Leakage Resilient eCK-Secure Key Exchange Protocol Without Random Oracles

Daisuke Moriyama
Institute of Information Security
2-14-1, Tsuruya-cho, Kanagawa-ku
Yokohama-shi, Kanagawa, 221-0835 Japan
dgs082101@iisec.ac.jp

Tatsuaki Okamoto
NTT
3-9-11 Midori-cho, Musashino-shi
Tokyo, 180-8585 Japan
okamoto.tatsuaki@lab.ntt.co.jp

ABSTRACT

This paper presents the first formalization of partial key leakage security of a *two-pass* two-party authenticated key exchange (AKE) protocol on the extended Canetti-Krawczyk (eCK) security model. Our formalization, λ-leakage resilient eCK security, is a (stronger) generalization of the eCK security model with enhanced by the notion of λ-leakage resilient security recently introduced by Akavia, Goldwasser and Vaikuntanathan. We present a PKI-based two-pass key exchange protocol with Hash Proof System (HPS), that is λ-leakage resilient eCK secure without random oracles.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public key cryptosystems

General Terms

Security, Theory

Keywords

authenticated key exchange, eCK security model, leakage resilient security, without random oracle

1. INTRODUCTION

1.1 Background

Recently a notion of *leakage resilient* security has been developed to capture information leakage. Akavia, Goldwasser and Vaikuntanathan [1] formalized the leakage against memory attack such that a malicious adversary can choose an arbitrary function and be responded with input the secret key under the constraint that the total output length of the functions is bounded by the security parameter. Based on the above framework, [8, 11, 17] proposed leakage resilient public key encryption schemes and leakage resilient signature schemes. [2, 3] showed a variant of the leakage resilience such that the leakage parameter is only dependent on the secret key size and independent from the system (public) parameter. This framework is generalization of the memory attack and called bounded retrieval model.

The existing security models for two-party authenticated key exchange (AKE) protocols such as the Canetti-Krawczyk (CK) security model [4] and the extended CK (eCK) security model [14] have already treated the security on secret key leakage. The CK model allows an adversary to issue a corrupt query to obtain the whole internal state including the static secret key after the target session was executed and a session state reveal query to obtain the corresponding (secret) session state information except the static secret key. The eCK model allows (some legal combinations of) static and ephemeral secret key leakages. However, in such existing security models for AKE, the security with the leakage of partial secret information has not been captured.

Alwen, Dodis and Wichs presented an efficient leakage resilient (PKI-based) authenticated key exchange (AKE) protocol in the random oracle model, where they introduced the leakage resilient security (in the sense of [1]) on the CK security model [3]. The key technique of their AKE protocol is to construct a leakage resilient signature scheme as a building block of their protocol. Such a signature-based AKE protocol however has the following shortcomings: (1) the number of interactions between two parties is at least three, i.e., no two-pass protocol is possible, and (2) there is a common weakness in the security as noted in [14], where an adversary can obtain random coins for the underlying signature scheme through a session state reveal query in the CK model (provided that the session state contains the random coins to generate signatures). If the underlying signature scheme is vulnerable by the random coin leakage, the AKE protocol should be insecure.

Dodis et.al. [7] proposed a framework to construct an AKE protocol that has leakage resilient security on the CK security model based on the leakage resilient public key encryption scheme or signature scheme in the standard model. Their construction has also shortcomings: (1) it is a three-pass (not "two-pass") protocol, and (2) the security against the Key Compromise Impersonation (KCI) attack is not formally ensured, that is one of the fundamental security properties for the AKE protocol.

To the best of our knowledge, no leakage resilient security notion (in the sense of [1]) for AKE has been formalized on the *extended Canetti-Krawczyk (eCK)* security model [14], that captures the resistance to the KCI attack and other several important attacks. Moreover, no leakage resilient *two-pass* AKE scheme has been presented.

ASIACCS '11, March 22–24, 2011, Hong Kong, China.

1.2 Our Result

This paper presents the first formalization of key leakage security (λ-*leakage resilience* [1]) of a PKI-based *two-pass* two-party authenticated key exchange (AKE) protocol on the *eCK* security model.

We here provide an example to show the difference of this security formalization and the eCK security model. Let a test session key be owned by Alice and the peer be Bob, and no matching session exists. Then, in our new λ-leakage resilient security model, at most λ *bits* of static secret key of Alice and Bob can be revealed to an adversary even if Alice's ephemeral secret key is revealed, while in the eCK security, *the whole* of Alice's static secret key should be kept secret if the ephemeral secret key of Alice is revealed to an adversary.

This paper presents a PKI-based two-pass AKE protocol that is λ-leakage resilient eCK secure against memory attack without random oracles. The proposed protocol is based on Hash Proof System (HPS) [5] and secure under the subset membership assumption, decision Diffie-Hellman assumption, collision resistant (CR) hash function family and pseudo-random function family with pairwise-independent random sources (πPRF family). We remark that our protocol employs an implementation trick, so-called NAXOS trick (see [14, 12, 16] for more discussion). Informally, this trick requires that the ephemeral public key X is computed with the hashing of ephemeral secret key x and static secret key a, i.e., $X := g^{H(x,a)}$. See Table 1 for comparison of the proposed protocol with existing eCK-secure AKE protocols.

2. PRELIMINARIES

2.1 Notation

When A is a probabilistic machine or algorithm, $A(x)$ denotes the random variable of the output of A on input x. $y \xleftarrow{\mathsf{R}} A(x)$ denotes that y is randomly selected from $A(x)$ according to its distribution. Then, $A(x) \to a$ indicates the event that A outputs a on input x if a is a value. When A is a set, $y \xleftarrow{\mathsf{U}} A$ means that y is uniformly selected from A. When A is a value, $y := A$ denotes that y is set as A.

2.2 Average min-entropy

Let X and Y be two random variables over a finite domain S. The statistical distance between X and Y is defined by $\mathrm{SD}(X,Y) := \frac{1}{2}\prod_{s\in S}|\Pr[X=s]-\Pr[Y=s]|$. If the statistical distance between X and Y is at most ϵ, then we say that these variables are ϵ-close. The min-entropy of a random variable X is $H_\infty(X) := -\log(\max_x \Pr[X=x])$. The average min-entropy [9] of a random variable X given Y is defined by $\tilde{H}_\infty(X \mid Y) := -\log\left(E_{y:=Y}\left[2^{-H_\infty(X|Y=y)}\right]\right)$ and Dodis et al. [9] proved the following property of average min-entropy.

Lemma 1. Let X, Y, Z be random variables and Y has at most 2^λ possible values. Then $\tilde{H}_\infty(X \mid Y) \geq \tilde{H}_\infty(X) - \lambda$.

