An Efficient Authenticated Key Exchange Protocol with a Tight Security Reduction

Jooyoung Lee and Choon Sik Park

The Attached Institute of Electronics and Telecommunications Research Institute
Yuseong-gu, Daejeon, Korea 305-390
{jlee05,csp}@ensec.re.kr

Abstract. In this paper, we present a new authenticated key exchange(AKE) protocol, called NETS, and prove its security in the extended Canetti-Krawczyk model under the random oracle assumption and the gap Diffie-Hellman(GDH) assumption. Our protocol enjoys a simple and tight security reduction compared to those of HMQV and CMQV without using the Forking Lemma. Each session of the NETS protocol requires only three exponentiations per party, which is comparable to the efficiency of MQV, HMQV and CMQV.

Key words: authenticated key exchange, Gap Diffie-Hellman assumption, extended Canetti-Krawzcyk model

1 Introduction

Key exchange protocols are cryptographic protocols that allow two parties that share no secret information in common to establish a secret key via public communication. The most famous example is the classic Diffie-Hellman(DH) key exchange protocol that marked the birth of modern cryptography [7]. However, the original DH protocol did not provide authentication of the communicating parties, suffering from active attacks such as a man-in-the-middle attack. The quest for an authenticated key exchange(AKE) protocol, that not only allows parties to compute a shared key but also ensures authenticity of the parties, has resulted in a considerable amount of research both in security models and protocols [3–5, 8–13, 19, 22].

Security models. A well known security model for authenticated key exchange is the Cannetti-Krawczyk(CK) model [5]. The CK model lies in a family of indistinguishability-based proof models including the Bellare-Rogaway model [4], Bellere-Pointcheval-Rogaway model [3], and their variants. Choo, Boyd and Hitchcock compared these various security models, and concluded that the CK model offers the strongest definition of security with some modification of the model [6]. However, all these models commonly based on a strong assumption that the adversary is not allowed to obtain certain secret information about the session that is being attacked. As a result, AKE protocols secure in the CK model still might be vulnerable to key-compromise impersonation(KCI) attacks or the leakage of ephemeral private keys(LEP). Especially, the latter attack might be practical in a scenario where the random number generator used by a party is compromised.

Krawczyk later provided a stronger version of the CK model that not only guarantees resilience to the above attacks, but also achieves weak perfect forward secrecy(wPFS) [9]. However, this stronger version still does not include attacks such as revelation of both ephemeral private keys or both static private keys. Recently, LaMacchia, Lauter and Mityagin proposed an elegant security model, called henceforth eCK model, that captures all these security properties [11]. Informally speaking, the only corruption powers that an adversary is not allowed

for in the eCK model are those that would trivially break an authentication protocol. For this reason, the eCK model is currently regarded as the strongest security model.

AKE protocols and security assumptions. One of the most famous and standardized AKE protocols is the MQV protocol proposed by Law, Menezes, Qu, Solinas and Vanstone [13]. However, the construction of the MQV protocol was not based on a formal security proof. In [9], Krawczyk slightly modified the MQV protocol to obtain the HMQV protocol. He presented a formal security proof of the HMQV protocol in the stronger version of the CK model, under the random oracle assumption, the gap Diffie-Hellman(GDH) assumption and the knowledge of exponent assumption(KEA1) [2]. The NAXOS protocol is the first AKE protocol whose security is established in the eCK model [11]. One shortcoming of the protocol is that it requires 4 exponentiations per party compared to 2.5 exponentiations for MQV and HMQV. Motivated by this observation, Ustaoğlu presented the CMQV protocol that achieves both efficiency and security [22]. The security proof of NAXOS and CMQV uses only the random oracle assumption and the GDH assumption, which are weaker than the security assumptions of HMQV.

Focusing on the attempts to weaken the security assumptions in the analysis of AKE protocols, Okamoto proposed an AKE protocol that is proven secure without the random oracle assumption [19]. However, the proof depends upon a rather strong assumption of the existence of πPRFs. The AKE protocol $\mathcal{TS}3$ proposed by Jeong, Katz and Lee [8] uses only the decisional Diffie-Hellman(DDH) assumption and the existence of secure message authentication codes, while we note that the security model is much weaker than the eCK model. In [15], Lee and Park proposed the NAXOS+ protocol and proved its security under the CDH assumption. A major drawback of NAXOS+ is that each session of the protocol requires 5 exponentiations per party, showing less efficiency compared to existing protocols.

Contribution. In [22], the author posed an open problem of finding an AKE protocol that "acheives the performance of MQV and at the same time enjoys a security reduction that is as tight as the security reduction for NAXOS." We provide for a partial answer to the problem by presenting a new AKE protocol, called NETS. NETS is proven secure in the extended Canetti-Krawczyk model under the random oracle assumption and the gap Diffie-Hellman(GDH) assumption. NETS enjoys a simple and tight security reduction, which is similar to the proof of NAXOS. Furthermore, each session of the NETS protocol requires only 3 exponentiations per party, which is comparable to the efficiency of MQV, HMQV and CMQV. Table 1 compares the existing AKE protocols in terms of efficiency, security and underlying assumptions. Here RO is short for the random oracle assumption, and the other abbreviations are described in the text.

