A Lightweight Generic Compiler for Authenticated Key Exchange from Non-interactive Key Exchange with Auxiliary Input

Zheng Yang[1], Chao Liu[1], Wanping Liu[1], Song Luo[1], Hua Long[1] and Shuangqing Li[2]
(Corresponding author: Hua Long)
School of Computer Science and Engineering, Chongqing University of Technology[1]
Chongqing 400054, China
College of Computer Science, Chongqing University[2]
Chongqing 400044, China
(Email: cqlongman@163.com)
(Received Sept. 3, 2015; revised and accepted Dec. 7 & Dec. 15, 2015)

Abstract

We introduce a new lightweight generic compiler that is able to transform any passively forward secure two-message key exchange (KE) protocols into authenticated key exchange (AKE) protocols with security in the presence of active adversaries who can reveal critical session specific information such as long-term or ephemeral secrets and can establish malicious parties. The compiler is built based on a new security notion regarding non-interactive key exchange with auxiliary input (NIKEA). The NIKEA is able to provide two security properties on the confidentiality and the unforgeability of shared key. Our new compiler is a very useful tool for the design of new AKE protocols in a modular and efficient way, that is suitable for resources constrained devices.

Keywords: Authenticated key exchange, non-interactive key exchange, protocol compiler, standard model

1 Introduction

Authenticated key exchange (AKE) is a cryptographic primitive which enables two parties to compute a session key with an assurance that the generated key is only known by these intended communication partners. In many application systems, AKE protocols usually serve as an important building block to protect the communication data over insure networks.

AKE Compilers. It is known to be a generic security strengthening transformation that pushes forward the modular design of AKE protocols. An interesting fashion of AKE compiler is to securely combine authentication protocols (AP) with passively secure key exchange protocols (KE) to yield AKE protocols that is referred as AP&KE style compiler [14] in the sequel, see the works in [14, 16, 18]. In this paper, we focus on a variant of this style AKE compiler where the implicit key authentication is guaranteed (instead of the explicit mutual authentication in previous works). Several advantages of AKE compilers are worth highlighting. First of all one could realize a AKE protocol with a rich collection of existing authentication and key exchange protocols which are specifically fit to a certain application scenario. On the second, a generic compiler would be very useful to avoid any modifications (which are often costly or error-prone in practice) in existing implementations of the input sub-protocols. To the last but not least, it could simplify the security analysis of the entire system, where the security of any resulting AKE protocol is directly inherited from the security proof of the AKE compiler.

While reviewing existing AKE compilers[14, 16, 18], we notice that they might be not efficient enough. Katz and Yung presented a generic compiler (which is referred to as KY compiler) for building group authenticated key exchange [16] based on passively secure group key exchange and digital signature. The KY compiler needs an additional communication round to the input protocol, in which each party chooses a random nonce and broadcasts it to its communication partners. In 2010, Jager et al. [14] introduced the first compiler (called as JKSS compiler) which accounts only for a constant number of additional messages (which is independent of the KE protocol) to be exchanged. But this scheme requires the KE protocol to output the session key to the compiler (unlike the KY compiler) and increases three additional communication

rounds in the compiler that might be not practical. Most recently, Li et al. [18] proposed three new AP&KE style compilers (which are referred to as LSYBS compilers). Unlike the KY and JKSS compilers, no nonce is required in the LSYBS compilers which instead rely on the entropy of the ephemeral keys of KE. As a result the LSYBS compilers are more round efficient than KY and JKSS compilers. However, we find out that all these compilers increase the communication rounds to the compiled KE protocols. Thus they might be not suitable for power constrained devices which need low latency of communication. In addition, the LSYBS compiler shows that if a KE protocol without long-term key is passively secure then each protocol message generated by the ephemeral generation function (EKGen) is unique. This uniqueness property is what enables the LSYBS to get rid of the random nonce used in previous compilers such as KY compiler and the compiler introduced by Jager et al. [14] (which will be referred to as JKSS compiler). However, their restriction on EKGen rules out a lot of key exchange protocol with long-term key. In this work we therefore try to broaden the range of KE that a AKE compiler can work on, i.e., without putting restriction on specific ephemeral key generation function.

Recently, Boyd et al. [4] and Cremers et al. [8] proposed two compilers respectively for two-message AKE protocols. However, these two compilers all aimed to compile (e)CK secure two-message protocols to achieve perfect forward secrecy without increasing protocol round and changing the internal execution of compiled protocols. However, they need very strong assumptions on the compiled protocols, i.e., they should be proved secure in the CK [5] model or the eCK model [17]. On the other hand, the (e)CK secure protocols without random oracles are inefficient. The computational costs of these two compilers are basically less computational efficient than above AP&KE style AKE compiler. But, to our best of knowledge, it is still an open question on how to build AP&KE style AKE compiler without increasing protocol round.

Non-interactive key exchange. Non-interactive key exchange (NIKE) is introduced to allow two parties to calculate a shared key based only on their long-term public keys without any interaction. NIKE has many real-world applications, e.g., establishing keys and enabling secure communications in mobile ad hoc and sensor networks where the energy cost of communication is prime concern [6, 10, 11]. The formal security of NIKE was studied by Freire et al. [10]. However, the limitation of NIKE is also obvious that it lacks of some important security properties of AKE, such as perfect forward secrecy or resilience of known key attacks. Once the long-term private key or the shared key of honest parties is leaked somehow then the security of the system cannot be guaranteed anymore. Hence we try to figure out a solution on key establishment to make a trade-off between round efficiency and AKE security properties.

Contributions. In this paper, we first present a new notion concerning non-interactive key exchange with auxiliary input (NIKEA). In contrast to ordinary NIKE [10], the share key of NIKEA is generated relying on long-term keys and an auxiliary input string aux (which could be for example timestamps, constants or other public information). Intuitively, the shared key generated with different aux would be distinct. Hence the leakage of some shared key may not affect the security of other shared key with distinct aux. This leads the confidential security property of NIKEA to be stronger than that of NIKE. Besides, the NIKEA has another interesting security property on unforgeability that adversary is unable to generated the shared key of uncorrupted honest parties with an auxiliary input aux that is not used by these parties before. A concrete NIKEA scheme is proposed, which is derived from the pairing-based NIKE scheme in [10]. Moreover we somehow optimize the algorithms to make it to be more efficient and practical. Namely we require the certificate authority to check the validity of registered public key rather than doing so in each execution of shared key generation. The new NIKEA scheme is proven secure without random oracles under standard assumptions.

On the next we introduce a new lightweight AP&KE style compiler that generically build secure AKE from secure NIKEA protocols and two-message passively forward secure two-message key exchange protocols. Namely we take the NIKEA as an authentication protocol. We observe that the forward secrecy property of KE would lead the message transcript of each session to be unique among its owner's sessions. One of the reasons that we choose NIKEA as our building block is that it can be efficiently realized. It is remarkable that the new compiler does not require any modifications in the underlying KE and underlying application based on such KE. It is thus easily applicable to existing systems what makes it to be very useful in real world applications. The main idea is to take the message outputted by each KE instance as the auxiliary input of NIKEA, where the generated shared key is used as one-time authentication token for such KE message. Unlike previous compilers [14, 16, 18], we do not increase any protocol round. All communication can be done within two moves. In addition, the generic compiler can also be efficiently instantiated for instance with concrete Diffie-Hellman based KE and NIKEA. Then the computational cost is approximately dominated by only three exponentiations. In a nutshell, the proposed compiler is suitable for resources constrained application environment (such as sensor networks). Furthermore, the security analysis of the compiler is given in the standard model, i.e., without assuming random oracles. The security result shows that our compiler satisfies well-known desirable security properties including resilience of chosen identity and public key attacks, known session key attacks and leakage of ephemeral secrets (from sessions non-related to test session), and provision of perfect forward secrecy. Although the resilience of key compromise impersonation attacks is not covered by our compiler, we

believe that it would still meet the security requirement in most applications.

Other Related Works. In our work an important AKE security property that we care about is the perfect forward secrecy. It is notorious that the PFS for TMAKE is non-trivial to achieve. 1n 2012, Cremers and Feltz [8] proposed a stronger security model (referred to as eCKw) to reformulate the wPFS notion based on a new concept so called *origin-session*. The resultant model is claimed to provide a slightly stronger form of wPFS than eCK model's. On the second, they further develop eCKw to model PFS that yields another new model (which is referred to as eCK-PFS). More interestingly, it is possible to transform any eCKw secure protocol (e.g. [23]) to be eCK-PFS secure using the signature based compiler in [8]. The implication relationship between eCK and eCKw models was studied in literature [8, 24]. In 2016, Yang and Zhang [25] introduced a new authenticated group key exchange (AGKE) model named g-eCK-PFS which particularly covers PFS. These above models (e.g. eCK-PFS and g-eCK-PFS) consider the security of AKE protocol in a very strong sense. This also leads the protocols being secure in these models to be inefficiency.

Some other GAKE protocols, for instance. [7, 9, 12, 19, 21, 22] have been recently proposed from different motivations. But the efficiency still needs to be optimized somehow. We stress that our construction idea can also be used in the group case to build efficient AGKE protocol. But the group NIKEA is required then.

2 Preliminaries and Definitions

In this section, we describe the cryptographic building blocks that will be used in the rest of Sections. The set of integers between 1 and n is denoted by $[n] = \{1, \ldots, n\}$. The notion $a \xleftarrow{\$} S$ denotes the action of sampling a uniformly random element a from a set S. Let '||' denote the operation concatenating two binary strings. Let $\mathcal{IDS}$ be an identity space.

2.1 Target Collision-Resistant Hash Functions

Let $\mathsf{TCRHF} : \mathcal{K}_{\mathsf{TCRHF}} \times \mathcal{M}_{\mathsf{TCRHF}} \to \mathcal{Y}_{\mathsf{TCRHF}}$ be a family of keyed-hash functions associated with key space $\mathcal{K}_{\mathsf{TCRHF}}$, message space $\mathcal{M}_{\mathsf{TCRHF}}$ and hash value space $\mathcal{Y}_{\mathsf{TCRHF}}$. The public key $hk_{\mathsf{TCRHF}} \in \mathcal{K}_{\mathsf{TCRHF}}$ of a hash function $\mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, \cdot)$ is generated by a PPT algorithm $\mathsf{TCRHF.KG}(1^\kappa)$ on input security parameter κ. If the hash key hk_{TCRHF} is obvious from the context, we write $\mathsf{TCRHF}(m)$ for $\mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, m)$.

