

# Strongly Secure Authenticated Key Exchange without NAXOS' approach

Minkyu Kim<sup>1</sup> \*, Atsushi Fujioka<sup>2</sup>, and Berkant Ustaoglu<sup>2</sup>

<sup>1</sup> ISaC and Department of Mathematical Sciences  
Seoul National University, Seoul 151-747, Korea  
minkyu97@snu.ac.kr

<sup>2</sup> NTT Information Sharing Platform Laboratories  
3-9-11 Midori-cho Musashino-shi Tokyo 180-8585, Japan  
{fujioka.atsushi, ustaoglu.berkant}@lab.ntt.co.jp

**Abstract.** LaMacchia, Lauter and Mityagin [15] proposed the extended Canetti-Krawczyk (eCK) model and an AKE protocol, called NAXOS. Unlike previous security models, the adversary in the eCK model is allowed to obtain ephemeral secret information related to the test session, which makes the security proof difficult. To overcome this NAXOS combines an ephemeral private key  $x$  with a static private key  $a$  to generate an ephemeral public key  $X$ ; more precisely  $X = g^{H(x,a)}$ . As a result, no one is able to query the discrete logarithm of  $X$  without knowing both the ephemeral and static private keys. In other words, the discrete logarithm of an ephemeral public key, which is typically the ephemeral secret, is hidden via an additional random oracle.

In this paper, we show that it is possible to construct eCK-secure protocol without the NAXOS' approach by proposing two eCK-secure protocols. One is secure under the GDH assumption and the other under the CDH assumption; their efficiency and security assurances are comparable to the well-known HMQV [12] protocol. Furthermore, they are at least as secure as protocols that use the NAXOS' approach but unlike them and HMQV, the use of the random oracle is minimized and restricted to the key derivation function.

**Keywords:** AKE, eCK model, NAXOS' approach, trapdoor test

## 1 Introduction

Using key exchange two parties can establish a common secret, called a session key, via a public communication channel. Diffie and Hellman [10] proposed the first key exchange protocol in which two parties exchange  $X = g^x$ ,  $Y = g^y$  and derive a session key from  $g^{xy} = Y^x = X^y$ . The original Diffie-Hellman (DH) protocol does not provide authentication and is vulnerable to active person-in-the-middle attacks. A key exchange protocol is authenticated key exchange

---

\* This work was done while the first author was visiting NTT Information Sharing Platform Laboratories. He was partially supported by NAP of Korea Research Council of Fundamental Science and Technology.

(AKE) if both parties are assured that only their intended peers can derive the session key.

Bellare and Rogaway [5] proposed the first security model and definition for authenticated key exchange that allows a rigorous analysis. Their model is indistinguishability based, where an adversary is required to differentiate between a random key and a session key. There have been several variations to the Bellare-Rogaway model and until recently, the Canetti-Krawczyk [8] (CK) model was regarded as one of the most significant BR modifications.

The CK model, however, fails to capture some desirable AKE properties. CK-secure protocols may still be vulnerable to key compromise impersonation (KCI) attack or may not be resilient to the leakage of ephemeral private keys (LEP). Resilience to LEP is motivated by scenarios where the session specific information is stored in an insecure place or the random number generator used by a party is corrupt.

To bring these attacks and properties within the scope of analysis, LaMacchia, Lauter and Mityagin changed the CK model, to the so called eCK model, and proposed NAXOS as an example of an eCK-secure protocol. Informally, the eCK aims to allow all adversary queries, except those that trivially break AKE protocols. In particular the eCK adversary is allowed to obtain ephemeral secret information related to the test session, which makes the security arguments in the eCK model difficult. To achieve eCK security, NAXOS requires that the ephemeral public key  $X$  be computed from an exponent made up by hashing an ephemeral private key  $x$  and the static private key  $a$ , more precisely,  $X = g^{H(x,a)}$  instead of  $X = g^x$ . In this paper generating ephemeral public key as  $X = g^{H(x,a)}$  is called *NAXOS' approach*. In NAXOS' approach no one is able to query the discrete logarithm of an ephemeral public key  $X$  without the pair  $(x, a)$ ; thus the discrete logarithm of  $X$  is hidden via an extra random oracle. Using NAXOS' approach many protocols [25, 11, 16, 17] were argued secure in the eCK model under the random oracle assumption. In the standard model, the only (to our knowledge) eCK-secure protocol is due to Okamoto [22]; it uses pseudo-random functions instead of hash functions.

In this paper, we show that it is possible to construct eCK-secure AKE protocols without NAXOS' approach by giving two example protocols. Protocol 1 relies on the Gap Diffie-Hellman and the random oracle assumptions. Protocol 2 is derived by applying the trapdoor technique introduced by Cash, Kiltz and Shoup [9] to Protocol 1, and thus uses Computational Diffie-Hellman assumption instead of the gap assumption.

Our protocols provide no less security assurances than protocols utilizing the NAXOS' approach in the sense that our analysis considers leakage of the discrete logarithm of ephemeral public keys. One advantage of this method (see [26]) is to reduce the risk of leaking the static private key, since the derivation of the ephemeral public key is independent from the static private key. This is in contrast to protocols that use the NAXOS' approach. In addition, unlike other eCK secure protocols and HMQV, which require at least two random oracles,

we minimize the use of the random oracle, by applying it only to the session key derivation.

**Organization.** In section 2, we recall the security assumptions and the trapdoor test, which we use in this paper. In section 3, we briefly outline the eCK model and then propose our new protocols with security arguments in sections 4 and 5. In section 6 we compare our protocols with other relevant protocols and conclude in section 7.

## 2 Preliminaries

Let  $G$  be a cyclic group of prime order  $q$  and generator  $g$ . Let  $\text{dl}_g : G \rightarrow \mathbf{Z}_q$  be the discrete logarithm (DL) function which takes an input  $X \in G$  and returns  $x \in \mathbf{Z}_q$  such that  $X = g^x$ . Define the computational Diffie-Hellman (CDH) function  $\text{dh}_g : G^2 \rightarrow G$  as  $\text{dh}_g(X, Y) = g^{\text{dl}_g(X)\text{dl}_g(Y)}$ , and the corresponding decisional predicate  $\text{ddh}_g : G^3 \rightarrow \{0, 1\}$  as a function which takes an input  $(X, Y, Z) \in G^3$  and returns 1 if  $Z = \text{dh}_g(X, Y)$  and 0 otherwise.

The advantage of an algorithm  $\mathcal{S}$  in solving the CDH problem,  $\text{Adv}^{\text{CDH}}(\mathcal{S})$ , is the probability that, given input  $X, Y$  selected uniformly at random from  $G$ ,  $\mathcal{S}$  returns  $\text{dh}_g(X, Y)$ . Similarly, the advantage of an algorithm  $\mathcal{S}$  in solving the Gap Diffie-Hellman (GDH) problem,  $\text{Adv}^{\text{GDH}}(\mathcal{S})$ , is the probability that, given input  $X, Y$  selected uniformly at random in  $G$  and oracle access to  $\text{ddh}_g(\cdot, \cdot, \cdot)$ ,  $\mathcal{S}$  returns  $\text{dh}_g(X, Y)$ .

We say that  $G$  satisfy the CDH (resp. GDH) assumption if no probabilistic polynomial-time bounded algorithm can solve the CDH (resp. GDH) problem on  $G$  with non-negligible advantage.

In the security argument of Protocol 2 we will use the following theorem, called the trapdoor test, (see [9] for theorem details).

**Theorem 1 (Trapdoor Test in [9]).** *Let  $G$  be a cyclic group of prime order  $q$ , generated by  $g \in G$ . Suppose  $X_1, r, s$  are mutually independent random variables, where  $X_1$  takes values in  $G$ , and each of  $r, s$  is uniformly distributed over  $\mathbf{Z}_q$ , and define the random variable  $X_2 := g^s / X_1^r$ . Further, suppose that  $\hat{Y}, \hat{Z}_1, \hat{Z}_2$  are random variables taking values in  $G$ , each of which is defined as some function of  $X_1$  and  $X_2$ . Then we have:*

1.  $X_2$  is uniformly distributed over  $G$ ;
2.  $X_1$  and  $X_2$  are independent;
3. if  $X_1 = g^{x_1}$  and  $X_2 = g^{x_2}$ , then the probability that the truth value of

$$\hat{Z}_1^r \hat{Z}_2 \stackrel{?}{=} \hat{Y}^s \tag{1}$$

does not agree with the truth value of

$$\hat{Z}_1 \stackrel{?}{=} \hat{Y}^{x_1} \wedge \hat{Z}_2 \stackrel{?}{=} \hat{Y}^{x_2} \tag{2}$$

is at most  $1/q$ ; moreover, if (2) holds, then (1) certainly holds.

### 3 Security Model

For further eCK details and explanations see [15].