2.3 Average-case strong extractor [9]

Let $k \in \mathbb{N}$ be a security parameter and $\mathsf{Ext} : \{0,1\}^{n(k)} \times \{0,1\}^{t(k)} \to \{0,1\}^{\ell(k)}$ be a function. We say that Ext is an efficient average-case (m,ϵ)-strong extractor if for all pairs of random variables (X,I) conditioned on $X \in \{0,1\}^{n(k)}$ and $\tilde{H}_\infty(X \mid I) \geq m$, $\mathrm{SD}((\mathsf{Ext}(X,S),S,I),(U_\ell,S,I)) \leq \epsilon$ where $S \xleftarrow{\mathsf{U}} \{0,1\}^{t(k)}$ and $U_\ell \xleftarrow{\mathsf{U}} \{0,1\}^{\ell(k)}$. We call S the extraction key.

2.4 The Decision Diffie-Hellman assumption

Let k be a security parameter and $\mathbb{G}$ be a group of prime order q with $|q| = k$. For all $k \in \mathbb{N}$, we define the two distributions $\mathbb{D}(k) := \{(\mathbb{G}, g_1, g_2, g_1^x, g_2^x) \mid (g_1,g_2) \in \mathbb{G}^2; x \in \mathbb{Z}_q\}$ and $\mathbb{R}(k) := \{(\mathbb{G}, g_1, g_2, y_1, y_2) \mid (g_1,g_2,y_1,y_2) \in \mathbb{G}^4\}$. The advantage of an algorithm $\mathcal{A}$ breaking the Decision Diffie-Hellman (DDH) problem is defined as

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(k) := \left| \begin{array}{l} \Pr[\mathcal{A}(1^k,\rho) \to 1 \mid \rho \xleftarrow{\mathsf{U}} \mathbb{D}(k)] \\ -\Pr[\mathcal{A}(1^k,\rho) \to 1 \mid \rho \xleftarrow{\mathsf{U}} \mathbb{R}(k)] \end{array} \right|.$$

We say that the DDH assumption holds in $\mathbb{G}$ if for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(k)$ is negligible in k.

2.5 Pseudo-Random Function (PRF)

Let k be a security parameter and F be a PRF family. The PRF family is associated with $\{\mathsf{Seed}_k\}_{k\in\mathbb{N}}$, $\{\mathsf{Dom}_k\}_{k\in\mathbb{N}}$ and $\{\mathsf{Rng}_k\}_{k\in\mathbb{N}}$. When we select $\Sigma \xleftarrow{\mathsf{R}} \mathsf{Seed}_k, \mathcal{D} \xleftarrow{\mathsf{R}} \mathsf{Dom}_k, \mathcal{R} \xleftarrow{\mathsf{R}} \mathsf{Rng}_k$ and $\sigma \xleftarrow{\mathsf{U}} \Sigma$, $F := \mathsf{F}^{k,\Sigma,\mathcal{D},\mathcal{R}}_\sigma$ is defined as $F : \mathcal{D} \to \mathcal{R}$. The advantage of an algorithm $\mathcal{A}^O$ breaking the PRF function with oracle access to O is defined as

$$\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\mathcal{A}}(k) := \left| \begin{array}{l} \Pr[\mathcal{A}^F(1^k,\mathcal{D},\mathcal{R}) \to 1] \\ -\Pr[\mathcal{A}^{RF}(1^k,\mathcal{D},\mathcal{R}) \to 1] \end{array} \right|$$

where $\Sigma \xleftarrow{\mathsf{R}} \mathsf{Seed}_k$, $\sigma \xleftarrow{\mathsf{U}} \Sigma$, $\mathcal{D} \xleftarrow{\mathsf{R}} \mathsf{Dom}_k, \mathcal{R} \xleftarrow{\mathsf{R}} \mathsf{Rng}_k, F := \mathsf{F}^{k,\Sigma,\mathcal{D},\mathcal{R}}_\sigma$ and $RF : \mathcal{D} \to \mathcal{R}$ is a truly random function. We say that F is a PRF family if for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\mathcal{A}}(k)$ is negligible in k.

2.6 Pseudo-Random Function with pairwise-independent random sources (πPRF)

The πPRF function family is introduced by Okamoto [18] that is one of the extension of PRF family. In a traditional PRF, a uniformly random seed is reused many times. But we concentrate on the case that there are many seeds derived from the cryptographic primitives. Note that these variables may be correlated and the PRF family no longer ensures the security. Nonetheless, πPRF states that if a specific variable σ_{i_0} (associated with 'seed') is pairwise-independent from the other variables, then the output of the function with σ_{i_0} is indistinguishable from random.

Let $f_\Sigma : I_\Sigma \to X_\Sigma$ be a deterministic polynomial-time algorithm where X_Σ is a set of random variables and I_Σ is a set of indices regarding Σ ($\Sigma \xleftarrow{\mathsf{R}} \mathsf{Seed}_k$). This algorithm takes as input index $i \in I_\Sigma$ and outputs $\sigma_i \in X_\Sigma$.

Now, we define the independence of the two variables that is derived from the function F_Σ. For $j = 0, \ldots, p(k)$, $i_j \in I_\Sigma$ are indices and $\sigma_{i_0}, \sigma_{i_1}, \ldots, \sigma_{i_{p(k)}}$ are random variables output by f_Σ. We say that σ_{i_0} is pairwisely independent from the other variables $(\sigma_{i_1}, \ldots, \sigma_{i_{p(k)}})$ if for any pair of $(\sigma_{i_0}, \sigma_{i_j})$ $(j = 1, \ldots, p(k))$, for any $(x,y) \in \Sigma^2$, we have $\Pr[\sigma_{i_0} \to x \ \wedge \ \sigma_{i_j} \to y] = 1/|\Sigma|^2$.

Consider a probabilistic polynomial-time algorithm $\mathcal{A}^{F,I_\Sigma}$ that issues oracle queries to F or RF. When $\mathcal{A}$ interacts with (F, I_Σ) and sends $(q_j, i_j) \in \mathcal{D} \times I_\Sigma$, the oracle computes $\sigma_{i_j} := f_\Sigma(i_j)$, chooses $\overline{\sigma}_j \xleftarrow{\mathsf{R}} \sigma_{i_j}$ and outputs $\mathsf{F}^{k,\Sigma,\mathcal{D},\mathcal{R}}_{\overline{\sigma}_j}(q_j)$

for each $j = 0, 1, \ldots, p(k)$. $\mathcal{A}^{RF,I_\Sigma}$ is same as $\mathcal{A}^{F,I_\Sigma}$ except the output of $\mathsf{F}^{k,\Sigma,\mathcal{D},\mathcal{R}}_{\overline{\sigma}_0}(q_0)$ is replaced by a truly random function $RF(q_0)$. The advantage of an algorithm $\mathcal{A}$ breaking the πPRF function is defined by

$$\mathsf{Adv}^{\pi\mathsf{PRF}}_{\mathsf{F},I_\Sigma,\mathcal{A}}(k) := \left| \begin{array}{l} \Pr[\mathcal{A}^{F,I_\Sigma}(1^k,\mathcal{D},\mathcal{R}) \to 1] \\ -\Pr[\mathcal{A}^{RF,I_\Sigma}(1^k,\mathcal{D},\mathcal{R}) \to 1] \end{array} \right|.$$

We say that F is a πPRF family if for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\pi\mathsf{PRF}}_{\mathsf{F},I_\Sigma,\mathcal{A}}(k)$ is negligible in k.

