# Modular Design of Role-Symmetric Authenticated Key Exchange Protocols

Yuting Xiao[1], Rui Zhang(✉)[1,2], and Hui Ma[1]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences

[2] School of Cyber Security, University of Chinese Academy of Sciences

`{xiaoyuting,r-zhang,mahui}@iie.ac.cn`

**Abstract.** Authenticated Key Exchange (AKE) is an important primitive in applied cryptography. Previously several strong models of AKE were introduced, e.g., CK, $\mathrm{CK}^+$, eCK and their extended versions considering perfect forward secrecy (PFS), (denoted by a "-PFS" suffix). These models provide different security guarantees and they are incomparable. Hence, one still lacks systematic understanding of the prerequisites for secure AKEs and a modular design of AKE protocols. In this paper, we investigate this issue in the context of One-Round Authenticated Key Exchange (ORKE), which is role-symmetric for players and only needs a single round to establish a session key.

Our treatments are as follows: First, we reformat the CK, CK-PFS, $\mathrm{CK}^+$, $\mathrm{CK}^+$-PFS, eCK and eCK-PFS models in the context of ORKE, some of which are formulated for the first time in the literature. Next, we introduce a new tool, Key-wise Recoverable Function (KRF). With merely black-box calls to KRFs, we build modular constructions for ORKEs. As an immediate application, many previous protocols can be explained naturally by the construction. We further build a protocol with CK, $\mathrm{CK}^+$, eCK, CK-PFS, $\mathrm{CK}^+$-PFS and eCK-PFS security simultaneously, by properly instantiating the underlying KRF. As a by-product, we have simplified proofs for a few known protocols, with non-standard assumptions avoidable.

**Keywords:** role-symmetric, one-round authenticated key exchange, key-wise recoverable functions, modular construction

## 1 Introduction

Authenticated Key Exchange (AKE) is a fundamental cryptographic primitive to set up secure channels between parties over an open network. In the last decades, many AKE protocols have been developed and used in practice, e.g., SSL/TLS, IPSec and SSH. Typically, in a two-party AKE protocol $\Pi$, each party possesses a long-term public/secret key pair. If any two parties want to negotiate a session key, they should select their own ephemeral keys, then exchange messages in a specific order (e.g., the initiator $\mathcal{I}$ starts by sending $M_1$, the responder $\mathcal{R}$

sends $M_2$, the initiator $\mathcal{I}$ sends $M_3$, and so on), finally compute the session key from four pieces of information, including their own long-term secret and ephemeral keys, the other party's long-term public key and the transcript (i.e., the concatenation of the identities of both sides and all transmitted messages).

We say that $\Pi$ is an $n$-round protocol if the maximum number of messages exchanged from an initiator to a responder during one protocol execution is $n$. We say a protocol is role-symmetric if both sides have equivalent roles, namely, neither side needs to wait for the other party's message to arrive. It is a significant useful property in practice, which greatly reduces latency. In the literature, several famous protocols are role-symmetric, e.g., MQV [20], HMQV [18], NAXOS [19], etc. In particular, these protocols only involve two messages in one round to establish a session key. Such protocols have attracted much attention due to their simplicity and their efficiency in terms of bandwidth usage. In this paper, we focus on this case, also known as One-Round Authenticated Key Exchanges (ORKEs).

Up to now, a number of security models have been introduced for AKEs. The first is the BR model introduced by Bellare and Rogaway [2], capturing an open network fully controlled by adversaries. It is an indistinguishability-based security definition. Any party executing one protocol instance is called a session. Adversaries are allowed to launch various interleaving attacks, corrupt parties' long-term keys and reveal session keys. These behaviours are formally modeled as performing Send$(\cdot)$, Corrupt$(\cdot)$ and SKReveal$(\cdot)$ queries on specific sessions. The security is defined via an experiment between an adversary and the challenger, where the adversary is allowed to adaptively ask above queries, and choose a target session in a Test$(\cdot)$ query that outputs a real or random session key according to a flipped coin. The adversary is said to win if it guesses the correct bit with non-negligible advantage over random guess. To avoid trivial success, the target session must be *fresh* throughout the experiment. Note that the *freshness* notion is important help us understand different security models, which in turn depends on the definitions of *session identifiers* (to identify sessions) and *matching-session* (to denote the session via the same execution instance with the target session), and mainly reflects the restrictions on the adversary's access to secret information of the target session. After that, several stronger security models were developed, namely, CK [7], eCK [19] and CK$^+$ [18]. In these models, the adversaries are allowed to obtain more secret information.

The CK model was introduced by Canetti and Krawczyk [7], which additionally allows an adversary to obtain the secret state of a specific session via the SSReveal$(\cdot)$ query. Since Canetti and Krawczyk did not explicitly specify what information is included in the session state, the claiming CK-security of a particular protocol should come with a careful pre-definition of it. The eCK (extended CK) model was introduced by LaMacchia, Lauter and Mityagin [19]. They replaced the SSReveal$(\cdot)$ query by the EphKReveal$(\cdot)$ query, which gives an adversary the power to corrupt the ephemeral key (i.e., the randomness used) of a specific session. As the name implies, the eCK model provides more security guarantees that are not originally considered by the CK model, e.g., weak perfect forward secrecy (wPFS) [1,18], as well as resistance to key-compromise impersonation attack (KCI) [23,15] and maximal exposure attack (MEX, where a non-trivial combination of the ephemeral and long-term keys of the target session and its matching session are exposed to the adversary). The $\mathrm{CK}^+$ model was originally used to capture the security properties of HMQV [18] and later reformatted by Fujioka et al. [11]. It seems like but not actually a combination of the CK and eCK models. In the $\mathrm{CK}^+$ model, an adversary can ask $\mathsf{SSReveal}(\cdot)$ queries and get a non-trivial combination of the ephemeral and long-term keys of the target session and its matching session.

In the literature, several works have developed the CK, $\mathrm{CK}^+$ and eCK models for capturing perfect forward secrecy (PFS). Boyd and Nieto [6] considered PFS for the CK model, which we prefer call the $\mathrm{CK}_{\mathrm{BN}}$ model. Yoneyama [27] proposed the $\mathrm{CK}^+$-$\mathrm{sFS}^{\mathrm{NSR}}$ model based on the $\mathrm{CK}^+$ model. Cremers and Feltz [9] also proposed the eCK-PFS model to capture the PFS for the eCK model. Note that, when considering PFS, both parties' long-term keys will eventually be exposed to the adversary, thus if it is also allowed to reveal the ephemeral secret of either party, the adversary would trivially win. To avoid trivial success, some less common constraints on the adversaries' behaviours were raised in the $\mathrm{CK}_{\mathrm{BN}}$ and $\mathrm{CK}^+$-$\mathrm{sFS}^{\mathrm{NSR}}$ models, e.g., $\mathsf{SSReveal}(\cdot)$ query is not allowed on any session between the owner and the peer of the target session. While in the eCK-PFS model, the notion of *origin-session* was proposed, which facilitates analysing and limiting the adversaries' behaviours in a more granular way, thus following the common manner defining security in the CK, CK+ and eCK models.

To date, only a few work attempted to investigate relations of the existing models. Cremers [8] noticed that, the original versions of the CK [7], $\mathrm{CK}^+$ [18] and eCK [19] models are incomparable, by showing a (somehow artificial) protocol provable secure in one model is insecure in other models. This accounts for why later work only considered security in a single model: e.g., [17,4,22] in the CK model, [21,16,25] in the eCK model, [24] in the $\mathrm{CK}^+$ model and [3,26] in the eCK-PFS model.

On the other hand, some subsequent works *did* make subtle changes to these models. For instance, Boyd et al. [4] redefined the session identifiers for the CK model using the concatenation of the messages sent and received by parties instead of a string required to be sent along with the message, which requires the definition of matching notion to be modified accordingly; in the $\mathrm{CK}^+$ model by Fujioka et al. [11,12], the definition of matching notion includes an extra restriction, i.e., two sessions must have non-equal role identifiers (denoting the actor of a session is an initiator or a responder); in the eCK-PFS model by Bergsma et al. [3], such restriction was dropped in defining the matching notion, differing from the original eCK and eCK-PFS models.

Hence a natural question arises whether insisting these different definitions of session identifiers and matching sessions really matters in practice? In addition, many different techniques and different assumptions were used for different schemes in different models, therefore, a systematic understanding of how to construct secure ORKEs is extremely necessary and helpful.

The above are exactly our motivations to revisit the CK, eCK, CK$^+$ and eCK-PFS models, in the context of ORKE as a first step, and develop a modular construction that can be proved secure in these models, respectively. As a result, we show there exists an ORKE protocol provably secure in different models, if its underlying building-blocks meet some natural security properties.

## 1.1 Our Results

**A Complete Set of Definitions for ORKE.** We present a succinct and comprehensible unification of the existing models in the context of ORKE. We also formally defined CK-PFS and CK$^+$-PFS models utilizing the notion of origin session. As the name suggests, these two models extend naturally the CK and CK$^+$ models by capturing perfect forward secrecy (PFS). Note that, they are stronger than the CK$_{\mathrm{BN}}$ and CK$^+$-sFS$^{\mathrm{NSR}}$ models, respectively. Combining with the existing CK, eCK, CK$^+$ and eCK-PFS models, we have a complete set of unified strong security definitions for ORKE.

**A New Tool KRF for Secure ORKEs.** We introduce a new tool, called Key-wise Recoverable Function (KRF). Using KRF and passively secure Key Exchange (KE) as building blocks, we give a modular construction (Fig. 6) and other extended variants, whose security holds in all the above-mentioned strong models by assuming the underlying KRF meets different security definitions.

**Unification of the Previous Works.** We note that our modular construction simultaneously captures the ideas behind several well-known constructions, including 2×KEM+DH [4] (CK security), NAXOS [19] (eCK security), HMQV [18] (CK$^+$ security) and BJS [3] (eCK-PFS security).

Independent from our work, Xue et al. [24] introduced a primitive called double-key key encapsulation mechanism (2-key KEM), based on which, they presented modular constructions to simplify the construction and analysis of CK$^+$-secure and eCK-secure AKEs. Compared with their work, our work has a wider range of application as the CK, CK-PFS, CK$^+$-PFS and eCK-PFS models were also taken into account. In addition, our modular constructions are role-symmetric, which makes it more suitable for some scenarios.

**Table 1.** Detailed Comparisons with Xue et al.

| Constructions | Role-Symmetric | Tools | Applicable Models | Unification of Previous Works | |
|---|---|---|---|---|---|
| Xue et al. [24] | N | 2-key KEM | CK, eCK | HMQV [18] | CK$^+$ |
| | | | | NAXOS [19] | eCK |
| | | | | Okamoto [21] | eCK |
| | | | | FSXY12,13 [11,12] | CK$^+$ |
| Our work | Y | KRF,KE | CK, CK$^+$, eCK<br>CK-PFS, CK$^+$-PFS, eCK-PFS | 2KEM+DH [4] | CK |
| | | | | HMQV [18] | CK$^+$ |
| | | | | NAXOS [19] | eCK |
| | | | | BJS [3] | eCK-PFS |

†† "N" denotes "no", and "Y" denotes "yes".

**New Results for ORKEs with KRFs.** We have the following new results:

- We observe that our modular construction using a same KRF achieves CK$^+$ (resp., CK$^+$-PFS) security and eCK (resp., eCK-PFS) security, which makes a protocol selection much easier in practice.
- By instantiating our modular construction with a proper KRF, we obtain a secure protocol in the CK-PFS model (first formulated in this work). Compared with the SIG(2KEM+DH) construction, an immediate scheme inspired by the work of Cremers and Feltz [9], this proposal is more efficient in terms of computation and bandwidth.
- Finally, we show that there is a KRF with full security (i.e., meeting all the security definitions), applying our modular construction, a secure ORKE is then acquired in all the known security models, namely, CK, CK$^+$, eCK, CK-PFS, CK$^+$-PFS and eCK-PFS.

## 2 Preliminary

In this section, we review some useful notations and notions.

**Notations.** For arbitrary $k \in \mathbb{N}$, $1^k$ denotes the string of $k$ ones. For an integer $m$, $[m] \overset{\text{def}}{=} \{1, 2, \ldots, m\}$. If $\mathcal{S}$ is a distribution, $x \leftarrow_{\$} \mathcal{S}$ denotes randomly choosing an element according to $\mathcal{S}$. If $\mathsf{A}$ is an algorithm, $\mathsf{A}(x; r) \rightarrow y$ denotes that $\mathsf{A}$ takes $x$ as input and $r$ as internal randomness returns output $y$. If $y$ is a variable, $y \leftarrow \mathsf{A}(x)$ denotes assigning the output of $\mathsf{A}$ with $x$ as input to $y$. A function $\mu(\cdot)$ is called negligible, if for every polynomial $p(\cdot)$, there exists some $\lambda_0$ such that $\mu(\lambda) \leq 1/p(\lambda)$, for every $\lambda > \lambda_0$.

**Passively Secure Key Exchange**. Here we define passively secure Key Exchange (KE) that is used without any long-term keys, which consists of two polynomial time algorithms: a probabilistic algorithm $\mathsf{KE.Gen}(1^\lambda) \rightarrow (pk, sk)$ that takes as input a security parameter $1^\lambda$ and returns a key pair $(pk, sk)$; a deterministic algorithm $\mathsf{KE.Key}(sk, pk') \rightarrow k$ that takes as input a secret key $sk$ and a public key $pk'$ and returns a key $k$. Let $\mathrm{CKey}(pk, pk')$ denote $k \leftarrow \mathsf{KE.Key}(sk, pk')$. Correctness requires that for any $(pk, sk) \leftarrow \mathsf{KE.Gen}(1^\lambda)$ and $(pk', sk') \leftarrow \mathsf{KE.Gen}(1^\lambda)$, $\mathrm{CKey}(pk, pk') = \mathrm{CKey}(pk', pk)$ holds.

**Definition 1 (Passive-Security).** *A KE scheme* $\mathsf{KE} = (\mathsf{KE.Gen}, \mathsf{KE.Key})$ *is called Passively Secure (PS), if for any PPT adversary* $\mathcal{A}$*, its advantage:*

$$\mathsf{Adv}^{\mathsf{PS}}_{\mathsf{KE},\mathcal{A}}(\lambda) \overset{def}{=} \Pr[b' = b : b' \leftarrow \mathcal{A}^{\mathrm{PExecute}(\cdot),\mathrm{PReveal}(\cdot),\mathrm{PTest}(\cdot)}(1^\lambda)] \leq \mu(\lambda),$$

*where* $\mathcal{A}$ *is allowed to adaptively query:*

- $\mathrm{PExecute}(i)$*: For unused identity* $i$*, compute* $(pk_i, sk_i) \leftarrow \mathsf{KE.Gen}(1^\lambda)$ *and* $(pk'_i, sk'_i) \leftarrow \mathsf{KE.Gen}(1^\lambda)$*, and return* $(pk_i, pk'_i)$*. Otherwise, do nothing.*
- $\mathrm{PReveal}(i)$*: If the identity* $i$ *has been used in previous* $\mathrm{PExecute}(\cdot)$ *queries, compute* $k_i \leftarrow \mathsf{KE.Key}(sk_i, pk'_i)$ *and return* $k_i$*.*

– PTest($i^*$): *This can be asked for only once. If* $b = 0$*, return the real key* $k_{i^*} \leftarrow \mathsf{KE.Key}(sk_{i^*}, pk'_{i^*})$*; else if* $b = 1$*, return a random key. Throughout the experiment,* PReveal($i^*$) *should never been queried.*

**Pseudo-Random Function.** Let $\mathcal{F} \overset{\text{def}}{=} \{\mathcal{F}_\lambda : \mathcal{S}_\lambda \times Dom_\lambda \rightarrow Rng_\lambda\}_{\lambda \in \mathbb{N}}$ define a function family with families of key spaces $\{\mathcal{S}_\lambda\}_{\lambda \in \mathbb{N}}$, domains $\{Dom_\lambda\}_{\lambda \in \mathbb{N}}$ and ranges $\{Rng_\lambda\}_{\lambda \in \mathbb{N}}$, where $\lambda$ denotes a security parameter.