In the eCK model, each party is a probabilistic polynomial-time Turing machine and is assigned a static public and private key pair together with a certificate that binds party's identity to its public key. We denote a party's identity  $\mathcal{A}, \mathcal{B}, \mathcal{C}, \dots$ <sup>3</sup>. We assume that, the certificate authority (CA) does not require proof of possession of the corresponding private key included in a certificate. However, CA verifies that the public key is in  $G^\times = G - \{\text{id}_G\}$ , where  $\text{id}_G$  is the identity element of  $G$ .

We outline the eCK model for two-pass Diffie-Hellman protocols, where two parties  $\mathcal{A}$  and  $\mathcal{B}$  exchange static and ephemeral public keys and thereafter compute a session key that depends on the exchanged public keys and identities of the parties.

**Session.** An invocation of a protocol is called a *session*. Session activation is made via an incoming message of the forms  $(\mathcal{I}, \mathcal{A}, \mathcal{B})$  or  $(\mathcal{R}, \mathcal{A}, \mathcal{B}, Y)$ . If  $\mathcal{A}$  was activated with  $(\mathcal{I}, \mathcal{A}, \mathcal{B})$ , then  $\mathcal{A}$  is called the session *initiator*; otherwise it is called the session *responder*. After activation, session initiator  $\mathcal{A}$  creates ephemeral public key  $X$  and sends  $(\mathcal{R}, \mathcal{B}, \mathcal{A}, X)$  to the session responder  $\mathcal{B}$ , who then prepares ephemeral public key  $Y$ , computes the session key and sends  $(\mathcal{I}, \mathcal{A}, \mathcal{B}, X, Y)$  to  $\mathcal{A}$ . Upon receiving  $(\mathcal{I}, \mathcal{A}, \mathcal{B}, X, Y)$ ,  $\mathcal{A}$  also computes a session key for the session  $\mathcal{A}$  owns. We say that a session is *completed* if its owner computes a session key.

If  $\mathcal{A}$  is the initiator of a session, the session is identified via  $(\mathcal{I}, \mathcal{A}, \mathcal{B}, X, \times)$  or  $(\mathcal{I}, \mathcal{A}, \mathcal{B}, X, Y)$ . For a responder  $\mathcal{A}$  the session is identified via  $(\mathcal{R}, \mathcal{A}, \mathcal{B}, Y, X)$ . The *matching session* of  $(\mathcal{I}, \mathcal{A}, \mathcal{B}, X, Y)$  is a session with identifier  $(\mathcal{R}, \mathcal{B}, \mathcal{A}, X, Y)$  and vice versa. In the remainder of the paper we will omit  $\mathcal{I}$  and  $\mathcal{R}$  since these "role markers" are implicitly defined from the order of ephemeral public keys.

**Adversary.** The adversary  $\mathcal{M}$  is modeled as a probabilistic Turing machine that controls all communications including session activation, performed via `Send(message)` query. The message has one of the following forms:  $(\text{pid}, \overline{\text{pid}})$ ,  $(\text{pid}, \overline{\text{pid}}, X)$ , or  $(\text{pid}, \overline{\text{pid}}, X, Y)$ , where  $\text{pid}$  and  $\overline{\text{pid}}$  are identities. Each party submits its responses to the adversary, who decides the global delivery order.

The adversary does not have immediate access to a party's private information. However, leakage of private information is captured via the following adversary queries:

- `EphemeralKeyReveal(sid)` The adversary obtains the ephemeral secret key associated with the session  $\text{sid}$ .

---

<sup>3</sup> In the eCK model the adversary selects these identifier strings.

- **SessionKeyReveal**(sid) The adversary obtains the session key for the session sid, provided that the session holds a session key.
- **StaticKeyReveal**(pid) The adversary learns the static secret key of the party pid.
- **EstablishParty**(pid)<sup>4</sup> This query allows the adversary to register a static public key on behalf of a party pid; the adversary totally controls that party. If a party pid is established by **EstablishParty**(pid) query issued by adversary, then we call the party *dishonest*. If not, we call the party *honest*. This query models malicious insider.

To define eCK security we need the following definition.

**Definition 1 (Freshness).** *Let  $\text{sid}^*$  be the session identifier of a completed session, owned by an honest party  $\mathcal{A}$  with peer  $\mathcal{B}$ , who is also honest. If the matching session exists, then let  $\overline{\text{sid}^*}$  be the session identifier of the matching session of  $\text{sid}^*$ . Define  $\text{sid}^*$  to be fresh if none of the following conditions hold:*

1. *Adversary issues a **SessionKeyReveal**( $\text{sid}^*$ ) or **SessionKeyReveal**( $\overline{\text{sid}^*}$ ) query (if  $\text{sid}^*$  exists)*
2.  *$\text{sid}^*$  exists and Adversary makes either of the following queries*
  - *both **StaticKeyReveal**( $\mathcal{A}$ ) and **EphemeralKeyReveal**( $\text{sid}^*$ ), or*
  - *both **StaticKeyReveal**( $\mathcal{B}$ ) and **EphemeralKeyReveal**( $\text{sid}^*$ )*
3.  *$\overline{\text{sid}^*}$  does not exist and Adversary makes either of the following queries*
  - *both **StaticKeyReveal**( $\mathcal{A}$ ) and **EphemeralKeyReveal**( $\text{sid}^*$ ), or*
  - ***StaticKeyReveal**( $\mathcal{B}$ )*

**Security Experiment.** Initially, the adversary  $\mathcal{M}$  is given a set of honest parties, for whom  $\mathcal{M}$  selects identifiers. Then the adversary makes any sequence of the queries described above. During the experiment,  $\mathcal{M}$  makes a special query **Test**( $\text{sid}^*$ ), where  $\text{sid}^*$  is a fresh session, and is given with equal probability either the session key held by  $\text{sid}^*$  or a random key; the query does not terminate the experiment. The experiment continues until  $\mathcal{M}$  makes a guess whether the key is random or not. The adversary *wins* the game if the test session  $\text{sid}^*$  is still fresh and if  $\mathcal{M}$  guess was correct.

**Definition 2 (eCK security).** *The advantage of the adversary  $\mathcal{M}$  in the AKE experiment with AKE protocol  $\Pi$  is defined as*

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

<sup>4</sup> Formally, this query is not available in the eCK model [15], where the adversary is only allowed to select identities of parties and establishes dishonest parties before starting the interaction with the parties. This does not present a deficiency in the model since the query gives the addition power to the adversary to decide (dishonest) party specific information after observing the behavior of honest parties.

We say that an AKE protocol  $\Pi$  is secure in the eCK model 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.
2. For any probabilistic polynomial-time bounded adversary  $\mathcal{M}$ ,  $\text{Adv}_{\Pi}^{\text{AKE}}(\mathcal{M})$  is negligible.

## 4 Protocol 1

In sections 4 and 5, we offer two eCK-secure protocols without NAXOS' approach. The following are parameters used in the protocol descriptions.

**Parameters.** Let  $k/2$  be the security parameter and  $G$  be a cyclic group with generator  $g$  and order a  $k$ -bit prime  $q$ . Let  $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$  be a cryptographic hash function modeled as a random oracle. Party  $\mathcal{A}$ 's static private key is a pair  $a_1, a_2 \in \mathbf{Z}_q^\times$  and his public key is the pair  $A_1 = g^{a_1}, A_2 = g^{a_2} \in G^\times$ . Similarly, the party  $\mathcal{B}$ 's static keys are  $b_1, b_2 \in \mathbf{Z}_q^\times, B_1 = g^{b_1}, B_2 = g^{b_2} \in G^\times$ .

### 4.1 Protocol 1 description

In the description,  $\mathcal{A}$  is the session initiator and  $\mathcal{B}$  is the session responder.

1.  $\mathcal{A}$  chooses at random an ephemeral private key  $x \in \mathbf{Z}_q^\times$ , computes the ephemeral public key  $X = g^x$  and sends  $(\mathcal{B}, \mathcal{A}, X)$  to  $\mathcal{B}$ .
2. Upon receiving  $(\mathcal{B}, \mathcal{A}, X)$ ,  $\mathcal{B}$  verifies that  $X \in G^\times$ . If so,  $\mathcal{B}$  chooses at random an ephemeral private key  $y \in \mathbf{Z}_q^\times$  and computes the ephemeral public key  $Y = g^y$ . After computing the shared secrets  $Z_1 = (XA_1)^{y+b_1}, Z_2 = (XA_2)^{y+b_2}$ , the session key  $\text{SK} = H(Z_1, Z_2, X, Y, \mathcal{A}, \mathcal{B})$  and sending  $(\mathcal{A}, \mathcal{B}, X, Y)$  to  $\mathcal{A}$ ,  $\mathcal{B}$  completes the session with session key SK.
3. Upon receiving  $(\mathcal{A}, \mathcal{B}, X, Y)$ ,  $\mathcal{A}$  checks if he owns a session with session identifier  $(\mathcal{A}, \mathcal{B}, X, \times)$ . If so,  $\mathcal{A}$  verifies that  $Y \in G^\times$  and computes  $Z_1 = (YB_1)^{x+a_1}, Z_2 = (YB_2)^{x+a_2}$  and completes the session  $(\mathcal{A}, \mathcal{B}, X, Y)$  with session key  $\text{SK} = H(Z_1, Z_2, X, Y, \mathcal{A}, \mathcal{B})$ .