2.7 Collision Resistant (CR) hash function

Let H associated with $\mathsf{KH}_{k\in\mathbb{N}}$, $\{\mathsf{Dom}_k\}_{k\in\mathbb{N}}$ and $\{\mathsf{Rng}_k\}_{k\in\mathbb{N}}$ be a family of collision resistant (CR) hash function indexed by a security parameter $k \in \mathbb{N}$. When $h \xleftarrow{\mathsf{R}} \mathsf{KH}_k$, $\mathcal{D} \xleftarrow{\mathsf{R}} \mathsf{Dom}_k$, and $\mathcal{R} \xleftarrow{\mathsf{R}} \mathsf{Rng}_k$, we obtain deterministic polynomial-time algorithm $H := \mathsf{H}^{k,\mathcal{D},\mathcal{R}}_h$ that computes $H : \mathcal{D} \to \mathcal{R}$. We define the advantage of an algorithm $\mathcal{A}$ breaking the CR hash function by

$$\mathsf{Adv}^{\mathrm{CR}}_{\mathsf{H},\mathcal{A}}(k) := \Pr\left[\begin{array}{l} (\rho_1,\rho_2) \in \mathcal{D}^2 \wedge \rho_1 \neq \rho_2 \wedge \mathsf{H}^{k,\mathcal{D},\mathcal{R}}_h(\rho_1) = \\ \mathsf{H}^{k,\mathcal{D},\mathcal{R}}_h(\rho_2) \mid \mathcal{A}(1^k,h,\mathcal{D},\mathcal{R}) \to (\rho_1,\rho_2) \end{array} \right],$$

where $\mathcal{D} \xleftarrow{\mathsf{R}} \mathsf{Dom}_k, \mathcal{R} \xleftarrow{\mathsf{R}} \mathsf{Rng}_k$, and $h \xleftarrow{\mathsf{R}} \mathsf{KH}_k$. H is a CR hash function family if for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{CR}}_{\mathsf{H},\mathcal{A}}(k)$ is negligible in k.

3. DEFINITIONS AND BACKGROUND

3.1 Hash Proof Systems (HPS)

3.1.1 Smooth Projective Hashing

The notion of hash proof systems is introduced by Cramer and Shoup [5] to design IND-CCA2 secure public key encryption schemes.

Let $\mathcal{SK}, \mathcal{PK}$ be sets and $\mu : \mathcal{SK} \to \mathcal{PK}$ be a function. Let $\mathcal{X}, \mathcal{W}, \mathcal{T}, \mathcal{K}$ be sets and $\Lambda_{sk} : \mathcal{X} \times \mathcal{T} \to \mathcal{K}$ be a hash function indexed by $sk \in \mathcal{SK}$. For a proper subset $\mathcal{L} \subset \mathcal{X}$, the hash function is said to be projective if the value of $\Lambda_{sk}(X, \mathsf{aux})$ is uniquely determined by $pk := \mu(sk)$, auxiliary input $\mathsf{aux} \in \mathcal{T}$ and witness $w \in \mathcal{W}$ for $X \in \mathcal{L}$. On the other hand, if $X \in \mathcal{X}\backslash\mathcal{L}$, $\Lambda_{sk}(X, \mathsf{aux})$ may not be computable from pk and X. Remark that $\Lambda_{sk}(X, \mathsf{aux})$ can be evaluated using sk for any $X \in \mathcal{X}$. Therefore, if $X \in \mathcal{L}$, there are two ways to compute $\Lambda_{sk}(X, \mathsf{aux})$: using sk or w.

Following [5], we define 2-universal and its slight variant which we call 2-universal′.

Definition 1. (2-universal) A projective hash function is ε-almost 2-universal if for all $X, X' \in \mathcal{X}\backslash\mathcal{L}$ with $X \neq X'$,

$$\mathrm{SD}\left(\begin{array}{c} (k, \Lambda_{sk}(X', \mathsf{aux}'), \Lambda_{sk}(X, \mathsf{aux})), \\ (pk, \Lambda_{sk}(X', \mathsf{aux}'), K) \end{array} \right) \leq \varepsilon$$

where $sk \xleftarrow{\mathsf{U}} \mathcal{SK}, pk := \mu(sk), \mathsf{aux}, \mathsf{aux}' \in \mathcal{T}$ and $K \xleftarrow{\mathsf{U}} \mathcal{K}$.

Definition 2. (2-universal′) A projective hash function is ε-almost 2-universal′ if for all $X, X' \in \mathcal{X}\backslash\mathcal{L}$ with $(X, \mathsf{aux}) \neq (X', \mathsf{aux})$,

$$\mathrm{SD}\left(\begin{array}{c} (pk, \Lambda_{sk}(X', \mathsf{aux}'), \Lambda_{sk}(X, \mathsf{aux})), \\ (pk, \Lambda_{sk}(X', \mathsf{aux}'), K) \end{array} \right) \leq \varepsilon$$

where $sk \xleftarrow{\mathsf{U}} \mathcal{SK}, pk := \mu(sk), \mathsf{aux}, \mathsf{aux}' \in \mathcal{T}$ and $K \xleftarrow{\mathsf{U}} \mathcal{K}$.

3.1.2 Hash Proof System (HPS)

Kurosawa and Desmedt [13] introduced a variant of hash proof system that consists of the following algorithms $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{EncKey}, \mathsf{Enc}', \mathsf{DecKey})$. The probabilistic algorithm $\mathsf{Setup}(1^k)$ generates $(\mathsf{group}, \mathcal{SK}, \mathcal{PK}, \mathcal{X}, \mathcal{L}, \mathcal{T}, \mathcal{W}, \mathcal{K}, \Lambda_{(\cdot)}, \mu)$ (group may contains some additional parameters). The key generation algorithm $\mathsf{Gen}(1^k; sk)$ takes as input $sk \in \mathcal{SK}$ and output $pk := \mu(sk) \in \mathcal{PK}$ using the function μ. The deterministic algorithm $\mathsf{Enc}(1^k; w)$ generates $X \in \mathcal{L}$ with a witness $w \in \mathcal{W}$ and EncKey outputs K with pk, w and auxiliary input aux (in contrast to the traditional hash proof system [6], [13] defined that $\mathcal{X}$ is independent from pk and simply generated by w). On the other hand, the probabilistic Enc' algorithm outputs $X \in \mathcal{X}\backslash\mathcal{L}$. The deterministic private evaluation algorithm $\mathsf{DecKey}(sk, X, \mathsf{aux})$ outputs K with sk, X and aux. For correctness, we require $\mathsf{Enckey}(pk, w, \mathsf{aux}) = \mathsf{DecKey}(sk, X, \mathsf{aux})$ if $X \in \mathcal{L}$. We assume that the above algorithms are efficiently computable.