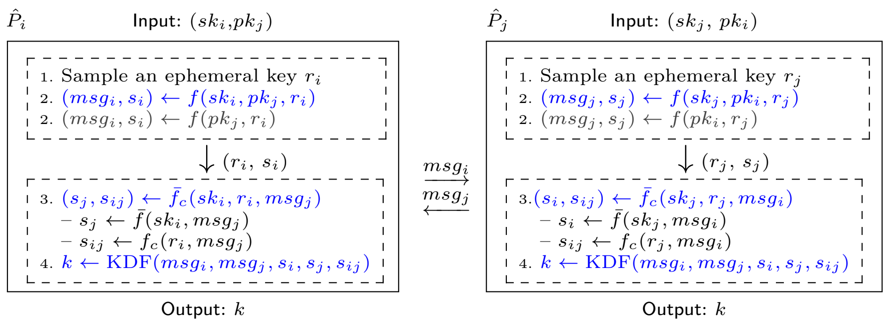
**Definition 2 (PRF).** *A function family* $\mathcal{F}$ *is called a secure Pseudo-Random Function (PRF) family if for any PPT adversary* $\mathcal{A}$*, its advantage*

$$\mathsf{Adv}^{\mathrm{PRF}}_{\mathcal{F},\mathcal{A}}(\lambda) \overset{def}{=} | \Pr[1 \leftarrow \mathcal{A}^{\mathcal{F}_\lambda(\cdot)}] - \Pr[1 \leftarrow \mathcal{A}^{\mathcal{RF}_\lambda(\cdot)}] | \leq \mu(\lambda),$$

*where* $\mathcal{RF}_\lambda(\cdot) : Dom_\lambda \rightarrow Rng_\lambda$ *is a truly random function family.*

## 3 Security Definitions for ORKEs

In this section, we unify the definitions of the CK [7,4,5], eCK [19,21], CK$^+$ [18,11,12] and eCK-PFS [9,3] models in the context of ORKE, and introduce the CK-PFS and CK$^+$-PFS models. We resemble the method defining security models used in [11], namely we formulate these models as follows: wPFS, PFS, KCI resistance, and MEX resistance are integrated into the experiments of considered models by exhaustively classifying leakage patterns. Such definitional treatment is convenient for capturing all required properties rigorously in each model, and greatly simplifies the security proofs in these models.



$\hat{P}_i$
Input: $(sk_i, pk_j)$
1. Sample an ephemeral key $r_i$
2. $(msg_i, s_i) \leftarrow f(sk_i, pk_j, r_i)$
2. $(msg_i, s_i) \leftarrow f(pk_j, r_i)$
$(r_i, s_i)$
3. $(s_j, s_{ij}) \leftarrow \bar{f}_c(sk_i, r_i, msg_j)$
– $s_j \leftarrow \bar{f}(sk_i, msg_j)$
– $s_{ij} \leftarrow f_c(r_i, msg_j)$
4. $k \leftarrow \mathrm{KDF}(msg_i, msg_j, s_i, s_j, s_{ij})$
Output: $k$
$msg_i$
$msg_j$
$\hat{P}_j$
Input: $(sk_j, pk_i)$
1. Sample an ephemeral key $r_j$
2. $(msg_j, s_j) \leftarrow f(sk_j, pk_i, r_j)$
2. $(msg_j, s_j) \leftarrow f(pk_i, r_j)$
$(r_j, s_j)$
3. $(s_i, s_{ij}) \leftarrow \bar{f}_c(sk_j, r_j, msg_i)$
– $s_i \leftarrow \bar{f}(sk_j, msg_i)$
– $s_{ij} \leftarrow f_c(r_j, msg_i)$
4. $k \leftarrow \mathrm{KDF}(msg_i, msg_j, s_i, s_j, s_{ij})$
Output: $k$

**Fig. 1.** A generic description of ORKE

We first present a generic description of ORKE to help us understand the security models. Assume each party $\hat{P}_i$ possesses a long-term public/secret key pair $(pk_i, sk_i)$, and will select an ephemeral key (i.e., the randomness $r$) in each execution instance. In general, we use three functions to abstract each party's local computations: (1) $f$ to generate the message sent to its peer party; (2) $\bar{f}_c$ to deal with the received message; (3) KDF to compute the session key. Take one execution instance between two parties $\hat{P}_i$ and $\hat{P}_j$ as an illustration (see Fig. 1). The function $f$ may take two forms. The first takes the party's own secret key as a partial input, while the second does not. We use them to capture different forms of existing protocols. For examples: in 2KEM+DH [4] and HMQV [18], each party's own secret key is not required to compute a sent message; while in NAXOS [19] and BJS [3], that is required. The function $\bar{f_c}$ can be subdivided into $\bar{f}$ and $f_c$. Note that, in ORKE, each party only sends a single message independent of the message sent by its peer party. Therefore, the usages of its long-term key and ephemeral key are different: the former is used to recover the embedded key material along with the received message, i.e., $s_i$ and $s_j$; the latter is used to negotiate a new piece of key material, i.e., $s_{ij}$.

**Syntax.** Let $\mathcal{P} = \{\hat{P}_1, \hat{P}_2, ..., \hat{P}_N\}$ be a finite set of $N$ parties' identities. A protocol $\Pi$ is a collection of $N$ interactive PPT Turing machines run by different parties. Each party can execute multiple protocol instances, called *sessions*, concurrently. Each session can only be activated once. The $i$-th session at $\hat{P}_U$ is denoted as $(\hat{P}_U, i) \in \mathcal{P} \times \mathbb{N}$. For each session $s$, a tuple of variables partially selected form the following lists will be set:

- $s_{actor}$: To denote the identity of the session's actor;
- $s_{peer}$: To denote the identity of the session's intended peer;
- $s_{sent}$: The concatenation of timely ordered messages sent by $s_{actor}$;
- $s_{recv}$: The concatenation of timely ordered messages received by $s_{actor}$;
- $s_{id}$: A string generated by $s_{actor}$ to explicitly identify the session and required to be sent along with the message;
- $s_{role}$: To denote the role of $s_{actor}$, e.g., initiator or responder.

These values will be determined once a session is activated or during the protocol execution. A session is called *completed* if it returns a session key then terminates normally. In previous works, to identify any two distinct sessions $s$ and $s'$ involved in the same instance, the notion of *matching-session* was defined:

- $s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor} \wedge s_{sid} = s'_{sid}$; *or*
- $s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor} \wedge s_{sent} = s'_{recv} \wedge s_{recv} = s'_{sent}$; *or*
- $s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor} \wedge s_{sent} = s'_{recv} \wedge s_{recv} = s'_{sent} \wedge s_{role} \neq s'_{role}$.

Among these notions: using an explicit string (i.e., $s_{sid}$) to identify a session is seldom adopted now; and in the role-symmetric setting, the variable $s_{role}$ cannot be utilized to determine whether two sessions are matched or not, since both sides are allowed to be the initiator. For these reasons, we adopt the second type. Besides, we will use the notion of *origin-session* introduced in [9], which is important to define "-PFS" models. A (possibly incomplete) session $s'$ is called an origin session for a completed session $s$ when $s'_{sent} = s_{recv}$.

**Matching-Session vs Origin-Session.** Take the right session in (b) from the three execution instances shown in Fig. 2 as an illustration, its matching session is thought to be non-existent (since that is not an honest session), but its origin session is thought to be existent, say the left session in (a). “The origin session of a session $s$ does not exist” means that $s_{recv}$ is not originated from an honest party but the adversary. In fact, two honest sessions are matched if and only if they are both origin sessions of each other.

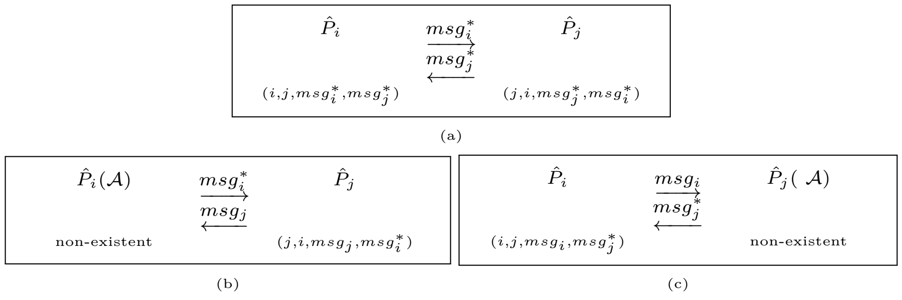


**Fig. 2.** Protocol execution instances with an adversary $\mathcal{A}$. (a) $\mathcal{A}$ passively observes. (b) $\mathcal{A}$ replays messages originated from $\hat{P}_i$. (c) $\mathcal{A}$ replays messages originated from $\hat{P}_j$.

**Oracle Queries.** The adversary is modeled as an interactive PPT Turing machine that controls all communications between parties, i.e., the adversary can eavesdrop, stop, delay and alter the messages passing over the channel. And it may be allowed to obtain session-specific secret information. These abilities are modeled via different oracle queries:

– **Send**$(s, m)$: This query models the adversary sending a message $m$ to a session $s$, and responses according to the protocol description. Abusing notations, the adversary is allowed to activate a sessuib $s$ with a peer $\hat{P}_U$ via a **Send**$(\hat{P}_U, s)$ query, or communicate with a session $s$ by sending a message $m$ on behalf of $\hat{P}_U$ via a **Send**$(\hat{P}_U, s, m)$ query.
– **Corrupt**$(\hat{P}_U)$: This query models long-term key (LTK) leakages, and returns the LTK of $\hat{P}_U$, which is denoted as LTK$[\hat{P}_U]$.
– **SKReveal**$(s)$: This query models session key (SK) leakages, and returns the SK of $s$ if it is *completed*, which is denoted as SK$[s]$.
– **EphKReveal**$(s)$: This query models ephemeral key (EphK) leakages, and returns the EphK (i.e., the randomness) of $s$, which is denoted as EphK$[s]$.
– **SSReveal**$(s)$: This query models session state (SS) leakages, and returns the SS of $s$ before it completes, which is denoted as SS$[s]$.
– **Test**$(s)$: This query does not model practical attacks, but is important for indistinguishability-based security definitions. A random coin $b$ is flipped: if $b = 0$, return SK$[s]$; else return a random key. This query can be issued for only once and must be on a session that is both *completed* and *fresh*. The notion of *freshness* is defined as in the last column of Table 2. Jumping ahead, in the experiment, the input of this query is called the adversary’s target session, and denoted as $s^*$ throughout this paper. In addition, we use $\bar{s}^*$ and $\tilde{s}^*$ to denote its intended matching session and origin session, respectively. If without any explicit statement, they are thought to be existent.

**Table 2.** Allowed Queries and Freshness in Different Models

| Model | Allowed Queries | Freshness |
|---|---|---|
| CK | Send$(\cdot)$<br>Corrupt$(\cdot)$<br>SSReveal$(\cdot)$<br>SKReveal$(\cdot)$<br>Test$(\cdot)$ | The adversary has never perform:<br>· SKReveal$(s^*)$ and SSReveal$(s^*)$;<br>· SKReveal$(\bar{s}^*)$ and SSReveal$(\bar{s}^*)$;<br>· Corrupt$(s^*_{peer})$ if the target's matching session does not exist. |
| CK-PFS | | The adversary has never perform:<br>· SKReveal$(s^*)$ and SSReveal$(s^*)$;<br>· SKReveal$(\bar{s}^*)$ and SSReveal$(\bar{s}^*)$;<br>· Corrupt$(s^*_{peer})$ if the matching session does not exist but the origin session does exist;<br>· Corrupt$(s^*_{peer})$ before the completion of $s^*$ if the target's origin session does not exist. |
| CK$^+$ | | The adversary has never perform:<br>· SKReveal$(s^*)$ and SSReveal$(s^*)$;<br>· SKReveal$(\bar{s}^*)$ and SSReveal$(\bar{s}^*)$;<br>It is limited to obtain one key combination as follows:<br>· LTK$[s^*_{actor}]$ and LTK$[\bar{s}^*_{actor}]$;<br>· EphK$[s^*]$ and EphK$[\bar{s}^*]$;<br>· LTK$[s^*_{actor}]$ and EphK$[\bar{s}^*]$;<br>· EphK$[s^*]$ and LTK$[\bar{s}^*_{actor}]$;<br>· LTK$[s^*_{actor}]$ if the target's matching session does not exist;<br>· EphK$[s^*]$ if the target's matching session does not exist. |
| CK$^+$-PFS | | The adversary has never perform:<br>· SKReveal$(s^*)$ and SSReveal$(s^*)$;<br>· SKReveal$(\bar{s}^*)$ and SSReveal$(\bar{s}^*)$;<br>It is limited to obtain one key combination as follows:<br>· LTK$[s^*_{actor}]$ and LTK$[\tilde{s}^*_{actor}]$;<br>· EphK$[s^*]$ and EphK$[\tilde{s}^*]$;<br>· LTK$[s^*_{actor}]$ and EphK$[\tilde{s}^*]$;<br>· EphK$[s^*]$ and LTK$[\tilde{s}^*_{actor}]$;<br>· LTK$[s^*_{actor}]$ and LTK$[s^*_{peer}]$ if the target's origin session does not exist, but the latter should be after the completion of $s^*$;<br>· EphK$[s^*]$ and LTK$[s^*_{peer}]$ if the target's origin session does not exist, but the latter should be after the completion of $s^*$. |
| eCK | Send$(\cdot)$<br>Corrupt$(\cdot)$<br>EphKReveal$(\cdot)$<br>SKReveal$(\cdot)$<br>Test$(\cdot)$ | The adversary has never perform:<br>· SKReveal$(s^*)$ and SKReveal$(\bar{s}^*)$;<br>· both Corrupt$(s^*_{actor})$ and EphKReveal$(s^*)$;<br>· both Corrupt$(\bar{s}^*_{actor})$ and EphKReveal$(\bar{s}^*)$;<br>· Corrupt$(s^*_{peer})$ if the target's matching session does not exist. |
| eCK-PFS | | The adversary has never perform:<br>· SKReveal$(s^*)$ and SKReveal$(\bar{s}^*)$;<br>· both Corrupt$(s^*_{actor})$ and EphKReveal$(s^*)$;<br>· both Corrupt$(\tilde{s}^*_{actor})$ and EphKReveal$(\tilde{s}^*)$;<br>· Corrupt$(s^*_{peer})$ before the completion of $s^*$ if the target's origin session does not exist. |

†† In the CK-PFS and CK$^+$-PFS models, SSReveal$(\cdot)$ is only forbade on the target session $s^*$ and its matching session $\bar{s}^*$, but still allowed on its origin session $\tilde{s}^*$. Under a special case that the target's matching session doesn't exist but its origin session exists. , the adversary may perform SSReveal$(\tilde{s}^*)$ to get SS$[\tilde{s}^*]$ that includes EphK$[\tilde{s}^*]$ and some other intermediates.

**Important Security Notions and the Experiment.** Before giving the formal security definition, we recall several important security goals for ORKEs:

– **Perfect Forward Secrecy (PFS)**: To guarantee the secrecy of older SKs, say any PPT adversary is unable to distinguish them from random keys, even when the LTKs of both parties are corrupted.
– **weak Perfect Forward Secrecy (wPFS)**, a weak version of PFS: To guarantee the secrecy of older SKs, whose negotiation processes were not thrust in, even when the LTKs of both parties are corrupted.

– **resistance to Key-Compromise Impersonation (KCI)**: To guarantee the secrecy of SKs under KCI attacks. In a KCI attack, an adversary corrupts a party $\hat{P}_i$ and tries to authenticate itself to $\hat{P}_i$ as some uncorrupted party $\hat{P}_j$. Once succeeds, it can compute the SK and break the secrecy trivially.
– **resistance to Maximal EXposure (MEX)**: To guarantee the secrecy of a SK under the disclosure of any pair of LTKs and EphKs of both parties in the session except for both the LTK and EphK of each party.

The formal security definition in each model is defined via a two-phase experiment played between a challenger and an adversary $\mathcal{A}$. In Phase-I, $\mathcal{A}$ may adaptively perform allowed oracle queries as collected in Table 2. At some point, $\mathcal{A}$ performs a Test$(\cdot)$ query on a target of its choice. In Phase-II, $\mathcal{A}$ can continue with its regular actions like in the first phase. Eventually, $\mathcal{A}$ outputs a guess bit $b'$ and halts. If $b = b'$ and the target session is kept *fresh* throughout the experiment, then $\mathcal{A}$ is determined as winning in the experiment.

**Defining the Output of SSReveal$(\cdot)$.** In Fig. 3, we illustrate the execution processes of $\hat{P}_i$ in Fig. 1 and the timing of the SSReveal$(\cdot)$ query may be allowed. The adversary may trivially win without any limitation.

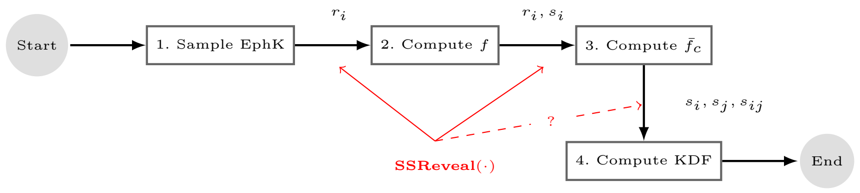


**Fig. 3.** An illustration of the execution processes

Consider the most extreme case that the SSReveal$(\cdot)$ query may return all internal states, i.e., $(r_i, s_i, s_j, s_{ij})$. Elaborate with the CK$^+$ experiment and the three execution instances shown in Fig. 2. Assume the adversary $\mathcal{A}$ chooses the left session in (a) as its target eventually, the session key materials of which we denoted as $s_i^*$, $s_j^*$ and $s_{ij}^*$. According to the definition, the right session in (b) and the left session in (c) are both not matched to the target session, thus $\mathcal{A}$ can perform SSReveal$(\cdot)$ queries on them to get $s_i^*$ and $s_j^*$, respectively. Besides, $\mathcal{A}$ can chose to reveal the EphK of the target's matching session, say the right session in (a), which helps $\mathcal{A}$ to obtain $s_{ij}^*$. By doing so, $\mathcal{A}$ can compute the target session key, thus trivially win.