Remark 1. We note that the input to the key derivation function is computed by simultaneous double exponentiation for MQV, HMQV and CMQV. As analyzed in [22], the computation can be speeded up via Shamir's algorithm, reducing the cost by 0.75 exponentiations on average. Unfortunately, the speed up does not apply to the NETS protocol.

Organization. In the next section, we present a formal description of the security assumptions and the eCK model. We describe the NETS protocol in Section 3, and prove its security in Section 4. Section 5 concludes.

Protocol	Efficiency	Security	Assumptions
MQV	2.5	unproven	-
HMQV	2.5	CK, wPFS, KCI, LEP	KEA1, GDH, RO
KEA+	3	CK, KCI	GDH, RO
NAXOS	4	eCK	GDH, RO
CMQV	3	eCK	GDH, RO
NAXOS+	5	eCK	CDH, RO
NETS	3	eCK	GDH, RO

Table 1. Protocol comparison

2 Preliminaries

2.1 Assumption

Let $\mathbb{G} = \langle g \rangle$ denote a multiplicative cyclic group of prime order q, generated by $g \in \mathbb{G}$. We define the following three functions, called respectively the discrete logarithm function, the computational Diffie-Hellman function, and the decisional Diffie-Hellman function.

$$\begin{aligned} \mathsf{DLOG} : \mathbb{G} &\longrightarrow \mathbb{Z}_q \\ U &\longmapsto u \text{ such that } U = g^u, \end{aligned} \tag{1}$$

$$\begin{aligned} \mathsf{CDH} : \mathbb{G}^2 &\longrightarrow \mathbb{Z}_q \\ (U, V) &\longmapsto g^{\mathsf{DLOG}(U) \cdot \mathsf{DLOG}(V)}, \end{aligned} \tag{2}$$

and

$$\begin{aligned} \mathsf{DDH} : \mathbb{G}^3 &\longrightarrow \mathbb{Z}_q \\ (U, V, W) &\longmapsto \begin{cases} 1, \text{if } \mathsf{CDH}(U, V) = W, \\ 0, \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

Let $\mathbb{G}^*$ denote the set of non-identity elements in $\mathbb{G}$. The *advantage* of an algorithm $\mathcal{S}$ in solving the discrete logarithm (DLOG) problem, $\mathrm{Adv}^{\mathsf{dlog}}(\mathcal{S})$, is defined to be the probability that, given $U \xleftarrow{\$} \mathbb{G}^*$, $\mathcal{S}$ correctly returns $\mathsf{DLOG}(U)$. Similarly, the advantage of an algorithm $\mathcal{S}$ in solving the gap Diffie-Hellman(GDH) problem, $\mathrm{Adv}^{\mathsf{gdh}}(\mathcal{S})$, is the probability that, given as input $(U, V) \xleftarrow{\$} (\mathbb{G}^*)^2$ and oracle access to $\mathsf{DDH}(\cdot, \cdot, \cdot)$, $\mathcal{S}$ correctly returns $\mathsf{CDH}(U, V)$. We say that $\mathbb{G}$ satisfies the GDH assumption if no feasible adversary can solve the GDH problem with non-negligible probability.

2.2 Extended Canetti-Krawzcyk Security Model

In this section, we describe the eCK model focusing on two pass AKE protocols in the random oracle model. We slightly modify the original model by introducing *explicit* session identifiers. We believe that the explicit identifiers would render a simpler description of AKE protocols, especially for those of three or more passes. We are using the same framework in [15]. For further details on the eCK model, we refer to [11, 22].

Parties. We fix a set $\mathbf{P}$ of n parties, each of which is modeled as a probabilistic Turing machine that makes queries to a set $\mathbf{H} = (H_1, \ldots, H_l)$ of independent random oracles. Each party $\mathcal{A} \in \mathbf{P}$ is represented by a distinct string of a fixed length, say μ. When $\mathcal{A} \in \mathbf{P}$ appears in a protocol flow or as an argument to a function, we mean the string which names the party. We assume that each party $\mathcal{A}$ stores a static public/private key pair $(pk_{\mathcal{A}}, sk_{\mathcal{A}})$, together with a certificate that binds the public key to that party. However, we do not assume that the certification authority(CA) requires parties to prove possession of their static private keys.

An instance of the protocol executed by a party is called a *session*. When a party $\mathcal{A}$ creates a new session, $\mathcal{A}$ chooses a distinct index i from a set $\mathbf{I}$ of indices, and assigns a session identifier $sid = (\mathcal{A}, i) \in \mathbf{SID}$ to the session, where we denote $\mathbf{SID} = \{0,1\}^{\mu} \times \mathbf{I}$. Creation of a session is made via an incoming message that has one of the forms $(\mathcal{A}, \mathcal{B}) \in \{0,1\}^{2\mu}$ and $(\mathcal{A}, (\mathcal{B}, j), Y) \in \{0,1\}^{\mu} \times \mathbf{SID} \times \{0,1\}^*$. When $\mathcal{A}$ creates a session on a message of the form $(\mathcal{A}, \mathcal{B})$, $\mathcal{A}$ is called the *initiator* of the session, otherwise the *responder* of the session. The behavior of a party $\mathcal{A}$ is illustrated in Fig. 1. Note that each session *sid* is associated with the following three variables.