Both parties compute the shared secrets  $Z_1 = g^{(x+a_1)(y+b_1)}, Z_2 = g^{(x+a_2)(y+b_2)}$  and therefore compute the same session key SK.

### 4.2 Protocol 1 security argument

**Theorem 2.** *If the GDH assumption holds in  $G$  and  $H$  is a random oracle, then the Protocol 1 is eCK-secure.*

*Proof.* Let  $\mathcal{M}$  be a polynomially bounded adversary against Protocol 1, that runs in time  $t(k)$ , activates at most  $n(k)$  honest parties, at most  $s(k)$  sessions and makes at most  $h(k)$  queries to the oracle  $H$ , where  $t(k)$ ,  $n(k)$ ,  $s(k)$ , and  $h(k)$  are polynomially bounded in  $k$ . Assume also that  $\text{Adv}_{\text{Protocol 1}}^{\text{AKE}}(\mathcal{M})$  is non-negligible. Since  $H$  is modeled as a random oracle, the adversary  $\mathcal{M}$  has only three ways to distinguish a session key of the test session from a random string.

- A1. Guessing attack:  $\mathcal{M}$  correctly guesses the session key.
- A2. Key replication attack:  $\mathcal{M}$  creates a session that is not matching to the test session, but has the same session key as the test session.
- A3. Forging attack:  $\mathcal{M}$  computes  $Z_1$  and  $Z_2$  used in the test session, and queries  $H$  with  $(Z_1, Z_2, X, Y, \mathcal{A}, \mathcal{B})$ .

Since  $H$  is a random oracle, the probability of guessing the output of  $H$  is  $\mathcal{O}(1/2^k)$ , which is negligible. Since non-matching sessions have different communicating parties or ephemeral public keys, key replication is equivalent to finding an  $H$ -collision; therefore the probability, that event A2 occurs, is  $\mathcal{O}(s(k)^2/2^k)$ , which is also negligible. Thus events A1 and A2 can be ruled out.

Let  $M$  be the event that  $\mathcal{M}$  wins the security game,  $H$  be the event that  $\mathcal{M}$  queries  $H$  with  $(Z_1, Z_2, X, Y, \mathcal{A}, \mathcal{B})$ , and  $\bar{H}$  the complementary event of  $H$ . Since  $H$  is a random oracle and events A1 and A2 were ruled out, we have  $\Pr[M|\bar{H}] = \frac{1}{2}$  except with negligible difference. Then

$$\Pr[M] = \Pr[M \wedge H] + \Pr[M|\bar{H}] \Pr[\bar{H}] \leq \Pr[M \wedge H] + \frac{1}{2}$$

$$\text{Adv}_{\text{Protocol 1}}^{\text{AKE}}(\mathcal{M}) \leq \Pr[M \wedge H] = \Pr[A3]$$

Since  $\text{Adv}_{\text{Protocol 1}}^{\text{AKE}}(\mathcal{M})$  is non-negligible,  $\Pr[A3]$  is also non-negligible.

Now, consider the following complementary sub-events of A3.

- E1. A3 occurs and the test session has no matching session.
- E2. A3 occurs and the test session has a matching session.

Then

$$\Pr[A3] = \Pr[E1] + \Pr[E2]$$

Consider also the following sub-events of E1 so that  $E1 = E1_a \vee E1_b$ .

- E1<sub>a</sub>. E1 occurs and  $\mathcal{M}$  does not reveal the ephemeral private key of the owner of the test session, but may query for the static private key of the test session owner.
- E1<sub>b</sub>. E1 occurs and the owner's static private key of the test session has never been revealed by  $\mathcal{M}$ , but may query for the ephemeral private key of the test session owner.

Consider also the following sub-events of E2 so that  $E2 = E2_a \vee E2_b \vee E2_c \vee E2_d$ .

- E2<sub>a</sub>. E2 occurs and  $\mathcal{M}$  does not reveal the ephemeral private keys of both the owner of the test session and its peer, but may query for the static private keys of the test session peers.

- E2<sub>b</sub>. E2 occurs and  $\mathcal{M}$  does not reveal the static private keys of both the test session and its matching session, but may query for the ephemeral private keys of the test session peers.
- E2<sub>c</sub>. E2 occurs and the owner’s ephemeral private key and the peer’s static private key of the test session have never been revealed by  $\mathcal{M}$ , but  $\mathcal{M}$  may query `StaticKeyReveal` with the identity of the test session owner and query `EphemeralKeyReveal` with the session matching to the test session.
- E2<sub>d</sub>. E2 occurs and the owner’s static private key and the peer’s ephemeral private key of the test session have never been revealed by  $\mathcal{M}$ , but  $\mathcal{M}$  may query `StaticKeyReveal` with the identity of the test session peer and `EphemeralKeyReveal` with the test session.

We then have

$$\Pr[\text{E1}] \leq \Pr[\text{E1}_a] + \Pr[\text{E1}_b] \tag{3}$$

$$\Pr[\text{E2}] \leq \Pr[\text{E2}_a] + \Pr[\text{E2}_b] + \Pr[\text{E2}_c] + \Pr[\text{E2}_d]. \tag{4}$$

We will show how to construct a GDH solver  $\mathcal{S}$  that uses a Protocol 1 adversary  $\mathcal{M}$ . The solver  $\mathcal{S}$  is given a CDH instance  $(U, V)$ , where  $U$  and  $V$  are selected uniform randomly in  $G$ , access to a  $\text{ddh}_g(\cdot, \cdot, \cdot)$  oracle and has to compute  $\text{dh}_g(U, V)$ . Without loss of generality in the analysis, we denote the test session owner and peer by  $\mathcal{A}$  and  $\mathcal{B}$ , respectively, and assume that  $\mathcal{A}$  is the initiator.

**Analysis of E1<sub>a</sub>.** We use  $\mathcal{M}$  to construct a GDH solver  $\mathcal{S}$  that succeeds with non-negligible probability provided that event E1<sub>a</sub> occurs.  $\mathcal{S}$  prepares  $n(k)$  honest parties, selects one party  $\mathcal{B}$  to whom  $\mathcal{S}$  assigns static public key  $B_1 = V, B_2 = V^r$ , where  $\mathcal{S}$  randomly chooses  $r \in \mathbf{Z}_q$ . The remaining  $n(k) - 1$  parties are assigned random static public and private key pairs.  $\mathcal{S}$  also chooses a session  $\text{sid}^*$ , owned by an honest party  $\mathcal{A}$ .

When  $\mathcal{M}$  activates sessions between honest peers  $\mathcal{S}$  follows the protocol description. Since  $\mathcal{S}$  knows static private keys of at least one peer, it can respond all queries faithfully. The only exception is the session  $\text{sid}^*$ , for which  $\mathcal{S}$  sets the ephemeral public key of  $\text{sid}^*$  to  $U$ , and chooses a random  $\zeta \in \{0, 1\}^k$  as the session key of  $\text{sid}^*$ .

The simulator has difficulty in responding queries related to  $\mathcal{B}$  because  $\mathcal{S}$  does not know the static private key of  $\mathcal{B}$ . More precisely, for sessions owned by  $\mathcal{B}$  with peer  $\mathcal{C}$  controlled by  $\mathcal{M}$ ,  $\mathcal{S}$  cannot compute shared secrets  $Z_1, Z_2$ , but may have to answer `SessionKeyReveal` queries. Note that  $\mathcal{M}$  can obtain session keys of these session by computing the shared secrets  $Z_1, Z_2$  and query  $H$ . If two values do not coincide, then  $\mathcal{S}$  fails its simulation. To handle this situations,  $\mathcal{S}$  prepares  $\text{R}^{\text{list}}$  with entries of the form  $(\text{pid}, \overline{\text{pid}}, W, W', \text{SK}) \in \{0, 1\}^* \times \{0, 1\}^* \times G^2 \times \{0, 1\}^k$ , which is maintained for consistent responses to  $H$  and `SessionKeyReveal` queries.

We next describe the action of  $\mathcal{S}$  when  $\mathcal{M}$  makes queries related to  $\mathcal{B}$ . In the following,  $Y$  is generated by the party  $\mathcal{B}$ . Recall that if the session identifier is  $(\mathcal{B}, \mathcal{C}, X, Y)$  (resp.  $(\mathcal{B}, \mathcal{C}, Y, X)$ ), then  $\mathcal{B}$  is the session responder (resp. initiator).

