Efficient Secure Authenticated Key Exchange Without NAXOS' Approach Based on Decision Linear Problem

Mojahed Ismail Mohamed[1,2(✉)], Xiaofen Wang[1], and Xiaosong Zhang[1]

[1] School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu, China
mmmmoj@hotmail.com, wangxuedou@sina.com, johnsonzxs@uestc.edu.cn
[2] Department of Electronic Engineering, Karary University, Omdurman, Sudan

Abstract. LaMacchia, Lauter and Mityagin [4] presents significant security model for Authenticated Key Exchange (AKE) protocols (eCK) which it is extending for Canetti-Krawczyk model (CK). They contrived a protocol secured in that model called NAXOS. eCK model allows adversary to obtain ephemeral secret information corresponding to the test session which complexify the security proof. To vanquish this NAXOS combines an ephemeral private key with a static private key to generate an ephemeral public in the form $X = g^{H(x,a)}$. As a consequence, the discrete logarithm of an ephemeral public key is hidden via an additional random oracle. In this paper we present AKE protocol secure in eCK model under Decision Linear assumption(DLIN) without using NAXOS trick with a fastened reduction, which reduce the risk of leaking the static private key, that because of 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. And minimize the use of the random oracle, by applying it only to the session key derivation. Moreover, each ephemeral and static key has its particular generator which gives tight security for the protocol.

Keywords: eCK model · AKE · Decision Linear assumption · NAXOS' approach

1 Introduction

An Authenticated Key Exchange protocol (AKE) allows two parties to end up with a shared secret key in secure and authenticated manner. The authentication problem deals with restraining adversary that actively controls the communication links used by legitimated parties. They may modify and delete messages in transit, and even inject false one or may controls the delays of messages.

In 1993, Bellare and Rogaway [1] provided the first formal treatment of entity authentication and authenticated key distribution appropriate to the distributed environment. In 1998, Bellare,Canetti, Mihir and Krawczyk [2] provided model

S. Guo et al. (Eds.): CollaborateCom 2015, LNICST 163, pp. 243–256, 2016.
DOI: 10.1007/978-3-319-28910-6_22

for studying session-oriented security protocols. They also introduce the "authenticator" techniques that allow for greatly simplifying the analysis of protocols. In addition, they proposed a definition of security of KE protocols rooted in the simulatability approach used to define security of multiparty computation. In 2002 Canetti and Krawczyk [3] presented their security model which had extended by LaMacchia, Lauter and Mityagin [4] model and proposed NAXOS protocol which is secure under their model. That model capture attacks resulting from leakage of ephemeral and long-term secret keys, defined by an experiment in which the adversary is given many corruption power for various key exchange sessions and most solve a challenge on a test session. This model doesn't give an adversary capability to trivially break an AKE protocol.

To acquire eCK security, NAXOS need that the ephemeral public key X is computed from an exponent result from 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's approach. In NAXOS's approach no one is capable 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 additional random oracle. Using NAXOS' approach many protocols [5–8] were claimed secure in the eCK model under the random oracle assumption. In the standard model, eCK-secure protocols were claimed secure in the eCK model as Okamoto [9]; they uses pseudo-random functions instead of hash functions.

Motivating Problem. (1) Design AKE-secure protocol without NAXOS trick to achieve two goals: (i) 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. (ii) Minimize the use of the random oracle, by applying it only to the session key derivation. Kim, Minkyu, Atsushi Fujioka, and Berkant Ustaolu [10] proposed two strongly secure authenticated key exchange without NAXOS approach, one of their protocol supposed to be secure under the GDH assumption and the other under the CDH assumption in random oracle model. (2) Design AKE-secure protocol secure under Decision Linear Assumption. Boneh, Boyen, and Shacham [11] introduced a decisional assumption, called Linear, intended to take the place of DDH in groups - in particular, bilinear groups [12] - where DDH is easy. For this setting, the Linear problem has desirable properties, as Boneh, Boyen and Shacham show: it is hard if DDH is hard, but, at least in generic groups [13], remains hard even if DDH is easy.

Contributions. We presents a concrete and practical AKE protocol that is eCK secure under Decisional linear assumption in the random oracle model. Our protocol does not rely on any NAXOS trick that yields a more efficient solution when it is implemented with secure device. We give tight proofs reducing eCK security of our protocol to break the used cryptographic primitives under random oracle.

In our protocol the ephemeral public key is containing of each peers generator, which result in two different discrete logarithm problem with two different generators, which increase the complexity.

In derivation of session key, each party will compute shared secret from ephemeral keys and static keys.

Organization. Section 2 reviews security definitions and state the hard problem. Section 3 gives brief for the eCK model. Section 4 proposes AKE-secure protocol with its security results. Section 5 compares our protocol with other related AKE protocols and shows its efficiency. And finally we draw the conclusion in Sect. 6.

2 Preliminaries

In this section we review security definitions we will use to construct our protocol.

2.1 The Decision Linear Diffie-Hellman Assumption

Let G be a cyclic group of prime order p and along with arbitrary generators u, v and h where

$$g, u, v, h \in G :< g >= G; u = g^{\alpha}; v = g^{\beta}; g^{\delta} = h; \alpha, \beta, \delta \in \mathbb{Z}_p^* \tag{1}$$

consider the following problem:

Decision Linear Problem in G [11]. Given $u, v, h, u^a, v^b, h^c \in G$ as input, output yes if $a + b = c$ and no otherwise.