We say that a hash proof system is 2-universal (resp., 2-universal′) if for all possible outcomes of the setup algorithm, the projective hash function is ε-almost 2-universal (resp., ε-almost 2-universal′) for some negligible fraction ε.

3.1.3 Subset Membership Problem

The subset membership problem is hard in HPS if $X \xleftarrow{\mathsf{R}} \mathsf{Enc}(1^k; w) \in \mathcal{L}$ is computationally indistinguishable from $X' \xleftarrow{\mathsf{R}} \mathsf{Enc}'(1^k) \in \mathcal{X}\backslash\mathcal{L}$. More formally, we define the advantage of an algorithm $\mathcal{A}$ breaking the subset membership problem as

$$\mathsf{Adv}^{\mathsf{SM}}_{\mathsf{HPS},\mathcal{A}}(k) := \left| \begin{array}{l} \Pr[\mathcal{A}(\mathcal{X}, \mathcal{L}, X) \to 1] \\ -\Pr[\mathcal{A}(\mathcal{X}, \mathcal{L}, X') \to 1] \end{array} \right|$$

where $(\mathcal{X}, \mathcal{L})$ is generated by $\mathsf{Setup}(1^k)$, $X \xleftarrow{\mathsf{R}} \mathsf{Enc}(1^k; w)$ and $X' \xleftarrow{\mathsf{R}} \mathsf{Enc}'(1^k)$.

Example : (DDH-based 2-universal′ hash proof system) We show an example of a 2-universal′ hash proof system based on Kurosawa-Desmedt key encapsulation mechanism [13]:

- $\mathcal{SK} := \mathbb{Z}_q^4$, $\mathcal{PK} := \mathbb{G}^2$, $\mathcal{X} := \mathbb{G}^2$, $\mathcal{T} := \mathbb{Z}_q$, $\mathcal{W} := \mathbb{Z}_q$, $\mathcal{K} := \mathbb{G}$.
- $\mathsf{group} := (\mathbb{G}, g_1, g_2, H)$ where $\mathbb{G}$ is a group of prime order q ($|q| = k$), $(g_1, g_2) \xleftarrow{\mathsf{U}} \mathbb{G}^2$ and $H : \mathbb{G}^2 \times \{0,1\}^* \to \mathcal{T}$ is a CR hash function.
- For $sk := (b_1, b_2, b_3, b_4) \in \mathcal{SK}$, Gen algorithm outputs $(B_1, B_2) := (g_1^{b_1} g_2^{b_2}, g_1^{b_3} g_2^{b_4})$.
- For $x \xleftarrow{\mathsf{U}} \mathcal{W}$, Enc algorithm outputs $(X_1, X_2) := (g_1^x, g_2^x) \in \mathcal{L}$.
- For $x \xleftarrow{\mathsf{U}} \mathcal{W}$ and $\mathsf{aux} := H(X_1, X_2, \mathsf{aux}')$ (aux' can be set as an arbitrary string), EncKey algorithm outputs $K := (B_1 B_2^{\mathsf{aux}})^x \in \mathcal{K}$.
- For $(X_1, X_2) \in \mathcal{X}$ and $\mathsf{aux} := H(X_1, X_2, \mathsf{aux}')$, DecKey algorithm outputs $K := X_1^{b_1 + \mathsf{aux} \cdot b_3} X_2^{b_2 + \mathsf{aux} \cdot b_4} \in \mathcal{K}$.

3.2 Authenticated Key Exchange Model

3.2.1 Basic initialization.

Let $\mathcal{A}, \mathcal{B}$ be the party's identity with static public keys A, B, respectively. The certified public key $\hat{A}$ ($\hat{B}$) binds each party's identity $\mathcal{A}$ ($\mathcal{B}$), static public key A (B) and its certificate. When a key exchange protocol between $\mathcal{A}$ and $\mathcal{B}$ is executed, party $\mathcal{A}$ is activated to execute an instance of the protocol called a session. The party executing the session is called the owner of the session and the other party is called the peer.

A session is uniquely determined by the session identifier of the form $\mathsf{sid} := (\mathsf{role}, \hat{A}, \hat{B}, X, Y)$, where $\mathsf{role} \in \{\text{initiator}, \text{responder}\}$ denotes the role of the owner of the session, owner $\hat{A}$ executes the session with peer $\hat{B}$, X is ephemeral public key output by $\mathcal{A}$ and Y is incoming ephemeral public key at the session. If there exists another session of the form $(\mathsf{role}', \hat{B}, \hat{A}, X, Y)$ where $\mathsf{role} \neq \mathsf{role}'$, this session is said to be matching session with $(\mathsf{role}, \hat{A}, \hat{B}, X, Y)$. If the party outputs the session key at the session, we call that the session is completed.

The adversary $\mathcal{M}$ is modeled as probabilistic polynomial-time Turing machine that controls all communications. All parties are activated by the adversary and the output messages are controlled by the adversary via $\mathsf{Send}(message)$ query. $\mathcal{M}$ can register static public key Z on behalf of party pid through $\mathsf{EstablishParty}(\mathsf{pid}, Z)$ query. If the adversary establishes party pid, the party is said to be dishonest. If a party is not dishonest, we call the party *honest*.

3.2.2 The extended Canetti-Krawczyk (eCK) security model

The eCK security model is proposed by LaMacchia, Lauter and Mityagin [14] and is based on the model in Section 3.2.1. Here, The adversary can issue the following queries:

- $\mathsf{SessionKeyReveal}(\mathsf{sid})$ The adversary obtains the session key for the completed session sid.
- $\mathsf{StaticKeyReveal}(\mathsf{pid})$ The adversary obtains the static secret key of party pid.
- $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$ The adversary obtains the ephemeral secret key for the session sid.

When adversary $\mathcal{M}$ issues a test query $\mathsf{Test}(\mathsf{sid}^*)$, the challenger flips a coin $\gamma \xleftarrow{\cup} \{0,1\}$. If $\gamma = 1$, $\mathcal{M}$ receives the actual session key SK^* of the test session sid^*. Otherwise, the challenger sends $\mathcal{M}$ a random key R^* where $R^* \xleftarrow{\cup} \{0,1\}^{|SK^*|}$. Finally, $\mathcal{M}$ guesses the flipped coin and outputs a bit $\gamma' \in \{0,1\}$. In the original eCK security, *fresh session* is defined as follows:

Definition 3. (fresh session of eCK security) Let $\mathsf{sid} := (\mathsf{role}, \hat{A}, \hat{B}, X^*, Y^*)$ be the session identifier executed by honest parties $\mathcal{A}$ and $\mathcal{B}$. If there exists the matching session to session sid, we denote the matching session as $\overline{\mathsf{sid}}$. Session sid is said to be fresh if none of the following conditions hold.