- **Send**( $\mathcal{B}, \mathcal{C}$ ):  $\mathcal{S}$  randomly selects  $y \in \mathbf{Z}_q^\times$ , computes  $Y = g^y$ , creates a new session with sid  $(\mathcal{B}, \mathcal{C}, Y, \times)$  and returns  $(\mathcal{C}, \mathcal{B}, Y)$  to  $\mathcal{M}$ .
- **Send**( $\mathcal{B}, \mathcal{C}, X$ ):  $\mathcal{S}$  randomly selects  $y \in \mathbf{Z}_q^\times$ , compute  $Y = g^y$ , creates a new session with sid  $(\mathcal{B}, \mathcal{C}, X, Y)$  and returns  $(\mathcal{C}, \mathcal{B}, X, Y)$  to  $\mathcal{M}$ .
- **Send**( $\mathcal{B}, \mathcal{C}, Y, X$ ):  $\mathcal{S}$  checks if  $\mathcal{B}$  owns a session with sid  $(\mathcal{B}, \mathcal{C}, Y, \times)$ . If not, the session is aborted; otherwise,  $\mathcal{S}$  updates sid to  $(\mathcal{B}, \mathcal{C}, Y, X)$ .
- **H**( $\cdot$ ):  $\mathcal{S}$  maintains an initially empty list  $\mathbf{H}^{\text{list}}$  with entries of the form  $(\hat{Z}_1, \hat{Z}_2, W, W', \text{pid}, \overline{\text{pid}}, \text{SK}) \in G^4 \times \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^k$  and simulates a random oracle in the usual way except for queries of the form  $(\hat{Z}_1, \hat{Z}_2, X, Y, \mathcal{C}, \mathcal{B})$  and  $(\hat{Z}_1, \hat{Z}_2, Y, X, \mathcal{B}, \mathcal{C})$ . When  $(\hat{Z}_1, \hat{Z}_2, X, Y, \mathcal{C}, \mathcal{B})$  (resp.  $(\hat{Z}_1, \hat{Z}_2, Y, X, \mathcal{B}, \mathcal{C})$ ) is queried,  $\mathcal{S}$  does one of the following.
  1. If  $(\hat{Z}_1, \hat{Z}_2, X, Y, \mathcal{C}, \mathcal{B}, \text{SK})$  (resp.  $(\hat{Z}_1, \hat{Z}_2, Y, X, \mathcal{B}, \mathcal{C}, \text{SK})$ )  $\in \mathbf{H}^{\text{list}}$  for some SK, then  $\mathcal{S}$  returns SK to  $\mathcal{M}$ .
  2. Otherwise,  $\mathcal{S}$  checks if there exists  $(\mathcal{B}, \mathcal{C}, X, Y, \text{SK})$  (resp.  $(\mathcal{B}, \mathcal{C}, Y, X, \text{SK})$ )  $\in \mathbf{R}^{\text{list}}$  such that  $\text{ddh}_g(XC_1, YB_1, \hat{Z}_1) = 1$  and  $\text{ddh}_g(XC_2, YB_2, \hat{Z}_2) = 1$ . If such a pair exists,  $\mathcal{S}$  returns SK from  $\mathbf{R}^{\text{list}}$ , and stores the new tuple  $(\hat{Z}_1, \hat{Z}_2, X, Y, \mathcal{C}, \mathcal{B}, \text{SK})$  (resp.  $(\hat{Z}_1, \hat{Z}_2, Y, X, \mathcal{B}, \mathcal{C}, \text{SK})$ ) in  $\mathbf{H}^{\text{list}}$ .
  3. If neither of the above two cases hold, then  $\mathcal{S}$  chooses  $\text{SK} \in \{0, 1\}^k$  at random, returns it to  $\mathcal{M}$  and stores the new tuple  $(\hat{Z}_1, \hat{Z}_2, X, Y, \mathcal{C}, \mathcal{B}, \text{SK})$  (resp.  $(\hat{Z}_1, \hat{Z}_2, Y, X, \mathcal{B}, \mathcal{C}, \text{SK})$ ) in  $\mathbf{H}^{\text{list}}$ .
- **SessionKeyReveal**( $\mathcal{B}, \mathcal{C}, X, Y$ ) or **SessionKeyReveal**( $\mathcal{B}, \mathcal{C}, Y, X$ ):  $\mathcal{S}$  maintains an initially empty list  $\mathbf{R}^{\text{list}}$  with entries of the form  $(\text{pid}, \overline{\text{pid}}, W, W', \text{SK}) \in \{0, 1\}^* \times \{0, 1\}^* \times G^2 \times \{0, 1\}^k$ . When **SessionKeyReveal**( $\mathcal{B}, \mathcal{C}, X, Y$ ) (resp.  $(\mathcal{B}, \mathcal{C}, Y, X)$ ) is queried,  $\mathcal{S}$  does one of the following.
  1. If there is no session with identifier  $(\mathcal{B}, \mathcal{C}, X, Y)$  (resp.  $(\mathcal{B}, \mathcal{C}, Y, X)$ ), the query is aborted.
  2. If  $(\mathcal{B}, \mathcal{C}, X, Y, \text{SK})$  (resp.  $(\mathcal{B}, \mathcal{C}, Y, X, \text{SK})$ )  $\in \mathbf{R}^{\text{list}}$  for some SK,  $\mathcal{S}$  returns SK to  $\mathcal{M}$ .
  3. Otherwise, go through  $\mathbf{H}^{\text{list}}$  to find  $(\hat{Z}_1, \hat{Z}_2, X, Y, \mathcal{C}, \mathcal{B}, \text{SK})$  (resp.  $(\hat{Z}_1, \hat{Z}_2, Y, X, \mathcal{B}, \mathcal{C}, \text{SK})$ ) satisfying  $\text{ddh}_g(XC_1, YB_1, \hat{Z}_1) = 1$  and  $\text{ddh}_g(XC_2, YB_2, \hat{Z}_2) = 1$ . If such a pair exists,  $\mathcal{S}$  returns SK, and stores the new tuple  $(\mathcal{B}, \mathcal{C}, X, Y, \text{SK})$  (resp.  $(\mathcal{B}, \mathcal{C}, Y, X, \text{SK})$ ) in  $\mathbf{R}^{\text{list}}$ .
  4. If none of the above three cases hold, then  $\mathcal{S}$  chooses  $\text{SK} \in \{0, 1\}^k$  at random, returns it to  $\mathcal{M}$  and stores the new tuple  $(\mathcal{B}, \mathcal{C}, X, Y, \text{SK})$  (resp.  $(\mathcal{B}, \mathcal{C}, Y, X, \text{SK})$ ) in  $\mathbf{R}^{\text{list}}$ .
- **EphemeralKeyReveal**( $\cdot$ ):  $\mathcal{S}$  responds to the query faithfully.
- **StaticKeyReveal**( $\mathcal{B}$ ) or **EstablishParty**( $\mathcal{B}$ ):  $\mathcal{S}$  aborts.
- **Test**(sid): If  $\text{sid} \neq \text{sid}^*$ ,  $\mathcal{S}$  aborts. Otherwise,  $\mathcal{S}$  randomly chooses  $\zeta \in \{0, 1\}^k$  and returns it to the adversary  $\mathcal{M}$ .

Provided that event  $E1_a$  occurs and  $\mathcal{M}$  selects the session  $\text{sid}^*$  as the test session with peer  $\mathcal{B}$ , the simulation does not fail; let  $Y$  denote the test session incoming ephemeral public key. If  $\mathcal{M}$  is successful with non-negligible probability it must have queried  $H$  with inputs  $\hat{Z}_1 = (YB_1)^{x^*+a_1}$  and  $\hat{Z}_2 = (YB_2)^{x^*+a_2}$ , where  $x^* \equiv \text{dl}_g(U) \bmod q$ , because  $\mathcal{S}$  sets the ephemeral public key  $X^*$  of  $\text{sid}^*$  as  $U$ . To solve the CDH instance,  $\mathcal{S}$  checks if there is an  $H$  query made by  $\mathcal{M}$  of the form  $(Z_1, Z_2, U, Y, \mathcal{A}, \mathcal{B})$ , such that  $\text{ddh}_g(UA_1, YB_1, Z_1) = 1$  and  $\text{ddh}_g(UA_2, YB_2, Z_2) = 1$ . If such an  $H$  query exists,  $\mathcal{S}$  computes<sup>5</sup>  $Z_1^* = Z_1/(YB_1)^{a_1}$  and  $Z_2^* = Z_2/(YB_2)^{a_2}$ . If  $Z_1, Z_2$  are correct, then since  $B_1 = V$  and  $\text{dl}_g(B_2) \equiv r \cdot \text{dl}_g(B_1) \bmod q$  we have,

$$Z_1^*/Z_2^* = (B_1/B_2)^x = U^{\text{dl}_g(B_1) - \text{dl}_g(B_2)}.$$

Therefore, by computing  $\left(Z_1^*/Z_2^*\right)^{1/(1-r)}$ ,  $\mathcal{S}$  can find  $U^{\text{dl}_g(B_1)} = \text{dh}_g(U, V)$ .