One can easily show that an algorithm for solving Decision Linear in G gives an algorithm for solving DDH in G. The converse is believed to be false. That is, it is believed that Decision Linear is a hard problem even in bilinear groups where DDH is easy. More precisely, we define the advantage of an algorithm $\mathcal{A}$ in deciding the Decision Linear problem in G as

$$\begin{aligned} AdvLinear_{\mathcal{A}} \stackrel{\text{def}}{=} \big|&\Pr\left[\mathcal{A}(u, v, h, u^a, v^b, h^{a+b}) = yes : u, v, h \leftarrow_{\$} G; a, b \leftarrow_{\$} \mathbb{Z}_p\right] \\ &- \Pr\left[\mathcal{A}(u, v, h, u^a, v^b, \gamma) = yes : u, v, \gamma \leftarrow_{\$} G; a, b \leftarrow_{\$} \mathbb{Z}_p\right]\big| \end{aligned} \tag{2}$$

The probability is over the uniform random choice of the parameters to $\mathcal{A}$, and over the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}(t, \epsilon)$-decides Decision Linear in G if $\mathcal{A}$ runs in time at most t, and $AdvLinear_{\mathcal{A}}$ is at least ϵ.

Definition 2.1. *We say that the (t, ϵ)-Decision Linear Assumption (DLIN) holds in G if no t-time algorithm has advantage at least ϵ in solving the Decision Linear problem in G.*

2.2 Linear Diffie-Hellman

Let $dl_u, dl_v : G \rightarrow \mathbb{Z}_p$ be the discrete logarithm (DL) functions which takes an input $X, Y \in G$ and returns $x, y \rightarrow \mathbb{Z}_p$ such that $X = v^x$ and $Y = u^y$. Define the Linear Diffie-Hellman functions $ldh : G^2 \rightarrow G$ as $ldh(A, B) = A^{dl_v(X)} B^{dl_u(Y)}$,$ldh(X, Y) = X^{dl_v(A)} Y^{dl_u(B)}$, and Decisional Linear predicate $DLIN_{u,v,h} : G^3 \rightarrow \{0, 1\}$ as a function which takes an input $(A, B, Z) \in G^3$ and returns 1 if

$$Z = A^{dl_v(X)} B^{dl_u(Y)} = h^{dl_v(X)+dl_u(Y)} \tag{3}$$

or in input $(X, Y, Z) \in G^3$ and returns 1 if

$$Z = X^{dl_v(A)} Y^{dl_u(B)} = h^{dl_v(X)+dl_u(Y)} \tag{4}$$

3 Security Model

In this section, eCK model is outlined [17]. In eCK model there are n different parties $P = P_1, \ldots, P_n$ running the KE protocol Π. Each party is possesses a pair of long-term static (private/public) keys together with a corresponding certificate issued by certifying authority. The protocol Π is executed between two parties, say $\mathcal{A}$ and $\mathcal{B}$, whose static public key are A and B respectively. These two parties exchange their ephemeral public keys X and Y, and obtain the same final session key.

Sessions. A party is activated by an outside call or an incoming message to execute the protocol Π. Each program of executing Π is modeled as an interactive probabilistic polynomial-time machine. We call a session an invocation of an instance of Π within a party. We assume that $\mathcal{A}$ is the session initiator and $\mathcal{B}$ is the session responder. Then $\mathcal{A}$ is activated by the outside call $(\mathcal{A}, \mathcal{B})$ or the incoming message $(\mathcal{A}, \mathcal{B}, Y)$. When activated by $(\mathcal{A}, \mathcal{B})$, $\mathcal{A}$ prepares an ephemeral public key X and stores a separate session state which includes all session-specific ephemeral information. The session identifier (denoted by sid) in $\mathcal{A}$ is initialized with $(\mathcal{A}, \mathcal{B}, X, -, \mathcal{I})$. After $\mathcal{A}$ is activated by $(\mathcal{A}, \mathcal{B}, Y)$ (receiving an appropriate message from responder), the session identifier is updated to $(\mathcal{A}, \mathcal{B}, X, Y, \mathcal{I})$. Similarly, the responder $\mathcal{B}$ is activated by the incoming message $(\mathcal{B}, \mathcal{A}, X)$. When activated, $\mathcal{B}$ also prepares an ephemeral public key Y and stores a separate session state, and the corresponding session identifier is $(\mathcal{B}, \mathcal{A}, Y, X, \mathcal{R})$. A $(\mathcal{B}, \mathcal{A}, Y, X, \mathcal{R})$ (if it exists) is said to be matching to the session $(\mathcal{A}, \mathcal{B}, X, Y, \mathcal{I})$ or $(\mathcal{A}, \mathcal{B}, X, -, \mathcal{I})$. For a session $(\mathcal{A}, \mathcal{B}, *, *, role)$, $\mathcal{A}$ is called the owner of the session while $\mathcal{B}$ is called the peer of the session. We say sid is complete if there is no symbol in sid.

Adversaries. The adversary $\mathcal{M}$ is also modeled as a probabilistic polynomial-time machine. $\mathcal{M}$ controls the whole communications between parties by sending arbitrary messages to the intended party on behalf of another party and receiving

the outgoing message from the communicating parties. In order to capture the possible attacks, $\mathcal{M}$ is allowed to make the following queries as well as H queries of (hash) random oracles.