In previous works [4,5,11,12,24], it is assumed that the intermediate values computed from the received message and own secret key will not be stored for a long time before computing the session key, which should be securely erased once the computation is over. That is equivalent to assume the 3rd and 4th steps shown in Fig. 1 and Fig. 3 are inseparable, thus the SSReveal$(\cdot)$ query is broken down once the party begins dealing with the received message. To make the security model definitions still meaningful in the role-symmetric and one-round setting, we also put such constraint on SSReveal$(\cdot)$ queries.

**Table 3.** The CK Model

| Case | the target session $s^*$ | | the matching session $\bar{s}^*$ | | Security |
|---|---|---|---|---|---|
| | EphK[$s^*$] | LTK[$s^*_{actor}$] | EphK[$\bar{s}^*$] | LTK[$s^*_{peer}$]=LTK[$\bar{s}^*_{actor}$] | |
| I | | ✓ | | ✓ | wPFS |
| II | | ✓ | – | | KCI |

**Table 4.** The CK-PFS Model

| Case | the target session $s^*$ | | the origin session $\tilde{s}^*$ | | Security |
|---|---|---|---|---|---|
| | EphK[$s^*$] | LTK[$s^*_{actor}$] | EphK[$\tilde{s}^*$] | LTK[$s^*_{peer}$]=LTK[$\tilde{s}^*_{actor}$] | |
| I | | ✓ | | ✓ | wPFS |
| II | | ✓ | ✓ | | KCI |
| III | | ✓ | – | $✓_\tau$ | KCI-PFS |

**Table 5.** The CK$^+$ Model

| Case | the target session $s^*$ | | the matching session $\bar{s}^*$ | | Security |
|---|---|---|---|---|---|
| | EphK[$s^*$] | LTK[$s^*_{actor}$] | EphK[$\bar{s}^*$] | LTK[$s^*_{peer}$]=LTK[$\bar{s}^*_{actor}$] | |
| I | ✓ | | | ✓ | MEX |
| II | ✓ | | ✓ | | MEX |
| III | | ✓ | | ✓ | wPFS |
| IV | | ✓ | ✓ | | MEX |
| V | ✓ | | – | | MEX |
| VI | | ✓ | – | | KCI |

**Table 6.** The CK$^+$-PFS Model

| Case | the target session $s^*$ | | the origin session $\tilde{s}^*$ | | Security |
|---|---|---|---|---|---|
| | EphK[$s^*$] | LTK[$s^*_{actor}$] | EphK[$\tilde{s}^*$] | LTK[$s^*_{peer}$]=LTK[$\tilde{s}^*_{actor}$] | |
| I | ✓ | | | ✓ | MEX |
| II | ✓ | | ✓ | | MEX$^+$ |
| III | | ✓ | | ✓ | wPFS |
| IV | | ✓ | ✓ | | MEX$^+$ |
| V | ✓ | | – | $✓_\tau$ | MEX-PFS |
| VI | | ✓ | – | $✓_\tau$ | KCI-PFS |

**Table 7.** The eCK Model

| Case | the target session $s^*$ | | the matching session $\bar{s}^*$ | | Security |
|---|---|---|---|---|---|
| | EphK[$s^*$] | LTK[$s^*_{actor}$] | EphK[$\bar{s}^*$] | LTK[$s^*_{peer}$]=LTK[$\bar{s}^*_{actor}$] | |
| I | ✓ | | | ✓ | MEX |
| II | ✓ | | ✓ | | MEX |
| III | | ✓ | | ✓ | wPFS |
| IV | | ✓ | ✓ | | MEX |
| V | ✓ | | – | | MEX |
| VI | | ✓ | – | | KCI |

**Table 8.** The eCK-PFS Model

| Case | the target session $s^*$ | | the origin session $\tilde{s}^*$ | | Security |
|---|---|---|---|---|---|
| | EphK[$s^*$] | LTK[$s^*_{actor}$] | EphK[$\tilde{s}^*$] | LTK[$s^*_{peer}$]=LTK[$\tilde{s}^*_{actor}$] | |
| I | ✓ | | | ✓ | MEX |
| II | ✓ | | ✓ | | MEX |
| III | | ✓ | | ✓ | wPFS |
| IV | | ✓ | ✓ | | MEX |
| V | ✓ | | – | $✓_\tau$ | MEX-PFS |
| VI | | ✓ | – | $✓_\tau$ | KCI-PFS |

†† The symbol ✓denotes that $\mathcal{A}$ is allowed to corrupt the key; – denotes empty value because the corresponding session does not exist at all; $✓_\tau$ denotes that $\mathcal{A}$ is allowed to corrupt the key, but should after the completion of the target session.

**Model Formulations.** We formulate the CK, CK-PFS, CK$^+$, CK$^+$-PFS, eCK and eCK-PFS models as in Tables 3 to 8. We use "KCI-PFS", "MEX-PFS" and "MEX$^+$" to distinguish them from the standard KCI and MEX notions. The first two are considered in the "-PFS" models, where if the origin session of the target session $s^*$ doesn't exist, LTK($s^*_{actor}$) is allowed to be corrupted after the completion of $s^*$. As for the notion of "MEX$^+$", it is only used in the CK$^+$-PFS model, where a spacial event may occur, i.e., the matching session of the target session doesn't exist but its origin session exists. Recall that, SSReveal($\cdot$) query is not forbade on the target's non-matching sessions, thus the EphK of the origin session may be corrupted. Note that the SSReveal($\cdot$) query does not merely return the EphK, which makes it different to the same numbered cases in the CK$^+$, eCK and eCK-PFS models.

In the literature, the CK$_{\mathrm{BN}}$ [6] and CK$^+$-sFS$^{\mathrm{NSR}}$ [27] were also introduced to capture PFS for the CK and CK$^+$ models, respectively. But they both didn't utilize the notion of origin-session. To avoid the trivial case that the adversary derives the EphK of the target's origin session and the LTK of the peer party at the same time, some constraints are required. In the CK$_{\mathrm{BN}}$ model, SSReveal($\cdot$) query should be forbade to capture PFS, otherwise one should back done to consider wPFS. In the CK$^+$-sFS$^{\mathrm{NSR}}$ model: if the target's matching session does not exist, the adversary is allowed to corrupt the owner of the target session, and also the peer party after the completion of $s^*$, but with a precondition that SSReveal($\cdot$) query is not allowed to any session between the owner and the peer of $s^*$; otherwise, the adversary is not allowed to corrupt the peer party at all.

We should emphasize, what are considered in the $\mathrm{CK_{BN}}$ model can be classified into the Case-I and Case-III in Table 4. Moreover, the CK-PFS model also take KCI resistance into consideration, which makes it stronger. Besides, what are considered in the $\mathrm{CK^{+}\text{-}sFS^{NSR}}$ model are also considered in the $\mathrm{CK^{+}}$-PFS model. But as we insist, the existence of the origin session does not imply the existence of the matching session, such that the cases considered in the $\mathrm{CK^{+}}$-PFS model cannot be fully covered by the $\mathrm{CK^{+}\text{-}sFS^{NSR}}$ model. Therefore, the $\mathrm{CK^{+}}$-PFS model is stronger than the $\mathrm{CK^{+}\text{-}sFS^{NSR}}$ model too.

**Differences Among These Models.** The key difference between "-PFS" models and others is that the formers allow $\mathrm{LTK}[s^{*}_{peer}]$ corruption after the completion of the target session, even its origin session does not exist. Recall that the existence of the target's matching session implies its origin session's existence, but not vice versa. Therefore, "-PFS" models consider more complex situations, e.g., the cases I and III in the $\mathrm{CK^{+}}$-PFS model are not allowed in the $\mathrm{CK^{+}}$ model. The CK and CK-PFS models differ from others: they do not consider the MEX attack and its variants. The eCK (resp., eCK-PFS) model differs from the CK and $\mathrm{CK^{+}}$(resp., CK-PFS and $\mathrm{CK^{+}}$-PFS) models: it does not allow the SSReveal$(\cdot)$ query, instead of the EphKReveal$(\cdot)$ query. The former not only returns EphKs, but also some intermediates. As shown in Fig. 1, computing these intermediates may involve the session owner's LTK, thus the leakage through the SSReveal$(\cdot)$ query is at least no smaller than the EphKReveal $(\cdot)$ query. We should emphasize that this statement is not absolute when other variants of SSReveal$(\cdot)$ out of this paper are considered, e.g., it merely returns intermediates derived from both the LTK and EphK through some one-way computations, the adversary may learn no more than directly asking the EphKReveal$(\cdot)$ query.

**Definition 3.** *A protocol* $\Pi$ *is called secure in a specific model if and only if for any* PPT *adversary* $\mathcal{A}$*, the following properties hold,*

- *Two honest parties complete matching sessions output the same key;*
- *The advantage* $\mathsf{Adv}^{\mathsf{Model}}_{\mathcal{A},\Pi}(\lambda) = |\Pr[b' = b] - 1/2|$ *that* $\mathcal{A}$ *wins in the experiment is negligible, where* $\mathsf{Model} \in \{\mathsf{CK}, \mathsf{CK\text{-}PFS}, \mathsf{CK}^{+}, \mathsf{CK}^{+}\text{-}\mathsf{PFS}, \mathsf{eCK}, \mathsf{eCK\text{-}PFS}\}$.

## 4 Our Modular Construction

In this section, we present some observations, motivated by which, we introduce a new tool KRF (Key-wise Recoverable Function) and our modular construction.

**Essential Observations.** Recall the abstraction in Fig. 1. To build a secure ORKE protocol, one should give proper implementations of $(f, \bar{f}_c = (\bar{f}, f_c)$, KDF). Among these functions, $f_c$ is used to negotiate a key material from both parties' EphKs. To the best of our knowledges, this is to achieve wPFS, an important security goal as we mentioned before. In fact, by itself, only passive attacks can be resisted. We can find its implementations easily, e.g., the typical Diffie-Hellman Key Exchange (DHKE). As for KDF, its functionality is just to derive a session key from already prepared key materials. Our essential goal is to give a modular understanding how to prepare these key materials. Put together $f$ that locates on the left (resp., right) and $\bar{f}$ that locates on the right (resp., left). The conceptual structure of ORKE can be abstracted as the "$2 \times (f, \bar{f}) + f_c + \text{KDF}$" paradigm.

## 4.1 Key-wise Recoverable Function (KRF)

How to implement $(f, \bar{f})$ becomes very important, which motivates us to define a new tool, namely Key-wise Recoverable Function (KRF). To give a proper definition for it is our starting point. Note that $(f, \bar{f})$ can be essentially viewed as an abstraction of an another type of key exchange (sometimes called One-Pass Key Exchange), where the initiator (e.g., $\hat{P}_i$) sends a single message to the responder (e.g., $\hat{P}_j$) without requiring response message: with $f$, $\hat{P}_i$ takes its own secret key $sk_i$, $\hat{P}_j$'s public key $pk_j$ and a randomness $r_i$ as input, and gets two output $(msg, s)$; with $\bar{f}$, $\hat{P}_j$ takes its own secret key $sk_j$, $\hat{P}_i$'s public key $pk_i$ and the received message $msg$ as input, can recover the secret $s$. Intuitively, to achieve an authenticated key establishment, it's well if such key exchange module satisfies the following properties:

1. $\hat{P}_j$ assures that the message $msg$ is indeed sent from the claimed $\hat{P}_i$;
2. $\hat{P}_i$ assures that only the intended $\hat{P}_j$ is able to compute the correct $s$.

These two properties inspired us to define *private evaluation* and *private recoverability* for KRF, respectively. Besides these, to determine whether more properties are required or not, we take a closer look at the CK, eCK, CK$^+$, CK-PFS, eCK-PFS and CK$^+$-PFS models. Recall that, to achieve security in these models, the key is to achieve wPFS and resistances to the KCI, MEX, MEX$^+$, MEX-PFS and KCI-PFS attacks. Among these goals, achieving wPFS can be achieved by properly implementing $f_c$.

To resist the MEX and MEX$^+$ attacks, it is required to assure that as long as one of the EphK (i.e., $r_i$) and the LTK (i.e.,$sk_i$) is kept secret, the adversary is unable to compute the correct $s$. To achieve this, we further define the notion *private recoverability* under different leakages, i.e., under the leakage of the randomness $r_i$ or the secret key $sk_i$. After defining *private recoverability* under the leakage of the secret key $sk_i$, we were able to resist the KCI attack. Consider a case that an adversary tries to authenticate itself to $\hat{P}_i$ as $\hat{P}_j$. Even the adversary has corrupted $sk_i$, basing on this property, it is unable to compute the correct $s$. To resist the MEX-PFS and KCI-PFS attacks, our idea is similar to [9,3]: if the adversary doesn't know the LTK of the target's peer, it is unable to originate a valid message to make the target session terminate normally without rejection. To achieve this, it is enough to define the *private evaluation* property.

Up to now, we have roughly considered all intended security goals. Next we formally define KRFs.

**Informal Description of KRF.** A KRF evaluates a set of function pairs $\{(f, \bar{f})\}$ indexed by an evaluation/re-evaluation key pair, e.g., $(\mathsf{ek}, \mathsf{rk})$, and their public keys are denoted as $\mathsf{epk}$ and $\mathsf{rpk}$, respectively. As shown in Fig. 4: on input $(\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2)$, $f_{\mathsf{ek}}(\cdot,\cdot,\cdot)$ outputs $(\mathsf{y}, \mathsf{w})$; for its paired function $\bar{f}_{\mathsf{rk}}(\cdot,\cdot,\cdot)$, it is able to recover $\mathsf{w}$ from $(\mathsf{epk}, \mathsf{x}_1, \mathsf{y})$. Here $\mathsf{x}_1$ captures some public input, which can also be set as empty if useless. A KRF may provide following security guarantees:

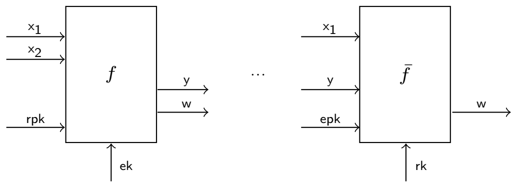


**Fig. 4.** An Illustration of KRF

– Private Evaluation: without $\mathsf{ek}$, any adversary is unable to generate a proper $(\mathsf{x}_1, \mathsf{y})$ pair such that $\bar{f}_{\mathsf{rk}}(\mathsf{epk}, \mathsf{x}_1, \mathsf{y}) \neq \perp$.
– Private Recoverability: without $\mathsf{rk}$, any adversary is unable to recover any information of $\mathsf{w}$ from $(\mathsf{x}_1, \mathsf{y})$ even the secret $\mathsf{ek}$ or $\mathsf{x}_2$ has been leaked.

**Formal Definition of KRF.** A Key-wise Recoverable Function (KRF) consists of the following three polynomial time algorithms:

– $\mathsf{KRF.Setup}(1^\lambda) \rightarrow \mathsf{pp}$: a probabilistic algorithm that takes as input a security parameter $1^\lambda$ and returns a common parameter $\mathsf{pp}$ that determines the key space $\mathcal{K} = (\mathcal{K}_0, \mathcal{K}_1)$ and four other spaces $(\mathcal{X}_1, \mathcal{X}_2, \mathcal{Y}, \mathcal{W})$;
– $\mathsf{KRF.KG}(\mathsf{pp}, \psi) \rightarrow (\mathsf{pk}, \mathsf{sk})$: a probabilistic algorithm that takes as input a common parameter $\mathsf{pp}$ and a signal bit $\psi \in \{0,1\}$, and returns a public/secret key pair $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{K}_\psi$;
– $\mathsf{KRF.Eval}(\psi, \mathsf{input}) \rightarrow \mathsf{output}$: a deterministic algorithm that evaluates $f$ or $\bar{f}$ according to the signal bit $\psi \in \{0,1\}$:
  - if $\psi = 0$, phrase $\mathsf{input}$ as a tuple of $(\mathsf{ek}, \mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2) \in \mathcal{K}_0 \times \mathcal{K}_1 \times \mathcal{X}_1 \times \mathcal{X}_2$ and evaluate $f_{\mathsf{ek}}(\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2)$ that outputs a tuple $(\mathsf{y}, \mathsf{w}) \in \mathcal{Y} \times \mathcal{W}$.
  - else if $\psi = 1$, phrase $\mathsf{input}$ as a tuple of $(\mathsf{rk}, \mathsf{epk}, \mathsf{x}_1, \mathsf{y}) \in \mathcal{K}_1 \times \mathcal{K}_0 \times \mathcal{X}_1 \times \mathcal{Y}$ and evaluate $\bar{f}_{\mathsf{rk}}(\mathsf{epk}, \mathsf{x}_1, \mathsf{y})$ that outputs an element $\mathsf{w} \in \mathcal{W}$ or a rejection symbol $\perp$ indicating false input.