- $\mathcal{M}$ issues either
  - $\mathsf{SessionKeyReveal}(\mathsf{sid})$, or
  - $\mathsf{SessionKeyReveal}(\overline{\mathsf{sid}})$ (if $\overline{\mathsf{sid}}$ exists).
- If $\overline{\mathsf{sid}}$ exists, then $\mathcal{M}$ issues either
  - $\mathsf{StaticKeyReveal}(\mathcal{A})$ and $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$, or
  - $\mathsf{StaticKeyReveal}(\mathcal{B})$ and $\mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}})$.
- If $\overline{\mathsf{sid}}$ does not exist, then $\mathcal{M}$ issues either
  - $\mathsf{StaticKeyReveal}(\mathcal{A})$ and $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$, or
  - $\mathsf{StaticKeyReveal}(\mathcal{B})$.

Definition 4. (eCK security) Let the test session sid^* be fresh where adversary $\mathcal{M}$ issues $\mathsf{Test}(\mathsf{sid}^*)$. Then, we define the advantage of $\mathcal{M}$ by

$$\mathsf{AdvAKE}^{\mathsf{eCK}}_{\mathcal{M}}(k) := |2 \cdot \Pr[\gamma' = \gamma] - 1|.$$

A key exchange protocol is eCK-secure if the following conditions hold.

- If two honest parties complete matching sessions, they compute the same session key.
- For any probabilistic polynomial-time adversary $\mathcal{M}$, $\mathsf{AdvAKE}^{\mathsf{eCK}}_{\mathcal{M}}$ is negligible in k.

3.2.3 Leakage resilient eCK security model

The recently developed leakage resilient security is formalized by Akavia et al. [1] so that the adversary can send arbitrary functions and receive the output of the functions that the functions take as input secret key with the constraint that the total output length of all the functions is bounded. The above leakage oracle has different property from the existing *reveal* oracle in the key exchange security model, so we provide the leakage resilient security in the eCK security model. Following the definition of [17], we allow the adversary to submit arbitrary leakage function f and party's identity pid that the function takes as input the static secret key of pid. The adversary can obtain the output of the function. We add the following query to the eCK security model:

- $\mathsf{StaticKeyLeakage}(f, \mathsf{pid})$ The adversary obtains $f(ssk)$ where ssk denotes the static secret key of the party pid.

Definition 5. (fresh session of leakage resilient eCK security) Let $\mathsf{sid} := (\mathsf{role}, \hat{A}, \hat{B}, X^*, Y^*)$ be the session executed by honest owner $\mathcal{A}$ and peer $\mathcal{B}$. Consider that sid is $\mathcal{A}$'s i-th session. We define the matching session of sid as $\overline{\mathsf{sid}}$, if it exists. We say that session sid is λ-leakage fresh if the following conditions hold.

- sid is fresh session in the sense of eCK security.
- Before the adversary activates $\mathcal{A}$'s i-th session, the total output length of all the functions that the adversary issues the $\mathsf{StaticKey\ Leakage}$ query to $\mathcal{A}$ and $\mathcal{B}$ is at most λ, respectively.
- After $\mathcal{A}$'s i-th session is activated, the adversary issues neither $\mathsf{StaticKeyLeakage}(\cdot, \mathcal{A})$ nor $\mathsf{StaticKeyLeakage}(\cdot, \mathcal{B})$.

Definition 6. (leakage resilient eCK security) Let the test session sid^* be λ-leakage fresh where adversary $\mathcal{M}$ issues $\mathsf{Test}(\mathsf{sid}^*)$. Then, we define the advantage of $\mathcal{M}$ by

$$\mathsf{AdvAKE}^{\lambda\text{-}\mathsf{eCK}}_{\mathcal{M}}(k) := |2 \cdot \Pr[\gamma' = \gamma] - 1|.$$

A key exchange protocol is λ-leakage resilient eCK-secure if the following conditions hold.

- If two honest parties complete matching sessions, they compute the same session key.
- For any probabilistic polynomial-time adversary $\mathcal{M}$, $\mathsf{AdvAKE}^{\lambda\text{-}\mathsf{eCK}}_{\mathcal{M}}(k)$ is negligible in k.

In the leakage-resilient security, the adversary cannot issue the leakage oracle to $\mathcal{A}$ nor $\mathcal{B}$ after the test session sid^* is activated, and we do not consider the ephemeral secret key leakage in this model. We show that these oracle queries do not make sense for the key exchange security model.

In [17], Noar and Segev formalized the leakage resilient public key encryption and indicated that the adversary cannot issue the leakage query after the challenge ciphertext is given to the adversary. This restriction follows from the fact that the adversary can encode the decryption algorithm and the challenge ciphertext into a leakage function, and he can obtain the specified bit that the challenger flipped at the challenge phase.

The same result is also applied to the leakage resilient security for AKE. If the test session is activated, the session key of the session is uniquely determined by the identities of the party, the owner's ephemeral and static secret key, the peer's static public key and the ephemeral public key input to the owner. If the adversary obtains ephemeral secret key of the test session through the reveal oracle, the adversary can encode the specification of the key derivation of the AKE protocol with the ephemeral secret and the other public information into a leakage function. Thus the adversary can win the game if the leakage oracle is allowed to issue after activating the test session.

In contrast, we do not allow the adversary to issue the ephemeral secret key leakage at any time. If the adversary wants to obtain the leakage of an ephemeral secret key, the adversary must specify the session identifier for the oracle query. But the session identifier is determined, the session key is also uniquely determined as denoted in the previous paragraph. So, if we allow the adversary to issue this query after activating the test session, the same attack described in the previous paragraph can also be mounted. Therefore, leakage resilience on the ephemeral secret key is not suitable for key exchange security model and we only focus on the static secret key leakage.

4. THE PROPOSED PROTOCOL

We show that a variant of Okamoto protocol [18] is leakage resilient eCK-secure AKE protocol. The main idea is derived from the Naor-Segev leakage resilient public key encryption [17]. In the proposed protocol, we apply the strong extractors in the computation of ephemeral public key and session key to satisfy the leakage resilient security.

Intuitively, each party runs Enc algorithm of the hash proof system and ephemeral Diffie-Hellman protocol. The output of the EncKey (or, DecKey) algorithm and ephemeral Diffie-Hellman tuple are used for the seed of the (pair-wise independent) hash functions via the strong extractor to hold the leakage resilient security. However, the strong extractor needs a uniformly random extraction key (and it must be used one-time), so the party exchanges it each other. They computes two different keys from the hash proof system (i.e., $K_{A,1}$ and $K_{A,2}$) and applies the strong extractor with each exchanged extraction key, so the procedure to output the session key is more complex than the original protocol.

We remark that the computation of the ephemeral public key also needs the strong extractor. Different from [18], an adversary can obtain whole of the ephemeral key and λ-bit of the static secret key in the leakage resilient eCK security model. Since the NAXOS trick treats the randomness used for the ephemeral public key as the output of the function of the ephemeral secret key and static secret key, we must take care of the leakage resilience not only for the session key but also in the computation process of the ephemeral public key.