EstablishParty($\mathcal{U}$): $\mathcal{M}$ Registers an arbitrary party $\mathcal{U}$ not in P, whose static public key is on $\mathcal{M}$s own choice. We call this kind of new registered parties dishonest ($\mathcal{M}$ totally controls the dishonest parties), while the parties in P are honest. We require that when $\mathcal{M}$ makes such query, the certifying authority should verify that the submitted static public key is in the appropriate group (to avoid small subgroup attack) and the proof that $\mathcal{M}$ knows the corresponding static private key.

Send($\mathcal{A}$,m): $\mathcal{M}$ sends the message m to party $\mathcal{A}$. Upon invocation $\mathcal{A}$ by m, the adversary obtains the outgoing message of $\mathcal{A}$.

EphemeralKeyReveal(sid): $\mathcal{M}$ obtains the ephemeral private key stored in the session state of session *sid*.

StaticKeyReveal(P_i): $\mathcal{M}$ learns the long-term static private key of an honest party P_i. In this case, P_i no longer seems honest.

SessionKeyReveal(sid): $\mathcal{M}$ obtains the session key for the session *sid* if the session has accepted, otherwise $\mathcal{M}$ obtains nothing.

Experiment. $\mathcal{M}$ is given the set P of honest parties, and makes whichever queries he wants. The final aim of the adversary is to distinguish a session key from a random string of the same length. Thus $\mathcal{M}$ selects a complete and fresh session *sid*, and makes a special query *Test(sid)*. This query can be queried only once, and the session *sid* is called test session. On this query, a coin b is flipped, if $b = 1$ $\mathcal{M}$ is given the real session key held by *sid*, otherwise $\mathcal{M}$ is given a random key drawn from the key space at random. $\mathcal{M}$ wins the experiment if he guesses the correct value of b. Of course, $\mathcal{M}$ can continue to make the above queries after the *Test* query; however the test session should remain fresh throughout the whole experiment.

Definition 3.1 (Fresh session). *Let sid be a complete session, owned by honest $\mathcal{A}$ with honest peer $\mathcal{B}$. If the matching session of sid exists, we let $\overline{sid}$ denote the session identifier of its matching session. sid is said to be fresh if none of the following events occurs:*

1. *$\mathcal{M}$ makes a* ***SessionKeyReveal(sid)*** *query or a* ***SessionKeyReveal***$(\overline{sid})$ *query if $\overline{sid}$ exists.*
2. *If $\overline{sid}$ exists, $\mathcal{M}$ makes either of the following queries:*
   (a) Both ***StaticKeyReveal***$(\mathcal{A})$ *and* ***EphemeralKeyReveal***(sid)*, or*
   (b) Both ***StaticKeyReveal***$(\mathcal{B})$ *and* ***EphemeralKeyReveal***$(\overline{sid})$*.*
3. *If $\overline{sid}$ does not exist, $\mathcal{M}$ makes either of the following queries:*

(a) Both ***StaticKeyReveal***($\mathcal{A}$) and ***EphemeralKeyReveal***(*sid*), or
(b) ***StaticKeyReveal***($\mathcal{B}$).

The eCK security notion can be described now.

Definition 3.2 (eCK security). *The advantage of the adversary $\mathcal{M}$ in the above experiment with respect to the protocol Π is defined as (b is the guessed value of coin by $\mathcal{M}$):*

$$Adv_{\Pi}^{AKE}(\mathcal{M}) = |2\Pr[b' = b] - 1| \tag{5}$$

The protocol Π is said to be secure if the following conditions hold:

1. *If two honest parties complete matching sessions, then they will both compute the same session key, except with a negligible probability.*
2. *The advantage of the adversary $\mathcal{M}$ is negligible.*

4 Protocol

Parameters. Let k to be defined as security parameter and G be as a cyclic group and g be its generator with order a k-bit prime p. Let users public key is a triple of generators $u, v, h \in G$. Parties $\mathcal{A}'s$,$\mathcal{B}'s$ static private key is $a, b \in \mathbb{Z}_p^*$,respectvly. Where $\mathcal{A}'s$ public key is $A = u^a$,$\mathcal{B}'s$ public key is $B = v^b$. Let $H : \{0,1\}^* \rightarrow \{0,1\}^k$ to be a hash function,for security proof will be modeled as a random oracle.

4.1 Protocol Description

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

1. $\mathcal{A}$ chooses randomly $x \in_R \mathbb{Z}_p^*$ as an ephemeral private key, computing $X = v^x$ as the ephemeral public key and then sends ($\mathcal{B}$,$\mathcal{A}$,X) to $\mathcal{B}$.
2. When $\mathcal{B}$ receiving ($\mathcal{B}$,$\mathcal{A}$,X), will verifies that $X \in G$. if so, will chooses randomly $y \in_R \mathbb{Z}_p^*$ as an ephemeral private key, computing $Y = u^y$ as the ephemeral public key and then sends ($\mathcal{A}$,$\mathcal{B}$,X,Y) to $\mathcal{A}$. Then $\mathcal{B}$ computing the shared secret $Z = X^b A^y$, the session $SK = H(Z, X, Y, \mathcal{A}, \mathcal{B})$ and competes the session.
3. When $\mathcal{A}$ receiving ($\mathcal{A}$,$\mathcal{B}$,X,Y), will checks whether he owns a session id with identifier *sid* ($\mathcal{A}$,$\mathcal{B}$,X,$\times$). if so, he verifies that $Y \in G$. if so, he computing the shared secret $Z = Y^a B^x$, the session $SK = H(Z, X, Y, \mathcal{A}, \mathcal{B})$ and competes the session.