*Correctness.* For any $\mathsf{pp} \leftarrow \mathsf{KRF.Setup}(1^\lambda)$, $(\mathsf{epk}, \mathsf{ek}) \leftarrow \mathsf{KRF.KG}(\mathsf{pp}, 0)$, $(\mathsf{rpk}, \mathsf{rk}) \leftarrow \mathsf{KG}(\mathsf{pp}, 1)$ and $(\mathsf{x}_1, \mathsf{x}_2) \in \mathcal{X}_1 \times \mathcal{X}_2$, $(\mathsf{y}, \mathsf{w}) \leftarrow \mathsf{KRF.Eval}(0, \mathsf{ek}, \mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2)$, $\mathsf{w}' \leftarrow \mathsf{KRF.Eval}(1, \mathsf{rk}, \mathsf{epk}, \mathsf{x}_1, \mathsf{y})$, it holds that $\mathsf{w} = \mathsf{w}'$ with overwhelming probability.

**Definition 4 (KRF).** *A KRF scheme KRF=(KRF.Setup, KRF.KG, KRF.Eval) is called Privately Evaluateable (PE), Privately Recoverable under the Leakage of Evaluation Key (PR-LEK) or Privately Recoverable under the Leakage of Full Input (PR-LEX), if for any PPT adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}^{\mathsf{PE/PR\text{-}LEK/PR\text{-}LX}}_{\mathsf{KRF},\mathcal{A}}(\lambda) \overset{def}{=} \Pr[\mathsf{Exp}^{\mathsf{PE/PR\text{-}LEK/PR\text{-}LX}}_{\mathsf{KRF},\mathcal{A}}(\lambda) = 1] \leq \mu(\lambda).$$

```
$\mathsf{Exp}^{\mathsf{PE}}_{\mathsf{KRF},\mathcal{A}}(\lambda)$ :
    $\mathsf{Initialization}(\lambda)$
    $(\mathsf{x}_1^*, \mathsf{y}^*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{info}, \mathsf{rk}^*)$
    if $\mathsf{KRF.Eval}(1, \mathsf{rk}^*, \mathsf{epk}^*, \mathsf{x}_1^*, \mathsf{y}^*) \neq \perp \wedge (\mathsf{x}_1^*, \mathsf{y}^*) \notin \mathcal{Q}$,
    return 1; else return 0
$\mathsf{Exp}^{\mathsf{PR\text{-}LEK}}_{\mathsf{KRF},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}(\lambda)$ :
    $\mathsf{Initialization}(\lambda)$
    $\mathsf{x}_1^* \leftarrow \mathcal{A}_1^{\bar{\mathcal{O}}(\cdot)}(\mathsf{info}, \mathsf{ek}^*)$, $\mathsf{x}_2^* \leftarrow \mathcal{X}_2$
    $(\mathsf{y}^*, \mathsf{w}_0^*) \leftarrow \mathsf{KRF.Eval}(0, \mathsf{ek}^*, \mathsf{rpk}^*, \mathsf{x}_1^*, \mathsf{x}_2^*)$
    $\mathsf{w}_1^* \leftarrow\$ \mathcal{W}$, $b \leftarrow\$ \{0,1\}$
    $b' \leftarrow \mathcal{A}_2^{\bar{\mathcal{O}}(\cdot)}(\mathsf{info}, \mathsf{ek}^*, \mathsf{x}_1^*, \mathsf{y}^*, \mathsf{w}_b^*)$
    if $b' = b$, return 1; else return 0
$\mathsf{Exp}^{\mathsf{PR\text{-}LX}}_{\mathsf{KRF},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}(\lambda)$ :
    $\mathsf{Initialization}(\lambda)$
    $\mathsf{x}_1^* \leftarrow \mathcal{A}_1^{\mathcal{O}(\cdot),\bar{\mathcal{O}}(\cdot)}(\mathsf{info})$, $\mathsf{x}_2^* \leftarrow \mathcal{X}_2$
    $(\mathsf{y}^*, \mathsf{w}_0^*) \leftarrow \mathsf{Eval}(0, \mathsf{ek}^*, \mathsf{rpk}^*, \mathsf{x}_1^*, \mathsf{x}_2^*)$
    $\mathsf{w}_1^* \leftarrow\$ \mathcal{W}$, $b \leftarrow\$ \{0,1\}$
    $b' \leftarrow \mathcal{A}_2^{\mathcal{O}(\cdot),\bar{\mathcal{O}}(\cdot)}(\mathsf{info}, \mathsf{x}_1^*, \mathsf{x}_2^*, \mathsf{y}^*, \mathsf{w}_b^*)$
    if $b' = b$, return 1; else return 0
```

```
$\mathsf{Initialization}(\lambda)$:
    $\mathcal{Q} \leftarrow \emptyset$, $\mathsf{pp} \leftarrow \mathsf{KRF.Setup}(1^\lambda)$
    $(\mathsf{epk}^*, \mathsf{ek}^*) \leftarrow \mathsf{KRF.KG}(\mathsf{pp}, 0)$
    $(\mathsf{rpk}^*, \mathsf{rk}^*) \leftarrow \mathsf{KRF.KG}(\mathsf{pp}, 1)$
    $\mathsf{info} \leftarrow (\mathsf{pp}, \mathsf{epk}^*, \mathsf{rpk}^*)$
$\mathcal{O}(\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2)$:
    if $(\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2) = (\mathsf{rpk}^*, \mathsf{x}_1^*, \mathsf{x}_2^*)$
        output $\perp$
    else if $(\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2) \neq (\mathsf{rpk}^*, \mathsf{x}_1^*, \mathsf{x}_2^*)$
        $(\mathsf{y}, \mathsf{w}) \leftarrow \mathsf{KRF.Eval}(0, \mathsf{ek}^*, \mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2)$
        if $\mathsf{rpk} = \mathsf{rpk}^*$
            $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathsf{x}_1, \mathsf{y})\}$
        output $(\mathsf{y}, \mathsf{w})$
$\bar{\mathcal{O}}(\mathsf{epk}, \mathsf{x}_1, \mathsf{y})$:
    if $(\mathsf{epk}, \mathsf{x}_1, \mathsf{y}) = (\mathsf{epk}^*, \mathsf{x}_1^*, \mathsf{y}^*)$
        output $\perp$
    else if $(\mathsf{epk}, \mathsf{x}_1, \mathsf{y}) \neq (\mathsf{epk}^*, \mathsf{x}_1^*, \mathsf{y}^*)$
        $\mathsf{w} \leftarrow \mathsf{KRF.Eval}(1, \mathsf{rk}^*, \mathsf{epk}, \mathsf{x}_1, \mathsf{y})$
        output $\mathsf{w}$
```

**Fig. 5.** The PE, PR-LEK and PR-LX experiments of KRF

## 4.2 A Modular Construction for ORKE

In this section, we introduce our modular construction. Two building blocks are used, i.e., a KRF scheme $\mathsf{KRF}$=($\mathsf{KRF.Setup}$, $\mathsf{KRF.KG}$, $\mathsf{KRF.Eval}$) that evaluates $f$ and $\bar{f}$ functions, and a KE scheme $\mathsf{KE} = (\mathsf{KE.Gen}, \mathsf{KE.Key})$ with randomness message $\mathcal{R}$. Our modular construction consists of the following three parts:
**Setup.** Generate $\mathsf{pp} \leftarrow \mathsf{KRF.Setup}(1^\lambda)$, select a collision resilient hash function $\mathsf{H}_0 : \{0,1\}^* \rightarrow \mathcal{X}_1$, and publish $(\mathsf{pp}, \mathsf{H}_0)$ as the system parameters.
**Long-term secrets.** Each party $\hat{P}_i$ is identified by an unique identifier $i \in [N]$ and in possession of two key pairs $(\mathsf{epk_i}, \mathsf{ek_i}) \leftarrow \mathsf{KRF.KG}(\mathsf{pp}, 0)$ and $(\mathsf{rpk_i}, \mathsf{rk_i}) \leftarrow \mathsf{KRF.KG}(\mathsf{pp}, 1)$. We assume all identifiers are comparable.
**Session execution.** To negotiate a session key, two parties, say $\hat{P}_i$ and $\hat{P}_j$ (with $i \leq j$), should execute as the description in Fig. 6.

| $\hat{P}_i(\mathsf{ek}_i, \mathsf{rk}_i)$ | | $\hat{P}_j(\mathsf{ek}_j, \mathsf{rk}_j)$ |
|---|---|---|
| $r_{i,1} \leftarrow\$ \mathcal{R}, r_{i,2} \leftarrow\$ \mathcal{X}_2$ | | $r_{j,1} \leftarrow\$ \mathcal{R}, r_{j,2} \leftarrow\$ \mathcal{X}_2$ |
| $(pk_i, sk_i) \leftarrow \mathsf{KE.Gen}(1^\lambda)$ | $\xrightarrow{pk_i, \mathsf{y}_i}$ | $(pk_j, sk_j) \leftarrow \mathsf{KE.Gen}(1^\lambda)$ |
| $\mathsf{x}_{i,1} \leftarrow \mathsf{H}_0(pk_i)$, $\mathsf{x}_{i,2} \leftarrow r_{i,2}$ | $\xleftarrow{pk_j, \mathsf{y}_j}$ | $\mathsf{x}_{j,1} \leftarrow \mathsf{H}_0(pk_j)$, $\mathsf{x}_{j,2} \leftarrow r_{j,2}$ |
| $(\mathsf{y}_i, \mathsf{w}_i) \leftarrow f_{\mathsf{ek}_i}(\mathsf{rpk}_j, \mathsf{x}_{i,1}, \mathsf{x}_{i,2})$ | | $(\mathsf{y}_j, \mathsf{w}_j) \leftarrow f_{\mathsf{ek}_j}(\mathsf{rpk}_i, \mathsf{x}_{j,1}, \mathsf{x}_{j,2})$ |
| $\mathsf{x}_{j,1} \leftarrow \mathsf{H}_0(pk_j)$ | | $\mathsf{x}_{i,1} \leftarrow \mathsf{H}_0(pk_i)$ |
| $\mathsf{w}'_j \leftarrow \bar{f}_{\mathsf{rk}_i}(\mathsf{epk}_j, \mathsf{x}_{j,1}, \mathsf{y}_j)$ | | $\mathsf{w}'_i \leftarrow \bar{f}_{\mathsf{rk}_j}(\mathsf{epk}_i, \mathsf{x}_{i,1}, \mathsf{y}_i)$ |
| Abort if $\mathsf{w}'_j = \perp$ | | Abort if $\mathsf{w}'_i = \perp$ |
| $k \leftarrow \mathsf{KE.Key}(sk_i, pk_j)$ | | $k' \leftarrow \mathsf{KE.Key}(sk_j, pk_i)$ |
| Let $T \leftarrow \mathsf{epk}_i \| \mathsf{rpk}_i \| \mathsf{epk}_j \| \mathsf{rpk}_j \| pk_i \| \mathsf{y}_i \| pk_j \| \mathsf{y}_j$ | | |
| $\hat{P}_i$ compute: $SK_i \leftarrow \mathrm{PRF}(\mathsf{w}_i, T) \oplus \mathrm{PRF}(\mathsf{w}'_j, T) \oplus \mathrm{PRF}(k, T)$ | | |
| $\hat{P}_j$ compute: $SK_j \leftarrow \mathrm{PRF}(\mathsf{w}'_i, T) \oplus \mathrm{PRF}(\mathsf{w}_j, T) \oplus \mathrm{PRF}(k', T)$ | | |

**Fig. 6.** Our modular construction

**Theorem 1.** *The modular construction shown in Fig. 6 instantiated by different* KEs *and* KRFs *yields different* ORKEs *in different models as shown in Table 9:*

**Table 9.** The main results of our modular construction

| Instantiations | Models | Requirements for the KE | Requirements for the KRF | | |
|---|---|---|---|---|---|
| | | | PE | PR-LEK | PR-LX |
| ORKEs | CK | PS | | ○ | |
| | CK-PFS | | ○ | ○ | |
| | CK$^+$ | | | ○ | ○ |
| | CK$^+$-PFS | | ○ | ○ | ○ |
| | eCK | | | ○ | ○ |
| | eCK-PFS | | ○ | ○ | ○ |

†† The symbol ○ denotes that the corresponding property is required.

**Table 10.** High-level proof strategies of our modular construction

| Models | Sub-Events | Sessions | | $\mathcal{A}$'s Knowledge | | | | | Unexposed | Reduction |
|---|---|---|---|---|---|---|---|---|---|---|
| | | matching session $\bar{s}^*$ | origin session $\tilde{s}^*$ | LTK[$s^*_{actor}$] | LTK[$s^*_{peer}$] | EphK[$s^*$] | EphK[$\bar{s}^*$] | EphK[$\tilde{s}^*$] | Intermediates | |
| CK | $\mathsf{CK}_1$ | $\exists$ | $\exists$ | | | $\times$ | $\times$ | | $k^*$ | PS |
| | $\mathsf{CK}_2$ | $\nexists$ | not sure | | $\times$ | $\times$ | | | $\mathsf{w}^*_i$ | PR-LEK |
| CK-PFS | $\mathsf{CK\text{-}PFS}_1$ | $\exists$ | $\exists$ | | | $\times$ | $\times$ | | $k^*$ | PS |
| | $\mathsf{CK\text{-}PFS}_2$ | $\nexists$ | $\exists$ | | $\times$ | $\times$ | | | $\mathsf{w}^*_i$ | PR-LEK |
| | $\mathsf{CK\text{-}PFS}_3$ | $\nexists$ | $\nexists$ | | $\checkmark_\tau$ | $\times$ | | | – | PE |
| CK$^+$ | $\mathsf{CK}^+_1$ | $\exists$ | $\exists$ | $\times$ | | | $\times$ | | $\mathsf{w}^*_j$ | PR-LEK |
| | $\mathsf{CK}^+_2$ | $\exists$ | $\exists$ | $\times$ | $\times$ | | | | $\mathsf{w}^*_i$ | PR-LX |
| | $\mathsf{CK}^+_3$ | $\exists$ | $\exists$ | | | $\times$ | $\times$ | | $k^*$ | PS |
| | $\mathsf{CK}^+_4$ | $\exists$ | $\exists$ | | $\times$ | $\times$ | | | $\mathsf{w}^*_i$ | PR-LEK |
| | $\mathsf{CK}^+_5$ | $\nexists$ | not sure | $\times$ | $\times$ | | | | $\mathsf{w}^*_i$ | PR-LX |
| | $\mathsf{CK}^+_6$ | $\nexists$ | not sure | | $\times$ | $\times$ | | | $\mathsf{w}^*_i$ | PR-LEK |
| CK$^+$-PFS | $\mathsf{CK^+\text{-}PFS}_1$ | not sure | $\exists$ | $\times$ | | | | $\times$ | $\mathsf{w}^*_j$ | PR-LEK |
| | $\mathsf{CK^+\text{-}PFS}_2$ | not sure | $\exists$ | $\times$ | $\times$ | | | | $\mathsf{w}^*_i$ | PR-LX |
| | $\mathsf{CK^+\text{-}PFS}_3$ | not sure | $\exists$ | | | $\times$ | | $\times$ | $k^*$ | PS |
| | $\mathsf{CK^+\text{-}PFS}_4$ | not sure | $\exists$ | | $\times$ | $\times$ | | | $\mathsf{w}^*_i$ | PR-LEK |
| | $\mathsf{CK^+\text{-}PFS}_5$ | $\nexists$ | $\nexists$ | | $\checkmark_\tau$ | | | | – | PE |
| eCK | $\mathsf{eCK}_1$ | $\exists$ | $\exists$ | $\times$ | | | $\times$ | | $\mathsf{w}^*_j$ | PR-LEK |
| | $\mathsf{eCK}_2$ | $\exists$ | $\exists$ | $\times$ | $\times$ | | | | $\mathsf{w}^*_i$ | PR-LX |
| | $\mathsf{eCK}_3$ | $\exists$ | $\exists$ | | | $\times$ | $\times$ | | $k^*$ | PS |
| | $\mathsf{eCK}_4$ | $\exists$ | $\exists$ | | $\times$ | $\times$ | | | $\mathsf{w}^*_i$ | PR-LEK |
| | $\mathsf{eCK}_5$ | $\nexists$ | not sure | $\times$ | $\times$ | | | | $\mathsf{w}^*_i$ | PR-LX |
| | $\mathsf{eCK}_6$ | $\nexists$ | not sure | | $\times$ | $\times$ | | | $\mathsf{w}^*_i$ | PR-LEK |
| eCK-PFS | $\mathsf{eCK\text{-}PFS}_1$ | not sure | $\exists$ | $\times$ | | | | $\times$ | $\mathsf{w}^*_j$ | PR-LEK |
| | $\mathsf{eCK\text{-}PFS}_2$ | not sure | $\exists$ | $\times$ | $\times$ | | | | $\mathsf{w}^*_i$ | PR-LX |
| | $\mathsf{eCK\text{-}PFS}_3$ | not sure | $\exists$ | | | $\times$ | | $\times$ | $k^*$ | PS |
| | $\mathsf{eCK\text{-}PFS}_4$ | not sure | $\exists$ | | $\times$ | $\times$ | | | $\mathsf{w}^*_i$ | PR-LEK |
| | $\mathsf{eCK\text{-}PFS}_5$ | $\nexists$ | $\nexists$ | | $\checkmark_\tau$ | | | | – | PE |

†† $\exists$ (resp., $\nexists$) denotes that the corresponding session does exists (resp., doesn't exist). $\times$ denotes that the corresponding LTK or EphK is always kept secret throughout the experiment.