Definition 1. TCRHF *is called* $(t_{\mathsf{TCRHF}}, \epsilon_{\mathsf{TCRHF}})$*-target-collision-resistant if for all* t_{TCRHF}*-time adversaries* $\mathcal{A}$ *it holds that*

$$\Pr\left[\begin{array}{c} hk_{\mathsf{TCRHF}} \xleftarrow{\$} \mathsf{TCRHF.KG}(1^\kappa), \\ m \xleftarrow{\$} \mathcal{M}_{\mathsf{TCRHF}}, \\ m' \leftarrow \mathcal{A}(1^\kappa, hk_{\mathsf{TCRHF}}, m), \\ m \neq m',\ m' \in \mathcal{M}_{\mathsf{TCRHF}}, \\ \mathsf{TCRHF}(m) = \mathsf{TCRHF}(m') \end{array}\right] \leq \epsilon_{\mathsf{TCRHF}},$$

where the probability is over the random bits of $\mathcal{A}$.

Normally target collision resistant functions can be realized with a specific cryptographic hash function such as MD5 and SHA.

2.2 Pseudo-Random Functions

Let $\mathsf{PRF} : \mathcal{K}_{\mathsf{PRF}} \times \mathcal{D}_{\mathsf{PRF}} \to \mathcal{R}_{\mathsf{PRF}}$ denote a family of deterministic functions, where $\mathcal{K}_{\mathsf{PRF}}$ is the key space, $\mathcal{D}_{\mathsf{PRF}}$ is the domain and $\mathcal{R}_{\mathsf{PRF}}$ is the range of PRF for security parameter κ. Let $\mathsf{RF} : \mathcal{D}_{\mathsf{PRF}} \to \mathcal{R}_{\mathsf{PRF}}$ be a stateful uniform random function which takes as input a distinct message $x \in \mathcal{D}_{\mathsf{PRF}}$, and outputs a random element $y \xleftarrow{\$} \mathcal{R}_{\mathsf{PRF}}$. The input message x of RF and its output y is one-to-one map.

Definition 2. *We say that* PRF *is a* $(t, \epsilon_{\mathsf{PRF}})$*-secure pseudo-random function family, if it holds that* $|\Pr[\mathsf{EXP}^{ind-cma}_{\mathsf{PRF},\mathcal{A}}(\kappa) = 1] - 1/2| \leq \epsilon_{\mathsf{PRF}}$ *for all adversaries* $\mathcal{A}$ *that make a polynomial number of oracle queries* q *while running in time at most* t *in the following experiment:*

$\mathsf{EXP}^{ind-cma}_{\mathsf{PRF},\mathcal{A}}(\kappa)$	$\mathcal{F}(b, x)$
$b \xleftarrow{\$} \{0,1\}, k \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$;	*If* $x \notin \mathcal{D}_{\mathsf{PRF}}$ *then return* $\perp$*;*
$b' \leftarrow \mathcal{A}^{\mathcal{F}(b,\cdot)}(\kappa)$;	*If* $b = 1$ *then return* $\mathsf{PRF}(k, x)$;
If $b = b'$ *then return 1;*	*Otherwise return* $\mathsf{RF}(x)$;
Otherwise return 0;	

where $\epsilon_{\mathsf{PRF}} = \epsilon_{\mathsf{PRF}}(\kappa)$ *is a negligible function in the security parameter* κ*, and the number of allowed queries* q *is bound by* t.

2.3 The Bilinear Decision Diffie-Hellman Assumption

We first briefly recall some of the basic properties of symmetric bilinear groups. The bilinear groups will be parametrized by a symmetric pairing parameter generator, denoted by $\mathsf{PG.Gen}$. This is a polynomial time algorithm that on input a security parameter 1^κ, returns the description of two multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of the same prime order p, generators g for $\mathbb{G}$, and a bilinear computable pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The formal description of of the properties of such pairing operation can be found in [3], which is omitted here.

Let $\mathcal{PG} : (\mathbb{G}, g, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathsf{PG.Gen}(1^\kappa)$ denote the description of symmetric bilinear groups. The Bilinear Decisional Diffie-Hellman (BDDH) problem [15] is stated as follows: given the tuple $(a, b, c, \gamma) \in \mathbb{Z}_p$ as input, and output yes if $e(g, g)^\gamma = e(g, g)^{abc}$ and no otherwise..

Definition 3. *We say that the* BDDH *problem relative to generator* PG.Gen *is* $(t, \epsilon_{\mathsf{BDDH}})$*-hard, if the probability bound* $|\Pr[\mathsf{EXP}^{bddh}_{\mathsf{PG.Gen},\mathcal{A}}(\kappa) = 1] - 1/2| \leq \epsilon_{\mathsf{BDDH}}$ *holds for all adversaries* $\mathcal{A}$ *running in probabilistic polynomial time* t *in the following experiment:*

$\mathsf{EXP}^{bddh}_{\mathsf{PG.Gen},\mathcal{A}}(\kappa)$
$\mathcal{PG} = (\mathbb{G}, g, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathsf{PG.Gen}(1^\kappa)$;
$(a, b, c, \gamma) \xleftarrow{\$} \mathbb{Z}_p^*$; $b \xleftarrow{\$} \{0, 1\}$;
if $b = 1$ $\Gamma \leftarrow e(g, g)^{abc}$, *otherwise* $\Gamma \leftarrow e(g, g)^\gamma$;
$b' \leftarrow \mathcal{A}(1^\kappa, \mathcal{PG}, g^a, g^b, g^c, \Gamma)$;
if $b = b'$ *then return 1, otherwise return 0;*

where $\epsilon_{\mathsf{BDDH}} = \epsilon_{\mathsf{BDDH}}(\kappa)$ *is a negligible function in the security parameter* κ.

2.4 Notations for Two-message KE

In a two-message AKE protocol (TMKE), each party may send a single 'message'. The key exchange procedure is done within two pass and a common shared session key is generated to be known only by session participants, which is shown in Figure 1.

A general TMKE protocol may consist of four polynomial time algorithms $(\mathsf{TMKE.ST}, \mathsf{TMKE.KG}, \mathsf{TMKE.MSG}, \mathsf{TMKE.SKG})$ with following semantics:

- $pms \leftarrow \mathsf{TMKE.ST}(1^\kappa)$: On input 1^κ, outputs pms, a set of system parameters.
- $(sk^{ke}_{\mathsf{ID}}, pk^{ke}_{\mathsf{ID}}) \xleftarrow{\$} \mathsf{TMKE.KG}(pms, \mathsf{ID})$: This algorithm takes as input system parameters pms and a party's identity $\mathsf{ID} \in \mathcal{IDS}$, and outputs a pair of long-term private/public key $(sk^{ke}_{\mathsf{ID}}, pk^{ke}_{\mathsf{ID}}) \in \{\mathcal{PK}, \mathcal{SK}\}$.
- $m_{\mathsf{ID}_1} \xleftarrow{\$} \mathsf{TMKE.MSG}(pms, sk^{ke}_{\mathsf{ID}_1}, \mathsf{ID}_2, pk^{ke}_{\mathsf{ID}_2}, r_{\mathsf{ID}_1}, m_{\mathsf{ID}_2})$: This algorithm takes as input system parameters pms and the sender ID_1's secret key $sk^{ke}_{\mathsf{ID}_1}$, the intended receiver ID_2's public key $pk^{ke}_{\mathsf{ID}_2}$, a randomness $r_{\mathsf{ID}_1} \xleftarrow{\$} \mathcal{R}_{\mathsf{TMKE}}$ and a message $m_{\mathsf{ID}_2} \in \mathcal{M}_{\mathsf{TMKE}}$ from party ID_2, and outputs a message $m_{\mathsf{ID}_1} \in \mathcal{M}_{\mathsf{TMKE}}$ to be sent, where $\mathcal{R}_{\mathsf{TMKE}}$ is the randomness space and $\mathcal{M}_{\mathsf{TMKE}}$ is message space. We remark that the secret key $sk^{ke}_{\mathsf{ID}_1}$ of sender, the identity ID_2 and public key $pk^{ke}_{\mathsf{ID}_2}$ of receiver are only optional for generating the message.[1]
- $K \leftarrow \mathsf{TMKE.SKG}(pms, sk^{ke}_{\mathsf{ID}_1}, \mathsf{ID}_2, pk^{ke}_{\mathsf{ID}_2}, r_{\mathsf{ID}_1}, m_{\mathsf{ID}_2})$: This algorithm take as the input system parameters pms and ID_1's secret key sk_{ID_1}, a public key $pk^{ke}_{\mathsf{ID}_2}$ of ID_2, a randomness $r_{\mathsf{ID}_1} \xleftarrow{\$} \mathcal{R}_{\mathsf{TMKE}}$ and a received message m_{ID_2} from party ID_2, and outputs session key $K \in \mathcal{K}_{\mathsf{TMKE}}$, where $\mathcal{K}_{\mathsf{TMKE}}$ is the session key space.