The two parties will compute the shared secret

$$\mathcal{B} : Z = X^b A^y = v^{x^b} u^{a^y} = h^{xb+ya} \tag{6}$$

$$\mathcal{A} : Z = Y^a B^x = u^{y^a} v^{b^x} = h^{xb+ya} \tag{7}$$

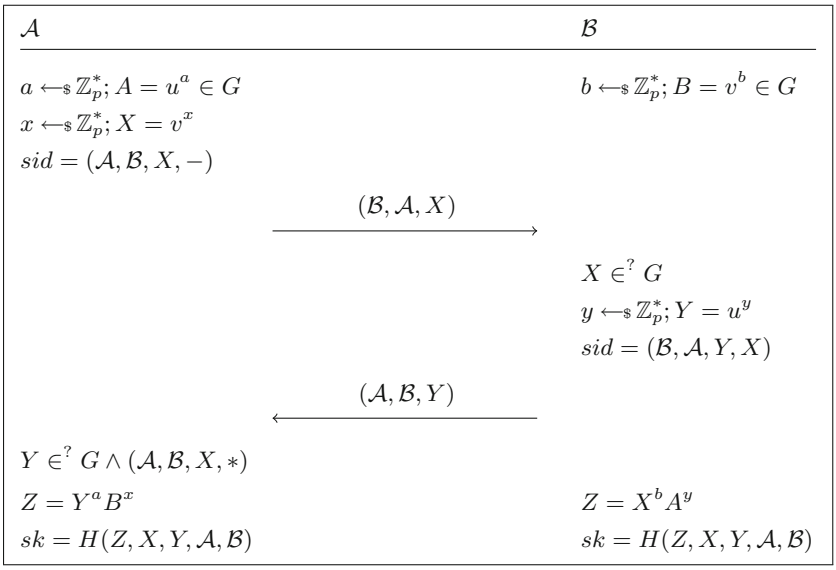

Fig. 1. Our Protocol

4.2 Protocol Security

Theorem 4.1. *If the DLIN assumption holds in G and H is modeled as a random oracle, the Protocol Π is eCK-secure.*

Proof. Let $\mathcal{M}$ be PPTadversary against protocol Π, then we assume $Adv_{\Pi}^{AKE}(\mathcal{M})$ is non-negligible. While H is modeled as a random oracle, there are only three ways for the adversary $\mathcal{M}$ to distinguish a session key - from a random string - of the test session.

- E1. Guessing attack: $\mathcal{M}$ will guesses the session key correctly.
- E2. Key replication attack: $\mathcal{M}$ success in creating a session with the same key to the test session and not matching to the test session.
- E3. Forging attack: $\mathcal{M}$ success in computing the value Z, and then makes queries to H with corresponding tuples $(Z, X, Y, \mathcal{A}, \mathcal{B})$.

The probability of E1 is $\mathcal{O}\big(1/2^k\big)$, which is negligible since H is a random oracle. E2 is same to computing an H-collision; thus the probability of E2 to occur is $\mathcal{O}\big((s(k)^2/2^k)\big)$, and it is a negligible probability. Thus events E1 and E2 can be omitted.

$$Adv_{\Pi}^{AKE}(\mathcal{M}) \leq \Pr[E3] \tag{8}$$

Now, we consider the following sub-events of E3.

- Event1. E3 occurs without existing of matching session for the test session.
- Event2. E3 occurs with existing of matching session for the test session.

Then

$$\Pr[E3] = \Pr[Event1] + \Pr[Event2] \tag{9}$$

We can get the following sub-events of Event1 so that Event1 = Even1.1 $\vee$ Event1.2.

- Event1.1. Event1 $\vee$ $\neg EphemeralKeyReveal(sid)$.
- Event1.2. Event1 $\vee$ $\neg StaticKeyReveal(owner)$.

Also we can get the following sub-events of Event2 so that Event2 = Event2.1 $\vee$ Event2.2 $\vee$ Event2.3 $\vee$ Event2.4.

- Event2.1. Event2 $\vee$ $\neg EphemeralKeyReveal(sid)$ $\vee$ $\neg EphemeralKeyReveal(\overline{sid})$.
- Event2.2. Event2 $\vee$ $\neg StaticKeyReveal(owner)$ $\vee$ $\neg StaticKeyReveal(peer)$.
- Event2.3. Event2 $\vee$ $\neg EphemeralKeyReveal(sid)$ $\vee$ $\neg StaticKeyReveal(peer)$.
- Event2.4. Event2 $\vee$ $\neg StaticKeyReveal(owner)$ $\vee$ $\neg EphemeralKeyReveal(\overline{sid})$.