Let $k \in \mathbb{N}$ be a security parameter and $\mathbb{G}$ be a group of prime order q with $|q| = k$. Pick a random generator g from $\mathbb{G}$. Let $\mathsf{HPS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{EncKey}, \mathsf{Enc}', \mathsf{DecKey})$ be 2-universal$'$ hash proof system. Run $\mathsf{Setup}(1^k)$ and obtain $\mathsf{params} := (\mathsf{group}, \mathcal{SK}, \mathcal{PK}, \mathcal{X}, \mathcal{L}, \mathcal{T}, \mathcal{W}, \mathcal{K}, \Lambda_{(\cdot)}, \mu)$. Consider that Π_k is a probabilistic space for a certified public key (such as $\hat{A}$) and choose CR hash function $H : (\Pi_k)^2 \times \mathcal{X}^2 \times \mathbb{G}^2 \times \{0,1\} \to \mathcal{T}$ from a CR hash function family H. We assume that λ be a bound on the amount of leakage. Let $\mathsf{Ext}_1 : \mathcal{SK} \times \{0,1\}^{t(k)} \to \{0,1\}^{t'(k)}$ be average-case $(|\mathcal{SK}| - \lambda, \epsilon_1)$-strong extractor and $\mathsf{Ext}_2 : \mathcal{K} \times \{0,1\}^{t(k)} \to \{0,1\}^{\ell(k)}$ be average-case $(|\mathcal{K}| - \lambda, \epsilon_2)$-strong extractor, and $\mathsf{Ext}_3 : \mathbb{G} \times \{0,1\}^{t(k)} \to \{0,1\}^{\ell(k)}$ be average-case $(|\mathbb{G}|, \epsilon_3)$-strong extractor. Select πPRF $F := \mathsf{F}^{k,\Sigma_{\mathsf{F}},\mathcal{D}_{\mathsf{F}},\mathcal{R}_{\mathsf{F}}}$ and PRFs $\bar{F} := \mathsf{F}^{k,\Sigma_{\bar{\mathsf{F}}},\mathcal{D}_{\bar{\mathsf{F}}},\mathcal{R}_{\bar{\mathsf{F}}}}$, $\tilde{F} := \mathsf{F}^{k,\Sigma_{\tilde{\mathsf{F}}},\mathcal{D}_{\tilde{\mathsf{F}}},\mathcal{R}_{\tilde{\mathsf{F}}}}$, $\hat{F} := \mathsf{F}^{k,\Sigma_{\hat{\mathsf{F}}},\mathcal{D}_{\hat{\mathsf{F}}},\mathcal{R}_{\hat{\mathsf{F}}}}$ from a πPRF family F and PRF families $\bar{\mathsf{F}}, \tilde{\mathsf{F}}, \hat{\mathsf{F}}$ where

$$\Sigma_{\mathsf{F}} := \{0,1\}^{\ell(k)}, \mathcal{D}_{\mathsf{F}} := (\Pi_k)^2 \times \mathcal{X}^2 \times \mathbb{G}^2 \times \{0,1\}^{2t(k)}, \mathcal{R}_{\mathsf{F}} := \{0,1\}^{\ell(k)}$$
$$\Sigma_{\bar{\mathsf{F}}} := \{0,1\}^{\ell(k)}, \mathcal{D}_{\bar{\mathsf{F}}} := (\Pi_k)^2 \times \mathcal{X}^2 \times \mathbb{G}^2 \times \{0,1\}^{2t(k)}, \mathcal{R}_{\bar{\mathsf{F}}} := \{0,1\}^{\ell(k)},$$
$$\Sigma_{\tilde{\mathsf{F}}} := \{0,1\}^{t'(k)}, \mathcal{D}_{\tilde{\mathsf{F}}} := \{0,1\}^{k}, \mathcal{R}_{\tilde{\mathsf{F}}} := \mathcal{W} \times \mathbb{Z}_q$$
$$\Sigma_{\hat{\mathsf{F}}} := \{0,1\}^{k}, \mathcal{D}_{\hat{\mathsf{F}}} := \{0,1\}^{k}, \mathcal{R}_{\hat{\mathsf{F}}} := \mathcal{W} \times \mathbb{Z}_q.$$

Note that we can easily construct $(\tilde{\mathsf{F}}, \hat{\mathsf{F}})$ from any PRF functions which output the bit-length more than $|\mathcal{W}| + |\mathbb{Z}_q|$. The public parameter of our protocol is $(\mathsf{params}, \mathbb{G}, q, g, H, \mathsf{Ext}_1, \mathsf{Ext}_2, \mathsf{Ext}_3, F, \bar{F}, \tilde{F}, \hat{F})$.

$\mathcal{A}$ selects static secret key $a \xleftarrow{\cup} \mathcal{SK}$ and computes static public key $A := \mathsf{Gen}(1^k; a)$. Similarly, $\mathcal{B}$ chooses static secret key $b \xleftarrow{\cup} \mathcal{SK}$ and computes static public key $B := \mathsf{Gen}(1^k; b)$.

We describe the proposed key exchange protocol between initiator $\mathcal{A}$ and responder $\mathcal{B}$. At the beginning of the session, $\mathcal{A}$ performs the following.

1. Select ephemeral secret key $(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) \xleftarrow{\cup} \{0,1\}^{t(k)} \times \{0,1\}^k \times \{0,1\}^k$.
2. Set $\tilde{a} := \mathsf{Ext}_1(a, \tilde{x}_1)$ and $(x, x_1) := \tilde{F}_{\tilde{a}}(\tilde{x}_2) + \hat{F}_{\tilde{x}_3}(1^k)$ (as two-dimensional vectors).
3. Compute $X := \mathsf{Enc}(1^k; x)$, $X_1 := g^{x_1}$ and select $t_A \xleftarrow{\cup} \{0,1\}^{t(k)}$.
4. Erase $(\tilde{a}, x, x_1)$ and set $\mathsf{sid}_A := (\mathsf{initiator}, \hat{A}, \hat{B}, X, X_1, t_A, \cdot)$.
5. Send $(\hat{B}, \hat{A}, X, X_1, t_A)$ to $\mathcal{B}$ (the ephemeral public key of $\mathcal{A}$ is (X, X_1, t_A)).

When $\mathcal{B}$ receives $(\hat{B}, \hat{A}, X, X_1, t_A)$, $\mathcal{B}$ checks that $X \in \mathcal{X}$ and $X_1 \in \mathbb{G}$. If the verification holds, $\mathcal{B}$ executes the following.