We say that the TMKE.SKG algorithm is correct, if for all $(sk^{ke}_{\mathsf{ID}_1}, pk^{ke}_{\mathsf{ID}_1}) \xleftarrow{\$} \mathsf{TMKE.KG}(\mathsf{ID}_1)$ and $(sk^{ke}_{\mathsf{ID}_2}, pk^{ke}_{\mathsf{ID}_2}) \xleftarrow{\$} \mathsf{TMKE.KG}(\mathsf{ID}_2)$, for all $r_{\mathsf{ID}_1}, r_{\mathsf{ID}_2} \xleftarrow{\$} \mathcal{R}_{\mathsf{TMKE}}$ and for all messages $m_{\mathsf{ID}_1} \xleftarrow{\$} \mathsf{TMKE.MSG}(sk^{ke}_{\mathsf{ID}_1}, \mathsf{ID}_2, pk^{ke}_{\mathsf{ID}_2}, r_{\mathsf{ID}_1}, \emptyset)$ and $m_{\mathsf{ID}_2} \xleftarrow{\$} \mathsf{TMKE.MSG}(sk^{ke}_{\mathsf{ID}_2}, \mathsf{ID}_1, pk^{ke}_{\mathsf{ID}_1}, r_{\mathsf{ID}_2}, m_{\mathsf{ID}_1})$, it holds that

$$\mathsf{TMKE.SKG}(sk_{\mathsf{ID}_1}, \mathsf{ID}_2, pk_{\mathsf{ID}_2}, r_{\mathsf{ID}_1}, m_{\mathsf{ID}_2}) = \mathsf{TMKE.SKG}(sk_{\mathsf{ID}_2}, \mathsf{ID}_1, pk_{\mathsf{ID}_1}, r_{\mathsf{ID}_2}, m_{\mathsf{ID}_1})$$

A the system initiation phase, the parameters would be generated as $pms \leftarrow \mathsf{TMKE.ST}(1^\kappa)$, where pms might be ignored in the description of other algorithms of TMKE for simplicity. The Figure 1 briefly illustrates the generic protocol execution of TMKE on input pms.

Please note that if in the above execution, if the party ID_2's message m_{ID_2} is generated to be independent of m_{ID_1} then the TMKE is a one-round AKE protocol, i.e. $m_{\mathsf{ID}_2} \xleftarrow{\$} \mathsf{TMKE.MSG}(sk^{ke}_{\mathsf{ID}_2}, \mathsf{ID}_1, pk^{ke}_{\mathsf{ID}_1}, r_{\mathsf{ID}_2}, \emptyset)$. The independence property of one-round AKE enables parties to run protocol instances simultaneously (which is a key feature of one-round protocol).

3 Non-Interactive Key Exchange with Auxiliary Input

In the subsection, we introduce a new security notion regarding non-interactive key exchange with auxiliary input (NIKEA).

3.1 Notions for Non-Interactive Key Exchange with Auxiliary Input

We consider a NIKEA scheme in the public key setting consists of three algorithms: NIKEA.Setup, NIKEA.KG and NIKEA.ShareKey associated with an identity space $\mathcal{IDS}$ and a shared key space $\mathcal{K}_{\mathsf{NIKEA}}$, in which those algorithms are defined as follows:

- $pms^{nikea} \leftarrow \mathsf{NIKEA.Setup}(1^\kappa)$: This algorithm takes as input a security parameter κ and outputs a set of system parameters pms^{nikea}. The parameters pms^{nikea} might be implicitly used by other algorithms for simplicity.
- $(sk_{\mathsf{ID}}, pk_{\mathsf{ID}}, \mathsf{pf}_{\mathsf{ID}}) \xleftarrow{\$} \mathsf{NIKEA.KG}(\mathsf{ID})$: This algorithm takes as input an identity ID, and outputs a pair of long-term secret/public key $(sk_{\mathsf{ID}}, pk_{\mathsf{ID}})$ and corresponding proof $\mathsf{pf}_{\mathsf{ID}}$ for key registration.
- $K \leftarrow \mathsf{NIKEA.ShareKey}(\mathsf{ID}_1, sk_{\mathsf{ID}_1}, \mathsf{ID}_2, pk_{\mathsf{ID}_2}, aux)$: This algorithm takes as input an identity ID_1, a secret key sk_{ID_1} along with another identity ID_2 and corresponding public key pk_{ID_2}, and an auxiliary input string $aux \in \{0, 1\}^*$, and outputs either a shared key $K \in \mathcal{K}_{\mathsf{NIKEA}}$ for the two parties, or a failure symbol $\perp$. This algorithm is assumed to always output $\perp$ if the input identities are not distinct.

[1] Please note that if ID_1 is initiator then $m_{\mathsf{ID}_2} = \emptyset$.

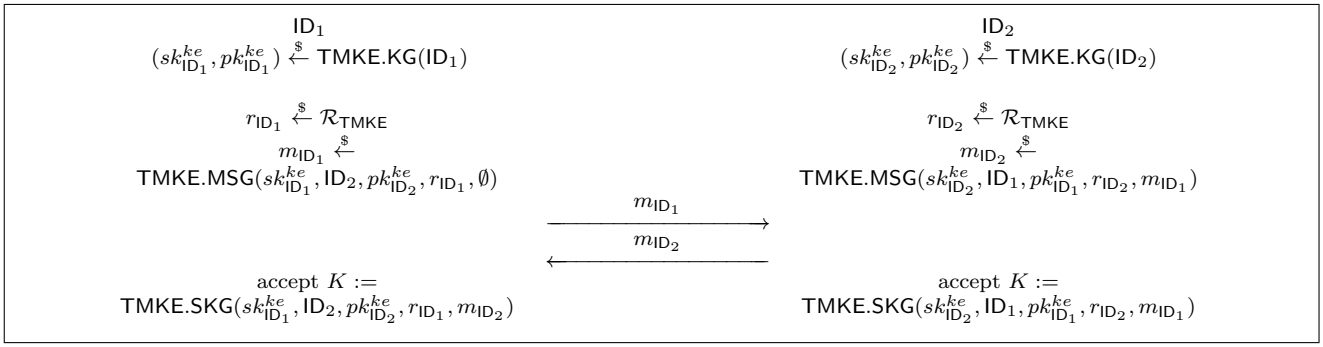

Figure 1: General TMKE protocol

For correctness, we require that, for a tuple of identities $(\mathsf{ID}_1, \mathsf{ID}_2)$, and corresponding key pairs $(sk_{\mathsf{ID}_1}, pk_{\mathsf{ID}_1})$ and $(sk_{\mathsf{ID}_2}, pk_{\mathsf{ID}_2})$ and the same aux, the algorithm $\mathsf{NIKEA.ShareKey}$ should satisfy the constraint:

– $\mathsf{NIKEA.ShareKey}(\mathsf{ID}_1, sk_{\mathsf{ID}_1}, \mathsf{ID}_2, pk_{\mathsf{ID}_2}, aux) =$ $\mathsf{NIKEA.ShareKey}(\mathsf{ID}_2, sk_{\mathsf{ID}_2}, \mathsf{ID}_1, pk_{\mathsf{ID}_1}, aux)$

One of the differences between the notion of NIKE in [10] and the above notion of NIKEA is that the $\mathsf{NIKEA.ShareKey}$ algorithm (in the later notion) requires an additional input aux for shared key generation. The NIKE in [10] can be seen as a special NIKEA with empty $aux = \emptyset$. In order to run the NIKEA correctly, two parties should share the aux somehow. For example the aux could be synchronized timestamps or constants or other public information.[2] In contrast to the NIKE, the NIKEA might be useful to generate either one-time shard key or authentication token for aux (see our upcoming AKE proposal). In a nutshell, the NIKEA can provide more functions than NIKE. In the following, we formally describe the security notion of NIKEA.

3.2 Security Definition for NIKEA

We describe the formal security model for two party PKI-based NIKEA protocols, that is modified from the CKS-like model [10] for NIKE. Besides we do slightly modification on modelling public key registration. Specifically, each party ID_i might be required to provide extra information (denoted by $\mathsf{pf}_{\mathsf{ID}_i}$) to prove that the registered public key is sound. Let $\{\mathsf{Honest}, \mathsf{Dishonest}\}$ be two vector lists. In order to formulate the capabilities of active adversaries against NIKEA, the adversaries are allowed to ask the following queries:

- $\mathsf{RegisterHonest}(\mathsf{ID})$: On input an identity $\mathsf{ID} \in \mathcal{IDS}$, if $\mathsf{ID} \notin \{\mathsf{Honest}, \mathsf{Dishonest}\}$ then $\mathcal{C}$ runs $\mathsf{NIKEA.KG}(pms^{nikea}, \mathsf{ID})$ to generate a long-term secret/public key pair $(sk_{\mathsf{ID}}, pk_{\mathsf{ID}}) \in (\mathcal{PK}, \mathcal{SK})$ and adds the tuple $(\mathsf{ID}, sk_{\mathsf{ID}}, pk_{\mathsf{ID}})$ into the list Honest, and returns pk to $\mathcal{A}$; as otherwise a failure symbol $\perp$ is returned. This query is allowed to be asked at most twice. Parties established by this query are called honest.

- $\mathsf{EstablishParty}(\mathsf{ID}_\tau, pk_{\mathsf{ID}_\tau}, \mathsf{pf}_{\mathsf{ID}_\tau})$: This query allows the adversary to register an identity ID_τ and a long-term public key pk_{ID_τ} on behalf of a party ID_τ, if the $\mathsf{ID}_\tau \notin \{\mathsf{Honest}, \mathsf{Dishonest}\}$ and pk_{ID_τ} is ensured to be sound by evaluating the non-interactive proof $\mathsf{pf}_{\mathsf{ID}_\tau}$. We only require that the proof is non-interactive in order to keep the model simple. Parties established by this query are called dishonest.

- $\mathsf{RevealKey}^{nikea}(\mathsf{ID}_1, \mathsf{ID}_2, aux)$: On input a tuple of identities $(\mathsf{ID}_1, \mathsf{ID}_2)$, $\mathcal{C}$ returns a failure symbol $\perp$ if both parties ID_1 and ID_2 are dishonest. Otherwise $\mathcal{C}$ runs $\mathsf{NIKEA.ShareKey}$ using the secret key of one of the honest parties in $(\mathsf{ID}_1, \mathsf{ID}_2)$ and the public key of the other party and the aux given by adversary, and returns the result to $\mathcal{A}$.

- $\mathsf{Test}^{nikea}(\mathsf{ID}_1, \mathsf{ID}_2, aux)$: Given two identities $(\mathsf{ID}_1, \mathsf{ID}_2)$ and string aux, the challenger $\mathcal{C}$ returns a failure symbol $\perp$ if one of following condition holds: (i) $\mathsf{ID}_1 = \mathsf{ID}_2$, (ii) $\mathsf{ID}_1 \notin \mathsf{Honest}$ or (iii) $\mathsf{ID}_2 \notin \mathsf{Honest}$. Otherwise the challenger $\mathcal{C}$ samples a random bit $b \xleftarrow{\$} \{0, 1\}$, and it answers this query in terms of the bit b. Specifically, if $b = 1$, $\mathcal{C}$ runs $\mathsf{NIKEA.ShareKey}$ using the secret key of ID_1 and the public key of ID_2 to obtain the shared key K_1; else if $b = 0$, the challenger generates a random key K_1. $\mathcal{C}$ returns K_b to adversary. This query can be queried only once.