We get

$$\Pr[Event1] \leq \Pr[Event1.1] + \Pr[Event1.2] \tag{10}$$

$$\Pr[Event2] \leq \Pr[Event2.1] + \Pr[Event2.2] + \Pr[Event2.3] + \Pr[Event2.4] \tag{11}$$

We will construct a solver for The Decision Linear Problem (DLIN) $\mathcal{S}$ that uses protocol Π and adversary $\mathcal{M}$. The solver $\mathcal{S}$ is given (U, V, Z), where U, V and Z are selected uniform randomly in G, access to $DLIN_{u,v,h}(.,.,.)$ oracle to solve Decision Linear problem. Without loss of generality, we denote $\mathcal{A}$ as the test session owner and $\mathcal{B}$ as the responder, and we assume that $\mathcal{A}$ to be the initiator.

Event1.1. We will use $\mathcal{M}$ to construct a DLOG solver $\mathcal{D}$ and DLIN solver $\mathcal{S}$ with non-negligible probability to succeeds. $n(k)$ honest parties will prepared by $\mathcal{S}$, then $\mathcal{S}$ will selects party $\mathcal{B}$ to assigns static public key B^* with v^{b*} where $b*$ is unknown. Random static public and private key pairs will be assigned to the remaining $n(k)-1$ parties. A session sid belongs to $\mathcal{A}$ will be chosen by $\mathcal{S}$. $\mathcal{S}$ will follows the protocol when activating honest parties and will responds faithfully to all queries as he knows static private keys of at least one peer. When the session be sid which has the ephemeral public key U setting by $\mathcal{S}$.

While $\mathcal{S}$ has know idea about the static private key of $\mathcal{B}$, it will be difficult to the simulator to responding queries related to $\mathcal{B}$. Precisely, $\mathcal{M}$ has difficult to compute shared secrets Z but may have to answer SessionKeyReveal queries for sessions owned by $\mathcal{S}$ with peer $\mathcal{C}$. To obtain session keys of these session, $\mathcal{M}$ computes the shared secrets Z and query H. $\mathcal{S}$ will fail its simulation when those two values did not coincide. To overcome this, R^{list} will be prepared by $\mathcal{S}$

with entries of the form $(P_i, P_j, W, W', SK) \in \{0,1\}^* \times \{0,1\}^* \times G^2 \times \{0,1\}^k$, this list will be used later in responses to H and SessionKeyReveal queries.

We will show the behavior of $\mathcal{S}$ when $\mathcal{M}$ sends queries related to $\mathcal{B}$. Party $\mathcal{B}$ generates Y. As we know if $(\mathcal{C}, \mathcal{B}, X, Y, \mathcal{I})(\mathcal{B}, \mathcal{C}, Y, X, \mathcal{R})$ is the session identifier, then $\mathcal{B}$ is the session responder $(\mathcal{I},\mathcal{R})$.

- Send($\mathcal{B}$,$\mathcal{C}$): y will be selected randomly by $\mathcal{S}$ to compute $Y = u^y$. then $sid = (\mathcal{B}, \mathcal{C}, Y, *, \mathcal{R})$ will be new session. $(\mathcal{C}, \mathcal{B}, Y)$ will be sent to $\mathcal{M}$.
- Send($\mathcal{B}$,$\mathcal{C}$,X): y will be selected randomly by $\mathcal{S}$ to compute $Y = u^y$. then $sid = (\mathcal{B}, \mathcal{C}, Y, X, \mathcal{R})$ will be new session. $(\mathcal{C}, \mathcal{B}, Y)$ will be sent to $\mathcal{M}$.
- Send($\mathcal{B}$,$\mathcal{C}$,Y,X): $\mathcal{S}$ will checks whether $\mathcal{B}$ owns session with $sid = (\mathcal{B}, \mathcal{C}, Y, *, \mathcal{R})$. if there is no session with this identifier then abort; otherwise, $\mathcal{S}$ will update the session identifier $sid = (\mathcal{B}, \mathcal{C}, Y, X, \mathcal{R})$.
- $H(.)$: $\mathcal{S}$ prepares an initially empty list H^{list} in the form $(\hat{Z}, W, W', P_i, P_j, SK) \in G^3 \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^k$ and will simulate a random oracle usually. If the query in the form $(\hat{Z}, X, Y, \mathcal{C}, \mathcal{B})$ and $(\hat{Z}, Y, X, \mathcal{B}, \mathcal{C})$ then $\mathcal{S}$ will response with one of the following.
  1. if $(\hat{Z}, X, Y, \mathcal{C}, \mathcal{B}, SK) \vee (\hat{Z}, Y, X, \mathcal{B}, \mathcal{C}, SK) \in H^{list}$ for some SK, then $\mathcal{S}$ return SK to $\mathcal{M}$.
  2. Otherwise, $\mathcal{S}$ checks if there exist $(X, Y, \mathcal{C}, \mathcal{B}, SK) \vee (Y, X, \mathcal{B}, \mathcal{C}, SK) \in R^{list}$ such that $DLIN_{u,v,h}(XA, YB, \hat{Z}) = 1$. If such relation exist, $\mathcal{S}$ returns SK from R^{list}, and stores the new tuple $(X, Y, \mathcal{C}, \mathcal{B}, SK) \wedge (Y, X, \mathcal{B}, \mathcal{C}, SK)$ in H^{list}.
  3. If neither of the above two cases hold, then $\mathcal{S}$ chooses $SK \in_R \{0,1\}^k$ at random, returns it to $\mathcal{M}$ and stores the new tuple $(X, Y, \mathcal{C}, \mathcal{B}, SK) \wedge (Y, X, \mathcal{B}, \mathcal{C}, SK)$ in H^{list}.
- SessionKeyReveal($\mathcal{B}, \mathcal{C}, Y, X$) or SessionKeyReveal($\mathcal{C}, \mathcal{B}, X, Y$): An initially empty list R^{list} will be prepared by $\mathcal{S}$ in the form $(P_i, P_j, W, W', SK) \in \{0,1\}^* \times \{0,1\}^* \times G^2 \times \{0,1\}^k$.
  When SessionKeyReveal($\mathcal{B}, \mathcal{C}, Y, X$) or SessionKeyReveal($\mathcal{C}, \mathcal{B}, X, Y$) is queried, $\mathcal{S}$ will response with one of the following.
  1. If there is no session with identifier $(\mathcal{C}, \mathcal{B}, X, Y)$ or $(\mathcal{B}, \mathcal{C}, Y, X)$ the query is aborted.
  2. If $(\mathcal{B}, \mathcal{C}, Y, X, SK) \vee (\mathcal{C}, \mathcal{B}, X, Y, SK) \in R^{list}$ for some SK, $\mathcal{S}$ returns SK to $\mathcal{M}$.
  3. Otherwise, go through H^{list} to find $(\hat{Z}, X, Y, \mathcal{C}, \mathcal{B}, SK)$ or $(\hat{Z}, Y, X, \mathcal{B}, \mathcal{C}, SK)$ satisfying $DLIN_{u,v,h}(XA, YB, \hat{Z}) = 1$. If such tuple is exist, $\mathcal{S}$ returns SK, and stores the new tuple $(\mathcal{B}, \mathcal{C}, Y, X, SK) \vee (\mathcal{C}, \mathcal{B}, X, Y, SK)$ in R^{list}.
  4. If none of the above three cases hold, the $\mathcal{S}$ choose $SK \in_R \{0,1\}^k$ at random, returns it to $\mathcal{M}$ and stores the new tuple $(\mathcal{B}, \mathcal{C}, Y, X, SK) \vee (\mathcal{C}, \mathcal{B}, X, Y, SK)$ in R^{list}.
- EphemeralKeyReveal(.): $\mathcal{S}$ faithfully will respond to the query.
- StaticKeyReveal($\mathcal{B}$) or EstablishParty($\mathcal{B}$): $\mathcal{S}$ aborts.
- Test(sid): if $sid \neq \overline{sid}$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ randomly choose $\zeta \in \{0,1\}^k$ at random, returns it to adversary $\mathcal{M}$.