- esk_{sid} (sometimes denoted $esk_{\mathcal{A}}$). Stores an ephemeral private key of λ bits used in the computation of the session key for a constant λ.
- T_{sid}. Records the transcript of the session. The variable takes its value from the set $\mathbf{T} = (\{0,1\}^{\mu})^2 \times \{init, resp\} \times (\{-\} \cup \{0,1\}^*)^2$, where "$-$" is a special symbol not in $\{0,1\}^*$ that means there is no message associated with the position. $T_{sid} = (\mathcal{A}, \mathcal{B}, role, X, Y)$ means that $\mathcal{A}$ executes the session with $\mathcal{B}$ as the second party. $role = init$ means that $\mathcal{A}$ is an initiator, and $role = resp$ a responder. X represents an outgoing message from $\mathcal{A}$, and Y an incoming message to $\mathcal{A}$. We call $\mathcal{A}$ the *owner* of the session, and $\mathcal{B}$ the *peer* of the session.
- K_{sid}. Stores the session key of λ bits.

We say that a session *sid* is *complete* if T_{sid} does not contain the symbol "$-$". For a complete session *sid* with $T_{sid} = (\mathcal{A}, \mathcal{B}, role, X, Y)$, its *matching session* is defined as a session sid^* such that $T_{sid^*} = (\mathcal{B}, \mathcal{A}, role', Y, X)$ or $(\mathcal{B}, \mathcal{A}, role', Y, -)$, and $role \neq role'$. An *equivalent session* of *sid* is defined as a session sid^e such that $T_{sid^e} = T_{sid}$.

Remark 2. For most of existing AKE protocols, the probability that a session *sid* has an equivalent session that is not *sid* itself is negligible. Similarly, each session has more than one matching session only with negligible probability.

Adversaries. An adversary $\mathcal{M}$ is modeled as a probabilistic Turing machine that makes oracle queries to honest parties as well as the set $\mathbf{H}$ of random oracles. It means that the adversary is able to control all communications between parties by sending arbitrary messages to a party on behalf of another party, obtaining responses from the party and/or making decisions about their delivery. In order to capture possible leakage of private information, we assume that an adversary is allowed to make the following additional oracle queries.

- $\mathsf{EstablishParty}(\mathcal{A})$. Registers an additional party $\mathcal{A}$ not in $\mathbf{P}$ together with a static public key on the adversary's own choice. The parties registered by $\mathsf{EstablishParty}(\cdot)$ queries are called *dishonest*, while the parties in $\mathbf{P}$ called *honest*. We assume that the adversary is able to totally control dishonest parties.
- $\mathsf{StaticKeyReveal}(\mathcal{A})$. Reveals a static private key of an honest party $\mathcal{A}$.

if $msg\_in = (\mathcal{A}, \mathcal{B})$ **then**
  Create a session $sid = (\mathcal{A}, i)$, Generate esk_{sid}
  Compute an ephemeral public key X, $T_{sid} \leftarrow (\mathcal{A}, \mathcal{B}, init, X, -)$
  return $(\mathcal{B}, (\mathcal{A}, i), X)$
else if $msg\_in = (\mathcal{A}, (\mathcal{B}, j), Y)$ **then**
  Create a session $sid = (\mathcal{A}, i)$, Generate esk_{sid}
  Compute an ephemeral public key X, $T_{sid} \leftarrow (\mathcal{A}, \mathcal{B}, resp, X, Y)$
  Compute K_{sid}
  return $((\mathcal{B}, j), (\mathcal{A}, i), X)$
else if $msg\_in = ((\mathcal{A}, i), (\mathcal{B}, j), Y)$ and $T_{(\mathcal{A},i)} = (\mathcal{A}, \mathcal{B}, init, X, -)$ for some X **then**
  $T_{(\mathcal{A},i)} \leftarrow (\mathcal{A}, \mathcal{B}, init, X, Y)$
  Compute K_{sid}

Fig. 1. Behavior of a party $\mathcal{A}$ on an incoming message msg_in

- $\mathsf{EphemeralKeyReveal}(sid)$. Reveals an ephemeral private key of a session sid owned by an honest party.
- $\mathsf{SessionKeyReveal}(sid)$. Reveals a session key of a complete session sid owned by an honest party.

Experiment. Initially, the adversary $\mathcal{M}$ is given a set $\mathbf{P}$ of honest parties. $\mathcal{M}$ makes any sequence of the oracle queries described above. At any time in the experiment, $\mathcal{M}$ selects a complete session $sid \in \mathbf{SID}$ owned by an honest party and makes a query $\mathsf{Test}(sid)$. On the query, that is made only once during the experiment, the experiment answers $C \leftarrow \mathsf{SessionKeyReveal}(sid)$ or $C \xleftarrow{\$} \{0,1\}^{\lambda}$ with the same probability. At the end of the experiment, $\mathcal{M}$ guesses whether the challenge C is random or not. If the test session is *clean* and if $\mathcal{M}$ makes a correct guess, then we say that $\mathcal{M}$ *wins* the experiment.