*Proof.* For simplicity, let $s^*_{actor} = i$ and $s^*_{peer} = j$, thus $\mathrm{SS}[s^*] = (r^*_{i,1}, r^*_{i,2}, \mathsf{w}^*_i)$ and $\mathrm{SS}[\bar{s}^*] = (r^*_{j,1}, r^*_{j,2}, \mathsf{w}^*_j)$ if $\bar{s}^*$ exists. Recall the formulations of the CK, CK-PFS, CK$^+$, CK$^+$-PFS, eCK and eCK-PFS models in Table 3 to 8. The adversary is allowed to corrupt different key combinations in different models and different cases. We split the statement into several events, covering all the possible behaviors of the adversary. Once the underlying KE and KRF meet proper security, no matter under which event, at least one of the three key materials $(\mathsf{w}^*_i, \mathsf{w}^*_j, k^*)$ would never be exposed. That helps to further prove the target session key $sk^* = \mathrm{PRF}(\mathsf{w}^*_i, T) \oplus \mathrm{PRF}(\mathsf{w}^*_j, T) \oplus \mathrm{PRF}(k^*, T)$ is pseudorandom by assuming the underlying PRF is secure.

As summarized in Table 10, the modular construction is secure in different models if they meet the corresponding requirements. Under the event $\mathsf{CK}_1$, the randomness selected by both sides are kept secret throughout the experiment. If the underling $\mathsf{KE}$ is PS secure, $k^*$ is always kept secret from the adversary. Under the event $\mathsf{CK}_2$, even the matching session of the target does not exist, the message $s^*_{recv}$ still might be an replay-message generated in other session (i.e., its origin-session does exist), $\mathcal{A}$ may have performed SSReveal$(\cdot)$ query on it, thus $(k^*, \mathsf{w}^*_j)$ may have been exposed to $\mathcal{A}$. But SSReveal$(\cdot)$ query on $s^*$ is forbid, thus SS$[s^*]$ is always kept secret from the adversary. Moreover, if the underling $\mathsf{KRF}$ is PR-LEK, $\mathsf{w}^*_i$ would never been exposed. Under the event $\mathsf{CK\text{-}PFS}_3$, if the underling $\mathsf{KRF}$ is PE, $\mathcal{A}$ is unable to generate a valid message to make $s^*$ accept before corrupting LTK$[s^*_{peer}]$, thus $s^*$ would always terminate with abort. The analyses under other events are essentially similar. Due to page limitations, more details should be found in the full version. □

In particular, we can also sum up the high-level proof strategies shown in Table 10 to get a simplified version of it as shown in Table 11.

**Table 11.** Simplified proof strategies of our modular construction

| Case | The origin session $\tilde{s}^*$ | $\mathcal{A}$'s knowledge | | | | Unexposed Intermediates | Reduce to |
|---|---|---|---|---|---|---|---|
| | | LTK$[s^*_{actor}]$ | LTK$[s^*_{peer}]$ | EphK$[s^*]$ | EphK$[\tilde{s}^*]$ | | |
| I | $\exists$ | $\times$ | | | $\times$ | $\mathsf{w}^*_j$ | PR-LEK |
| II | | | | $\times$ | $\times$ | $k^*$ | PS |
| III | not sure | $\times$ | $\times$ | | | $\mathsf{w}^*_i$ | PR-LX |
| IV | | | $\times$ | $\times$ | | $\mathsf{w}^*_i$ | PR-LEK |
| V | $\nexists$ | | $\checkmark_\tau$ | | | $-$ | PE |

## 4.3 Two Enhanced Versions of Our Modular Construction

In this section, we present two enhanced versions of our modular construction to reduce the randomness used, thus to decrease the communication and computation overheads to some extent. In particular, the same randomness will be used for both the $\mathsf{KRF}$ and $\mathsf{KE}$ modules.

| $\hat{P}_i$ $(\mathsf{ek}_i, \mathsf{rk}_i)$ | | $\hat{P}_j$ $(\mathsf{ek}_j, \mathsf{rk}_j)$ |
|---|---|---|
| $\mathsf{x}_{i,2} \leftarrow\!\$\; \mathcal{X}_2$ | $\xrightarrow{\mathsf{y}_i}$ | $\mathsf{x}_{j,2} \leftarrow\!\$\; \mathcal{X}_2$ |
| $(\mathsf{y}_i, \mathsf{w}_i) \leftarrow f_{\mathsf{ek}_i}(\mathsf{rpk}_j, -, \mathsf{x}_{i,2})$ | $\xleftarrow{\mathsf{y}_j}$ | $(\mathsf{y}_j, \mathsf{w}_j) \leftarrow f_{\mathsf{ek}_j}(\mathsf{rpk}_i, -, \mathsf{x}_{j,2})$ |
| $\mathsf{w}'_j \leftarrow \bar{f}_{\mathsf{rk}_i}(\mathsf{epk}_j, -, \mathsf{y}_j)$ | | $\mathsf{w}'_i \leftarrow \bar{f}_{\mathsf{rk}_j}(\mathsf{epk}_i, -, \mathsf{y}_i)$ |
| Abort if $\mathsf{w}'_j = \perp$ | | Abort if $\mathsf{w}'_i = \perp$ |
| $k \leftarrow \mathrm{CKey}(\mathsf{y}_i, \mathsf{y}_j)$ | | $k' \leftarrow \mathrm{CKey}(\mathsf{y}_j, \mathsf{y}_i)$ |
| Let $T \leftarrow \mathsf{epk}_i \| \mathsf{rpk}_i \| \mathsf{epk}_j \| \mathsf{rpk}_j \| \mathsf{y}_i \| \mathsf{y}_j$ | | |
| $\hat{P}_i$ compute: $SK_i \leftarrow \mathrm{PRF}(\mathsf{w}_i, T) \oplus \mathrm{PRF}(\mathsf{w}'_j, T) \oplus \mathrm{PRF}(k, T)$ | | |
| $\hat{P}_j$ compute: $SK_j \leftarrow \mathrm{PRF}(\mathsf{w}'_i, T) \oplus \mathrm{PRF}(\mathsf{w}_j, T) \oplus \mathrm{PRF}(k', T)$ | | |

**Fig. 7.** Our first enhanced modular construction.

**The first enhanced construction.** As shown in Fig. 7, the $\mathsf{KRF}$ output $\mathsf{y}$ is used as a $\mathsf{KE}$ public key $pk$, and its specific input $\mathsf{x}_1$ is set as an empty string.

**Theorem 2.** *Theorem 1 holds for the modular construction shown in Fig. 7 if Simulatability holds for the underlying $\mathsf{KRF}$ and $\mathsf{KE}$, i.e., for any $\mathsf{pp} \leftarrow \mathsf{KRF.Setup}(1^\lambda)$, $(\mathsf{epk}, \mathsf{ek}) \leftarrow \mathsf{KRF.KG}(\mathsf{pp}, 0)$, $(\mathsf{rpk}, \mathsf{rk}) \leftarrow \mathsf{KRF.KG}(\mathsf{pp}, 1)$ and $\mathsf{x}_1 \in \mathcal{X}_1$, there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that*

*i for any PPT algorithm $\mathcal{D}$, the following equality holds:*

$$\Pr[\mathsf{x}_2 \leftarrow_\$ \mathcal{X}_2, (\mathsf{y}, \mathsf{w}) \leftarrow f_{\mathsf{ek}}(\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2), pk \leftarrow \mathcal{S}_1(\mathsf{epk}, \mathsf{rpk}, \mathsf{x}_1, \mathsf{y}) : \mathcal{D}(pk) = 1] \\ = \Pr[(pk, sk) \leftarrow \mathsf{KE.Gen}(1^\lambda) : \mathcal{D}(pk) = 1];$$

*ii for any PPT algorithm $\hat{\mathcal{D}}$, the following equality holds:*

$$\Pr[(pk, sk) \leftarrow \mathsf{KE.Gen}(1^\lambda), (\mathsf{y}, \mathsf{w}) \leftarrow \mathcal{S}_2(pk, \mathsf{epk}, \mathsf{ek}, \mathsf{rpk}, \mathsf{rk}, \mathsf{x}_1) : \hat{\mathcal{D}}(\mathsf{y}, \mathsf{w}) = 1] \\ = \Pr[\mathsf{x}_2 \leftarrow_\$ \mathcal{X}_2, (\mathsf{y}, \mathsf{w}) \leftarrow f_{\mathsf{ek}}(\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2) : \hat{\mathcal{D}}(\mathsf{y}, \mathsf{w}) = 1].$$

*Proof.* During the security proof, no matter in which case shown in Table 11, the adversary's view should be perfectly simulated. For those sessions that are non-origin sessions of the target session, executes honestly according to the protocol description; as for the target session and its origin session, embed different challenges according different reduction strategies as follows:

1. for Case-I that LTK$[s^*_{actor}]$ and EphK$[\tilde{s}^*]$ are kept secret, set LTK$[s^*_{actor}]$ as $\mathsf{rpk}^*$ and $\mathsf{y}^*_j$ as the PR-LEK challenge $\mathsf{y}^*$.
2. for Case-II that EphK$[s^*]$ and EphK$[\tilde{s}^*]$ are kept secret, perform PExecute$(\cdot)$ query to get a PS challenge $(pk^*_i, pk^*_j)$.
3. for Case-III that LTK$[s^*_{actor}]$ and LTK$[s^*_{peer}]$ are kept secret, set LTK$[s^*_{actor}]$ as $\mathsf{epk}^*$, LTK$[s^*_{peer}]$ as $\mathsf{rpk}^*$, and $\mathsf{y}^*_i$ as the PR-LX challenge $\mathsf{y}^*$.
4. for Case-IV that LTK$[s^*_{peer}]$ and EphK$[s^*]$ are kept secret, set LTK$[s^*_{peer}]$ as $\mathsf{rpk}^*$ and $\mathsf{y}^*_i$ as the PR-LEK challenge $\mathsf{y}^*$.
5. for Case-V that LTK$[s^*_{peer}]$ are not corrupted before the target session completes, set LTK$[s^*_{peer}]$ as $\mathsf{epk}^*$, once the adversary is able to make the target session accepts, out put its message as a solution of PE experiment.

For the modular construction shown in Fig. 6 that uses independent randomness, the two parts of $s^*_{sent} = (pk^*_i, \mathsf{y}^*_i)$ or $s^*_{recv} = (pk^*_j, \mathsf{y}^*_j)$ can be simulated separately. In particular, one part is set with the corresponding challenge, while the another part is generated honestly. But for the enhanced modular construction shown in Fig 7 that uses the same randomness for both the $\mathsf{KRF}$ and $\mathsf{KE}$ modules, above simulation strategies 1–4 cannot work any more. Technically, if *Simulatability* is satisfied, we only need to make some minor changes to keep the original reduction strategies work: for Case-I, invoke $\mathcal{S}_1$ to get $pk^*_j$; for Case-II, invoke $\mathcal{S}_2$ two times to get corresponding $(\mathsf{y}^*_i, \mathsf{w}^*_i)$ and $(\mathsf{y}^*_j, \mathsf{w}^*_j)$; for Case-III, use the exposed randomness in the PR-LX security experiment to compute $pk^*_i$ directly; for Case-IV, invoke $\mathcal{S}_1$ to get $pk^*_i$. □

**The second enhanced construction.** We first introduce the notion of KE-simulatable KRF, whose security experiments are defined as in Fig. 8. Simulatability is inherently required: a KE public key $pk$ can be directly used as a KRF output $\mathsf{y}$, and vice versa. In addition, the computation of $\mathsf{w}$ is allowed to be delayed until some $\mathsf{x}_1$ is specified.

```
$\mathsf{Exp}^{\mathsf{SPR\text{-}LEK}}_{\mathsf{KRF},\mathcal{A}}(\lambda)$ :
    $\mathsf{Initialization}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{KE.Gen}(1^\lambda)$
    $\mathsf{x_2}^* \leftarrow sk$, $\mathsf{y}^* \leftarrow pk$
    $\mathsf{x}_1^* \leftarrow \mathcal{A}_1^{\bar{\mathcal{O}}(\cdot)}(\mathsf{info}, \mathsf{ek}^*, \mathsf{y}^*)$
    $\mathsf{w}_0^* \leftarrow \mathsf{KRF.Eval}(0, \mathsf{ek}^*, \mathsf{rpk}^*, \mathsf{x}_1^*, \mathsf{x_2}^*)$
    $\mathsf{w}_1^* \leftarrow\$ \mathcal{W}$, $b \leftarrow\$ \{0,1\}$
    $b' \leftarrow \mathcal{A}_2^{\bar{\mathcal{O}}(\cdot)}(\mathsf{info}, \mathsf{ek}^*, \mathsf{x}_1^*, \mathsf{y}^*, \mathsf{w}_b^*)$
    if $b' = b$, return 1; else return 0
```

```
$\mathsf{Exp}^{\mathsf{SPR\text{-}LX}}_{\mathsf{KRF},\mathcal{A}}(\lambda)$ :
    $\mathsf{Initialization}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{KE.Gen}(1^\lambda)$
    $\mathsf{x_2}^* \leftarrow sk$, $\mathsf{y}^* \leftarrow pk$
    $\mathsf{x}_1^* \leftarrow \mathcal{A}_1^{\mathcal{O}(\cdot),\bar{\mathcal{O}}(\cdot)}(\mathsf{info}, \mathsf{y}^*)$
    $\mathsf{w}_0^* \leftarrow \mathsf{KRF.Eval}(0, \mathsf{ek}^*, \mathsf{rpk}^*, \mathsf{x}_1^*, \mathsf{x}_2^*)$
    $\mathsf{w}_1^* \leftarrow\$ \mathcal{W}$, $b \leftarrow\$ \{0,1\}$
    $b' \leftarrow \mathcal{A}_2^{\mathcal{O}(\cdot),\bar{\mathcal{O}}(\cdot)}(\mathsf{info}, \mathsf{x}_1^*, \mathsf{x}_2^*, \mathsf{y}^*, \mathsf{w}_b^*)$
    if $b' = b$, return 1; else return 0
```

**Fig. 8.** The SPR-LEK and SPR-LX experiments of KE-simulatable KRF

**Definition 5 (KE-simulatable KRF).** *Given a KE scheme $\mathsf{KE} = (\mathsf{KE.Gen}, \mathsf{KE.Key})$, a scheme KRF=(KRF.Setup, KRF.KG, KRF.Eval) is called KE-simulatable KRF with SPR-LEK or SPR-LX security, if for any PPT stateful adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, its advantage*

$$\mathsf{Adv}^{\mathsf{SPR\text{-}LEK/SPR\text{-}LX}}_{\mathsf{KRF},\mathcal{A}}(\lambda) \overset{def}{=} \Pr[\mathsf{Exp}^{\mathsf{SPR\text{-}LEK/SPR\text{-}LX}}_{\mathsf{KRF},\mathcal{A}}(\lambda) = 1] \leq \mu(\lambda).$$

If taking a KE-simulatable KRF as the building block, our modular construction in Fig. 6 can be enhanced as in Fig. 9.

| $\hat{P}_i$ $(\mathsf{ek}_i, \mathsf{rk}_i)$ | | $\hat{P}_j$ $(\mathsf{ek}_j, \mathsf{rk}_j)$ |
|---|---|---|
| $(pk_i, sk_i) \leftarrow \mathsf{KE.Gen}(1^\lambda)$ | $\xrightarrow{pk_i}$ $\xleftarrow{pk_j}$ | $(pk_j, sk_j) \leftarrow \mathsf{KE.Gen}(1^\lambda)$ |
| $\mathsf{x}_{i,1} \leftarrow \mathsf{H}_0(pk_j)$, $\mathsf{x}_{j,1} \leftarrow \mathsf{H}_0(pk_i)$ | | $\mathsf{x}_{i,1} \leftarrow \mathsf{H}_0(pk_j)$, $\mathsf{x}_{j,1} \leftarrow \mathsf{H}_0(pk_i)$ |
| $\mathsf{w}_i \leftarrow f_{\mathsf{ek}_i}(\mathsf{rpk}_j, \mathsf{x}_{i,1}, sk_i)$ | | $\mathsf{w}_j \leftarrow f_{\mathsf{ek}_j}(\mathsf{rpk}_i, \mathsf{x}_{j,1}, sk_j)$ |
| $\mathsf{w}'_j \leftarrow \bar{f}_{\mathsf{rk}_i}(\mathsf{epk}_j, \mathsf{x}_{j,1}, pk_j)$ | | $\mathsf{w}'_i \leftarrow \bar{f}_{\mathsf{rk}_j}(\mathsf{epk}_i, \mathsf{x}_{i,1}, pk_i)$ |
| $k \leftarrow \mathsf{KE.Key}(sk_i, pk_j)$ | | $k' \leftarrow \mathsf{KE.Key}(sk_j, pk_i)$ |
| Let $T \leftarrow \mathsf{epk}_i \Vert \mathsf{rpk}_i \Vert \mathsf{epk}_j \Vert \mathsf{rpk}_j \Vert pk_i \Vert pk_j$ | | |
| $\hat{P}_i$ compute: $SK_i \leftarrow \mathrm{PRF}(\mathsf{w}_i, T) \oplus \mathrm{PRF}(\mathsf{w}'_j, T) \oplus \mathrm{PRF}(k, T)$ | | |
| $\hat{P}_j$ compute: $SK_j \leftarrow \mathrm{PRF}(\mathsf{w}'_i, T) \oplus \mathrm{PRF}(\mathsf{w}_j, T) \oplus \mathrm{PRF}(k', T)$ | | |

**Fig. 9.** Our second enhanced modular construction

**Theorem 3.** *The second enhanced modular construction shown in Fig. 9 instantiated by PS KE and different KE-simulatable KRFs yields different ORKEs in different models as in Table 12.*

*Proof.* Note that CK-PFS, CK$^+$ and eCK-PFS models are not considered here. We can prove this enhanced modular construction's security using the simplified proof strategies as shown in Table 13.