When event Event1.1 occurs and the session sid^* selected by $\mathcal{M}$ as the test session corresponding to peer $\mathcal{B}$, then no failure in the simulation will be happened. Now let's take U and V as incoming ephemeral public key and outgoing ephemeral public key for the test session. Let $\mathcal{B}$'s public key to be B^* ($b^* \equiv dl_v(B^*)modp$). Now to let $\mathcal{M}$ with non-negligible probability he must had queried H with inputs $\hat{Z} = U^a {B^*}^{x^*}$ ($x^* \equiv dl_v(U)modp$). But as long the ephemeral public key of sid^* is V, then $\mathcal{M}$ runs $StaticKeyReveal(\mathcal{A})$, $SessionKeyReveal(sid^*)$ and get a, Z^* respectively, then $\mathcal{M}$ computes ${B^*}^{x^*} = \frac{Z^*}{U^a}$ which solve the linear DH of (V, B^*) and computes U^a which solve the linear DH of (U, A).

As we have previously that $\mathcal{M}$ has probability greater than $\frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right)$ can solve DLIN, and at least $\frac{1}{s(k)n(k)}$ the test session is sid^* with peer $\mathcal{B}$ ($\frac{1}{n(k)}$ to pick the correct party $\mathcal{B}$ and $\frac{1}{s(k)}$ to pick the correct session). Thus, the advantage of $\mathcal{S}$ is

$$\begin{aligned} Adv^{DLIN}(\mathcal{S}) &\geq \left(\frac{1}{s(k)n(k)}.Adv_{\Pi}^{AKE}(\mathcal{M}) - Adv^{DLOG}(\mathcal{D})\right) \\ &\geq \frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right) \end{aligned} \tag{12}$$

Event1.2. In this event $n(k)$ honest parties will be prepared by $\mathcal{S}$, and two of them will be selected distinctly. Assume $\mathcal{A}$ and $\mathcal{B}$ the two parties selected distinctly, where U and V corresponds to their static public keys respectively. Then the rest of $n(k) - 2$ parties will assigned to random static and private key pairs. $\mathcal{M}$ will follow the protocol description whenever he activates sessions corresponding to any honest party except $\mathcal{A}$ and $\mathcal{B}$. When he activates sessions corresponding to $\mathcal{A}$ and $\mathcal{B}$, then he will do as in Event1.1.

When the session sid^* selected by $\mathcal{M}$ as the test session corresponding to peer $\mathcal{A}$ and $\mathcal{B}$, let $\mathcal{M}$ with non-negligible probability he must had queried H with inputs $Z = Y^{dl_u(A=U)} B^x$.

To do so, $\mathcal{M}$ runs $EphemeralKeyReveal(sid^*)$, $SessionKeyReveal(sid^*)$ and get x, Z^* respectively, then $\mathcal{M}$ computes $Y^{a^*} = \frac{Z^*}{B^x}$ which solve the linear DH of (Y, A) and computes B^x which solve the linear DH of (X, B).