Definition 1. *Let sid be a complete session with $T_{sid} = (\mathcal{A}, \mathcal{B}, role, X, Y)$, and let sid^* and sid^e denote a matching session and an equivalent session of sid, respectively. sid is called* clean *if $\mathcal{A}$ and $\mathcal{B}$ are honest parties, and if none of the following conditions hold.*

1. *$\mathcal{M}$ makes a $\mathsf{SessionKeyReveal}(sid^e)$ query,*
2. *$\mathcal{M}$ makes a $\mathsf{SessionKeyReveal}(sid^*)$ query, if sid^* exists,*
3. *$\mathcal{M}$ makes $\mathsf{StaticKeyReveal}(\mathcal{A})$ and $\mathsf{EphemeralKeyReveal}(sid^e)$ queries,*
4. *$\mathcal{M}$ makes $\mathsf{StaticKeyReveal}(\mathcal{B})$ and $\mathsf{EphemeralKeyReveal}(sid^*)$ queries, if sid^* exists,*
5. *$\mathcal{M}$ makes a $\mathsf{StaticKeyReveal}(\mathcal{B})$ query, if sid^* doesn't exists.*

Definition 2 (eCK security [11]). *The* advantage *of the adversary $\mathcal{M}$ in the AKE experiment with AKE protocol Π is defined as*

$$\mathrm{Adv}_{\Pi}^{\mathsf{ake}}(\mathcal{M}) = \mathsf{Pr}[\mathcal{M} \textit{ wins}] - \frac{1}{2}.$$

We say that an AKE protocol is secure in eCK model if matching sessions compute the same session keys and no efficient adversary $\mathcal{M}$ has more than a negligible advantage in winning the above experiment.

3 NETS AKE protocol

The NETS AKE protocol uses a group $\mathbb{G} = \langle g \rangle$ of prime order q such that the GDH assumption holds, and two hash functions $H_1 : \{0,1\}^\lambda \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \to \{0,1\}^\lambda$, where $\mathbb{Z}_q^* = \mathbb{Z}_q \backslash \{0\}$, λ is a constant such that $q = O(2^\lambda)$, and H_1 and H_2 are modeled as independent random oracles. For the simplicity of description, we assume there is a natural embedding from $\mathbb{G}$ to $\{0,1\}^*$. Using the framework of an AKE protocol illustrated in Section 2.2, the NETS AKE protocol is described as follows.

Static public/private key pair. For each party $\mathcal{A}$, a static private key $sk_\mathcal{A} \in \mathbb{Z}_q^*$ is assigned uniformly at random. The corresponding public key is defined as $pk_\mathcal{A} = g^{sk_\mathcal{A}} \in \mathbb{G}^*$. When a party registers its static public key to the CA, the CA checks if the public key is contained in $\mathbb{G}^*$, and issues the associated certificate. We assume that each party learns the other's public key via the certificate whenever a new session is initiated.

Ephemeral public/private key pair. For each session sid, a party $\mathcal{A}$ generates an ephemeral private key $esk_{sid} \in \{0,1\}^\lambda$ uniformly at random. The corresponding public key is defined as $X = g^{H_1(esk_\mathcal{A}, sk_\mathcal{A})} \in \mathbb{G}$. We assume that each party checks if the other's ephemeral public key is contained in $\mathbb{G}^*$ on receipt of the key. Note that the value $H_1(esk_\mathcal{A}, sk_\mathcal{A})$ is not regarded as an ephemeral key, since $H_1(esk_\mathcal{A}, sk_\mathcal{A})$ can be computed from $esk_\mathcal{A}$ and $sk_\mathcal{A}$ whenever needed.

Session key. In the end of a session sid such that $T_{sid} = (\mathcal{A}, \mathcal{B}, role, X, Y)$, its session key is defined as $K_{sid} = H_2(\sigma)$, where

$$\sigma = (\mathsf{CDH}(pk_\mathcal{A} X, pk_\mathcal{B} Y), \mathsf{CDH}(X, Y), X, Y, \mathcal{A}, \mathcal{B}) \,,$$

if $role = init$, and

$$\sigma = (\mathsf{CDH}(pk_\mathcal{A} X, pk_\mathcal{B} Y), \mathsf{CDH}(X, Y), Y, X, \mathcal{B}, \mathcal{A}) \,,$$

if $role = resp$. Fig. 2 depicts the protocol in a simplified form.

Remark 3. The "public key validation" prevents potential leakage of private information as shown in invalid-curve attacks [1] and small subgroup attacks [14, 17]. Especially, Menezes and Ustaoğlu demonstrates the importance of public key validation by investigating the effects of omitting public-key validation in HMQV and MQV [16, 17]. As long as the underlying group is cryptographically strong, public key validation provides some assurance that computations involving its static and ephemeral private keys do not reveal any information about the key itself [16, 17, 22].

4 Security of NETS protocol

In this section, we present a formal security proof of NETS protocol under the GDH assumption and the random oracle assumption.