1. for Case-I that LTK$[s^*_{actor}]$ and EphK$[\tilde{s}^*]$ are kept secret, set LTK$[s^*_{actor}]$ as $\mathsf{rpk}^*$ and $pk^*_j$ as the SPR-LEK challenge $\mathsf{y}^*$.

**Table 12.** The main results of our second enhanced modular construction

| Instantiations | Models | Requirements for the KE | Requirements for the KRF | |
|---|---|---|---|---|
| | | | SPR-LEK | SPR-LX |
| ORKEs | CK | PS | ○ | |
| | CK$^+$ | | ○ | ○ |
| | eCK | | ○ | ○ |

**Table 13.** The simplified proof strategies of our second enhanced modular construction

| Case | matching session $\bar{s}^*$ | $\mathcal{A}$'s knowledge | | | | Unexposed Intermediates | Reduce to |
|---|---|---|---|---|---|---|---|
| | | LTK$[s^*_{actor}]$ | LTK$[s^*_{peer}]$ | EphK$[s^*]$ | EphK$[\bar{s}^*]$ | | |
| I | $\exists$ | × | | | × | $\mathsf{w}^*_j$ | SPR-LEK |
| II | | | | × | × | $k^*$ | PS |
| III | $\nexists$ | × | × | | | $\mathsf{w}^*_i$ | SPR-LX |
| IV | | | × | × | | $\mathsf{w}^*_i$ | SPR-LEK |

2. for Case-II EphK$[s^*]$ and EphK$[\tilde{s}^*]$ are kept secret, perform PExecute$(\cdot)$ query to get two public keys $pk^*_i$ and $pk^*_j$, and compute $\mathsf{w}^*_i$ and $\mathsf{w}^*_j$ using LTK$[s^*_{peer}]$ and LTK$[s^*_{actor}]$, respectively.
3. for Case-III that LTK$[s^*_{actor}]$ and LTK$[s^*_{peer}]$ are kept secret, set LTK$[s^*_{actor}]$ as $\mathsf{epk}^*$, LTK$[s^*_{peer}]$ as $\mathsf{rpk}^*$, and $pk^*_i$ as the SPR-LX challenge $\mathsf{y}^*$.
4. for Case-IV that LTK$[s^*_{peer}]$ and EphK$[s^*]$ are kept secret, set LTK$[s^*_{peer}]$ as $\mathsf{rpk}^*$ and $pk^*_i$ as the SPR-LEK challenge $\mathsf{y}^*$. □

## 5 Unification of Previous Constructions

Here, we show that several well-known constructions can be viewed as special cases in our (enhanced) modular construction, including 2KEM+DH [4] (Sec. 5.1), HMQV [18] (Sec. 5.2), NAXOS [19] (Sec. 5.3) and BJS [3] (Sec. 5.4).

### 5.1 2KEM+DH

2KEM+DH was proved secure in the CK model. In 2KEM+DH, the KRF is initiated by $\mathsf{KRF}_{\mathsf{2KEM+DH}}$ in Fig. 10. Let $\mathsf{KEM}$=($\mathsf{KEM.Gen}$, $\mathsf{KEM.Enc}$, $\mathsf{KEM.Dec}$) be a KEM with randomness space $\mathcal{R}$. Here $\varpi$ denotes a fixed public string.

```
$\mathsf{KRF}_{\mathsf{2KEM+DH}}.\mathsf{KG}(\mathsf{pp}, \psi)$:
    if $\psi = 0$
        return $(\mathsf{epk}, \mathsf{ek}) \overset{\text{def}}{=} (\varpi, \varpi)$
    else if $\psi = 1$
        $(ek, dk) \leftarrow \mathsf{KEM.Gen}(1^\lambda)$
        return $(\mathsf{rpk}, \mathsf{rk}) \overset{\text{def}}{=} (ek, dk)$

$\mathsf{KRF}_{\mathsf{2KEM+DH}}.\mathsf{Eval}(\psi, \mathsf{input})$:
    if $\psi = 0$
        $(\varpi, \mathsf{rpk}, -, \mathsf{x}_2) \leftarrow \mathsf{input}$
        $(c, k) \leftarrow \mathsf{KEM.Enc}(\mathsf{rpk}; \mathsf{x}_2)$
        return $(\mathsf{y}, \mathsf{w}) \overset{\text{def}}{=} (c, k)$
    else if $\psi = 1$
        $(\mathsf{rk}, \varpi, -, \mathsf{y}) \leftarrow \mathsf{input}$
        $k \leftarrow \mathsf{KEM.Dec}(\mathsf{rk}, \mathsf{y})$
        return $\mathsf{w} \overset{\text{def}}{=} k$
```

**Fig. 10.** The $\mathsf{KRF}_{\mathsf{2KEM+DH}}$ implied by 2KEM+DH [4]

**Theorem 4.** *If KEM is IND-CCA, $\mathsf{KRF}_{\mathsf{2KEM+DH}}$ shown in Fig. 10 is PR-LEK.*

| $\hat{P}_i(ek_i, dk_i)$ | | $\hat{P}_j(ek_j, dk_j)$ |
|---|---|---|
| $x \leftarrow_\$ \mathbb{Z}_p, X \leftarrow g^x$ <br> $(c_i, k_i) \leftarrow \mathsf{KEM.Enc}(ek_j)$ | $\xrightarrow{c_i, X}$ <br> $\xleftarrow{c_j, Y}$ | $y \leftarrow_\$ \mathbb{Z}_p, Y \leftarrow g^y$ <br> $(c_j, k_j) \leftarrow \mathsf{KEM.Enc}(ek_i)$ |
| $k'_j \leftarrow \mathsf{KEM.Dec}(dk_i, c_j)$ | | $k'_i \leftarrow \mathsf{KEM.Dec}(dk_j, c_i)$ |
| Let $T \leftarrow ek_i \| ek_j \| c_i \| X \| c_j \| Y$ <br> $\hat{P}_i$ compute $sk_i \leftarrow \mathrm{PRF}(k_i, T) \oplus \mathrm{PRF}(k'_j, T) \oplus \mathrm{PRF}(Y^x, T)$ <br> $\hat{P}_j$ compute $sk_j \leftarrow \mathrm{PRF}(k'_i, T) \oplus \mathrm{PRF}(k_j, T) \oplus \mathrm{PRF}(X^y, T)$ | | |

**Fig. 11.** $\mathcal{P}_{\text{2KEM+DH}}$: apply $\mathsf{KRF}_{\mathsf{2KEM+DH}}$ and DHKE into our modular construction. Let $\mathbb{G}$ be a group of prime order $p$ with a generator $g$.

*Proof.* It is quite easy to prove that Theorem 4 holds. Since the PR-LEK challenge is in fact an IND-CCA challenge, and the $\bar{\mathcal{O}}(\cdot)$ oracle can be perfectly simulated using the underlying decryption oracle. Once the adversary is able to win in the experiment with non-negligible advantage, the IND-CCA security is also broken. Due to page limitations, we drop the details here. $\square$

## 5.2 HMQV

HMQV was proved secure in the CK$^+$ model. In HMQV, the KRF is initiated by $\mathsf{KRF}_{\mathsf{HMQV}}$ in Fig. 12. Let $\mathbb{G}$ be a group of prime order $p$ with $g$ as a generator, $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$ and $\bar{\mathsf{H}} : \mathbb{Z}_p \to \mathbb{Z}_p$ be two hash functions.

```
KRF_HMQV.KG(pp, $\psi$):
    if $\psi = 0$
        $a \leftarrow_\$ \mathbb{Z}_p,\ A \leftarrow g^a$
        return $(\mathsf{epk}, \mathsf{ek}) \overset{\text{def}}{=} (A, a)$
    else if $\psi = 1$
        $b \leftarrow_\$ \mathbb{Z}_p,\ B \leftarrow g^b$
        return $(\mathsf{rpk}, \mathsf{rk}) \overset{\text{def}}{=} (B, b)$
```

```
KRF_HMQV.Eval($\psi$, input):
    if $\psi = 0$
        $(\mathsf{ek}, \mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2) \leftarrow \mathsf{input}$
        $\mathsf{y} \leftarrow g^{\mathsf{x}_2},\ d \leftarrow \mathsf{H}(\mathsf{y}, \mathsf{rpk})$
        $e \leftarrow \mathsf{H}(\mathsf{x}_1, \mathsf{epk}),\ \mathsf{w} \leftarrow \bar{\mathsf{H}}(\mathsf{rpk}^{e(d \cdot \mathsf{ek} + \mathsf{x}_2)})$
        return $(\mathsf{y}, \mathsf{w})$
    else if $\psi = 1$
        $(\mathsf{rk}, \mathsf{epk}, \mathsf{x}_1, \mathsf{y}) \leftarrow \mathsf{input}$:
        $d \leftarrow \mathsf{H}(\mathsf{y}, \mathsf{rpk}),\ e \leftarrow \mathsf{H}(\mathsf{x}_1, \mathsf{epk})$
        return $\mathsf{w} \overset{\text{def}}{=} \bar{\mathsf{H}}((\mathsf{epk}^d \mathsf{y})^{e \cdot \mathsf{rk}})$
```

**Fig. 12.** The $\mathsf{KRF}_{\mathsf{HMQV}}$ implied by HMQV [18]

| $\hat{P}_i(A = g^a, a)$ | | $\hat{P}_j \quad (B = g^b, b)$ |
|---|---|---|
| $x \leftarrow_\$ \mathbb{Z}_p, X \leftarrow g^x$ | $\xleftarrow{X}$ <br> $\xrightarrow{Y}$ | $y \leftarrow_\$ \mathbb{Z}_p, Y \leftarrow g^y$ |
| $d \leftarrow \mathsf{H}(X, B), e \leftarrow \mathsf{H}(Y, A)$ <br> $sk_i \leftarrow \mathsf{H}_1((YB^e)^{da+x}, A, B, X, Y)$ | | $d \leftarrow \mathsf{H}(X, B), e \leftarrow \mathsf{H}(Y, A)$ <br> $sk_j \leftarrow \mathsf{H}_1((XA^d)^{eb+y}, A, B, X, Y)$ |

**Fig. 13.** $\mathcal{P}_{\text{HMQV}}$: apply $\mathsf{KRF}_{\mathsf{HMQV}}$ into our second enhanced modular construction. The required PRF is replaced by a RO $\mathrm{H}_1$, which covers the internal $\bar{\mathrm{H}}$.

**Theorem 5.** *If the GDH problem holds in $\mathbb{G}$, $\mathsf{H}$ and $\bar{\mathsf{H}}$ are modeled as random oracles, $\mathsf{KRF}_{\mathsf{HMQV}}$ shown in Fig. 12 is both SPR-LEK and SPR-LX.*

*Proof.* First we can see that $\mathsf{KRF}_{\mathsf{HMQV}}$ and DHKE meet *Simulatability*, and the corresponding simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ can be constructed as follows:

```
$\mathcal{S}_1(\mathsf{epk}, \mathsf{rpk}, \mathsf{x}_1, \mathsf{y})$:
    $pk \leftarrow \mathsf{y}$
  return $pk$
```

```
$\mathcal{S}_2(pk, \mathsf{epk}, \mathsf{ek}, \mathsf{rpk}, \mathsf{rk}, \mathsf{x}_1)$:
    $\mathsf{y} \leftarrow pk,\ d \leftarrow \mathsf{H}(\mathsf{y}, \mathsf{rpk}),\ e \leftarrow \mathsf{H}(\mathsf{x}_1, \mathsf{epk})$
    $\mathsf{w} \leftarrow \bar{\mathsf{H}}((\mathsf{epk}^d \mathsf{y})^{e \cdot \mathsf{rk}})$
  return $(\mathsf{y}, \mathsf{w})$
```

For any PPT adversary $\mathcal{A}_1$ against $\mathsf{Exp}^{\mathsf{SPR\text{-}LEK}}_{\mathsf{KRF},\mathcal{A}_1}$, we build an algorithm that simulates this experiment with these changes:

1. given a GDH challenge $(X, Y)$, set $\mathsf{rpk}^* \leftarrow X$, $\mathsf{y}^* \leftarrow Y$ and $\mathsf{w}_0^* \leftarrow_\$ \mathcal{W}$;
2. initialize two empty lists $\mathcal{L}_{\bar{\mathsf{H}}}$ and $\mathcal{L}_{\bar{f}}$;
3. for a $\bar{\mathsf{H}}(\mathsf{input})$ query:
   (a) if $\exists\ (\mathsf{input}, h) \in \mathcal{L}_{\bar{\mathsf{H}}}$, return $h$;
   (b) else if $\exists\ ((\mathsf{epk}, \mathsf{x}_1, \mathsf{y}), \mathsf{w}) \in \mathcal{L}_{\bar{f}}$ s.t. $\mathrm{CDH}(X, (\mathsf{y} \cdot \mathsf{epk}^d)^e) = \mathsf{input}$, where $d \leftarrow \mathsf{H}(\mathsf{y}, X)$ and $e \leftarrow \mathsf{H}(\mathsf{x}_1, \mathsf{epk})$, return $\mathsf{w}$ and record $(\mathsf{input}, \mathsf{w})$ into $\mathcal{L}_{\bar{\mathsf{H}}}$;
   (c) otherwise, return $h \leftarrow_\$ \mathbb{Z}_p$ and record $(\mathsf{input}, \mathsf{h})$ into $\mathcal{L}_{\bar{\mathsf{H}}}$.
4. for an $\bar{\mathcal{O}}(\mathsf{epk}, \mathsf{x}_1, \mathsf{y})$ query:
   (a) if $\exists\ ((\mathsf{epk}, \mathsf{x}_1, \mathsf{y}), \mathsf{w}) \in \mathcal{L}_{\bar{f}}$, return $\mathsf{w}$.
   (b) else if $\exists\ (V, h) \in \mathcal{L}_{\bar{\mathsf{H}}}$ s.t. $\mathrm{CDH}(X, (\mathsf{y} \cdot \mathsf{epk}^d)^e) = V$, where $d \leftarrow \mathsf{H}(\mathsf{y}, X)$ and $e \leftarrow \mathsf{H}(\mathsf{x}_1, \mathsf{epk})$, return $h$ and record $((\mathsf{epk}, \mathsf{x}_1, \mathsf{y}), h)$ into $\mathcal{L}_{\bar{f}}$.
   (c) otherwise, return $\mathsf{w} \leftarrow_\$ \mathbb{Z}_p$ and record $((\mathsf{epk}, \mathsf{x}_1, \mathsf{y}), \mathsf{w})$ into $\mathcal{L}_{\bar{f}}$.
5. if $\mathcal{A}_1$ has never queried on the correct value in a $\bar{\mathrm{H}}$ query, it cannot win in the experiment. Such that there must exist a tuple $(J, \mathsf{w}) \in \mathcal{L}_{\bar{\mathsf{H}}}$ and the value $J^{1/e^*}/X^{d^* \cdot \mathsf{ek}^*}$ is a solution of the GDH problem instance, where $d^* \leftarrow \mathsf{H}(Y, X)$ and $e^* \leftarrow \mathsf{H}(\mathsf{x_1}^*, \mathsf{epk}^*)$.