Security Experiment for Confidentiality $\mathsf{EXP}^{\mathsf{NIKEA}, ind-cma}_{\mathsf{NIKEA}, \mathcal{A}}(\kappa)$: On input security parameter κ, the security experiment is proceeded as a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ based on a non-interactive key exchange protocol with auxiliary input NIKEA, where the following steps are performed:

1) The $\mathcal{C}$ first runs $pms^{nikea} \leftarrow \mathsf{NIKEA.Setup}(1^\kappa)$ and gives pms^{nikea} to adversary $\mathcal{A}$.

[2] For example, the aux can be periodically distributed by certain trusted key management center.

2) The adversary $\mathcal{A}$ may interact with challenger $\mathcal{C}$ with $\mathsf{RegisterHonest}$, $\mathsf{EstablishParty}$, $\mathsf{RevealKey}^{nikea}$ queries as defined above.

3) Eventually, the adversary may terminate with outputting a bit b'.

4) At the end, the experiment returns 1 if all following conditions hold: (i) the adversary $\mathcal{A}$ has issued a Test^{nikea} query on input $(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*)$ in either identity order, (ii) Both parties $\mathsf{ID}_1^*, \mathsf{ID}_2^* \in \mathsf{Honest}$, (iii) $\mathcal{A}$ has not issued $\mathsf{RevealKey}^{nikea}$ query with input $(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*)$ in either identity order, and $b = b'$; Otherwise 0 is returned.

Definition 4. *A two party NIKEA protocol Σ is called $(t, \epsilon_{\mathsf{NIKEA\text{-}IND}})$-shared-key-secure if it holds that $|\Pr[\mathsf{EXP}^{\mathsf{NIKEA},ind-cma}_{\Sigma,\mathcal{A}}(\kappa) = 1] - 1/2| \leq \epsilon_{\mathsf{NIKEA\text{-}IND}}$ for all adversaries $\mathcal{A}$ running within time t in the above security experiment and for some negligible probability $\epsilon_{\mathsf{NIKEA\text{-}IND}} = \epsilon_{\mathsf{NIKEA\text{-}IND}}(\kappa)$ in the security parameter κ.*

The above security definition provide a stronger security guarantee than the CKS-light security [10], that allows the adversary to ask $\mathsf{RevealKey}^{nikea}$ queries to under attacked parties (as long as these queries have distinct inputs to Test^{nikea} query's).[3] In other words, the leaked shared key would not affect the shared key with distinct aux.

On the next we show another interesting security property of NIKEA, i.e., the unforgeability of the shared key associated with aux. Informally speaking the adversary is unable to output a shared key which is not generated by uncorrupted honest parties. A NIKE scheme combines with an authentication protocol (e.g., the one based on message authentication code) might fulfill the same security attribute as NIKEA. But this is not efficient and would require more security assumptions (comparing to using NIKEA).

Security Experiment for Unforgeability $\mathsf{EXP}^{\mathsf{NIKEA},euf-cma}_{\mathsf{NIKEA},\mathcal{A}}(\kappa)$: On input security parameter κ, the security experiment is proceeded as a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ based on a non-interactive key exchange protocol with auxiliary input NIKEA, where the following steps are performed:

1) The $\mathcal{C}$ first run $pms^{nikea} \leftarrow \mathsf{NIKEA.Setup}(1^\kappa)$ and gives pms^{nikea} to adversary $\mathcal{A}$.

2) The adversary $\mathcal{A}$ may interact with challenger $\mathcal{C}$ with $\mathsf{RegisterHonest}$, $\mathsf{EstablishParty}$, $\mathsf{RevealKey}^{nikea}$ queries as defined above.

3) Eventually, the adversary may terminate with outputting a tuple $(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*, K^*)$.

4) At the end, the experiment returns 1 if all following conditions hold: (i) both parties ID_1^* and ID_2^* are honest, (ii) $\mathcal{A}$ has not issued $\mathsf{RevealKey}^{nikea}$ query on input $(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*)$ in either identity order, and (iii) $K^* = \mathsf{NIKEA.ShareKey}(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*)$; Otherwise 0 is returned.

Definition 5. *A two party NIKEA protocol Σ is called $(t, \epsilon_{\mathsf{NIKEA\text{-}EUF}})$-unforgeable-secure if it holds that $|\Pr[\mathsf{EXP}^{\mathsf{NIKEA},euf-cma}_{\Sigma,\mathcal{A}}(\kappa) = 1] - 1/2| \leq \epsilon_{\mathsf{NIKEA\text{-}EUF}}$ for all adversaries $\mathcal{A}$ running within time t in the above security experiment and for some negligible probability $\epsilon_{\mathsf{NIKEA\text{-}EUF}} = \epsilon_{\mathsf{NIKEA\text{-}EUF}}(\kappa)$ in the security parameter κ.*

Lemma 1. *Assume the NIKEA protocol Σ is $(t, \epsilon_{\mathsf{NIKEA\text{-}IND}})$-shared-key-secure, then it is also $(t, \epsilon_{\mathsf{NIKEA\text{-}EUF}})$-unforgeable-secure provided that $t \approx t'$ and $\epsilon_{\mathsf{NIKEA\text{-}EUF}} \leq \epsilon_{\mathsf{NIKEA\text{-}IND}}$.*

Proof. Suppose that there exists an adversary $\mathcal{A}_1$ which can win the unforgeability security experiment with output $(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*, K^*)$ with overwhelming probability, then we could construct an adversary $\mathcal{A}_2$ using $\mathcal{A}_1$ to break the confidential property of Σ with the same advantage. Technically, $\mathcal{A}_2$ simulates the EUF-CMA security experiment for $\mathcal{A}_1$, and it forwards all queries from $\mathcal{A}_1$ to the challenger $\mathcal{C}$ in its own experiment and returns the corresponding answers to $\mathcal{A}_1$. Note that the triple $(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*)$ was never queried by $\mathcal{A}_1$ to $\mathsf{RevealKey}^{nikea}$. When $\mathcal{A}_1$ outputs a $(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*, K^*)$, then $\mathcal{A}_2$ asks the $\mathsf{Test}^{nikea}(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*)$ with obtaining a test key K_b. If $K_b = K^*$ then $\mathcal{A}_2$ would know that the K_b is the real key with probability at least $\epsilon_{\mathsf{NIKEA\text{-}EUF}}$. □

3.3 A Concrete NIKEA Scheme

We here introduce a pairing-based NIKEA scheme which is derived from the pairing based NIKE in [10]. In this variant, the chameleon hash function used in [10] is replaced with a target collision resistant hash function TCRHF (to lower the assumption). In particular we make use of a pseudo-random function to not only generate the final shared key but also bind the identities and an auxiliary string to such shared key. This also enables us to deal with the chosen identity and public key attacks modelled by $\mathsf{EstablishParty}$ query. In addition, we require the trusted public key bulletin (such as Certificate Authority) to check the validity of registered public key rather than doing so in each $\mathsf{NIKEA.ShareKey}$ execution for efficiency consideration. This change would lead the NIKEA scheme to be more suitable to power constrained devices. It is noticeable only one exponentiation is required in the $\mathsf{NIKEA.ShareKey}$ algorithm of our modified scheme that is more efficient than the original one [10] which requires three pairings operations.

The concrete algorithms of our new NIKEA scheme between two parties ID_1 and ID_2 are defined as follows:

- $\mathsf{NIKEA.Setup}(1^\kappa)$. On input security parameter 1^κ, this algorithm is proceeded as the follows: (i) Run

[3] Note that if the query $\mathsf{EstablishParty}(\mathsf{ID}_\tau, pk_{\mathsf{ID}_\tau}, \mathsf{pf}_{\mathsf{ID}_\tau})$ is asked with $\mathsf{pf} = \emptyset$ and the query $\mathsf{RevealKey}^{nikea}(\mathsf{ID}_1, \mathsf{ID}_2, aux)$ is asked with $aux = \emptyset$, then the above model equals to the CKS-light model [10]. The number of $\mathsf{EstablishParty}$ queries is bound by the time t.

$\mathcal{PG} = (\mathbb{G}, g, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathsf{PG.Gen}(1^\kappa)$, and generate random values $u, u_0, u_1, u_2 \xleftarrow{\$} \mathbb{G}$; (ii) Run $hk_{\mathsf{TCRHF}} \xleftarrow{\$} \mathsf{TCRHF.KG}(1^\kappa)$; (iii) Return system parameters $pms^{nikea} := (hk_{\mathsf{TCRHF}}, u, u_0, u_1, u_2)$.

- $\mathsf{NIKEA.KG}(\mathsf{ID})$. On input a party's identity $\mathsf{ID} \in \mathcal{IDS}$, the key generation algorithm does the following steps: (i) Choose one random element $sk_{\mathsf{ID}} \xleftarrow{\$} \mathbb{Z}_p^*$ as its secret keys, and (ii) Compute corresponding public key $pk_{\mathsf{ID}} := e(u, g^{sk_{\mathsf{ID}}})$, and generate proof $\mathsf{pf} := (g^{sk_{\mathsf{ID}}}, (u_0 u_1^{h_{\mathsf{ID}}} u_2^{h_{\mathsf{ID}}^2})^{sk_{\mathsf{ID}}})$ where $h_{\mathsf{ID}} = \mathsf{TCRHF}(g^{sk_{\mathsf{ID}}})$. Then the public key is registered if $e(u_0 u_1^{h_{\mathsf{ID}}} u_2^{h_{\mathsf{ID}}^2}, g^{sk_{\mathsf{ID}}}) = e(u_0 u_1^{h_{\mathsf{ID}}} u_2^{h_{\mathsf{ID}}^2})^{sk_{\mathsf{ID}}}, g)$ and $e(g^{sk_{\mathsf{ID}}}, u) = pk_{\mathsf{ID}}$.
- $K \xleftarrow{\$} \mathsf{NIKEA.ShareKey}(\mathsf{ID}_1, sk_{\mathsf{ID}_1}, \mathsf{ID}_2, pk_{\mathsf{ID}_2}, aux)$. Given the private key sk_{ID_1} of party ID_1, party ID_2's public key pk_{ID_2} and auxiliary input string aux, the ID_1 generates the shared key $K := \mathsf{PRF}(pk_{\mathsf{ID}_2}^{sk_{\mathsf{ID}_1}}, \mathsf{ID}_1 || \mathsf{ID}_2 || aux)$.