With probability at least  $\frac{1}{s(k)n(k)}$ , the test session is  $\text{sid}^*$  with peer  $\mathcal{B}$ . Thus, the advantage of  $\mathcal{S}$  is

$$\text{Adv}^{\text{GDH}}(\mathcal{S}) \geq \frac{1}{s(k)n(k)} \Pr[E1_a]. \quad (5)$$

**Analysis of  $E1_b$ .**  $\mathcal{S}$  prepares  $n(k)$  honest parties, selects two distinct parties, say  $\mathcal{A}$  and  $\mathcal{B}$ , and assigns  $\mathcal{A}$ 's and  $\mathcal{B}$ 's static public keys as  $A_1 = U, A_2 = U^s$  and  $B_1 = V, B_2 = V^r$ , respectively, where  $r$  and  $s$  are random elements of  $\mathbf{Z}_q$ . The remaining  $n(k) - 2$  parties are assigned random static and private key pairs. If  $\mathcal{M}$  activates sessions owned by any honest party except  $\mathcal{A}$  and  $\mathcal{B}$ , then  $\mathcal{S}$  follows the protocol description. The parties  $\mathcal{A}$  and  $\mathcal{B}$  are simulated as in the case  $E1_a$ .

If  $\mathcal{M}$  selected the session  $\text{sid}^*$  as the test session with owner  $\mathcal{A}$  and peer  $\mathcal{B}$ , this simulation does not fail provided that the event  $E1_b$  occurs. If  $\mathcal{M}$  is successful with non-negligible probability, it must have queried  $H$  with inputs of the form  $Z_1 = (YB_1)^{x+\text{dl}_g(A_1)}$ ,  $Z_2 = (YB_2)^{x+\text{dl}_g(A_2)}$ . To solve CDH,  $\mathcal{S}$  checks if there is an  $H$  query made by  $\mathcal{M}$  of the form  $(Z_1, Z_2, X, Y, \mathcal{A}, \mathcal{B})$ , such that  $\text{ddh}_g(XA_1, YB_1, Z_1) = 1$  and  $\text{ddh}_g(XA_2, YB_2, Z_2) = 1$ . If such an  $H$  query exists,  $\mathcal{S}$  computes  $Z_1^* = Z_1/(YB_1)^x, Z_2^* = Z_2/(YB_2)^x$ . Since  $(Z_2^*)^{1/s} = ((YB_2)^{\text{dl}_g(A_2)})^{1/s} = (YB_2)^{\text{dl}_g(A_1)}$ ,

$$Z_1^*/(Z_2^*)^{1/s} = A_1^{\text{dl}_g(B_1) - \text{dl}_g(B_2)}.$$

Therefore, from  $\left(Z_1^*/(Z_2^*)^{1/s}\right)^{1/(1-r)}$ ,  $\mathcal{S}$  can find  $A_1^{\text{dl}_g(B_1)} = \text{dh}_g(U, V)$ .

With probability at least  $\frac{1}{n(k)^2}$ ,  $\mathcal{M}$  will select a test session with owner and peer  $\mathcal{A}$  and  $\mathcal{B}$ , respectively. Thus the advantage of  $\mathcal{S}$  is

$$\text{Adv}^{\text{GDH}}(\mathcal{S}) \geq \frac{1}{n(k)^2} \Pr[E1_b]. \quad (6)$$

<sup>5</sup> Note that the computation requires the knowledge of  $a_1$  and  $a_2$ , and therefore it must be the case that  $\mathcal{A} \neq \mathcal{B}$ .

**Analysis of E2<sub>a</sub>.**  $\mathcal{S}$  prepares  $n(k)$  honest parties, and assigns random static public and private key pairs for these parties.  $\mathcal{S}$  also chooses two session  $\overline{\text{sid}^*}$ ,  $\overline{\text{sid}^*}$ . Let  $\mathcal{A}$  be the owner of  $\overline{\text{sid}^*}$  and  $\mathcal{B}$  owner of  $\overline{\text{sid}^*}$ .  $\mathcal{S}$  sets the ephemeral public key of  $\overline{\text{sid}^*}$  to be  $U$  and of  $\overline{\text{sid}^*}$  to be  $V$ . Hence  $\mathcal{S}$ 's simulation for  $\mathcal{M}$  can fail only if  $\mathcal{M}$  issues `EphemeralKeyReveal` against  $\overline{\text{sid}^*}$  or  $\overline{\text{sid}^*}$ .

Provided that  $\mathcal{M}$  selects the session  $\overline{\text{sid}^*}$  as the test session with owner  $\mathcal{A}$  and peer  $\mathcal{B}$  and  $\overline{\text{sid}^*}$  as its matching session, and event E2<sub>a</sub> occurs, then the simulation does not fail. If  $\mathcal{M}$  is successful with non-negligible probability it must have queried  $H$  with  $Z_1 = (YB_1)^{\text{dl}_g(U)+a_1}$ ,  $Z_2 = (YB_2)^{\text{dl}_g(U)+a_2}$ . To solve the CDH instance,  $\mathcal{S}$  checks if there is an  $H$  query  $(Z_1, Z_2, U, V, \mathcal{A}, \mathcal{B})$ , such that  $\text{ddh}_g(UA_1, VB_1, Z_1) = 1$  and  $\text{ddh}_g(UA_2, VB_2, Z_2) = 1$ . If such an  $H$  query exists,  $\mathcal{S}$  computes  $\text{dh}_g(U, V)$  by computing  $Z_1 / (U^{b_1} V^{a_1} A_1^{b_1})$ .

With probability  $\frac{1}{s(k)^2}$ ,  $\mathcal{M}$  selects  $\overline{\text{sid}^*}$  as the test session and  $\overline{\text{sid}^*}$  as its matching session. Thus, the advantage of  $\mathcal{S}$  is

$$\text{Adv}^{\text{GDH}}(\mathcal{S}) \geq \frac{1}{s(k)^2} \Pr[\text{E2}_a]. \quad (7)$$

**Analysis of E2<sub>b</sub>, E2<sub>c</sub> and E2<sub>d</sub>.** For event E2<sub>b</sub>, E2<sub>c</sub>, E2<sub>d</sub>,  $\mathcal{S}$ 's simulation is similar to E1<sub>b</sub>, E1<sub>a</sub>, E1<sub>a</sub>, respectively. We omit the details and provide only the conclusion:

$$\text{Adv}^{\text{GDH}}(\mathcal{S}) \geq \frac{1}{n(k)^2} \Pr[\text{E2}_b] \quad (8)$$

$$\text{Adv}^{\text{GDH}}(\mathcal{S}) \geq \frac{1}{s(k)n(k)} \Pr[\text{E2}_c] \quad (9)$$

$$\text{Adv}^{\text{GDH}}(\mathcal{S}) \geq \frac{1}{s(k)n(k)} \Pr[\text{E2}_d]. \quad (10)$$

Combining equations (5), (6), (7), (8), (9), and (10), the advantage of  $\mathcal{S}$  is

$$\text{Adv}^{\text{GDH}}(\mathcal{S}) \geq \max \left\{ \frac{1}{s(k)n(k)} \Pr[\text{E1}_a], \frac{1}{n(k)^2} \Pr[\text{E1}_b], \frac{1}{s(k)^2} \Pr[\text{E2}_a], \frac{1}{n(k)^2} \Pr[\text{E2}_b], \frac{1}{s(k)n(k)} \Pr[\text{E2}_c], \frac{1}{s(k)n(k)} \Pr[\text{E2}_d] \right\}.$$

Since  $\Pr[\text{A3}]$  is non-negligible, from (3), (4), at least one of  $\Pr[\text{E1}_a], \dots, \Pr[\text{E2}_d]$  is non-negligible, and therefore  $\text{Adv}^{\text{GDH}}(\mathcal{S})$  is non-negligible. During the simulation,  $\mathcal{S}$  performs group exponentiations, queries the DDH oracle, and simulates  $H$ . All of these take polynomially bounded time because a group exponentiation takes time  $\mathcal{O}(k)$  and  $t(k), n(k), s(k), h(k)$  are polynomial in  $k$ . Therefore, the running time of  $\mathcal{S}$  is polynomially bounded. Hence,  $\mathcal{S}$  is a polynomial-time algorithm that solves the GDH problem in  $G$  with non-negligible probability, which contradicts the assumed security of GDH problem in  $G$ . This completes the argument.

## 5 Protocol 2

### 5.1 Protocol 2 description

Protocol 2 is similar to Protocol 1 and follows below. The difference between the two protocols is that Protocol 2 computes two additional shared secrets. In the description,  $\mathcal{A}$  is the session initiator and  $\mathcal{B}$  session responder.