Let $\mathcal{M}$ be an AKE adversary against NETS who runs in time t and involves n honest parties. We let k denote the number of sessions that $\mathcal{M}$ activates in honest parties, and let

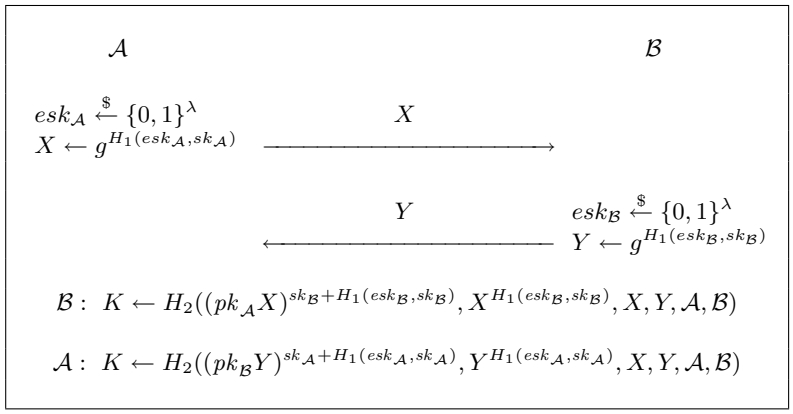

Fig. 2. NETS AKE protocol

m_1 and m_2 respectively denote the number of queries that $\mathcal{M}$ makes to H_1 and H_2. We now start by observing that since the session key of the test session is computed as $K = H_2(\sigma)$ for some 6-tuple σ, the adversary has only two ways to distinguish K from a random string.

- $A1$: At some point, $\mathcal{M}$ queries H_2 on the same 6-tuple σ.
- $A2$: $\mathcal{M}$ forces the establishment of another session key that is computed as $K = H_2(\sigma)$ for the same 6-tuple σ, and then reveal the session key.

The input σ determines the transcript of the session; The session should be either a matching session or an equivalent session of the test session. Since $\mathcal{M}$ is not allowed to make queries to reveal any session key of the matching sessions or equivalent sessions, event $A2$ could not occur. Therefore we conclude that

$$\mathrm{Adv}^{\mathsf{ake}}_{NETS}(\mathcal{M}) \leq \mathsf{Pr}[A1] + \frac{1}{2}\left(1 - \mathsf{Pr}[A1]\right) - \frac{1}{2}, \tag{4}$$

or

$$2\,\mathrm{Adv}^{\mathsf{ake}}_{NETS}(\mathcal{M}) \leq \mathsf{Pr}[A1]. \tag{5}$$

Now we define the following three events.

- $E1$: There exists an honest party $\mathcal{B}$ such that $\mathcal{M}$ makes a random oracle query $H_1(*, sk_{\mathcal{B}})$, either before or without making a query $\mathsf{StaticKeyReveal}(\mathcal{B})$.
- $E2$: $E1$ does not occur, $A1$ occurs and the test session has a matching session.
- $E3$: $E1$ does not occur, $A1$ occurs and the test session has no matching session.

Then we have

$$\mathsf{Pr}[A1] \leq \mathsf{Pr}[E1] + \mathsf{Pr}[E2] + \mathsf{Pr}[E3]. \tag{6}$$

Case 1. In this case, we suppose $\mathsf{Pr}[E1] = \max(\mathsf{Pr}[E1], \mathsf{Pr}[E2], \mathsf{Pr}[E3])$, and construct a DLOG solver $\mathcal{S}$ that uses the adversary $\mathcal{M}$ as a subroutine. Given a challenge $V \in \mathbb{G}$, $\mathcal{S}$ prepares n parties. One party, we call $\mathcal{B}$, is selected at random and assigned static public key $pk_{\mathcal{B}} = V$. The remaining $n-1$ parties are assigned random static key pairs. The solver $\mathcal{S}$ initiates $\mathcal{M}$ on this set of parties and awaits the actions of $\mathcal{M}$. When $\mathcal{M}$ makes a query to any party except $\mathcal{B}$, $\mathcal{S}$ follows the protocol description. $\mathcal{S}$ is also able to respond to the queries $\mathsf{EstablishParty}(\cdot)$ and $\mathsf{Test}(\cdot)$ faithfully. Thus we only describe how $\mathcal{S}$ simulates the party $\mathcal{B}$ and responds to the other oracle queries including the random oracles H_1 and H_2.