Similarly, for any PPT adversary $\mathcal{A}_2$ against $\mathsf{Exp}^{\mathsf{SPR\text{-}LX}}_{\mathsf{KRF},\mathcal{A}_2}$, we build an algorithm that simulates this experiment with these changes:

1. given a GDH challenge $(X, Y)$, set $(\mathsf{epk}^*, \mathsf{rpk}^*) \leftarrow (X, Y)$ and $\mathsf{w_0}^* \leftarrow_\$ \mathcal{W}$;
2. initialize three empty lists $\mathcal{L}_{\bar{\mathsf{H}}}$, $\mathcal{L}_f$ and $\mathcal{L}_{\bar{f}}$;
3. for a $\bar{\mathsf{H}}(\mathsf{input})$ query:
   (a) if $\exists\ (\mathsf{input}, h) \in \mathcal{L}_{\bar{\mathsf{H}}}$, return $h$;
   (b) else if $\exists\ ((\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2), \mathsf{w}) \in \mathcal{L}_f$ s.t. $\mathrm{CDH}(X, \mathsf{rpk}^{e \cdot d}) = \mathsf{input}/\mathsf{rpk}^{e \cdot \mathsf{x}_2}$, where $d \leftarrow \mathsf{H}(\mathsf{y}, \mathsf{rpk})$ and $e \leftarrow \mathsf{H}(\mathsf{x}_1, X)$, return $\mathsf{w}$ and record $(\mathsf{input}, \mathsf{w})$ into $\mathcal{L}_{\bar{\mathsf{H}}}$;
   (c) else if $\exists\ ((\mathsf{epk}, \mathsf{x}_1, \mathsf{y}), \mathsf{w}) \in \mathcal{L}_{\bar{f}}$ s.t. $\mathrm{CDH}(Y, (\mathsf{y} \cdot \mathsf{epk}^d)^e) = \mathsf{input}$, where $d \leftarrow \mathsf{H}(\mathsf{y}, Y)$ and $e \leftarrow \mathsf{H}(\mathsf{x}_1, \mathsf{epk})$, return $\mathsf{w}$ and record $(\mathsf{input}, \mathsf{w})$ into $\mathcal{L}_{\bar{\mathsf{H}}}$;
   (d) otherwise, return $h \leftarrow_\$ \mathbb{Z}_p$ and record $(\mathsf{input}, h)$ into $\mathcal{L}_{\bar{\mathsf{H}}}$.
4. for an $\mathcal{O}(\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2)$ query, compute $\mathsf{y} \leftarrow g^{\mathsf{x}_2}$:
   (a) if $\exists\ ((\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2), \mathsf{w}) \in \mathcal{L}_{\mathsf{F}}$, return $(\mathsf{y}, h)$;
   (b) else if $\exists\ (V, h) \in \mathcal{L}_{\bar{\mathsf{H}}}$ s.t. $\mathrm{CDH}(X, \mathsf{rpk}^{e \cdot d}) = V/\mathsf{rpk}^{e \cdot \mathsf{x}_2}$, where $d \leftarrow \mathsf{H}(\mathsf{y}, \mathsf{rpk})$ and $e \leftarrow \mathsf{H}(\mathsf{x}_1, X)$, return $(\mathsf{y}, h)$ and record $((\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2), h)$ into $\mathcal{L}_f$;
   (c) otherwise, return $(\mathsf{y}, \mathsf{w} \leftarrow_\$ \mathbb{Z}_p)$ and record $((\mathsf{rpk}, \mathsf{x}_1, \mathsf{x}_2), \mathsf{w})$ into $\mathcal{L}_f$.
5. for an $\bar{\mathcal{O}}(\mathsf{epk}, \mathsf{x}_1, \mathsf{y})$ query:
   (a) if $\exists\ ((\mathsf{epk}, \mathsf{x}_1, \mathsf{y}), \mathsf{w}) \in \mathcal{L}_{\bar{f}}$, return $\mathsf{w}$.
   (b) else if $\exists\ (U, h) \in \mathcal{L}_{\bar{\mathsf{H}}}$ s.t. $\mathrm{CDH}(Y, (\mathsf{y} \cdot \mathsf{epk}^d)^e) = U$, where $d \leftarrow \mathsf{H}(\mathsf{y}, Y)$ and $e \leftarrow \mathsf{H}(\mathsf{x}_1, \mathsf{epk})$, return $h$ and record $((\mathsf{epk}, \mathsf{x}_1, \mathsf{y}), h)$ into $\mathcal{L}_{\bar{f}}$.
   (c) otherwise, return $\mathsf{w} \leftarrow_\$ \mathbb{Z}_p$ and record $((\mathsf{epk}, \mathsf{x}_1, \mathsf{y}), \mathsf{w})$ into $\mathcal{L}_{\bar{f}}$.

6. Similarly, if $\mathcal{A}_2$ is able to win in the experiment, there must exist a tuple $(J, \mathsf{w}) \in \mathcal{L}_{\bar{\mathsf{H}}}$ and the value $(J^{1/e^*}/Y^{\mathsf{x}_2^*})^{1/d^*}$ is a solution of the GDH problem instance, where $d^* \leftarrow \mathsf{H}(\mathsf{y}^*, Y)$ and $e^* \leftarrow \mathsf{H}(\mathsf{x_1}^*, X)$. □

Note that to agree on a session key, the following equation should hold, where computing $B^{e(da+x)}$ and $A^{d(eb+y)}$ can be viewed as invoking $\mathsf{KRF}_{\mathsf{HMQV}}.\mathsf{Eval}(0, (a, B, Y, x))$ and $\mathsf{KRF}_{\mathsf{HMQV}}.\mathsf{Eval}(0, (b, A, X, y))$, respectively. But the common part $A^{edb} = B^{eda}$ is computed for only once.

$$\begin{aligned}(YB^e)^{da+x} &= Y^x \cdot Y^{da} \cdot \boxed{B^{e(da+x)}} \\ &= X^y \cdot A^{dy} \cdot A^{edb} \cdot X^{eb} \\ &= X^y \cdot \boxed{A^{d(eb+y)}} \cdot X^{eb} \\ &= (XA^d)^{eb+y}\end{aligned}$$

## 5.3 NAXOS

NAXOS was proved secure in the eCK model. In NAXOS, the KRF is initiated by $\mathsf{KRF}_{\mathsf{NAXOS}}$ in Fig. 14. Let $\mathbb{G}$ be a group of prime order $p$ with $g$ as a generator, $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$ and $\bar{\mathsf{H}} : \mathbb{Z}_p \to \{0,1\}^\lambda$ be two hash functions.

```
$\mathsf{KRF}_{\mathsf{NAXOS}}.\mathsf{KG}(\mathsf{pp}, \psi)$:
    if $\psi = 0$
        $a \leftarrow\$ \mathbb{Z}_p,\ A \leftarrow g^a$
        return $(\mathsf{epk}, \mathsf{ek}) \overset{\text{def}}{=} (A, a)$
    else if $\psi = 1$
        $b \leftarrow\$ \mathbb{Z}_p,\ B \leftarrow g^b$
        return $(\mathsf{rpk}, \mathsf{rk}) \overset{\text{def}}{=} (B, b)$
```

```
$\mathsf{KRF}_{\mathsf{NAXOS}}.\mathsf{Eval}(\psi, \mathsf{input})$:
    if $\psi = 0$
        $(\mathsf{ek}, \mathsf{rpk}, -, \mathsf{x}_2) \leftarrow \mathsf{input}$
        $\mathsf{y} \leftarrow g^{\mathsf{H}(\mathsf{ek}, \mathsf{x}_2)}$
        $\mathsf{w} \leftarrow \bar{\mathsf{H}}(\mathsf{epk}, \mathsf{rpk}^{\mathsf{H}(\mathsf{ek}, \mathsf{x}_2)})$
        return $(\mathsf{y}, \mathsf{w})$
    else if $\psi = 1$
        $(\mathsf{rk}, \mathsf{epk}, -, \mathsf{y}) \leftarrow \mathsf{input}$
        $\mathsf{w} \leftarrow \bar{\mathsf{H}}(\mathsf{epk}, \mathsf{y}^{\mathsf{rk}})$
        return $\mathsf{w}$
```

**Fig. 14.** The $\mathsf{KRF}_{\mathsf{NAXOS}}$ implied by NAXOS [19]

| $\hat{P}_i \quad (A = g^a, a)$ | | $\hat{P}_j \quad (B = g^b, b)$ |
|---|---|---|
| $r \leftarrow\$ \{0,1\}^\lambda,\ X \leftarrow g^{\mathsf{H}(a,r)}$ | $\xrightarrow{X}$ $\xleftarrow{Y}$ | $r' \leftarrow\$ \{0,1\}^\lambda,\ Y \leftarrow g^{\mathsf{H}(b,r')}$ |
| $sk_i \leftarrow \mathsf{H}_1(Y^a, B^{\mathsf{H}(a,r)}, Y^{\mathsf{H}(a,r)}, A, B)$ | | $sk_j \leftarrow \mathsf{H}_1(A^{\mathsf{H}(b,r')}, X^b, X^{\mathsf{H}(b,r')}, A, B)$ |

**Fig. 15.** $\mathcal{P}_{\text{NAXOS}}$: apply $\mathsf{KRF}_{\mathsf{NAXOS}}$ into our first enhanced modular construction. The required PRF is replaced by a RO $\mathrm{H}_1$, which covers the internal $\bar{\mathrm{H}}$.

**Theorem 6.** *If the GDH problem holds in $\mathbb{G}$, $\mathsf{H}$ and $\bar{\mathsf{H}}$ are modeled as random oracles, $\mathsf{KRF}_{\mathsf{NAXOS}}$ shown in Fig. 14 is both PR-LEK and PR-LX.*

*Proof.* For any PPT adversary $\mathcal{A}$ against the $\mathsf{Exp}^{\mathsf{PR\text{-}LEK}}_{\mathsf{KRF},\mathcal{A}}$ or $\mathsf{Exp}^{\mathsf{PR\text{-}LX}}_{\mathsf{KRF},\mathcal{A}}$, we build an algorithm simulating the corresponding experiments with these changes:

1. given a GDH challenge $(X, Y)$, set $\mathsf{rpk}^* \leftarrow X$, $\mathsf{y}^* \leftarrow Y$ and $\mathsf{w}_0^* \leftarrow\$ \mathcal{W}$;

2. initialize three empty lists $\mathcal{L}_{\mathsf{H}}$, $\mathcal{L}_{\bar{\mathsf{H}}}$ and $\mathcal{L}_{\mathsf{y}}$;
3. for a $\mathsf{H}(\mathsf{input})$ query:
   (a) if $\mathsf{input} = (\mathsf{ek}^*, \mathsf{x_2}^*)$, terminate the simulation with failure;
   (b) else if $\exists\ (\mathsf{input}, h) \in \mathcal{L}_{\mathsf{H}}$, return $h$;
   (c) otherwise, return $h \leftarrow_{\$} \mathbb{Z}_p$ and record $(\mathsf{input}, h)$ into $\mathcal{L}_{\mathsf{H}}$.
4. for a $\bar{\mathsf{H}}(\mathsf{epk}, Z)$ query:
   (a) if $\exists\ ((\mathsf{epk}, Z), h) \in \mathcal{L}_{\bar{\mathsf{H}}}$, return $h$;
   (b) else if $\mathrm{CDH}(X, Y) = Z$, halt and output $Z$ as the solution;
   (c) else if $\exists\ (\mathsf{y}, -, \mathsf{w}) \in \mathcal{L}_{\mathsf{y}}$ s.t. $\mathrm{CDH}(\mathsf{y}, X) = Z$, return $\mathsf{w}$. In addition, update corresponding records in $\mathcal{L}_{\mathsf{y}}$ and $\mathcal{L}_{\bar{\mathsf{H}}}$;
   (d) otherwise, returns $h \leftarrow_{\$} \{0,1\}^{\lambda}$ and record $((\mathsf{epk}, Z), h)$ into $\mathcal{L}_{\bar{\mathsf{H}}}$.
5. for an $\bar{\mathcal{O}}(\mathsf{epk}, -, \mathsf{y})$ query:
   (a) if $\mathsf{y} = Y$, return $\mathsf{w} \leftarrow_{\$} \{0,1\}^{\lambda}$ and record $((\mathsf{epk}, -), \mathsf{w})$ into $\mathcal{L}_{\bar{\mathsf{H}}}$;
   (b) else if $\exists\ ((\mathsf{epk}, Z), h) \in \mathcal{L}_{\bar{\mathsf{H}}}$ s.t. $\mathrm{CDH}(X, \mathsf{y}) = Z$, return $h$;
   (c) otherwise, return $\mathsf{w} \leftarrow_{\$} \{0,1\}^{\lambda}$, record $((\mathsf{epk}, -), \mathsf{w})$ into $\mathcal{L}_{\bar{\mathsf{H}}}$ and $(\mathsf{y}, -, \mathsf{w})$ into $\mathcal{L}_{\mathsf{y}}$, respectively.
6. if $\mathcal{A}$ is able to win in either experiment, there must exist a tuple $((\mathsf{epk}^*, J),$ $\mathsf{w}) \in \mathcal{L}_{\bar{\mathsf{H}}}$ and the value $J$ is a solution of the GDH problem instance.

If $\mathcal{A}$ has queried $\mathsf{H}$ on $(\mathsf{ek}^*, \mathsf{x_2}^*)$, the simulation fails. However, $\mathcal{A}$ just has a partial knowledge of the input, i.e., $\mathsf{ek}^*$ (resp., $\mathsf{x_2}^*$) when it is attempting to break the PR-LEK security (resp., the PR-LX security). Such bad event only occurs with negligible probability. □

Note that $\mathsf{KRF_{NAXOS}}$ and DHKE also meet *Simulatability*, and the corresponding simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ can be constructed as follows:

| $\mathcal{S}_1(\mathsf{epk}, \mathsf{rpk}, -, \mathsf{y})$: | $\mathcal{S}_2(pk, \mathsf{epk}, \mathsf{ek}, \mathsf{rpk}, \mathsf{rk}, \mathsf{x_1})$: |
|---|---|
| $pk \leftarrow \mathsf{y}$ | $\mathsf{y} \leftarrow pk$, $\mathsf{w} \leftarrow \bar{\mathsf{H}}(\mathsf{epk}, \mathsf{y}^{\mathsf{rk}})$ |
| **return** $pk$ | **return** $(\mathsf{y}, \mathsf{w})$ |

## 5.4 BJS

BJS was proved secure in the eCK-PFS model. In BJS, the KRF is initiated by $\mathsf{KRF_{BJS}}$ in Fig. 16. Let $\mathsf{NIKE}$=($\mathsf{NIKE.Gen}$, $\mathsf{NIKE.Key}$) be a NIKE with randomness space $\mathcal{R}$ and $\mathsf{SIG}$=($\mathsf{SIG.Gen}$, $\mathsf{SIG.Sign}$, $\mathsf{SIG.Vrfy}$) be a deterministic signature.

**Theorem 7.** *If $\mathsf{NIKE}$ is CKS-light secure and $\mathsf{SIG}$ is EUF-CMA, $\mathsf{KRF_{BJS}}$ shown in Fig. 16 is PE, PR-LEK and PR-LX.*

*Proof.* $\mathsf{KRF_{BJS}}$ is PE, since $\mathsf{y} \overset{def}{=} (pk^{\mathsf{t}}, \sigma)$ is actual a message/signature pair, any PPT adversary $\mathcal{A}_1$ is unable to output such a fresh and valid pair without breaking the EUF-CMA-security of the underlying $\mathsf{SIG}$. Note that the $\mathcal{O}(\cdot)$ oracle can be perfectly emulated using the underlying singing oracle.