As we have previously that $\mathcal{M}$ has probability greater than $\frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right)$ can solve DLIN, and at least $\frac{1}{s(k)n^2(k)}$ the test session is sid^* with owner $\mathcal{A}$ and peer $\mathcal{B}$ ($\frac{1}{n(k)}$ to pick the correct party $\mathcal{A}$ and $\frac{1}{n(k)}$ to pick the correct party $\mathcal{B}$ $\frac{1}{s(k)}$ to pick the correct session). Thus, the advantage of $\mathcal{S}$ is

$$\begin{aligned} Adv^{DLIN}(\mathcal{S}) &\geq \left(\frac{1}{s(k)n^2(k)}.Adv_{\Pi}^{AKE}(\mathcal{M}) - Adv^{DLOG}(\mathcal{D})\right) \\ &\geq \frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right) \end{aligned} \tag{13}$$

Event2.1. In this event $n(k)$ honest parties will be prepared by $\mathcal{S}$ with random static public/private keys. Two session sid, sid^* corresponding to $\mathcal{A}$ as the owner

of sid and $\mathcal{B}$ as the owner of sid^* by $\mathcal{S}$. Let V,U be the ephemeral public key of sid,sid^* respectively.

When Event2.1 occurs, there is no failure in the simulation and $\mathcal{M}$ must success with non-negligible probability to query H with $Z = U^a B^{dl_v(V)}$.

To do so, $\mathcal{M}$ runs $StaticKeyReveal(\mathcal{A}), SessionKeyReveal(sid^*)$ and get a, Z^* respectively, then $\mathcal{M}$ computes $B*^{X^*} = \frac{Z^*}{U^a}$ which solve the linear DH of (V, B^*) and computes U^a which solve the linear DH of (U, A).

As we have previously that $\mathcal{M}$ has probability greater than $\frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right)$ can solve DLIN, and at least $\frac{1}{s^2(k)}$ the test session is sid^* with its matching session sid ($\frac{1}{s(k)}$ to pick the sid for party $\mathcal{A}$ and $\frac{1}{s^2(k)}$ to pick its matching session sid^* for party $\mathcal{B}$). Thus, the advantage of $\mathcal{S}$ is

$$\begin{aligned} Adv^{DLIN}(\mathcal{S}) &\geq \left(\frac{1}{s^2(k)}.Adv_{\Pi}^{AKE}(\mathcal{M}) - Adv^{DLOG}(\mathcal{D})\right) \\ &\geq \frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right) \end{aligned} \tag{14}$$

Event2.2. For Event2.2 $\mathcal{S}$s simulation is similar to Event1.2. Hence:

As we have previously that $\mathcal{M}$ has probability greater than $\frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right)$ can solve DLIN, and at least $\frac{1}{s(k)n^2(k)}$ the test session is sid^* with owner $\mathcal{A}$ and peer $\mathcal{B}$ ($\frac{1}{n(k)}$ to pick the correct party $\mathcal{A}$ and $\frac{1}{n(k)}$ to pick the correct party $\mathcal{B}$ $\frac{1}{s(k)}$ to pick the correct session). Thus, the advantage of $\mathcal{S}$ is

$$\begin{aligned} Adv^{DLIN}(\mathcal{S}) &\geq \left(\frac{1}{s(k)n^2(k)}.Adv_{\Pi}^{AKE}(\mathcal{M}) - Adv^{DLOG}(\mathcal{D})\right) \\ &\geq \frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right) \end{aligned} \tag{15}$$

Event2.3 and Event2.4. For event2.3, event2.4 $\mathcal{S}$s simulation is similar to Event1.1. Hence:

As we have previously that $\mathcal{M}$ has probability greater than $\frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right)$ can solve DLIN, and at least $\frac{1}{s(k)n(k)}$ the test session is sid^* with peer $\mathcal{B}$ ($\frac{1}{n(k)}$ to pick the correct party $\mathcal{B}$ and $\frac{1}{s(k)}$ to pick the correct session). Thus, the advantage of $\mathcal{S}$ is

$$\begin{aligned} Adv^{DLIN}(\mathcal{S}) &\geq \left(\frac{1}{s(k)n(k)}.Adv_{\Pi}^{AKE}(\mathcal{M}) - Adv^{DLOG}(\mathcal{D})\right) \\ &\geq \frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right) \end{aligned} \tag{16}$$

Recombining Eqs. (12), (13), (14), (15) and (16), $\mathcal{S}$'s advantage will be

$$Adv^{DLIN}(\mathcal{S}) \geq max \left\{ \begin{array}{r} \frac{1}{s(k)n(k)}.Adv_{\Pi}^{AKE}(\mathcal{M}) - Adv^{DLOG}(\mathcal{D}), \\ \frac{1}{s(k)n^2(k)}.Adv_{\Pi}^{AKE}(\mathcal{M}) - Adv^{DLOG}(\mathcal{D}), \\ \frac{1}{s^2(k)}.Adv_{\Pi}^{AKE}(\mathcal{M}) - Adv^{DLOG}(\mathcal{D}), \\ \frac{1}{2} + \mathcal{O}\left(\frac{q^2(k)}{p}\right) \end{array} \right\} \quad (17)$$