– Simulation of $\mathcal{B}$: On the incoming messages $(\mathcal{B}, \mathcal{A})$ and $(\mathcal{B}, (\mathcal{A}, j), X)$, $\mathcal{S}$ creates a new session $sid = (\mathcal{B}, i)$ and selects a random ephemeral key $esk_{(\mathcal{B},i)} \in \{0,1\}^\lambda$. $\mathcal{S}$ selects $Y_i \in \mathbb{G}$ independently and uniformly at random, except in the case of $esk_{(B,i)} = esk_{(B,i')}$ for some previous session (B, i'). For the exception, $\mathcal{S}$ sets $Y_i = Y_{i'}$. Then $\mathcal{S}$ presents $\mathcal{M}$ with the outgoing messages $(\mathcal{A}, (\mathcal{B}, i), Y_i)$ for $(\mathcal{B}, \mathcal{A})$ and $((\mathcal{A}, j), (\mathcal{B}, i), Y_i)$ for $(\mathcal{B}, (\mathcal{A}, j), X)$, respectively. On the incoming message $((\mathcal{B}, i), (\mathcal{A}, j), X)$, $\mathcal{S}$ makes no response.
– $\mathsf{StaticKeyReveal}(\mathcal{A})$: If $\mathcal{A} = \mathcal{B}$, then $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ submits the static private key that $\mathcal{S}$ has generated in the initialization phase.
– $\mathsf{EphemeralKeyReveal}(sid)$: $\mathcal{S}$ submits the value esk_{sid} selected for sid.
– $\mathsf{SessionKeyReveal}(sid)$: $\mathcal{S}$ submits the value K_{sid} selected as follows.
  - If sid is not owned by $\mathcal{B}$, then $\mathcal{S}$ is able to compute the input σ such that $K_{sid} = H_2(\sigma)$. Then $\mathcal{S}$ computes $H_2(\sigma)$ following the recipe of H_2 described below.
  - If sid is owned by $\mathcal{B}$, then $\mathcal{S}$ checks if there is a previous session with the same transcript as sid, in which case the previous session key is selected as K_{sid}.
  - Otherwise, let $T_{sid} = (\mathcal{B}, \mathcal{A}, role, Y_i, X)$ and $role = resp$. Then the session key should be computed as $K_{sid} = H_2(\sigma)$, where
$$\sigma = (\mathsf{CDH}(pk_{\mathcal{A}}X, pk_{\mathcal{B}}Y_i), \mathsf{CDH}(X, Y_i), X, Y_i, \mathcal{A}, \mathcal{B})\,.$$
    $\mathcal{S}$ checks if H_2 was queried with $(W_1, W_2, X, Y_i, \mathcal{A}, \mathcal{B})$ such that
$$\mathsf{DDH}(pk_{\mathcal{A}}X, pk_{\mathcal{B}}Y_i, W_1) = 1 \text{ and } \mathsf{DDH}(X, Y_i, W_2) = 1,$$
    in which case the answer to that query is selected as the session key. If no such query was made, then $\mathcal{S}$ assigns a random value to K_{sid}. For the case $role = init$, K_{sid} is selected in an analogous way.
– $H_1(x, y)$: $\mathcal{S}$ checks if $g^y \stackrel{?}{=} V$. If this is the case, $\mathcal{S}$ succeeds in computing $\mathsf{DLOG}(V)$. Otherwise, $\mathcal{S}$ submits a random value by lazy sampling.
– $H_2(\sigma)$: $\mathcal{S}$ submits the value $H_2(\sigma)$ selected as follows.
  - $\mathcal{S}$ checks if H_2 was queried with the same input before, in which case the answer to the previous query is selected as $H_2(\sigma)$.
  - If $\sigma = (W_1, W_2, Z_1, Z_2, \mathcal{A}, \mathcal{B})$ for some $\mathcal{A}$, W_j and Z_j, $j = 1, 2$, then $\mathcal{S}$ checks if $\mathsf{SessionKeyReveal}(\cdot)$ was queried with any sid of $T_{sid} = (\mathcal{B}, \mathcal{A}, resp, Y_i, X)$ such that
$$\mathsf{DDH}(pk_{\mathcal{A}}X, pk_{\mathcal{B}}Y_i, W_1) = 1 \text{ and } \mathsf{DDH}(X, Y_i, W_2) = 1,$$
    in which case the session key is selected as $H_2(\sigma)$. For the case $\sigma = (W_1, Z_1, W_2, Z_2, \mathcal{B}, \mathcal{A})$ for some $\mathcal{A}$, W_j and Z_j, $j = 1, 2$, $H_2(\sigma)$ is selected in an analogous way.
  - Otherwise, $\mathcal{S}$ selects a random value as $H_2(\sigma)$.

Now $\mathcal{S}$ perfectly simulates the $\mathcal{M}$'s environment until $\mathcal{M}$ queries $\mathsf{StaticKeyReveal}(\mathcal{B})$. Since in a perfect simulation $\mathcal{M}$ will query $H_1(*, \mathsf{DLOG}(V))$ without first issuing a $\mathsf{StaticKeyReveal}(\mathcal{B})$ query with probability $\mathsf{Pr}[E1]/n$, we can estimate the success probability of $\mathcal{S}$ as follows.

$$\mathrm{Adv}^{\mathsf{dlog}}(\mathcal{S}) \geq \frac{\mathsf{Pr}[E1]}{n} \geq \frac{2}{3n}\,\mathrm{Adv}^{\mathsf{ake}}_{NETS}(\mathcal{M}). \tag{7}$$

Note that the solver $\mathcal{S}$ runs in time

$$t_{\mathcal{S}} = O(t + \underbrace{n\lambda + k\lambda}_{\substack{\text{preparation of}\\ \text{public keys}}} + \overbrace{m_1\lambda + m_1^2 + k^2}^{\text{simulation of } H_1} + \underbrace{m_2 k + m_2^2 + k^2}_{\substack{\text{simulation of } H_2\\ \text{and } \mathsf{SessionKeyReveal}(\cdot)}}), \tag{8}$$

since a group exponentiation takes time $O(\lambda)$.

Case 2. In this case, we suppose $\mathsf{Pr}[E2] = \max(\mathsf{Pr}[E1], \mathsf{Pr}[E2], \mathsf{Pr}[E3])$, and construct a Diffie-Hellman solver $\mathcal{S}$ that uses the adversary $\mathcal{M}$ as a subroutine. Given challenges $U, V \in \mathbb{G}$, $\mathcal{S}$ prepares n parties and assigns random static key pairs. $\mathcal{S}$ faithfully responds to any query from $\mathcal{M}$, with the only difference being that $\mathcal{S}$ chooses two sessions sid and sid^* independently and uniformly at random, and sets their ephemeral public keys as

$$U = g^{H_1(esk_{sid}, sk_{\mathcal{A}})} \text{ and } V = g^{H_1(esk_{sid^*}, sk_{\mathcal{B}})}. \tag{9}$$

Here we suppose sessions sid and sid^* are owned by $\mathcal{A}$ and $\mathcal{B}$, respectively.