Theorem 1. *Suppose the Bilinear Decisional Diffie-Hellman problem is $(t, \epsilon_{\mathsf{BDDH}})$-hard in $\mathcal{PG}$, the hash function TCRHF is $(t, \epsilon_{\mathsf{TCRHF}})$-target-collision-resistant and the pseudo-random function family is $(t, \epsilon_{\mathsf{PRF}})$-shared-key-secure as defined above. Then the proposed NIKEA scheme is $(t', \epsilon_{\mathsf{NIKEA\text{-}IND}})$-secure provided that $t \approx t'$ and $\epsilon_{\mathsf{NIKEA\text{-}IND}} \leq \epsilon_{\mathsf{TCRHF}} + \epsilon_{\mathsf{BDDH}} + \epsilon_{\mathsf{PRF}}$.*

The proof of this theorem is presented in Appendix A.

4 Security Model for Authenticated Key Exchange

In this section we present a security model for authenticated key exchange (AKE) that is extended from the model by Bellare and Rogaway [1] with additionally formulating the active attacks on chosen identity and public key attacks, known session key, leakage of ephemeral secret and perfect forward secrecy. In this model, the active adversary is provided with an 'execution environment' which emulates the real world execution of AKE protocols.

Execution Environment. In the execution environment, we fix a set of honest parties $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_\ell\}$ for $\ell \in \mathbb{N}$, where ID_i ($i \in [\ell]$) is the identity of a party which is chosen uniquely from space $\mathcal{IDS}$. Each identity is associated with a long-term key pair $(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_i}) \in (\mathcal{SK}, \mathcal{PK})$ for authentication. Each honest party ID_i can sequentially and concurrently execute the protocol multiple times with different intended partners, this is characterized by a collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$ for $d \in \mathbb{N}$.[4] Oracle π_i^s behaves as party ID_i carrying out a process to execute the s-th protocol instance (session), which has access to the long-term key pair $(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_i})$ and to all other public keys. Moreover, we assume each oracle π_i^s maintains a list of independent internal state variables: (i)pid_i^s – storing the identities and public keys of session participants which are sorted lexicographically in terms of identity, including ID_i; (ii) Φ_i^s – denoting the decision $\Phi_i^s \in \{\mathtt{accept}, \mathtt{reject}\}$; (iii)$\rho_i^s$ – denoting the role $\rho_i^s \in \{Initiator(I), Responder(R)\}$; (iv) sT_i^s – recording the transcript of messages sent by oracle π_i^s; (v) rT_j^t – recording the transcript of messages received by oracle π_i^s.

All those variables of each oracle are initialized with empty string which is denoted by the symbol $\emptyset$. At some point, each oracle π_i^s may complete the execution always with a decision state $\Phi_i^s \in \{\mathtt{accept}, \mathtt{reject}\}$.

Adversarial Model. An adversary $\mathcal{A}$ in our model is a PPT Turing Machine taking as input the security parameter 1^κ and the public information (e.g., generic description of above environment), which may interact with these oracles by issuing the following queries.

- $\mathsf{Execute}(ID_1, s_1, \mathsf{ID}_2, s_2)$: This query allows adversary to execute the protocol among unused oracles $\{\pi_i^{s_i}\}_{1 \leq i \leq 2}$, and responds with the transcript of the execution. The $\mathsf{pid}_i^{s_i}$ of each instance is set to $\{\mathsf{ID}_1, \mathsf{ID}_2\}$. We will write $\mathsf{Execute}(\mathsf{ID}_1, \mathsf{ID}_2)$ for short, where the identities are sorted lexicographically.
- $\mathsf{Send}(\mathsf{ID}_i, s, m)$: The adversary can use this query to send any message m of his own choice to oracle π_i^s. The oracle π_i^s will respond the next message m^* (if any) to be sent according to the protocol specification and its internal states. Oracle π_i^s would be initiated via sending the oracle the first message $m = (\top, \mathsf{pid}_i^s)$ consisting of a special initialization symbol $\top$ and a variable storing partner identities.
- $\mathsf{RevealKey}(\mathsf{ID}_i, s)$: Oracle π_i^s responds with the session key if $\Phi_i^s = \mathtt{accept}$.
- $\mathsf{RevealState}(\mathsf{ID}_i, s)$: Oracle π_i^s responds with randomness used to generate the session key of this oracle.
- $\mathsf{Corrupt}(\mathsf{ID}_i)$: Oracle π_i^1 responds with the long-term secret key sk_{ID_i} of party ID_i if $i \in [\ell]$; otherwise a failure symbol $\perp$ is returned.
- $\mathsf{EstablishParty}(\mathsf{ID}_\tau, pk_{\mathsf{ID}_\tau}, \mathsf{pf}_{\mathsf{ID}_\tau})$: This query allows the adversary to register an identity ID_τ ($\ell < \tau$ and $\tau \in \mathbb{N}$) and a static public key pk_{ID_τ} on behalf of a party ID_τ. Parties established by this query are called dishonest. This query is proceeded similarly to the one described in Section 3.2.
- $\mathsf{Test}(\mathsf{ID}_i, s)$: If the oracle has state $\Phi \neq \mathtt{accept}$, then the oracle π_i^s returns some failure symbol $\perp$. Otherwise it flips a fair coin b, samples a random element K_0 from key space $\mathcal{K}_{\mathsf{AKE}}$, and sets K_1 to the real session key of oracle π_i^s. Finally the key K_b is returned.

[4] An oracle in this paper might be alternatively written as $\pi_{\mathsf{ID}_i}^s$ which is conceptually equivalent to π_i^s.

Secure AKE Protocols. In order to denote the situation that two oracles are engaged in an on-line communication, we first define two notions regarding partnership, i.e. *matching sessions* (MS) and origin session (OS) [8], where the MS is used to formulate the security related to RevealKey query, and the OS is used to formulate the security related to RevealState and Corrupt queries.

Definition 6 (Matching sessions)**.** *We say that π_i^s has a* matching session *to π_j^t, if $\mathsf{pid}_i^s = \mathsf{pid}_j^t$, $\rho_i^s \neq \rho_j^t$, $rT_j^t = sT_i^s$ and $sT_j^t = rT_i^s$. The π_j^t is said to be the partner-oracle of π_i^s.*

Definition 7 (Origin session)**.** *We say that π_i^s has a* origin session *to π_j^t, if $rT_j^t = sT_i^s$. The π_i^s is said to be the origin-oracle of π_i^s.*

Correctness. We say an authenticated key exchange (AKE) protocol Π is correct, if two oracles π_i^s and π_j^t accept with matching sessions, then both oracles hold the same session key.

For the security definition, we need the notion of *freshness* of an oracle. in the sequel, we give two freshness definitions. Let π_i^s be an accepted oracle, π_j^t be an oracle (if it exists) having matching session to π_i^s, and π_l^v be an oracle (if it exists) having origin session to π_i^s.

Definition 8 (Passive Freshness)**.** *The oracle π_i^s is said to be* KE*-fresh if the following condition is held:*

- *$\mathcal{A}$ queried either* $\mathsf{RevealKey}(\pi_i^s)$ *or* $\mathsf{RevealKey}(\pi_j^t)$ *(if π_j^t exists).*

Definition 9 (Active Freshness)**.** *The oracle π_i^s is said to be* AKE*-fresh if none of the following conditions holds:*

1) *$\mathcal{A}$ queried* $\mathsf{EstablishParty}(\mathsf{ID}_j, pk_{\mathsf{ID}_j})$ *to some party $\mathsf{ID}_j \in \mathsf{pid}_i^s$.*
2) *$\mathcal{A}$ queried either* $\mathsf{RevealKey}(\pi_i^s)$ *or* $\mathsf{RevealKey}(\pi_j^t)$ *(if π_j^t exists).*
3) *$\mathcal{A}$ queried either* $\mathsf{Corrupt}(\mathsf{ID}_i)$ *or* $\mathsf{Corrupt}(\mathsf{ID}_j)$ *to some party $\mathsf{ID}_j \in \mathsf{pid}_i^s$.*
4) *$\mathcal{A}$ queried either* $\mathsf{RevealState}(\pi_i^s)$ *or* $\mathsf{RevealState}(\pi_l^v)$ *(if π_l^t exists).*

Let $M \in \{\mathsf{KE}, \mathsf{AKE}\}$ be a variable to denote two distinct security experiments.

Security Experiment $\mathsf{EXP}_{\Pi,\mathcal{A}}^M(\kappa)$: On input security parameter 1^κ, the security experiment is proceeded as a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ based on (A)KE protocol Π, where the following steps are performed:

1) At the beginning of the game, the challenger $\mathcal{C}$ implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$, and generates ℓ long-term key pairs and corresponding proof $(pk_{\mathsf{ID}_i}, sk_{\mathsf{ID}_i}, \mathsf{pf}_{\mathsf{ID}_i})$ for all honest parties ID_i for $i \in [\ell]$ where the identity $\mathsf{ID}_i \in \mathcal{IDS}$ of each party is chosen uniquely. $\mathcal{C}$ gives adversary $\mathcal{A}$ all identities, public keys and proofs $\{(\mathsf{ID}_1, pk_{\mathsf{ID}_1}, \mathsf{pf}_{\mathsf{ID}_1}), \ldots, (\mathsf{ID}_\ell, pk_{\mathsf{ID}_\ell}, \mathsf{pf}_{\mathsf{ID}_\ell})\}$ as input.
2) If $M = \mathsf{KE}$, then $\mathcal{A}$ is allowed to ask a polynomial number of queries: Execute, Corrupt and RevealKey.
3) If $M = \mathsf{AKE}$, then $\mathcal{A}$ is allowed to ask a polynomial number of queries: Send, Execute, RevealState, Corrupt, EstablishParty and RevealKey.
4) At some point, $\mathcal{A}$ may issue a $\mathsf{Test}(\pi_i^s)$ query on an oracle π_i^s during the game with only once.
5) At the end of the game, the $\mathcal{A}$ may terminate with returning a bit b' as its guess for b of Test query.
6) Finally, 1 is returned if all following conditions hold:
   - $\mathcal{A}$ has issued a Test query to a M-fresh oracle π_i^s without failure,
   - $\mathcal{A}$ returned a bit b' which equals to b of Test-query;

   Otherwise 0 is returned.