1. Select ephemeral secret key $(\tilde{y}_1, \tilde{y}_2, \tilde{y}_3) \xleftarrow{\cup} \{0,1\}^{t(k)} \times \{0,1\}^k \times \{0,1\}^k$.
2. Set $\tilde{b} := \mathsf{Ext}_1(b, \tilde{y}_1)$ and $(y, y_1) := \tilde{F}_{\tilde{b}}(\tilde{y}_2) + \hat{F}_{\tilde{y}_3}(1^k)$ (as two-dimensional vectors).
3. Compute $Y := \mathsf{Enc}(1^k; y)$, $Y_1 := g^{y_1}$ and select $t_B \xleftarrow{\cup} \{0,1\}^{t(k)}$.
4. Erase $(\tilde{b}, y, y_1)$ and set $\mathsf{sid}_B := (\mathsf{responder}, \hat{B}, \hat{A}, X, X_1, t_A, Y, Y_1, t_B)$.
5. Send $(\hat{A}, \hat{B}, X, X_1, t_A, Y, Y_1, t_B)$ to $\mathcal{A}$ (the ephemeral public key of $\mathcal{B}$ is (Y, Y_1, t_B)).

Upon receiving $(\hat{A}, \hat{B}, X, X_1, t_A, Y, Y_1, t_B)$, $\mathcal{A}$ checks that she has the incomplete session of form sid_A and verifies that $Y \in \mathcal{X}$ and $Y_1 \in \mathbb{G}$. If the conditions hold, $\mathcal{A}$ updates the SID to $\mathsf{sid}_A := (\mathsf{initiator}, \hat{A}, \hat{B}, X, X_1, t_A, Y, Y_1, t_B)$.

$\mathcal{A}$ computes session key SK_A as follows:

1. Set $\mathsf{s} := (\hat{A}, \hat{B}, X, X_1, t_A, Y, Y_1, t_B)$.
2. Compute $\mathsf{aux}_i := H(\mathsf{s} \| i)$ for $i = \{1, 2, 3, 4\}$.
3. Compute $K_{A,1} := \mathsf{EncKey}(B, x, \mathsf{aux}_1) \oplus \mathsf{DecKey}(a, Y, \mathsf{aux}_2)$, $K_{A,2} := \mathsf{EncKey}(B, x, \mathsf{aux}_3) \oplus \mathsf{DecKey}(a, Y, \mathsf{aux}_4)$ and $K_{A,3} := Y_1^{x_1}$.
4. Set $\sigma_{A,1} := \mathsf{Ext}_2(K_{A,1}, t_A), \sigma_{A,2} := \mathsf{Ext}_2(K_{A,2}, t_B)$ and $\sigma_{A,3} := \mathsf{Ext}_3(K_{A,3}, t_A \oplus t_B))$.
5. Output $SK_A := F_{\sigma_{A,1}}(\mathsf{s}) \oplus F_{\sigma_{A,2}}(\mathsf{s}) \oplus \bar{F}_{\sigma_{A,3}}(\mathsf{s})$.

Along with $\mathcal{A}$'s computation, $\mathcal{B}$ sets $\mathsf{s}, \mathsf{aux}_i$ $(i = \{1, 2, 3, 4\})$ and computes $K_{B,1} := \mathsf{DecKey}(b, X, \mathsf{aux}_1) \oplus \mathsf{EncKey}(y, A, \mathsf{aux}_2)$, $K_{B,2} := \mathsf{DecKey}(b, X, \mathsf{aux}_3) \oplus \mathsf{EncKey}(y, A, \mathsf{aux}_4)$ and $K_{B,3} := X_1^{y_1}$. $\mathcal{B}$ sets $\sigma_{B,1} := \mathsf{Ext}_2(K_{B,1}, t_A)$, $\sigma_{B,2} := \mathsf{Ext}_2(K_{B,2}, t_B)$, $\sigma_{B,3} := \mathsf{Ext}_3(K_{B,3}, t_A \oplus t_B)$ and outputs the session key $SK_B := F_{\sigma_{B,1}}(\mathsf{s}) \oplus F_{\sigma_{B,2}}(\mathsf{s}) \oplus \bar{F}_{\sigma_{B,3}}(\mathsf{s})$.

From the definition of the hash proof system, we have

$$\begin{aligned}
\mathsf{EncKey}(B, x, \mathsf{aux}_1) &= \mathsf{DecKey}(b, X, \mathsf{aux}_1) \\
\mathsf{DecKey}(a, Y, \mathsf{aux}_2) &= \mathsf{EncKey}(y, A, \mathsf{aux}_2) \\
\mathsf{EncKey}(B, x, \mathsf{aux}_3) &= \mathsf{DecKey}(b, X, \mathsf{aux}_3) \\
\mathsf{DecKey}(a, Y, \mathsf{aux}_4) &= \mathsf{EncKey}(y, A, \mathsf{aux}_4)
\end{aligned}$$

and two party computes $K_{A,1} = K_{B,1}$, $K_{A,2} = K_{B,2}$ and $K_{A,3} = K_{B,3} = g^{x_1 y_1}$ (we call these keys as shared key). These variables are extracted by the exchanged extraction key (t_A, t_B) and we obtain $\sigma_{A,i} = \sigma_{B,i}$ for $i \in \{1, 2, 3\}$. Therefore, two party can compute the same session key $SK_A = SK_B$.

Unlike [18], each party computes two different values $K_{A,1}$ and $K_{A,2}$, and extracts the two variables with the exchanged extraction key to compute the secure session key. This stems from the fact that even in the presence of an active adversary, at least one of the two extraction keys is uniformly chosen by herself and it can be used to extract the proper randomness. Note that the other extraction key may be chosen by the adversary and it does not derive the sufficient randomness when we use it as the extraction key for the strong extractor.

For example, setting $k := 256$, the order of the group g is $|q| = k = 256$ (the order of the group for the DDH-based hash proof system is also set as 256-bits). If we consider that 56 bits of the secret key may be leaked ($\lambda = 56$) and $\epsilon_1 = \epsilon_2 = \epsilon_3 = 40$, we can extract 160 bits of the uniformly random variable and obtain 160 bits of the session key.

Theorem 1. Suppose that HPS is 2-universal$'$ hash proof system, the DDH assumption holds in $\mathbb{G}$, Ext_1 is average-case $(|\mathcal{SK}| - \lambda, \epsilon_1)$-strong extractor, Ext_2 is average-case $(|\mathcal{K}| - \lambda, \epsilon_2)$-strong extractor, Ext_3 is average-case $(|\mathbb{G}|, \epsilon_3)$-strong extractor, F is πPRF family with index $\{(I_\Sigma, f_\Sigma)\}_{k \in \mathbb{N}}$, where $I_\Sigma := \{(V, \mathsf{aux}, R, t) \mid (X, \mathsf{aux}, K, t) \in \mathcal{X} \times \mathcal{W} \times \mathcal{K} \times \{0,1\}^{t(k)}\}$ and $f_\Sigma : (V, \mathsf{aux}, R, t) \mapsto \mathsf{Ext}_2(\mathsf{DecKey}(r, V, \mathsf{aux}) \oplus K, t)$ with $r \xleftarrow{\cup} \mathcal{SK}$, $\bar{\mathsf{F}}, \tilde{\mathsf{F}}, \hat{\mathsf{F}}$ are PRF hash function family and H is CR hash function family. Then the proposed protocol is λ-leakage resilient eCK-secure where $\lambda := \min\{\log |\mathcal{SK}| - \omega(\log k) - t'(k), \log |\mathcal{K}| - \omega(\log k) - \ell(k)\}$.