1.  $\mathcal{A}$  chooses at random an ephemeral private key  $x \in \mathbf{Z}_q^\times$ , computes the ephemeral public key  $X = g^x$  and sends  $(\mathcal{B}, \mathcal{A}, X)$  to  $\mathcal{B}$ .
2. Upon receiving  $(\mathcal{B}, \mathcal{A}, X)$ ,  $\mathcal{B}$  verifies that  $X \in G^\times$ . If so,  $\mathcal{B}$  chooses at random an ephemeral private key  $y \in \mathbf{Z}_q^\times$ , computes the ephemeral public key  $Y = g^y$ . After computing the shared secrets  $Z_1 = (XA_1)^{y+b_1}$ ,  $Z_2 = (XA_1)^{y+b_2}$ ,  $Z_3 = (XA_2)^{y+b_1}$ ,  $Z_4 = (XA_2)^{y+b_2}$ , the session key  $\text{SK} = H(Z_1, Z_2, Z_3, Z_4, X, Y, \mathcal{A}, \mathcal{B})$  and sending  $(\mathcal{A}, \mathcal{B}, X, Y)$  to  $\mathcal{A}$ ,  $\mathcal{B}$  completes the session with session key SK.
3. Upon receiving  $(\mathcal{A}, \mathcal{B}, X, Y)$ ,  $\mathcal{A}$  checks if he owns a session with identifier  $\text{sid} = (\mathcal{A}, \mathcal{B}, X, \times)$ . If so,  $\mathcal{A}$  verifies  $Y \in G^\times$  and computes  $Z_1 = (YB_1)^{x+a_1}$ ,  $Z_2 = (YB_2)^{x+a_1}$ ,  $Z_3 = (YB_1)^{x+a_2}$ ,  $Z_4 = (YB_2)^{x+a_2}$  and completes the session  $\text{sid} = (\mathcal{A}, \mathcal{B}, X, Y)$  with session key  $\text{SK} = H(Z_1, Z_2, Z_3, Z_4, X, Y, \mathcal{A}, \mathcal{B})$ .

Both parties compute the same  $Z_1 = g^{(x+a_1)(y+b_1)}$ ,  $Z_2 = g^{(x+a_1)(y+b_2)}$ ,  $Z_3 = g^{(x+a_2)(y+b_1)}$ ,  $Z_4 = g^{(x+a_2)(y+b_2)}$  and therefore compute the same session key SK.

### 5.2 Security proof

**Theorem 3.** *If the CDH assumption for  $G$  holds and  $H$  is a random oracle, then the Protocol 2 is eCK-secure.*

*Proof.* The security proof of Protocol 2 is similar to that of Protocol 1; only the differences are explained here. Let  $\mathcal{M}$  be a polynomially bounded adversary against Protocol 2, that runs in time  $t(k)$ , activates at most  $n(k)$  honest parties, at most  $s(k)$  sessions and makes at most  $h(k)$  queries to the oracle  $H$ , where  $t(k)$ ,  $n(k)$ ,  $s(k)$  and  $h(k)$  are polynomially bounded in  $k$ . Assume also that  $\mathcal{M}$  succeeds with non-negligible advantage. As the case of Protocol 1, the adversary has only three ways to distinguish a session key of a test session from a random string: guess, key replication or forging attack. Since  $H$  is a random oracle guessing and key replication occur only with negligible probability.

We use the same events and notation as in the security proof of Protocol 1. In event A3,  $\mathcal{M}$  computes  $Z_1, Z_2, Z_3$  and  $Z_4$  used in the test session and queries  $H$  with  $(Z_1, Z_2, Z_3, Z_4, X, Y, \mathcal{A}, \mathcal{B})$ . As in the security proof of Protocol 1 we show how to construct a CDH solver  $\mathcal{S}$ .

In the  $\mathcal{S}$ 's simulations of environment of  $\mathcal{M}$ , the most important point is to maintain consistency between  $H$  and **SessionKeyReveal** queries when  $\mathcal{S}$  does

not know static private key of the honest party that is activated. Such situations occur when  $\mathcal{S}$  embeds the CDH instance into the honest party's static public key. So, if  $\mathcal{M}$  queries  $H$  with  $(\hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4, X, Y, \mathcal{C}, \mathcal{B})$  or  $(\hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4, Y, X, \mathcal{B}, \mathcal{C})$ , then  $\mathcal{S}$  has to be able to check the correctness of  $\hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4$ .

We now explain how  $\mathcal{S}$  maintains the consistency. Let  $\mathcal{B}$  be an honest party whose static public key is  $B_1 = V, B_2 = g^s/V^r$ , where  $s$  and  $r$  are randomly selected from  $\mathbf{Z}_q$  by  $\mathcal{S}$ . Let  $\mathcal{C}$  be a party (not necessarily an honest one) whose static public key is  $C_1, C_2$ . When  $\mathcal{M}$  queries  $H$  with  $(\hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4, X, Y, \mathcal{C}, \mathcal{B})$  or  $(\hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4, Y, X, \mathcal{B}, \mathcal{C})$ , we may assume that there is a session with identifier  $(\mathcal{B}, \mathcal{C}, X, Y)$  or  $(\mathcal{B}, \mathcal{C}, Y, X)$ . Otherwise, it is sufficient for  $\mathcal{S}$  to return a random string to  $\mathcal{M}$ . Suppose there is a session with identifier  $(\mathcal{B}, \mathcal{C}, X, Y)$ , since  $\mathcal{B}$  is honest,  $Y$  is generated by  $\mathcal{S}$ , so  $\text{dl}_g(Y)$  is known to  $\mathcal{S}$ , who can compute

$$\bar{Z}_1 = \hat{Z}_1/(XC_1)^y, \bar{Z}_2 = \hat{Z}_2/(XC_1)^y, \bar{Z}_3 = \hat{Z}_3/(XC_2)^y, \bar{Z}_4 = \hat{Z}_4/(XC_2)^y$$

The values  $\hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4$  are generated according to the protocol if and only if  $\bar{Z}_1 = (XC_1)^{\text{dl}_g(B_1)}$ ,  $\bar{Z}_2 = (XC_1)^{\text{dl}_g(B_2)}$ ,  $\bar{Z}_3 = (XC_2)^{\text{dl}_g(B_1)}$  and  $\bar{Z}_4 = (XC_2)^{\text{dl}_g(B_2)}$ . The algorithm  $\mathcal{S}$  can check if  $\bar{Z}_1, \bar{Z}_2, \bar{Z}_3$  and  $\bar{Z}_4$  are generated according to the protocol specifications by verifying

$$\bar{Z}_1^r \bar{Z}_2 = (XC_1)^s, \bar{Z}_3^r \bar{Z}_4 = (XC_2)^s;$$

the verification holds with probability at least  $(1 - \frac{1}{q})^2$  when  $r, s$  are randomly chosen from  $\mathbf{Z}_q^\times$  (see Theorem 1). When there is a session with identifier  $(\mathcal{B}, \mathcal{C}, Y, X)$ , similar verification can be performed. In this case,  $\bar{Z}_1 = \hat{Z}_1/(XC_1)^y$ ,  $\bar{Z}_2 = \hat{Z}_2/(XC_2)^y$ ,  $\bar{Z}_3 = \hat{Z}_3/(XC_1)^y$ ,  $\bar{Z}_4 = \hat{Z}_4/(XC_2)^y$  and  $\mathcal{S}$  checks if  $\bar{Z}_1^r \bar{Z}_3 = (XC_1)^s$  and  $\bar{Z}_2^r \bar{Z}_4 = (XC_2)^s$ .

To complete the security proof, an explanation of how to embed and solve CDH instance in the cases E1<sub>a</sub>, E1<sub>b</sub>, E2<sub>a</sub>, E2<sub>b</sub>, E2<sub>c</sub> and E2<sub>d</sub> is still needed.