With probability $2\,\mathsf{Pr}[E2]/k^2$, one of the two sessions is the test session, the other its matching session, and $\mathcal{M}$ queries H_2 with σ that contains $\mathsf{CDH}(U, V)$ as a substring. However, we have to exclude the possibility that $\mathcal{M}$ distinguishes the simulated experiment from a true experiment. The only way to do so is to query $(esk_{sid}, sk_{\mathcal{A}})$ or $(esk_{sid^*}, sk_{\mathcal{B}})$ to H_1, and check if the equalities (9) hold. Since sid is a clean session, $\mathcal{M}$ should query for at most one of the values in each pair $(esk_{sid}, sk_{\mathcal{A}})$ and $(esk_{sid^*}, sk_{\mathcal{B}})$. If $\mathcal{M}$ reveals esk_{sid} via an $\mathsf{EphemeralKeyReveal}(\cdot)$ query, then $\mathcal{M}$ cannot query $(esk_{sid}, sk_{\mathcal{A}})$ to H_1 since we exclude the possibility of event $E1$. Without an $\mathsf{EphemeralKeyReveal}(\cdot)$ query, $\mathcal{M}$ cannot obtain any information about esk_{sid} since esk_{sid} is used in only one session. The same argument applying to esk_{sid^*}, we conclude that the probability that $\mathcal{M}$ correctly makes a random guess for one of esk_{sid} and esk_{sid^*}, and then query the random oracle H_1 with $(esk_{sid}, sk_{\mathcal{A}})$ or $(esk_{sid^*}, sk_{\mathcal{B}})$ is at most $(k + m_1)/2^{\lambda-1}$. Now the success probability and the run time of $\mathcal{S}$ are estimated as

$$\mathrm{Adv}^{\mathsf{cdh}}(\mathcal{S}) \geq \frac{2\,\mathsf{Pr}[E2]}{k^2}\left(1 - \frac{k + m_1}{2^{\lambda-1}}\right) \geq \frac{4\,\mathrm{Adv}^{\mathsf{ake}}_{NETS}(\mathcal{M})}{3k^2}\left(1 - \frac{k + m_1}{2^{\lambda-1}}\right), \tag{10}$$

and

$$t_{\mathcal{S}} = O\left(t + n\lambda + k\lambda + m_1^2 + m_2^2 + k^2\right), \tag{11}$$

respectively.

Case 3. In this case, we suppose $\mathsf{Pr}[E3] = \max(\mathsf{Pr}[E1], \mathsf{Pr}[E2], \mathsf{Pr}[E3])$, and construct a CDH solver $\mathcal{S}$ that uses the adversary $\mathcal{M}$ as a subroutine. Given challenges $U, V \in \mathbb{G}$, $\mathcal{S}$ prepares n parties. One party, we call $\mathcal{B}$, is selected at random and assigned static public key $pk_{\mathcal{B}} = V$. The remaining $n-1$ parties are assigned random static key pairs. $\mathcal{S}$ also chooses a session sid uniformly at random, and sets its ephemeral public key as $U = g^{H_1(esk_{sid}, sk_{\mathcal{A}})}$. The simulation of $\mathcal{S}$ is exactly the same as Case 1, with the only difference being that $\mathcal{S}$ does not check if it obtained $\mathsf{DLOG}(V)$ on a query $H_1(x, y)$, and $\mathcal{S}$ aborts if $\mathsf{SessionKeyReveal}(\cdot)$ is queried with sid.

Now $\mathcal{S}$ perfectly simulates the random oracle H_2. In a perfect simulation of $\mathcal{M}$'s environment, with probability $\mathsf{Pr}[E3]/nk$, sid is the test session with a peer $\mathcal{B}$, and $\mathcal{M}$ queries H_2 with σ that contains $(\mathsf{CDH}(pk_{\mathcal{A}}U, VY_i), \mathsf{CDH}(U, Y_i))$ as a substring. Since

$$\mathsf{CDH}(pk_{\mathcal{A}}U, VY_i) = \mathsf{CDH}(pk_{\mathcal{A}}, V)\mathsf{CDH}(pk_{\mathcal{A}}, Y_i)\mathsf{CDH}(U, V)\mathsf{CDH}(U, Y_i), \tag{12}$$

and $\mathsf{CDH}(pk_{\mathcal{A}}, V)$ and $\mathsf{CDH}(pk_{\mathcal{A}}, Y_i)$ are obtained using the exponent $sk_{\mathcal{A}}$, $\mathcal{S}$ can compute $\mathsf{CDH}(U, V)$. The only remaining possibility that we should exclude is that $\mathcal{M}$ queries $(esk_{sid}, sk_{\mathcal{A}})$ to H_1, and check if the equality $U = g^{H_1(esk_{sid}, sk_{\mathcal{A}})}$ holds. This event occurs with probability at most $(k + m_1)/2^{\lambda-1}$, on the condition that event $E3$ occurs with the test

session *sid* and $\mathcal{B}$ is the peer of *sid*. Therefore we can estimate the success probability of $\mathcal{S}$ as follows.