Definition 10 (Session Key Security)**.** *We say that a correct key exchange protocol Π is (M, t, ϵ)-secure, if for any $\mathcal{A}$ runs the M security experiment within time t while having advantage $\epsilon = \epsilon(\kappa)$ in terms security parameter κ, it holds that*

- *If two oracles π_i^s and π_j^t accept with matching sessions, then except for ϵ the following conditions must be satisfied: (i) the oracle π_i^s has a unique matching session at party ID_j, and (ii) the oracle π_j^t has a unique matching session at party ID_i.*
- *If a* Test *query has been issued to a M-fresh oracle π_i^s, then the probability holds that $|\Pr[\mathsf{EXP}_{\Pi,\mathcal{A}}^M(\kappa) = 1] - 1/2| < \epsilon$.*

It is not hard to see that the KE security provides forward secrecy property in presence of passive adversary. Since we allow the adversary to Corrupt the long-term keys (if any) of principles.

5 Compiler for two-move AKE Protocol from NIKEA

In this section we propose a generic compiler that transforms a passively forward secure two-move TMKE protocol to a AKE protocol based on NIKEA. The resulting AKE protocol can provide the AKE security as modelled in Section 4 that covers a lot of well-known security attributes such as resilience of leakage ephemeral keys (from sessions not 'associated' with test session) and chosen identity and public key attacks (such as the unknown key share attacks or small sub-group attacks), and provision of perfect forward secrecy.

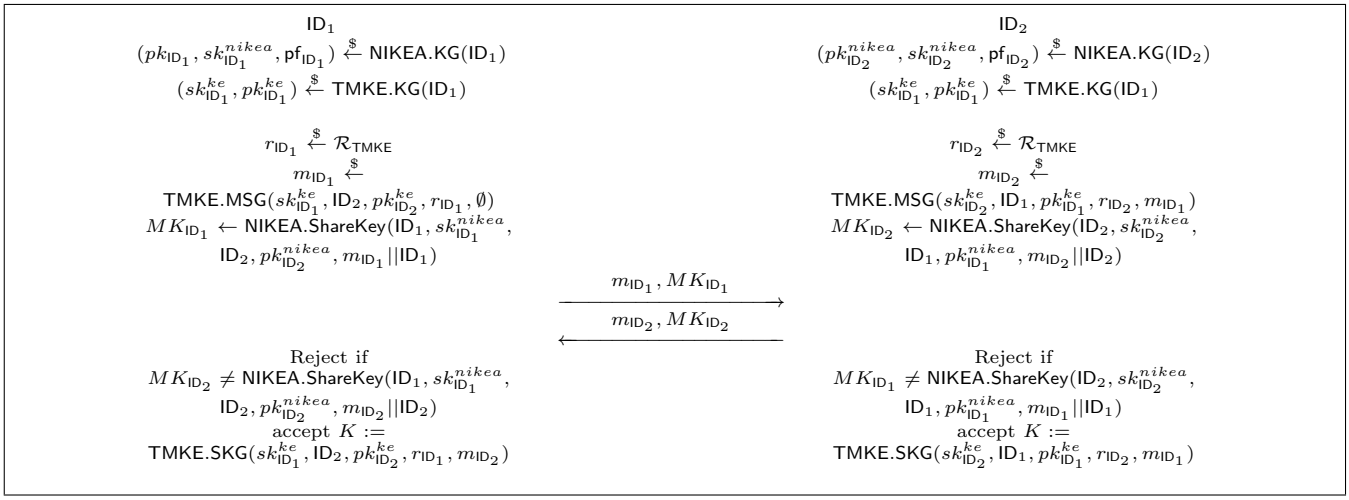

Figure 2: AKE protocol from NIKEA

Protocol Description. The compiler takes as input the following building blocks: (i)passively forward secure two-message key exchange protocol TMKE = ($\mathsf{TMKE.ST}$,$\mathsf{TMKE.KG}$,$\mathsf{TMKE.MSG}$,$\mathsf{TMKE.SKG}$), and CKS-like secure non-interactive key exchange scheme NIKEA = ($\mathsf{NIKEA.Setup}$, $\mathsf{NIKEA.KG}$, $\mathsf{NIKEA.ShareKey}$).

The parameters of the compiler consist of values generated by $pms^{nikea} \leftarrow \mathsf{NIKEA.Setup}(1^\kappa)$ and $pms^{ke} \leftarrow \mathsf{TMKE.ST}(1^\kappa)$. The generic compiler between two parties is shown as Figure 2.

Remark 1. In the above compiler, the NIKEA is used as a tool to authenticate the outgoing message of KE without changing it. Hence the KE and underlying application based on TMKE would have no 'awareness' on the increased compiler. Instead, security can be established by simply 'adding' the implementation of the compiler to the system. We stress that the computations on authentication tokens MK_{ID_1} and MK_{ID_2} can use the same shared key material (i.e., the $(pk_{\mathsf{ID}_2}^{nikea})^{sk_{\mathsf{ID}_1}^{nikea}}$). If one realizes the TMKE and NIKEA with Diffie-Hellman key exchange and the NIKEA presented in Section 3.2 respectively, then the overall computation cost would be approximately dominated by only three regular exponentiations (that is quite efficient). Moreover, the size of secret key is also very short that only one element in pairing group $\mathbb{G}$ is required. Such performance makes the resulting AKE protocol to be appealing to the resource constrained application environment (such as sensors networks).

The resilience of key compromise impersonation attacks is not covered by our compiler. In order to modify our compiler for achieving KCI resilience, a way is to use signature-based authentication protocol instead of NIKEA. But the computation cost will increase also. We leave out this as future work.

Comparisons. We summarize the comparisons between our proposal and some well known AKE compilers without random oracles in Table 1, i.e., the signature-based JKSS compiler [14] and the signature-based LSYBS compiler [18] which are referred to as JKSS_{SIG} and $\text{LSYBS}_{\text{SIG}}$ respectively. We instantiate the signature scheme in those compilers with the concrete one called $\text{Sig}_{\text{SRSA}}[\text{H}_{\text{cfs}}]$ [13] which is overall efficient on signing and verifying operations. Whereas the passive secure KE protocol in all compilers would be instantiated with the traditional Diffie-Hellman key exchange protocol.

Our comparisons are given from the following perspectives: (i) security assumptions; (ii) the number of exchanged messages sent by a party; (iii) overall computation cost of considered protocol; (iv) the communication round. Let 'DDH' denote the Decisional Diffie-Hellman assumption and 'SRSA' denote the strong RSA assumption. Let MAC denote the message authentication code. Let 'Exp' denote the regular exponentiation. In addition, we ignore the cost of PRF and MAC in the comparison.

Security Analysis. In the following, we are going to show that the new compiler is secure without appealing to random oracles.

Theorem 2. *Suppose the* TMKE *protocol is* $(\mathsf{TMKE}, t, \epsilon_{\mathsf{KE}})$*-secure and the NIKEA is* $(t, \epsilon_{\mathsf{NIKEA\text{-}EUF}})$*-unforgeable-secure. Then the proposed AKE compiler is* $(\mathsf{AKE}, t', \epsilon_{\mathsf{AKE}})$*-secure, provided that* $t \approx t'$ *and* $\epsilon_{\mathsf{AKE}} \leq d\ell \cdot \epsilon_{\mathsf{KE}} + d\ell^2 \cdot (\epsilon_{\mathsf{NIKEA\text{-}EUF}} + (d+1) \cdot \epsilon_{\mathsf{KE}})$.

We present the proof of this theorem in Appendix B.

6 Conclusions

We have presented a new security notion on non-interactive key exchange with auxiliary input (NIKEA). One of the advantages of NIKEA is that two parties may have a number of shared keys which are generated by different auxiliary input *aux*. We have also shown another interesting security property of NIKEA regarding unforgeability that adversary is unable to generate the

Table 1: Comparison

	Security Assumption	Message Length	Computation Cost	Communication Round
$JKSS_{SIG}$ [14]	SRSA, DDH, PRF, MAC	$9\mathbb{G}$	4 Exp	4
$LSYBS_{SIG}$ [18]	SRSA, DDH	$7\mathbb{G}$	4 Exp	2
Ours	BDDH, DDH, PRF	$2\mathbb{G}$	3 Exp	1

shared key of uncorrupted honest parties with an auxiliary input *aux* that is not used by these parties before. Based on such property, we have proposed a new lightweight AP&KE style compiler that generically build secure AKE from secure NIKEA protocols and passively secure two-move key exchange protocols without long-term keys. The new compiler is superior to previous similar works on perspectives of both communication and computation costs. Hence it is suitable for resources constrained application environment. As for a future work, it might be interesting to extend the idea of our compiler to group case for efficiency consideration, i.e. to build AGKE protocol from passively secure GKE and group NIKEA.

Acknowledgments

This study was supported by Scientific and Technological Research Program of Chongqing Municipal Education Commission (Grant No. KJ1500918, KJ1500904, KJ1401307), National Natural Science Foundation of China (Grant No. 11547148, 61503052), Research Project of Humanities and Social Sciences of Ministry of Education of China (Grant No. 15YJC790061), and Natural Science Foundation of Chongqing City (cstc2014jcyjA40024, cstc2013jcyjA40019).

References

[1] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Advances in Cryptography (Crypto'93)*, LNCS 773, pp. 232–249, Springer, 1993.

[2] M. Bellare and P. Rogaway, "The security of triple encryption and a framework for code-based game-playing proofs," in *Advances in Cryptography (Eurocrypt'06)*, LNCS 4004, pp. 409–426, Springer, 2006.