**E1<sub>a</sub> case.** Suppose that the test session is  $(\mathcal{A}, \mathcal{B}, X, Y)$ , where  $\mathcal{A} \neq \mathcal{B}$ ,  $X = U$ , and party  $\mathcal{B}$ 's static public key is  $B_1 = V, B_2 = g^s/V^r$  with randomly chosen  $r, s \in \mathbf{Z}_q^\times$ . In the event E1<sub>a</sub>, the ephemeral public key  $Y$  is controlled by  $\mathcal{M}$ . If  $\mathcal{M}$  is successful with non-negligible probability, it must have queried  $H$  with inputs of the form  $Z_1 = (YB_1)^{\text{dl}_g(U)+a_1}$ ,  $Z_2 = (YB_2)^{\text{dl}_g(U)+a_1}$ ,  $Z_3 = (YB_1)^{\text{dl}_g(U)+a_2}$  and  $Z_4 = (YB_2)^{\text{dl}_g(U)+a_2}$ . From these values,  $\mathcal{S}$  can compute  $Z_1^* = Z_1/(YB_1)^{a_1}$ ,  $Z_2^* = Z_2/(YB_2)^{a_1}$ ; note that

$$Z_1^*/Z_2^* = U^{\text{dl}_g(B_1) - \text{dl}_g(B_2)} = U^{(1+r)\text{dl}_g(B_1) - s}$$

because  $r\text{dl}_g(B_1) + \text{dl}_g(B_2) \equiv s \pmod{q}$ . Therefore, from  $\left((Z_1^*/Z_2^*) \cdot U^s\right)^{1/(1+r)}$ ,  $\mathcal{S}$  can compute  $U^{\text{dl}_g(B_1)} = \text{dh}_g(U, V)$ . With probability at least  $\frac{1}{s(k)n(k)}$ ,  $U$  is the test session outgoing ephemeral public key and  $\mathcal{B}$  is the test session peer. Since the probability that some trapdoor test yields an incorrect answer is at most  $2h(k)/q$ , the advantage of  $\mathcal{S}$  is

$$\text{Adv}^{\text{CDH}}(\mathcal{S}) \geq \frac{1}{s(k)n(k)} \Pr[\text{E1}_a] - \frac{2h(k)}{q}. \quad (11)$$

**E1<sub>b</sub> case.** Suppose that the test session is  $(\mathcal{A}, \mathcal{B}, X, Y)$ , where  $\mathcal{A} \neq \mathcal{B}$  and  $\mathcal{A}$ 's static public key is  $A_1 = U, A_2 = g^{s'}/U^{r'}$  and  $\mathcal{B}$ 's static public key is  $B_1 = V, B_2 = g^s/V^r$  with randomly chosen  $r, s, r', s' \in \mathbf{Z}_q^\times$ . In the event  $\text{E1}_b$ ,  $Y$  is controlled by  $\mathcal{M}$ , but  $\mathcal{S}$  selects  $X$  and so  $\mathcal{S}$  knows  $\text{dl}_g(X)$ . If  $\mathcal{M}$  is successful with non-negligible probability, it must have queried  $H$  with inputs of the form  $Z_1 = (YB_1)^{\text{dl}_g(U)+\text{dl}_g(A_1)}$ ,  $Z_2 = (YB_2)^{\text{dl}_g(U)+\text{dl}_g(A_1)}$ ,  $Z_3 = (YB_1)^{\text{dl}_g(U)+\text{dl}_g(A_2)}$  and  $Z_4 = (YB_2)^{\text{dl}_g(U)+\text{dl}_g(A_2)}$ . From these values,  $\mathcal{S}$  can compute  $Z_1^*/Z_2^* = Z_1/(YB_1)^x, Z_2^* = Z_2/(YB_2)^x$ ; note that

$$Z_1^*/Z_2^* = A_1^{\text{dl}_g(B_1)-\text{dl}_g(B_2)} = A_1^{(1+r)\text{dl}_g(B_1)-s}$$

Therefore, from  $\left((Z_1^*/Z_2^*) \cdot A_1^s\right)^{1/(1+r)}$ ,  $\mathcal{S}$  can compute  $A_1^{\text{dl}_g(B_1)} = \text{dh}_g(U, V)$ . With probability at least  $\frac{1}{n(k)^2}$ , the test session peers are  $\mathcal{A}$  and  $\mathcal{B}$ , and hence the advantage of  $\mathcal{S}$  is

$$\text{Adv}^{\text{CDH}}(\mathcal{S}) \geq \frac{1}{n(k)^2} \Pr[\text{E1}_b] - \frac{2h(k)}{q}. \quad (12)$$

**E2<sub>a</sub> case.** Suppose that the test session and its matching sessions are  $(\mathcal{A}, \mathcal{B}, X, Y)$  and  $(\mathcal{B}, \mathcal{A}, X, Y)$ , respectively, where  $X = U$  and  $Y = V$  (the case where  $X = V$  and  $Y = U$  is similar). The simulator  $\mathcal{S}$  knows the static private key of all honest parties including  $\mathcal{A}$  and  $\mathcal{B}$ . If  $\mathcal{M}$  is successful with non-negligible probability, it must have queried  $H$  with inputs of the form  $Z_1 = (YB_1)^{\text{dl}_g(U)+a_1}$ ,  $Z_2 = (YB_2)^{\text{dl}_g(U)+a_1}$ ,  $Z_3 = (YB_1)^{\text{dl}_g(U)+a_2}$  and  $Z_4 = (YB_2)^{\text{dl}_g(U)+a_2}$ . From these,  $\mathcal{S}$  can obtain  $\text{dh}_g(U, V)$  by computing  $Z_1/(U^{b_1}V^{a_1}A_1^{b_1})$ . With probability at least  $\frac{1}{s(k)^2}$ , the test session has ephemeral public keys  $U$  and  $V$ , and hence the advantage of  $\mathcal{S}$  is

$$\text{Adv}^{\text{CDH}}(\mathcal{S}) \geq \frac{1}{s(k)^2} \Pr[\text{E2}_a]. \quad (13)$$

**E2<sub>b</sub>, E2<sub>c</sub>, E2<sub>d</sub> cases.** For cases  $\text{E2}_b, \text{E2}_c$  and  $\text{E2}_d$ , the arguments are similar to  $\text{E1}_b, \text{E1}_a$ , and  $\text{E1}_a$ , respectively; therefore

$$\text{Adv}^{\text{CDH}}(\mathcal{S}) \geq \frac{1}{n(k)^2} \Pr[\text{E2}_b] - \frac{2h(k)}{q} \quad (14)$$

$$\text{Adv}^{\text{CDH}}(\mathcal{S}) \geq \frac{1}{s(k)n(k)} \Pr[\text{E2}_c] - \frac{2h(k)}{q} \quad (15)$$

$$\text{Adv}^{\text{CDH}}(\mathcal{S}) \geq \frac{1}{s(k)n(k)} \Pr[\text{E2}_d] - \frac{2h(k)}{q} \quad (16)$$

Combining equations (11), (12), (13), (14), (15), and (16), the advantage of  $\mathcal{S}$  is

$$\text{Adv}^{\text{CDH}}(\mathcal{S}) \geq \max \left\{ \frac{1}{s(k)n(k)} \Pr[\text{E1}_a], \frac{1}{n(k)^2} \Pr[\text{E1}_b], \frac{1}{s(k)^2} \Pr[\text{E2}_a], \right. \\ \left. \frac{1}{n(k)^2} \Pr[\text{E2}_b], \frac{1}{s(k)n(k)} \Pr[\text{E2}_c], \frac{1}{s(k)n(k)} \Pr[\text{E2}_d] \right\} - \frac{2h(k)}{q}.$$

As the security proof of Protocol 1, if  $\text{Adv}_{\text{Protocol 2}}^{\text{AKE}}(\mathcal{M})$  is non-negligible, then  $\Pr[\text{A3}]$  is also non-negligible and thus at least one of  $\Pr[\text{E1}_a], \dots, \Pr[\text{E2}_d]$  is non-negligible. Therefore,  $\text{Adv}^{\text{CDH}}(\mathcal{S})$  is non-negligible. Moreover, during the simulation,  $\mathcal{S}$  performs group exponentiations and simulates  $H$ , all of which take polynomially bounded in  $k$  time. Thus, the running time of  $\mathcal{S}$  is bounded by a polynomial in  $k$  time. Therefore,  $\mathcal{S}$  is a polynomial-time algorithm that solves the CDH problem in  $G$  with non-negligible advantage, which contradicts the hardness of the CDH problem in  $G$ . This concludes the argument.

*Remark 1.* In the security argument of Protocol 1 and Protocol 2, for simplicity we do not allow the test session to be of the form  $(\mathcal{A}, \mathcal{B}, X, Y)$ , where  $\mathcal{A} = \mathcal{B}$ . However, if we allow the session of the form  $(\mathcal{A}, \mathcal{A}, X, Y)$ , then the arguments can be modified to solve the Square computational Diffie-Hellman (SCDH) problem. The SCDH problem is given  $X \in G$ , compute  $X^{\text{dl}_g(X)^2}$ . More precisely, in Protocol 1 this case is reduced to solve SCDH problem given DDH oracle, and in Protocol 2 reduced to solve SCDH problem. Also, note that CDH problem is equivalent to SCDH problem in prime order cyclic group  $G$ , see [2].

## 6 Comparison

In this section, we compare our protocols with other related PKI-based two-pass AKE protocols in terms of underlying assumption, computational efficiency and security model. In Table 1 number of exponentiation in  $G$ , number of static public keys in terms of group elements and number of ephemeral public key in terms of group elements are denoted by E, sPK and ePK, respectively. All protocols are eCK secure except for HMQV, which is a modification of MQV [13]. It is secure in a modified CK [8] model and has additional security properties like resistance to KCI attack, wPFS, and resistance to LEP under GDH and knowledge of exponent assumptions (KEA1) [3].