$$\begin{aligned}\mathrm{Adv}^{\mathsf{cdh}}(\mathcal{S}) &\geq \frac{\mathsf{Pr}[E3]}{nk}\left(1-\frac{k+m_1}{2^{\lambda-1}}\right)\\ &\geq \frac{2\,\mathrm{Adv}^{\mathsf{ake}}_{NETS}(\mathcal{M})}{3nk}\left(1-\frac{k+m_1}{2^{\lambda-1}}\right). \end{aligned} \tag{13}$$

The solver $\mathcal{S}$ runs in time

$$t_{\mathcal{S}} = O\left(t + n\lambda + k\lambda + m_1^2 + m_2 k + m_2^2 + k^2\right). \tag{14}$$

Remark 4. When we compute $\mathsf{CDH}(U,V)$ from the equality (12), we can cancel the term $\mathsf{CDH}(U,Y_i)$ since $\mathsf{CDH}(U,Y_i)$ is included in the key derivation function. Due to this property, we are not required to use the Forking lemma.

To summarize, we obtain the following theorem.

Theorem 1. *If H_1 and H_2 are independent random oracles, and $\mathbb{G}$ is a group where the CDH assumption holds, then the NETS AKE protocol satisfies eCK security.*

5 Conclusion

We presented a new authenticated key exchange protocol, called NETS, and prove its security in the extended Canetti-Krawczyk model under the random oracle assumption and the gap Diffie-Hellman assumption. Each session of the NETS protocol requires only three exponentiations per party, which is comparable to the efficiency of MQV, HMQV and CMQV. A main advantage of NETS over these protocols is a simple and tight security proof. NETS can be easily modified to a three-pass variant to provide for key confirmation and full perfect forward secrecy. However, we note that NETS does not have a natural one-pass variant.

References

1. Antipa A., Brown D., Menezes A., Struik R., Vanstone S.: Validation of elliptic curve public keys. In: Desmedt Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 211–223. Springer, Heidelberg (2003)
2. Bellare M., Palacio A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004)
3. Bellare M., Pointcheval D., Rogaway P.: Authenticated key exchange secure against dictionary attacks. In: Pitzmann B. (ed.) EUROCRYPT 2001. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2001)
4. Bellare M., Rogaway P.: Entity authentication and key distribution. In: Stinson D.R. (ed.) CRYPTO 1993, LNCS, vol. 773, pp. 110–125. Springer, Heidelberg (1994)
5. Canetti R., Krawczyk H.: Analysis of key-exchange protocols and their use for building secure channels. In: Piftzmann B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001). Full version available at http://eprint.iacr.org/2001/040
6. Choo K.K., Boyd C., Hitchcock Y.: Examining indistinguishability-based proof models for key establishment protocols. In: Roy B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
7. Diffie W., Hellman H.: New directions in cryptography. IEEE transactions of Information Theory, vol. 22(6): 644–654 (1976)
8. Jeong, I.R., Katz, J., Lee, D.H.: One-round protocols for two-party authenticated key exchange. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 220–232. Springer, Heidelberg (1999)

9. 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). Full version available at http://eprint.iacr.org/2005/176.
10. Kudla, C., Paterson, K.C.: Modular security proofs for key agreement protocols. Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549–565 (2005)
11. LaMacchia K., Lauter K., Mityagin 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)
12. Lauter K., Mityagin, A.: Security analysis of KEA authenticated key exchange. In: Yung M., Dodis Y., Kiayias A., Malkin T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006)
13. Law L, Menezes A, Qu M., Solinas J., Vanstone S.: An efficient protocol for authenticated key agreement. Designs, Codes and Cryptography, vol. 28:119–134 (2003)
14. Lim C., Lee P.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Desmedt Y.G. (ed.) CRYPTO 1994. LNCS, vol. 1294, pp. 249–263. Springer, Heidelberg (1994)
15. Lee J., Park J.: Authenticated Key Exchange Secure under the Computational Diffie-Hellman Assumption. Preprint (2008)
16. Menezes A.: Another look at HMQV. Journal of Mathematical Cryptology, vol. 1(1): 148–175 (2007)
17. Menezes A., Ustaoğlu B.: On the importance of public-key validation in the MQV and HMQV key agreement protocols. In: Lange T. (ed.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 133–147. Springer, Heidelberg (2006)
18. Menezes A., Ustaoğlu B.: Comparing the pre- and post-specified peer models for key agreement. In: Mu Y., Susilo W., Seberry J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 53–68. Springer, Heidelberg (2008)
19. Okamoto T.: Authenticated key exchange and key encapsulation in the standard model. Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 474–484 (2007)
20. Okamoto T., Poincheval D.: The Gap-Problems: A new class of problems for the security of cryptogrphic schemes. In: Kim K. (ed.) PKC 2001, LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
21. Poincheval D., Stern J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology, vol. 13(3): 361–396 (2000)
22. Ustaoğlu B.: Obtaining a secure and efficient key agreement protocol for (H)MQV and NAXOS. Designs, Codes and Cryptography, vol. 46(3): 329–342 (2008)