $\mathsf{InterReveal}$ query is available to the attackers. We summarize these attackers as Fig.2. For the completeness of the proof, a patch is presented in the following to bound the success probability of this kind of attackers to be negligible.

Let $\mathcal{PM}$ be an adversary to break seCK-security of TMQV in SA_2 or SA_{1+2} which issues the queries in Fig. 2 at some points of its run. He may execute these queries polynomial many times. Let polynomial m be an upper bound on the number of Z_j chosen at Step 2. For simplicity, we assume $\mathcal{PM}$ always chooses m Z_j at Step 2.

1. The same as Step 1 in Fig. 2;
2. Issue an arbitrary number of H_1 oracle queries on $(\mathcal{I}, \hat{B}, \hat{P}_i, Y_i, Z_{l,j})$ for every $Z_{l,j} \in \{Z_{l,1}, \cdots, Z_{l,m}\}$, and record the answers;
3. Issue $\mathsf{Send}(sid, \hat{B}, \hat{P}_i, Y_i, Z_{l,i_l})$;
4. Issue $\mathsf{InterReveal}(sid)$.

Figure 3: Modified Attacking Queries.

For a $\mathcal{PM}$, let $\mathcal{PM}^*$ be an attacker perform the same as $\mathcal{PM}$, but with an input vector

$$\mathbf{v} = \begin{pmatrix} i_1 & Z_{1,1} & Z_{1,2} & \cdots & Z_{1,m} \\ i_2 & Z_{2,1} & Z_{2,2} & \cdots & Z_{2,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ i_k & Z_{k,1} & Z_{k,2} & \cdots & Z_{k,m} \end{pmatrix},$$

where k denotes the number of $\mathcal{PM}$'s runs of the sequence shown in Fig. 2, and $(Z_{l,1}, \cdots, Z_{l,m})$ denotes the queried random group elements chosen by $\mathcal{PM}$ in Step 2 of his l-th run ($l \in [1,k]$), and Z_{l,i_l} denotes the group element chosen by $\mathcal{PM}$ in Step 3 of his l-th run, and using the modified attacking sequence in Fig. 3. Notice that $\mathcal{PM}^*$ is polynomial if $\mathcal{PM}$ is.

Let $\mathbf{V}$ be the set of input vectors. For $\mathbf{v} \in \mathbf{V}$, we say that $\mathcal{PM}^*(\mathbf{v})$ matches $\mathcal{PM}$ if, for all $l \in [1,k]$, $\mathcal{PM}$ executes attacking queries in Fig. 2 and chooses $\{Z_{l,1}, \cdots, Z_{l,m}\}$ in Step 2 and uses Z_{l,i_l} in Step 3. In this case, $\Pr[\mathcal{PM} \text{ succeeds}] = \Pr[\mathcal{PM}^*(\mathbf{v}) \text{ succeeds}]$.

We say $\mathbf{v} \in \mathbf{V}$ possible if there is nonzero probability that $\mathcal{PM}^*(\mathbf{v})$ matches $\mathcal{PM}$. Let $Poss(\mathbf{V})$ be the set of these possible vectors. One can see the success probability of $\mathcal{PM}$ is bounded by

$$\Pr[\mathcal{PM} \text{ succeeds}] \leq \max_{\mathbf{v} \in Poss(\mathbf{V})} \{\Pr[\mathcal{PM}^*(\mathbf{v}) \text{ succeeds}]\}$$

We prove $\Pr[\mathcal{PM}^*(\mathbf{v}) \text{ succeeds}]$ is negligible for all $\mathbf{v} \in Poss(\mathbf{V})$. To this end, we modify the simulation of $\mathsf{Send}(\hat{B}, \hat{P}_i)$ for the l-th run as following:

- Select $s_{1b}, s_{2b} \xleftarrow{\$} \mathbb{Z}_q^*$ and $d \xleftarrow{\$} \{0,1\}^{|q|/2}$ uniformly at random, and compute $y_i = \frac{rs_{1b} + s_{2b} - ds}{r+1}$ and $Y_i = g^{y_i}$, and embed d in H_1 oracle list as $d = H_1(\mathcal{I}, \hat{B}, \hat{P}_i, Y_i, Z_{l,i_l})$.

Now when $\mathsf{InterReveal}(\mathcal{I}, \hat{B}, \hat{P}_i, Y_i, Z_{l,i_l})$ is issued, we can return the correct (s_{1b}, s_{2b}) pre-computed in the modified simulation of $\mathsf{Send}(\hat{B}, \hat{P}_i)$ such that the relation $(Y_i = g^{s_{1b}}/B_1^d) \wedge (Y_i = g^{s_{2b}}/B_2^d)$ always holds. The other parts of simulation are the same as SA_2 and SA_{1+2} respectively. $\Pr[\mathcal{PM}^*(\mathbf{v}) \text{ succeeds}]$ is the same as the probability concluded in SA_2 and SA_{1+2}, and $\Pr[\mathcal{PM} \text{ succeeds}]$ is upper bounded by the negligible probability. Thus, the proof of SA_2 and SA_{1+2} is consistent, after the patch is applied.

B.5 Overall Analysis

By the analysis shown in B.1 to B.4, the success probability of $\mathcal{S}$ is

$$\mathbf{Succ}_{\mathbb{G}}^{cdh}(\mathcal{S}) \geq \max\{\frac{1}{n(\lambda)s(\lambda)^2}, \frac{1}{s(\lambda)^2 h_2(\lambda)}, \frac{C}{n(\lambda)^2 s(\lambda) h_1(\lambda)}, \frac{C}{n(\lambda)s(\lambda)h_1(\lambda)}\}\mathbf{Adv}_{\text{TMQV}}^{seCK}(\mathcal{M})$$

where $\mathbf{Adv}^{seCK}_{\mathrm{TMQV}}(\mathcal{M})$ is the advantage of $\mathcal{M}$ to attack the seCK-security of TMQV. The running time of $\mathcal{S}$ is polynomial if that of $\mathcal{M}$ is polynomial.

Moreover, if $\mathbf{Adv}^{seCK}_{\mathrm{TMQV}}(\mathcal{M})$ is non-negligible then $\mathbf{Succ}^{cdh}_{\mathbb{G}}(\mathcal{S})$ is also non-negligible, which contradicts the CDH assumption. Therefore no polynomially bounded adversary succeeds in breaking seCK-security of TMQV with non-negligible probability.

Jiaxin Pan is a master candidate in Computer Science. His research interests include cryptography and network security.

Libin Wang received his PhD degree from Shanghai Jiaotong University. He is an associate professor of South China Normal University. His research interests include cryptography and network security.