While $Adv_{\Pi}^{AKE}(\mathcal{M})$ is non-negligible then Pr[E3] is non-negligible, and hence from (9),(10) and (11), with non-negligible probability one of these events will happen Event1.1, ..., Event2.4, and therefore $Adv^{GDH}(\mathcal{S})$ is non-negligible. $\mathcal{S}$, $\mathcal{D}$ and H simulation will take polynomially bounded time, $\mathcal{O}(k)$ and $t(k), n(k), s(k), h(k), q(k)$ are polynomial in k. Therefore, $Adv_{\Pi}^{AKE}(\mathcal{M})$ has polynomially bounded running time. Hence, $\mathcal{S}$ is a polynomial-time algorithm that solves the $DLIN$ problem in G with non-negligible, which contradicts the assumed security of $DLIN$ problem in G. This completes the argument.

5 Efficiency

Comparison between our protocol with other related AKE protocols in terms of based assumption, computational efficiency and security model will be discussed in this section. In Table 1 number of exponentiation in G (E), number of static public keys (SPK) and number of ephemeral public key (EPK). Table 1 shows the group exponentiations count; Okamoto's protocol is secure in the standard model, but the proof relies on an existence of PRF family. In the security proof of HMQV and CMQV, the reduction argument is less tight since the Forking Lemma [14] is essential for the arguments. KFU [10]-P1 use two static public key in computation. Our protocol in Table 1, have tighter security reductions and do not use the Forking Lemma and just use one static public key in computation.

Table 1. Protocols Comparison

Protocol	Computation	Security model	Assumption	NAXOS approch	SPK/EPK
Okamoto [9]	8E	eCK	Standard	Yes	2/3
HMQV [15]	2.5E	CK, wPFS,KCI, LEP	GDH, KEA1, RO	No	1/1
CMQV [5]	3E	eCK	RO, GDH	Yes	1/1
NAXOS [15]	4E	eCK	RO, GDH	Yes	1/1
NETS [8]	3E	eCK	RO, GDH	Yes	1/1
SMEN [16]	6E	eCK	RO, GDH	No	2/2
KFU [10]	3E	eCK	RO, GDH	No	2/1
Our	3E	eCK	DLIN, RO	No	1/1

It clear that our protocol has same computation efficiency and security model with NETS, CMQV and KFU-P1, but it differs from them in base assumption. Moreover our protocol and KFU distinguish with no NAXOS' approach. But KFU use two static public key in computation.

Our work showed possibility of constructing eCK-secure AKE protocols not relaying on NAXOS' approach, thus we showed that our protocol is secure though revealing of ephemeral private key while leaking of the static private key is not decrease which give it more practically advance.

Moreover, our protocols uses single random oracle which is opposed to two of HMQV and CMQV and which gives it advance on them. We only use random oracle in the session key derivation.

In addition, our protocol uses decision linear assumption with tight security proof.

6 Conclusions

In this paper we present new efficient eCK-secure AKE protocol without relying on NAXOS approach. We presents secure protocol in eCK model under Decision Linear assumption(DLIN) without using NAXOS trick with a fastened reduction.Our protocol reduce the risk of leaking the static private key, that because of the derivation of the ephemeral public key is independent from the static private key. Moreover, each ephemeral and static key has its particular generator which gives tight security for the protocol. Our protocol has strong definition with more efficiency and minimizing uses of random oracle, by applying it only to the session key derivation. Moreover, we present AKE-secure protocol rely on decision linear assumption with tight security proof.

References

1. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
2. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. ACM (1998)
3. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
4. Lauter, K., Mityagin, A., LaMacchia, B.A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
5. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol for (H)MQV and NAXOS. Des. Codes Crypt. **46**(3), 329–342 (2008). http://www.eprint.iacr.org/2007/123
6. Huang, H., Cao, Z.: Strongly secure authenticated key exchange protocol based on computational Diffie-Hellman problem. In: Inscrypt (2008)

7. Lee, J., Park, J.: Authenticated key exchange secure under the computational Diffie-Hellman assumption. http://www.eprint.iacr.org/2008/344
8. Lee, J., Park, C.: An efficient key exchange protocol with a tight security reduction. http://www.eprint.iacr.org/2008/345
9. Okamoto, T.: Authenticated key exchange and key encapsulation in the standard model. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 474–484. Springer, Heidelberg (2007)
10. Kim, M., Fujioka, A., Ustaoğlu, B.: Strongly secure authenticated key exchange without NAXOS' approach. In: Takagi, T., Mambo, M. (eds.) Advances in Information and Computer Security. LNCS, vol. 5824, pp. 174–191. Springer, Heidelberg (2009)
11. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
12. Joux, A., Nguyen, K.: Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. J. Cryptology **16**(4), 239–247 (2003)
13. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
14. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology **13**(3), 361–396 (2000)
15. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
16. Wu, J., Ustaoglu, B.: Efficient Key Exchange with Tight Security Reduction. Technical report CACR 2009–23, University of Waterloo (2009). http://www.cacr.math.uwaterloo.ca/techreports/2009/cacr200923.pdf
17. Li, H., ChuanKun, W.: CMQV+: an authenticated key exchange protocol from CMQV. Sci. China Inf. Sci. **55**(7), 1666–1674 (2012)