[3] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[4] C. Boyd and J. G. Nieto, "On forward secrecy in one-round key exchange," in *13th IMA International Conference Cryptography and Coding (IMACC'11*, LNCS 7089, pp. 451–468, Springer, 2011.

[5] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Advances in Cryptography (Eurocrypt'01)*, LNCS 2045, pp. 453–474, Springer, 2001.

[6] Ç. Çapar, D. Goeckel, K. G. Paterson, E. A. Quaglia, D. Towsley and M. Zafer, "Signal-flow-based analysis of wireless security protocols," *Information and Computation*, vol. 226, pp. 37–56, 2013.

[7] T. Y. Chang, M. S. Hwang and W. P. Yang, "A communication-efficient three-party password authenticated key exchange protocol," *Information Sciences*, vol. 181, no. 1, pp. 217–226, 2011.

[8] C. Cremers and M. Feltz, "Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal," *Designs, Codes and Cryptography*, vol. 74, no. 1, pp. 183–218, 2015.

[9] Z. Eslami, M. Noroozi and S. K. Rad, "Provably secure group key exchange protocol in the presence of dishonest insiders," *International Journal of Network Security*, vol. 18, no. 1, pp. 33–42, 2016.

[10] E. S. V. Freire, D. Hofheinz, E. Kiltz and K. G. Paterson, "Non-interactive key exchange," in *Public Key Cryptography*, pp. 254–271, 2013.

[11] R. Gennaro, S. Halevi, H. Krawczyk, T. Rabin, S. Reidt and S. D. Wolthusen, "Strongly-resilient and non-interactive hierarchical key-agreement in manets," in *Computer Security (Esorics'08)*, pp. 49–65, Springer, 2008.

[12] D. He, C. Chen, M. Ma, S. Chan and J. Bu, "A secure and efficient password-authenticated group key exchange protocol for mobile ad hoc networks," *International Journal of Communication Systems*, vol. 26, no. 4, pp. 495–504, 2013.

[13] D. Hofheinz, T. Jager and E. Kiltz, "Short signatures from weaker assumptions," in *Advances in Cryptology (Asiacrypt'11)*, pp. 647–666, Springer, 2011.

[14] T. Jager, F. Kohlar, S. Schaege and J. Schwenk, "Generic compilers for authenticated key exchange," in *Advances in Cryptography (Asiacrypt'10)*, LNCS 6477, pp. 232–249, Springer, 2010.

[15] A. Joux, "A one round protocol for tripartite diffie–Hellman," in *Algorithmic Number Theory*, pp. 385–393, Springer, 2000.

[16] J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," *Journal of Cryptology*, vol. 20, no. 1, pp. 85–113, 2007.

[17] B. A. LaMacchia, K. Lauter and A. Mityagin, "Stronger security of authenticated key exchange," in *Provable Security*, pp. 1–16, Springer, 2007.

[18] Y. Li, S. Schaege, Z. Yang, C. Bader and J. Schwenk, "New modular compilers for authenticated key exchange," in *International Conference on Applied*

Cryptography and Network Security, vol. 8479, pp. 1–18, Springer, 2014.

[19] Y. Li, D. Chen, W. Li, G. Wang and S. Paul, "A hybrid authenticated group key agreement protocol in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.

[20] V. Shoup, "Sequences of games: A tool for taming complexity in security proofs," Cryptology ePrint Archive, Report 2004/332, 2004. (`http://eprint.iacr.org/`)

[21] T. Y. Wu, Y. M. Tseng and T. T. Tsai, "A revocable id-based authenticated group key exchange protocol with resistant to malicious participants," *Computer Networks*, vol. 56, no. 12, pp. 2994–3006, 2012.

[22] C. Xu, H. Guo, Z. Li and Y. Mu, "New construction of affiliation-hiding authenticated group key agreement," *Security and Communication Networks*, vol. 6, no. 6, pp. 723–734, 2013.

[23] Z. Yang and W. Yang, "A practical strongly secure one-round authenticated key exchange protocol without random oracles," *Security and Communication Networks*, vol. 8, no. 6, pp. 1118–1131, 2015.

[24] Z. Yang, W. Yang, L. Zhu and D. Zhang, "Towards modelling perfect forward secrecy in two-message authenticated key exchange under ephemeral-key revelation," *Security and Communication Networks*, vol. 8, no. 18, pp. 3356–3371, 2015.

[25] Z. Yang and D. Zhang, "Towards modelling perfect forward secrecy for one-round group key exchange," *International Journal of Network Security*, vol. 18, no. 2, pp. 304–315, 2016.

Appendix A: Proof of Theorem 1

The proof will be given using a gamed based approach as in [2, 20]. Let break_δ be the event that the $\mathcal{A}$ correctly guesses the bit b sampled by the Test^{nikea} query in Game δ. Let $\mathsf{Adv}_\delta := \Pr[\mathsf{break}_\delta] - 1/2$ denote the advantage of $\mathcal{A}$ in Game δ.

Game 0 This is the original game with adversary $\mathcal{A}$. Thus we have that $\Pr[\mathsf{break}_0] - 1/2 = \epsilon_{\mathsf{NIKEA\text{-}IND}} = \mathsf{Adv}_0$.

Game 1 In this game we want to make sure that the received ephemeral keys are correctly formed. Technically the challenger proceeds exactly as before, but it aborts if there exist two distinct long-term public keys W and N such that $\mathsf{TCRHF}(W) = \mathsf{TCRHF}(V)$. According to the security property of underlying hash function, the above abortion event might occur with probability $\epsilon_{\mathsf{TCRHF}}$. Thus we have $\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \epsilon_{\mathsf{TCRHF}}$.

Game 2 In this game proceeds as the previous one, but we replace the key material $\beta^* = pk_{\mathsf{ID}_1^*}^{sk_{\mathsf{ID}_2^*}}$ of PRF with random value $\widetilde{\beta^*}$ where ID_1^* and ID_2^* are test parties. If there exists an adversary $\mathcal{A}$ which can distinguish the Game 2 from Game 1 then we can use it to construct an efficient distinguisher $\mathcal{D}$ to solve the BDDH problem as follows. Given a BDDH challenge instance $(\bar{g}, v, w, z, \Gamma)$, $\mathcal{D}$ sets $g := \bar{g}$, $u := z$, $pk_{\mathsf{ID}_1^*} := e(v, u)$ and $pk_{\mathsf{ID}_2^*} = e(w, u)$, and computes $h_{\mathsf{ID}_1} = \mathsf{TCRHF}(pk_{\mathsf{ID}_1^*})$ and $h_{\mathsf{ID}_2^*} = \mathsf{TCRHF}(pk_{\mathsf{ID}_2^*})$. Let $p(h) = p_0 + p_1 h + p_2 h^2 = (h - h_{\mathsf{ID}_1^*})(h - h_{\mathsf{ID}_2^*})$ be a polynomial of degree 2 over $\mathbb{Z}_p$ such that $p(h_{\mathsf{ID}_1^*}) = p(h_{\mathsf{ID}_2^*}) = 0$. Let $q(h) = q_0 + q_1 h + q_2 h^2$ be random polynomials of degree 2 over $\mathbb{Z}_p$. $\mathcal{D}$ next sets $u_0 = u^{p_0} g^{q_0}$, $u_1 = u^{p_1} g^{q_1}$ and $u_2 = u^{p_2} g^{q_2}$. $\mathcal{D}$ then answers the following queries:

- $\mathsf{RegisterHonest}(\hat{\mathsf{ID}})$: This query is simulated as the original one, except for the public keys for test parties ID_1^* and ID_2^* which are generated as above. In particular we have that $\mathsf{pf}_{\mathsf{ID}_1^*} = (v, v^{q(h_{\mathsf{ID}_1^*})})$ and $\mathsf{pf}_{\mathsf{ID}_2^*} = (w, w^{q(h_{\mathsf{ID}_2^*})})$, where $q(h_{\mathsf{ID}_1^*})$ and $q(h_{\mathsf{ID}_2^*})$ are known values. These are correct proofs for public keys of parties ID_1^* and ID_2^*, since $p(h_{\mathsf{ID}_1^*}) = p(h_{\mathsf{ID}_2^*}) = 0$.

- $\mathsf{EstablishParty}(\mathsf{ID}_\tau, pk_{\mathsf{ID}_\tau})$. Upon receiving a public key pk_{ID_τ} and an identity ID_τ from $\mathcal{A}$, the public key is registered if ID_τ has not been registered before and correspond proof $\mathsf{pf}_{\mathsf{ID}_\tau} = (\mathsf{pf}_{\mathsf{ID}_\tau,1}, \mathsf{pf}_{\mathsf{ID}_\tau,2})$ are evaluated correctly as protocol specification.

- $\mathsf{RevealKey}^{nikea}(\mathsf{ID}_1, \mathsf{ID}_2, aux)$. We assume this query is legitimate, otherwise $\mathcal{D}$ aborts. As for the case that there exists an honest user, say $\mathsf{ID}_2 \in \{\mathsf{ID}_1^*, \mathsf{ID}_2^*\}$ then $\mathcal{D}$ computes session key as
  $K = \mathsf{PRF}(e((\frac{\mathsf{pf}_{\mathsf{ID}_1,2}}{\mathsf{pf}_{\mathsf{ID}_1,1}^{q(h_{\mathsf{ID}_1})}})^{\frac{1}{p(h_{\mathsf{ID}_1})}}, pk_{\mathsf{ID}_2}), \mathsf{ID}_1 || \mathsf{ID}_2 || aux)$,
  where $h_{\mathsf{ID}_1} = \mathsf{TCRHF}(pk_{\mathsf{ID}_1})$.

- $\mathsf{Test}^{nikea}(\mathsf{ID}_1^*, \mathsf{ID}_2^*, aux^*)$: $\mathcal{D}$ returns $K^* = \mathsf{PRF}(\Gamma, \mathsf{ID}_1^* || \mathsf{ID}_2^* || aux^*)$.