When comparing computational efficiency, we do not take into account public-key validation, which is a necessary procedure to prevent potential leakage of private information similar to invalid-curve attacks [1] and small subgroup attacks [14]; see also [19, 21].

Table 1 presents the naive group exponentiations count; the numbers in parentheses reflect exponentiations using speedup techniques from [18, §2.3] and [20, Alg. 14.88]. The reduced numbers follow from: (i) HMQV, CMQV, and Okamoto's protocol can use simultaneous exponentiation [20, Alg. 14.88]; and (ii) NAXOS, NAXOS+, Huang-Cao protocol, and Protocol 2 have the same

base and can save time when applying Right-to-Left binary method. More precisely, in our Protocol 2, from the point of view of protocol initiator,  $Z_1, Z_3$  and  $Z_2, Z_4$  have the same base  $YB_1$  and  $YB_2$ , respectively. Thus, when applying Right-to-Left binary method the value  $(YB_1)^{2^i}$  (resp.  $(YB_2)^{2^i}$ ) can be reused for  $Z_1, Z_3$  (resp.  $Z_2, Z_4$ ). Similar arguments apply to NAXOS, NAXOS+ and Huang-Cao’s protocol.

Protocol	Computation	Security Model	Assumption	NAXOS approach	Num. of sPK/ePK
Okamoto [22]	8E (4.14E)	eCK	$\pi$ PRF, DDH, Standard	O	2/3
HMQV [12]	2.5E (2.17E)	CK, wPFS, KCI, LEP	KEA1, GDH, RO	×	1/1
CMQV [25]	3E (2.17E)	eCK	GDH, RO	O	1/1
NAXOS [15]	4E (3.17E)	eCK	GDH, RO	O	1/1
NETS [17]	3E	eCK	GDH, RO	O	1/1
SMEN <sup>-</sup> [26]	6E (2.46E)	eCK	GDH, RO	×	2/2
Protocol 1	3E	eCK	GDH, RO	×	2/1
NAXOS+ [16]	5E (3.34E)	eCK	CDH, RO	O	1/1
Huang-Cao [11]	5E (4.17E)	eCK	CDH, RO	O	2/1
Protocol 2	5E (3.34E)	eCK	CDH, RO	×	2/1

**Table 1.** Protocol Comparison

Okamoto’s protocol is secure in the standard model, but the proof depends on a rather strong assumption of the existence of  $\pi$ PRF family. In the security proof of HMQV and CMQV, the reduction argument is less tight since the Forking Lemma [24] is essential for the arguments. In comparison, the rest of the protocols in Table 1, including Protocol 1 and Protocol 2, have tighter security reductions and do not use the Forking Lemma.

**No NAXOS’ approach.** As shown in Table 1, Protocol 1 has the same characteristic as NETS and Protocol 2 has the same characteristic as NAXOS+ in computation efficiency, security model, and underlying assumption. The difference is that our protocols dispense with NAXOS’ approach, at the expense of an additional group element in the static key. SMEN<sup>-</sup> also has features similar to Protocol 1: it is eCK-secure in the random oracle model under the GDH assumption, does not use NAXOS’ approach and a static public key is a pair of group elements. It achieves better computational performance (2.46 vs 3 exponentiations), but requires that the an ephemeral key constitutes of two group elements. Therefore it provides a trade-off between computation and communication efficiencies.

We showed that it is possible to construct eCK-secure AKE protocols without using NAXOS' approach, so our protocols are secure even when the discrete logarithm of the ephemeral public key is revealed. As pointed out in [26], protocols that do not rely on NAXOS' approach decrease the risk of leaking the static private key in comparison with protocols that utilize the NAXOS' approach. This feature makes protocols like ours, SMEN<sup>-</sup> and HMQV more practical.

Another advantage of our protocols is the use of single random oracle as opposed to two for HMQV and CMQV. The random oracle is needed for the session key derivation, which is typical way to attain indistinguishability in random oracle model. As pointed in [7], although protocols secure in the random oracle model produce assurance for the scheme's correctness, there may remain some fear since concrete hash function instantiations differ from a truly random function. In the sense of minimal reliance on random oracles, our protocols and SMEN<sup>-</sup> are the best among protocols in Table 1.

## 7 Conclusion

The extended Canetti-Krawczyk (eCK) definition introduced by LaMacchia, Lauter and Mityagin is a strong security model for authenticated key exchange. This paper presented two eCK-secure AKE protocols without using NAXOS' approach. As a result, our protocols provide strong security assurances without compromising too much on efficiency. In addition, we minimized the reliance on the random oracle for the security argument and were able to utilize the trap-door test to assume only computational assumptions.

## References

1. A. Antipa, D. Brown, A. Menezes, R. Struik, S. Vanstone, "Validation of elliptic curve public keys," PKC 2003, LNCS 2567, pp. 211-223, 2003.
2. F. Bao, R. H. Deng and H. Zhu, "Variations of Diffie-Hellman Problem," ICICS 2003, LNCS 2836, pp. 301-312, 2003.
3. M. Bellare and A. Palacio, "The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols," Crypto 2004, LNCS 3152, pp.273-289, 2004.
4. M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," Eurocrypt 2001, LNCS 1807, pp. 139-155, 2001.
5. M. Bellare and P. Rogaway, "Entity authentication and key distribution," Crypto 1993, LNCS 773, pp. 110-125, 1993.
6. M. Bellare and P. Rogaway, "Provably secure session key distribution: the three party case," STOC 1995, pp. 57-66, 1995.
7. M. Bellare and P. Rogaway, "Minimizing the use of random oracles in authenticated encryption schemes," ICICS 1997, LNCS 1334, pp. 1-16, 1997.
8. R. Canetti, H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," Eurocrypt 2001, LNCS 2045, pp. 453-474, 2001.
9. D. Cash, E. Kiltz, and V. Shoup, "The twin diffie-hellman problem and applications," Eurocrypt 2008, LNCS 4965, pp. 127-145, 2008.
10. W. Diffie and H. Hellman, "New directions in cryptography," IEEE transactions of Information Theory, Vol. 22(6), pp. 644-654, 1976.

11. H. Huang and Z. Cao, "Strongly secure authenticated key exchange protocol based on computational Diffie-Hellman problem," *Inscrypt* 2008.
12. H. Krawczyk, "HMQR: A high-performance secure Diffie-Hellman protocol," *Crypto 2005*, LNCS 3621, pp. 546-566, 2005.
13. L. Law, A. Menezes, M. Qu, J. Solinas, S. Vanstone, "An efficient protocol for authenticated key agreement," *Designs, Codes and Cryptography*, Vol. 28, pp.119-134, 2003.
14. C. Lim, P. Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup," *Crypto 1994*, LNCS 1294, pp.249-263, 1994.
15. B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," *ProvSec 2007*, LNCS 4784, pp. 1-16, 2007.
16. J. Lee and J. Park, "Authenticated key exchange secure under the computational Diffie-Hellman assumption," <http://eprint.iacr.org/2008/344>.
17. J. Lee and C. Park, "An efficient key exchange protocol with a tight security reduction," <http://eprint.iacr.org/2008/345>.
18. D. M'Raihi and D. Naccache, "Batch exponentiation: a fast DLP-based signature generation strategy," *CCS '96: Proceedings of the 3rd ACM conference on Computer and communications security*, pp. 58-61, 1993.
19. A. Menezes, "Another look at HMQV," *Journal of Mathematical Cryptology*, Vol. 1(1), pp. 47-64, 2007.
20. A. Menezes, P. van Oorschot, and S. Vanstone, "Handbook of applied cryptography," CRC Press, Boca Raton, Florida, USA, 1997.
21. A. Menezes, B. Ustaoglu, "On the importance of public-key validation in the MQV and HMQV key agreement protocols," *Indocrypt 2006*, LNCS 4329, pp. 133-147, 2006.
22. T. Okamoto, "Authenticated key exchange and key encapsulation in the standard model," *Asiacrypt 2007*, LNCS 4833, pp.474-484, 2007.
23. T. Okamoto, D. Pointcheval, "The Gap-Problems: A new class of problems for the security of cryptographic schemes," *PKC 2001*, LNCS 1992, pp. 104-118, 2001.
24. D. Pointcheval and J. Stern, "Security Arguments for Digital Signatures and Blind Signatures," *J. of Cryptology*, Vol 13(3), pp. 361-396, 2000.
25. B. Ustaoglu, "Obtaining a secure and efficient key agreement protocol for (H)MQV and NAXOS," *Designs, Codes and Cryptography*, Vol. 46(3), pp. 329-342, 2008. Extended version available at <http://eprint.iacr.org/2007/123>.
26. J. Wu and B. Ustaoglu, "Efficient Key Exchange with Tight Security Reduction," *Technical Report CACR 2009-23*, University of Waterloo, 2009. Available at <http://www.cacr.math.uwaterloo.ca/techreports/2009/cacr2009-23.pdf>