We omit the security proof of this theorem due to space limitations and give a brief sketch of it here. Consider that an adversary chooses sid_A as the test session. To ensure the leakage resilient security, each party computes the ephemeral public key using the strong extractor Ext_1. Even if an adversary obtains λ-bits of the static secret key of $\mathcal{A}$ through the $\mathsf{StaticKeyLeakage}$ query, we have the following:

$$\begin{aligned}
&\langle (x, x_1), \tilde{a}, (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) \mid f(a) \rangle \\
&= \langle \tilde{F}_{\tilde{a}}(\tilde{x}_2) + \hat{F}_{\tilde{x}_3}(1^k), \mathsf{Ext}_1(a, \tilde{x}_1), (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) \mid f(a) \rangle \\
&\approx_{\epsilon_1} \langle \tilde{F}_{\tilde{a}}(\tilde{x}_2) + \hat{F}_{\tilde{x}_3}(1^k), \tilde{a}' \xleftarrow{\cup} \{0,1\}^{t'(k)}, (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) \mid f(a) \rangle \\
&\approx \langle (x', x_1') \xleftarrow{\cup} \mathcal{W} \times \mathbb{Z}_q, \tilde{a}' \xleftarrow{\cup} \{0,1\}^{t'(k)}, (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) \mid f(a) \rangle.
\end{aligned}$$

Note that the adversary can only obtain $\mathcal{A}$'s static secret key a or ephemeral secret key $(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$, so either one of the two seeds which input to the PRF function is unknown to the adversary and it derives uniformly random variables in $\mathcal{W} \times \mathbb{Z}_q$.

If there exists a matching session to the test session sid_A, $g^{x_1 y_1}$ is indistinguishable from random element in $\mathbb{G}$ under the DDH assumption and PRF $\bar{F}$ derives the uniformly random string. Otherwise, we can change $\mathsf{EncKey}(B, x, \mathsf{aux}_1)$ to $\mathsf{DecKey}(b, X, \mathsf{aux}_1)$ in the computation of $K_{A,1}$ and the subset membership problem provides $X \in \mathcal{L}$ is indistinguishable from $X \in \mathcal{X} \backslash \mathcal{L}$. Then,

1. $K_{A,1}$ is pairwisely independent from the other shared keys (in particular, computed by $\mathcal{A}$ and $\mathcal{B}$) if the adversary obtains no information about b since the hash proof system is 2-universal$'$.
2. Even if the adversary issues leakage oracle query and obtains partial information about $\mathcal{B}$'s static secret key, average min-entropy of $\mathsf{DecKey}(b, X, \mathsf{aux}_1)$ conditioned on all public information, leakage information on b and any one of the other shared keys is at least $|\mathcal{K}| - \lambda$.
3. By applying the strong extractor Ext_2, the extracted variable from $K_{A,1}$ becomes pairwisely independent from other variables even in the condition that the adversary obtains λ-bits information about $\mathcal{B}$'s static secret key.

Therefore, $\mathcal{A}$ derives a pairwisely independent variable in the computation of the session key and the πPRF derives the session key that is indistinguishable from truly random string. Note that we require 2-universal$'$ (not 2-universal)

Table 1: Comparison with Existing eCK-Secure Two-Pass AKE Protocols

	CMQV [20]	SEB [19]	Okamoto [18]	MO [16]	Proposed
Assumptions	GDH	GDH	DDH,CR, πPRF	DDH,CR, πPRF	DDH,SM,CR, πPRF
Implementation trick	NAXOS	-	NAXOS	-	NAXOS
Random oracle	Yes	Yes	No	No	No
λ-leakage security	No	No	No	No	Yes

hash proof system since the adversary can send $X \in \mathcal{X}\backslash\mathcal{L}$ to $\mathcal{B}$ and issues session key reveal query to $\mathcal{B}$ if the session is not matching to sid_A.

5. SECURITY AND COMPLEXITY EVALUATION

In Table 1, we compare the security of the proposed protocols with several existing eCK-secure two-pass AKE protocols, CMQV [20], Sarr-Elbaz-Bajard (SEB) [19], Okamoto [18] and Moriyama-Okamoto (MO) [16]. In comparison with the previous protocols, the proposed protocols satisfies λ-leakage resilient security in the eCK security model.

6. REFERENCES

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC 2009*, volume 5444 of *LNCS*, pages 474–495, 2009.

[2] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 113–134, 2010.

[3] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO 2009*, volume 5677 of *LNCS*, pages 36–54, 2009.

[4] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474, 2001.

[5] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64, 2002.

[6] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, 33(1):167–226, 2003.

[7] Y. Dodis, K. Haralambiev, A. Lopez-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631, 2010.

[8] Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In *STOC 2009*, pages 621–630, 2009.

[9] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2009.

[10] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum:. Leakage-resilient signatures. In *TCC 2010*, volume 5978 of *LNCS*, pages 343–360, 2010.

[11] J. Katz. Signature schemes with bounded leakage resilience. In *ASIACRYPT 2009*, volume 5978 of *LNCS*, pages 703–720, 2009.

[12] M. Kim, A. Fujioka, and B. Ustaoglu. Strongly secure authenticated key exchange without NAXOS' approach. In *IWSEC 2009*, volume 5824 of *LNCS*, pages 174–191, 2009.

[13] K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 426–442, 2004.

[14] B. A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *ProvSec 2007*, volume 4784 of *LNCS*, pages 1–16, 2007.

[15] S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In *TCC 2004*, volume 2951 of *LNCS*, pages 278–296, 2004.

[16] D. Moriyama and T. Okamoto. An eck-secure authenticated key exchange protocol without random oracles. In *Provsec 2009*, volume 5848 of *LNCS*, pages 154–167, 2009.

[17] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO 2009*, volume 5677 of *LNCS*, pages 18–35, 2009.

[18] T. Okamoto. Authenticated key exchange and key encapsulation without random oracles. Cryptology ePrint Archive, Report 2007/473, 2007.

[19] A. P. Sarr, P. Elbaz-Vincent, and J.-C. Bajard. An efficient authenticated diffie-hellman protocol. Cryptology ePrint Archive, Report 2009/408, 2009.

[20] B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, 46(3):329–342, 2008. Cryptology ePrint Archive, Report 2007/123.

[21] B. Ustaoglu. Comparing session state reveal and ephemeral key reveal for diffie-hellman protocols. In *Provsec 2009*, volume 5848 of *LNCS*, pages 183–197, 2009.

[22] J. Wu and B. Ustaoglu. Efficient key exchange with tight security reduction. Cryptology ePrint Archive, Report 2009/288, 2009.

[23] S. Wu and Y. Zhu. A framework for authenticated key exchange in the standard model. In *ISPEC 2009*, volume 5451 of *LNCS*, pages 207–218, 2009.