This completes our simulation correctly. If $\Gamma = \mathsf{BDDH}(v, w, z)$, then the simulation is equivalent to Game 1; otherwise the simulation is equivalent to Game 2. At the end, $\mathcal{D}$ returns what $\mathcal{A}$ returns to BDDH challenger. If $\mathcal{A}$ can distinguish the real key from the random value, that implies $\mathcal{D}$ solves the BDDH problem. We therefore obtain that $\mathsf{Adv}_1 \leq \mathsf{Adv}_2 + \epsilon_{\mathsf{BDDH}}$.

Game 3 In this game, the function $\mathsf{PRF}(\widetilde{\beta^*}, \cdot)$ computed in Test^{nikea} query is changed to a truly random function $\mathsf{RF}(\cdot)$. As the secret seed $\widetilde{\beta^*}$ is set to a truly random value due to previous game. If there exists an efficient adversary $\mathcal{A}$ who can distinguish the Game 3 from Game 2 with non-negligible advantage. Then we can construct an efficient algorithm $\mathcal{B}$ using $\mathcal{A}$ to break the security of PRF. In terms of the security of PRF, we have that $\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{PRF}}$. Note that in this game the session key returned by Test^{nikea} query is totally a truly random value which is independent to the bit b. Thus the probability that the adversary wins the game is $\mathsf{Adv}_3 = 0$.

Collect all the probabilities in above games, this theorem is proved.

Appendix B: Proof of Theorem 2

Let break_δ be the event that the $\mathcal{A}$ correctly guesses the bit b sampled by the Test-query in Game δ. Let $\mathsf{Adv}_\delta := \Pr[\mathsf{break}_\delta] - 1/2$ denote the advantage of $\mathcal{A}$ in Game δ. Let oracle $\pi_i^{s^*}$ denote the fresh test oracle and let $\pi_j^{t^*}$ denote the oracle having matching conversation to $\pi_i^{s^*}$.

Game 0 This is the original security game. We have that $\Pr[\mathsf{break}_0] - 1/2 = \epsilon_{\mathsf{AKE}} = \mathsf{Adv}_0$.

Game 1 This game proceeds exactly as the previous game but the challenger aborts if it fails to guess the test oracle $\pi_i^{s^*}$ and its intended communication partner ID_j. Since there are ℓ honest parties and d oracles for each party, the probability that the adversary guesses correctly is at least $1/(d\ell^2)$. Thus we have that $\mathsf{Adv}_0 \leq d\ell^2 \cdot \mathsf{Adv}_4$.

Game 2 In this game, the challenger proceeds exactly as previous game but it raises an abort event $\mathsf{abort}_{\mathsf{token}}$ that: the challenger aborts if test oracle accepts the incoming authentication token MK_{ID_j} which is not sent from its origin oracle. Due to the unforgeability of NIKEA, $\Pr[\mathsf{abort}_{\mathsf{token}}] \leq \epsilon_{\mathsf{NIKEA\text{-}EUF}}$. Thus we have that $\mathsf{Adv}_4 \leq \mathsf{Adv}_2 + \epsilon_{\mathsf{NIKEA\text{-}EUF}}$. In this game, we have that the test oracle must have origin oracle.

Game 3 In this game we are going to show that the test oracle has an origin oracle at each intended communication partner. The challenger proceeds as previous game but it aborts if either: (i) every (sub)-message (which could be, for instances, Diffie-Hellman key and identity) in the message transcript $m_i^{s^*}$ of test oracle $\pi_i^{s^*}$ has been sampled by some other oracle; or (ii) every (sub)-message in the message transcript $m_{\mathsf{ID}_j}^{t^*}$ of origin oracle $\pi_i^{s^*}$ (of test oracle) has been sampled by other oracle.

If the above abort event occurs in a non-negligible probability then there exists an adversary $\mathcal{B}$ which can break the KE security of P by running $\mathcal{A}$. The simulation of $\mathcal{B}$ is proceeded as follows:

- At the initiation phase, $\mathcal{B}$ first implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$. All long-term public/private key pairs for each honest user ID_i are generated and all public keys are given to adversary as input.
- Meantime, $\mathcal{B}$ generates the protocol messages for each oracle as protocol specification and answers all oracle queries honest except for the test oracle and its origin oracle.
- As for the test oracle and its partner oracles, $\mathcal{B}$ queries $\mathcal{C}_{\mathsf{KE}}$ for asking a $\mathsf{Execute}$ query to obtain T^*_{KE} and a Test query to obtain the session key $K^*_{b,\mathsf{KE}}$ of that $\mathsf{Execute}$ query. $\mathcal{B}$ simulates the test oracle chosen by $\mathcal{A}$ and its origin oracle using the transcript T^*_{KE} and $K^*_{b,\mathsf{KE}}$. $\mathcal{A}$ may keep asking oracle queries.
- $\mathcal{B}$ answers those oracle queries using secrets of her own choice. In particular the $\mathcal{B}$ would generate the signatures of oracles based on corresponding long-term key chosen by herself.

Assume that the adversary $\mathcal{A}$ leads two oracles $\pi_h^{v^*}$ (which is either test oracle or its origin oracle) and π_j^t to output the same message m^* without matching sessions. This means that the ephemeral secret key $esk_h^{v^*}$ (which equals to $esk_{\mathsf{ID}_j}^t$) used to generate the message m^* is known by $\mathcal{B}$. Since the oracle π_j^t is simulated by $\mathcal{B}$ honestly as protocol specification, i.e., esk_j^t is chosen by $\mathcal{B}$. Hence the $\mathcal{B}$ could break the KE security of P by computing the session key of test oracle using ephemeral secret $esk_h^{v^*}$. We therefore have that $\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{KE}}$.

This game also implies that each oracle π_i^s has a unique origin oracle. Hence the adversary $\mathcal{A}$ can not exploit $\mathsf{RevealKey}$ query to win the game.

Game 4 This game proceeds exactly as the previous game but the challenger aborts if it fails to guess the origin oracle $\pi_j^{t^*}$. Thus we have that $\mathsf{Adv}_3 \leq d \cdot \mathsf{Adv}_4$.

Game 5 Finally, we replace the key k^* of the test oracle $\pi_i^{s^*}$ and its partner oracle $\pi_j^{t^*}$ (if it exists) with the random value $\widetilde{k^*}$. Note that the KE protocol instance can be seen as being executed between the test oracle and its origin oracle due to Game 2. If there exists an adversary $\mathcal{A}$ which can distinguish this game from the previous game, then we use it to construct an algorithm $\mathcal{B}$ to break the passive security of key exchange protocol as follows. Assume that the adversary $\mathcal{B}$ interacts with the challenger $\mathcal{C}_{\mathsf{KE}}$. More specifically, $\mathcal{B}$ simulates the challenger in this game for $\mathcal{A}$ which is illustrated as follows:

- At the beginning, $\mathcal{B}$ implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$. All long-term public/private key pairs for each honest user ID_i are generated, and all public keys are given to adversary.
- Meantime, $\mathcal{B}$ generates the protocol messages for each oracle as protocol specification and answers all oracle queries honest except for the test oracle and its origin oracle.
- $\mathcal{B}$ queries a $\mathsf{Execute}$ query with obtaining message transcript T^* from $\mathcal{C}_{\mathsf{KE}}$, where one of the oracles involved in this query will be chosen as test oracle. $\mathcal{B}$ simulates the messages of test oracle and its origin oracle based on the transcript T^*. As for the $\mathsf{Test}(\pi_i^{s^*})$ query from $\mathcal{A}$, $\mathcal{B}$ answers it using the result of $\mathsf{Test}(\pi_i^{s^*})$ query returned by $\mathcal{C}_{\mathsf{KE}}$. If the test oracle $\pi_i^{s^*}$ has no matching session but has origin-oracle $\pi_j^{t^*}$ then the session key of $\pi_j^{t^*}$ can be computed using the ephemeral secret key (which is simulated by $\mathcal{B}$) of the origin-oracle of $\pi_j^{t^*}$. Please note that $\pi_j^{t^*}$ must also have origin-oracle due to the security of NIKEA.
- Eventually, $\mathcal{B}$ returns the bit b' from $\mathcal{A}$ to $\mathcal{C}_{\mathsf{KE}}$.

The simulation of $\mathcal{B}$ is perfect since $\mathcal{B}$ can always correctly answer all queries from $\mathcal{A}$. If $\mathcal{A}$ is able to correctly answer the bit b of Test-query with non-negligible probability, so does the adversary $\mathcal{B}$. Hence we obtain that $\mathsf{Adv}_4 \leq \mathsf{Adv}_5 + \epsilon_{\mathsf{KE}}$. In this game, the response to the Test query always consists of a random key, which is independent to the bit b flipped in the Test query. Thus we have $\mathsf{Adv}_5 = 0$. This theorem is proved by putting together of probabilities from above games.

Zheng Yang received the Master degree in Computer Science from Chongqing University in 2009. He got the doctor degree in IT-security from Ruhr-University Bochum, Germany in 2013. He is a researcher at Chongqing University of Technology, China. His main research interests include information security and cryptography.

Chao Liu received his B.S. degree and Ph.D. degree in Computer Science and Technology from Chongqing University in 2006 and 2013, respectively. He has been an instructor at School of Computer Science and Engineering, Chongqing University of Technology, Chongqing, China. His current research interests include impulsive systems, switched systems, neural networks, and network security.

Wanping Liu received the Ph.D. degree from the College of Computer Science, Chongqing University, China, in 2014. He is currently working in the School of Computer Science and Engineering, Chongqing University of Technology. His current research interests include discrete dynamical systems, mathematical modeling and network security.

Song Luo received the Ph.D. degree from the College of Computer Science, Peking University, China, in 2011. He is currently working in the School of Computer Science and Engineering, Chongqing University of Technology. His current research interests include Cryptology, and Network Security.

Hua Long received the Ph.D. degree from the College of Computer Science, Chongqing University, China, in 2010. He is currently working in the School of Computer Science and Engineering, Chongqing University of Technology. His current research interests include natural language processing, software engineering, pattern recognition, and network security.

Shuangqing Li received the Ph.D. degree from the College of Computer Science, Chongqing University, China, in 2010. He is currently working in the College of Computer Science, Chongqing University. His current research interests include software engineering, pattern recognition, cloud computing, and network security.