$\mathsf{KRF_{BJS}}$ is also PR-LEK (resp., PR-LX), since the underlying $\mathsf{NIKE}$ is CKS-light secure, and $\mathsf{w} = \mathrm{CKey}(\mathsf{epk}[1], \mathsf{rpk}) \oplus \mathrm{CKey}(pk^{\mathsf{t}}, \mathsf{rpk})$, thus any PPT adversary $\mathcal{A}_2$ is unable to distinguish it from a random value without knowing $(\mathsf{ek}[1], sk^{\mathsf{t}})$ or $\mathsf{rk}$. However, if $\mathcal{A}_2$ is attempting to break the PR-LEK-security

```
$\mathsf{KRF_{BJS}.KG}(\mathsf{pp}, \psi)$:
  if $\psi = 0$
    $(pk^{\mathsf{s}}, sk^{\mathsf{s}}) \leftarrow \mathsf{SIG.Gen}(1^\lambda)$
    $(pk^{\mathsf{n}}, sk^{\mathsf{n}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda)$
    **return**
      $(\mathsf{epk}, \mathsf{ek}) \overset{\text{def}}{=} ((pk^{\mathsf{s}}, pk^{\mathsf{n}}), (sk^{\mathsf{s}}, sk^{\mathsf{n}}))$
  else if $\psi = 1$
    $(pk^{\mathsf{n}}, sk^{\mathsf{n}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda)$
    **return** $(\mathsf{rpk}, \mathsf{rk}) \overset{\text{def}}{=} (pk^{\mathsf{n}}, sk^{\mathsf{n}})$
```

```
$\mathsf{KRF_{BJS}.Eval}(\psi, \mathsf{input})$:
  if $\psi = 0$
    $(\mathsf{ek}, \mathsf{rpk}, -, \mathsf{x}_2) \leftarrow \mathsf{input}$
    $(pk^{\mathsf{t}}, sk^{\mathsf{t}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda; \mathsf{x}_2)$
    $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{ek}[0], pk^{\mathsf{t}})$
    $\mathsf{w} \leftarrow \mathsf{NIKE.Key}(\mathsf{ek}[1], \mathsf{rpk}) \oplus \mathsf{NIKE.Key}(sk^{\mathsf{t}}, \mathsf{rpk})$
    **return** $(\mathsf{y} \overset{\text{def}}{=} (pk^{\mathsf{t}}, \sigma), \mathsf{w})$
  else if $\psi = 1$
    $(\mathsf{rk}, \mathsf{epk}, -, \mathsf{y}) \leftarrow \mathsf{input}$, $(pk^{\mathsf{t}}, \sigma) \leftarrow \mathsf{y}$
    if $\mathsf{SIG.Vrfy}(\mathsf{epk}[0], pk^{\mathsf{t}}, \sigma) \neq 1$, **return** $\perp$
    else **return**
      $\mathsf{w} \overset{\text{def}}{=} \mathsf{NIKE.Key}(\mathsf{rk}, \mathsf{epk}[1]) \oplus \mathsf{NIKE.Key}(\mathsf{rk}, pk^{\mathsf{t}})$
```

**Fig. 16.** The $\mathsf{KRF_{BJS}}$ from BJS [3]

| $\hat{P}_i \ (pk_i^{\mathsf{n}}, pk_i^{\mathsf{s}}), (sk_i^{\mathsf{n}}, sk_j^{\mathsf{s}})$ | | $\hat{P}_j \ (pk_j^{\mathsf{n}}, pk_j^{\mathsf{s}}), (sk_j^{\mathsf{n}}, sk_j^{\mathsf{s}})$ |
|---|---|---|
| $(pk_i^{\mathsf{t}}, sk_i^{\mathsf{t}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda)$ | $\xrightarrow{pk_i^{\mathsf{t}}, \sigma_i}$ | $(pk_j^{\mathsf{t}}, sk_j^{\mathsf{t}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda)$ |
| $\sigma_i \leftarrow \mathsf{SIG.Sign}(sk_i^{\mathsf{s}}, pk_i^{\mathsf{t}})$ | $\xleftarrow{pk_j^{\mathsf{t}}, \sigma_j}$ | $\sigma_B \leftarrow \mathsf{SIG.Sign}(sk_j^{\mathsf{s}}, pk_j^{\mathsf{t}})$ |
| Abort if $\mathsf{SIG.Vrfy}(pk_j^{\mathsf{s}}, \sigma_j) \neq 1$ | | Abort if $\mathsf{SIG.Vrfy}(pk_i^{\mathsf{s}}, \sigma_i) \neq 1$ |
| $k_{\mathsf{n,n}} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{n}}, pk_j^{\mathsf{n}})$ | | $k'_{\mathsf{n,n}} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{n}}, pk_i^{\mathsf{n}})$ |
| $k_{\mathsf{n,t}} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{n}}, pk_j^{\mathsf{t}})$ | | $k'_{\mathsf{n,t}} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{t}}, pk_i^{\mathsf{n}})$ |
| $k_{\mathsf{t,n}} = \mathsf{NIKE.Key}(sk_i^{\mathsf{t}}, pk_j^{\mathsf{n}})$ | | $k'_{\mathsf{t,n}} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{n}}, pk_i^{\mathsf{t}})$ |
| $k_{\mathsf{t,t}} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{t}}, pk_j^{\mathsf{t}})$ | | $k'_{\mathsf{t,t}} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{t}}, pk_i^{\mathsf{t}})$ |
| Let $T \leftarrow pk_i^{\mathsf{n}} \| pk_i^{\mathsf{s}} \| pk_j^{\mathsf{n}} \| pk_j^{\mathsf{s}} \| pk_i^{\mathsf{t}} \| \sigma_i \| pk_j^{\mathsf{t}} \| \sigma_j$ | | |
| $\hat{P}_i$ compute $sk_i \leftarrow \mathrm{PRF}(k_{\mathsf{n,n}}, T) \oplus \mathrm{PRF}(k_{\mathsf{n,t}}, T) \oplus \mathrm{PRF}(k_{\mathsf{t,n}}, T) \oplus \mathrm{PRF}(k_{\mathsf{t,t}}, T)$ | | |
| $\hat{P}_j$ compute $sk_j \leftarrow \mathrm{PRF}(k'_{\mathsf{n,n}}, T) \oplus \mathrm{PRF}(k'_{\mathsf{n,t}}, T) \oplus \mathrm{PRF}(k'_{\mathsf{t,n}}, T) \oplus \mathrm{PRF}(k'_{\mathsf{t,t}}, T)$ | | |

**Fig. 17.** $\mathcal{P}_{\mathrm{BJS}}$: apply $\mathsf{KRF_{BJS}}$ into our first enhanced modular construction.

(resp., PR-LX-security) of $\mathsf{KRF_{BJS}}$, it can only learn $\mathsf{ek}$ (resp., $sk^{\mathsf{t}}$ derived from $\mathsf{x}_2$). Note that by setting the pair of public keys $(pk^t, \mathsf{rpk})$ (resp., $(\mathsf{epk}[1], \mathsf{rpk})$) as the target two honestly registered keys, and the $\bar{\mathcal{O}}(\cdot)$ (and $\mathcal{O}(\cdot)$) oracle can be perfectly emulated using the underlying CorruptReveal$(\cdot)$ oracle. Due to page limitations, we drop the details here. $\square$

Note that it is easy to conclude that NIKE implies passively secure KE. In addition, $\mathsf{KRF_{BJS}}$ and $\mathsf{NIKE}$ meet *Simulatability*, and the corresponding simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ can be constructed as follows:

```
$\mathcal{S}_1(\mathsf{epk}, \mathsf{rpk}, -, \mathsf{y})$:
  $(pk^{\mathsf{t}}, \sigma) \leftarrow \mathsf{y}$
  **return** $pk \overset{def}{=} pk^{\mathsf{t}}$
```

```
$\mathcal{S}_2(pk, \mathsf{epk}, \mathsf{ek}, \mathsf{rpk}, \mathsf{rk}, -)$:
    $pk^{\mathsf{t}} \leftarrow pk$, $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{ek}[0], pk^{\mathsf{t}})$
    $\mathsf{w} \leftarrow \mathsf{NIKE.Key}(\mathsf{ek}[1], \mathsf{rpk}) \oplus \mathsf{NIKE.Key}(\mathsf{rk}, pk^{\mathsf{t}})$
  **return** $(\mathsf{y} \overset{def}{=} (pk^{\mathsf{t}}, \sigma), \mathsf{w})$
```

## 6 Further Results for ORKEs

In this section, we give some new results regarding ORKEs by applying our main results in this paper.

### 6.1 A Protocol with CK-PFS Security

According to our main results in Theorem 1, we can get the following result:

**Corollary 1.** *According to the modular construction in* Fig. 6*, one protocol instantiation proved secure in the CK$^+$ (resp., CK$^+$-PFS) model is also secure in the eCK (resp., eCK-PFS) model, and vice versa.*

Thus, our modular construction can be instantiated in the new CK$^+$-PFS model using the the protocol $\mathcal{P}_{\text{BJS}}$ illustrated in Fig. 17. As an application of our modular construction, we present an another protocol secure in the CK-PFS model in this section.

**An Immediate Construction.** Inspired by previous works [6,27,9,3,26], we can immediately get a construction in the CK-PFS model by applying a compiler to an 2KEM+DH in the CK model (see Fig 11) using an EUF-CMA deterministic signature $\mathsf{SIG}$=($\mathsf{SIG.Gen}$,$\mathsf{SIG.Sign}$,$\mathsf{SIG.Vrfy}$), which we denote as SIG(2KEM+DH) (see Fig. 18). It is easy to prove its security in the CK-PFS model. We drop the details here.

| $\hat{P}_i(ek_i, dk_i)(sk_i, vk_i)$ | | $\hat{P}_j(ek_j, dk_j)(sk_j, vk_j)$ |
|---|---|---|
| $x \leftarrow\$ \mathbb{Z}_p, X \leftarrow g^x$ <br> $(c_i, k_i) \leftarrow \mathsf{KEM.Enc}(ek_j)$ <br> $\sigma_i \leftarrow \mathsf{SIG.Sign}(sk_i, c_i \| X)$ | $\xrightarrow{c_i, X, \sigma_i}$ <br> $\xleftarrow{c_j, Y, \sigma_j}$ | $y \leftarrow\$ \mathbb{Z}_p, Y \leftarrow g^y$ <br> $(c_j, k_j) \leftarrow \mathsf{KEM.Enc}(ek_i)$ <br> $\sigma_j \leftarrow \mathsf{SIG.Sign}(sk_j, c_j \| Y)$ |
| Abort if $\mathsf{SIG.Vrfy}(vk_j, c_j \| Y, \sigma_j) \neq 1$ <br> $k'_j \leftarrow \mathsf{KEM.Dec}(dk_i, c_j)$ | | Abort if $\mathsf{SIG.Vrfy}(vk_i, c_i \| X, \sigma_i) \neq 1$ <br> $k'_i \leftarrow \mathsf{KEM.Dec}(dk_j, c_i)$ |
| Let $T \leftarrow ek_i \| vk_i \| ek_j \| vk_j \| c_i \| X \| \sigma_i \| c_j \| Y \| \sigma_j$ <br> $\hat{P}_i$ compute $sk_i \leftarrow \text{PRF}(k_i, T) \oplus \text{PRF}(k'_j, T) \oplus \text{PRF}(Y^x, T)$ <br> $\hat{P}_j$ compute $sk_j \leftarrow \text{PRF}(k_i, T) \oplus \text{PRF}(k'_j, T) \oplus \text{PRF}(X^y, T)$ | | |

**Fig. 18.** The protocol SIG(2KEM+DH)

**A New Construction from Our Modular Construction.** According to our main results in Theorem 1, the key to achieve secure ORKE in the CK-PFS model is to construct a KRF that is both PE and PR-LEK. On another side, we have proved in Theorem 7 that the $\mathsf{KRF}_{\mathsf{BJS}}$ (see Fig. 16) is PE, PR-LEK and PR-LX. It is quite nature to build a new scheme using the similar idea behind the construction of $\mathsf{KRF}_{\mathsf{BJS}}$ by reducing some unnecessary secrets and computations. Let $\mathsf{SIG}$=($\mathsf{SIG.Gen}$, $\mathsf{SIG.Sign}$, $\mathsf{SIG.Vrfy}$) be a deterministic signature and $\mathsf{NIKE}$ be a NIKE. We first give a construction $\mathsf{KRF}_{\mathsf{new}}$ as in Fig. 19, then apply it into our modular construction to derive an ORKE protocol $\mathcal{P}_{\text{new}}$ as in Fig. 20.

**Theorem 8.** *If $\mathsf{SIG}$ is EUF-CMA and $\mathsf{NIKE}$ is secure in the CKS-light model with randomness space $\mathcal{R}$, $\mathsf{KRF}_{\mathsf{new}}$ shown in Fig.* 19 *is both PE and PR-LEK.*

*Proof.* It is also easy to prove this theorem. First, $\mathsf{KRF}_{\mathsf{BJS}}$ is PE, since $\mathsf{y} \overset{def}{=} (pk^{\mathsf{t}}, \sigma)$ is actual a message/signature pair, any PPT adversary $\mathcal{A}_1$ is unable to output such a fresh and valid pair without breaking the EUF-CMA-security of the underlying $\mathsf{SIG}$. Note that the $\mathcal{O}_f$ oracle can be perfectly emulated using the underlying singing oracle.

```
$\mathsf{KRF_{new}.KG}(\mathsf{pp}, \psi)$:
    if $\psi = 0$
        $(vk, sk) \leftarrow \mathsf{SIG.Gen}(1^\lambda)$
        return $(\mathsf{epk}, \mathsf{ek}) \overset{\text{def}}{=} (vk, sk)$
    else if $\psi = 1$
        $(pk^{\mathsf{n}}, sk^{\mathsf{n}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda)$
        return $(\mathsf{rpk}, \mathsf{rk}) \overset{\text{def}}{=} (pk^{\mathsf{n}}, sk^{\mathsf{n}})$
```

```
$\mathsf{KRF_{new}.Eval}(\psi, \mathsf{input})$:
    if $\psi = 0$
        $(\mathsf{ek}, \mathsf{rpk}, -, \mathsf{x}_2) \leftarrow \mathsf{input}$
        $(pk^{\mathsf{t}}, sk^{\mathsf{t}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda; \mathsf{x}_2)$
        $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{ek}, pk^{\mathsf{t}})$
        $\mathsf{w} \leftarrow \mathsf{NIKE.Key}(sk^{\mathsf{t}}, \mathsf{rpk})$
        return $(\mathsf{y} \overset{\text{def}}{=} (pk^{\mathsf{t}}, \sigma), \mathsf{w})$
    else if $\psi = 1$
        $(\mathsf{rk}, \mathsf{epk}, -, (pk^{\mathsf{t}}, \sigma)) \leftarrow \mathsf{input}$
        $s \leftarrow \mathsf{SIG.Vrfy}(\mathsf{epk}, pk^{\mathsf{t}}), \sigma)$
        if $s \neq 1$, return $\perp$
        else return $\mathsf{w} \overset{\text{def}}{=} \mathsf{NIKE.Key}(\mathsf{rk}, pk^{\mathsf{t}})$
```

**Fig. 19.** Our proposal $\mathsf{KRF_{new}}$

| $\hat{P}_i(pk_i^{\mathsf{n}}, pk_i^{\mathsf{s}}), (sk_i^{\mathsf{n}}, sk_i^{\mathsf{s}})$ | | $\hat{P}_j(pk_j^{\mathsf{n}}, pk_j^{\mathsf{s}}), (sk_j^{\mathsf{n}}, sk_j^{\mathsf{s}})$ |
|---|---|---|
| $(pk_i^{\mathsf{t}}, sk_i^{\mathsf{t}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda)$ <br> $\sigma_i \leftarrow \mathsf{SIG.Sign}(sk_i^{\mathsf{s}}, pk_i^{\mathsf{t}})$ | $\xrightarrow{pk_i^{\mathsf{t}}, \sigma_i}$ <br> $\xleftarrow{pk_j^{\mathsf{t}}, \sigma_j}$ | $(pk_j^{\mathsf{t}}, sk_j^{\mathsf{t}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda)$ <br> $\sigma_j \leftarrow \mathsf{SIG.Sign}(sk_j^{\mathsf{s}}, pk_j^{\mathsf{t}})$ |
| Abort if $\mathsf{SIG.Vrfy}(pk_j^{\mathsf{s}}, \sigma_j) \neq 1$ <br> $k_{\mathsf{n,t}} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{n}}, pk_j^{\mathsf{t}})$ <br> $k_{\mathsf{t,n}} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{t}}, pk_j^{\mathsf{n}})$ <br> $k_{\mathsf{t,t}} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{t}}, pk_j^{\mathsf{t}})$ | | Abort if $\mathsf{SIG.Vrfy}(pk_i^{\mathsf{s}}, \sigma_i) \neq 1$ <br> $k'_{\mathsf{n,t}} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{t}}, pk_i^{\mathsf{n}})$ <br> $k'_{\mathsf{t,n}} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{n}}, pk_i^{\mathsf{t}})$ <br> $k'_{\mathsf{t,t}} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{t}}, pk_i^{\mathsf{t}})$ |
| Let $T \leftarrow pk_i^{\mathsf{n}}\|\|pk_i^{\mathsf{s}}\|\|pk_j^{\mathsf{n}}\|\|pk_j^{\mathsf{s}}\|\|pk_i^{\mathsf{t}}\|\|\sigma_i\|\|pk_j^{\mathsf{t}}\|\|\sigma_j$ <br> $\hat{P}_i$ compute $sk_i \leftarrow \mathrm{PRF}(k_{\mathsf{n,t}}, T) \oplus \mathrm{PRF}(k_{\mathsf{t,n}}, T) \oplus \mathrm{PRF}(k_{\mathsf{t,t}}, T)$ <br> $\hat{P}_j$ compute $sk_j \leftarrow \mathrm{PRF}(k'_{\mathsf{n,t}}, T) \oplus \mathrm{PRF}(k'_{\mathsf{t,n}}, T) \oplus \mathrm{PRF}(k'_{\mathsf{t,t}}, T)$ | | |

**Fig. 20.** $\mathcal{P}_{\text{new}}$: apply $\mathsf{KRF_{new}}$ into our first enhanced modular construction.

Second, $\mathsf{KRF_{BJS}}$ is PR-LEK, since the underlying $\mathsf{NIKE}$ is CKS-light secure, and $\mathsf{w} = \mathrm{CKey}(pk^{\mathsf{t}}, \mathsf{rpk})$, any PPT adversary $\mathcal{A}_2$ is unable to distinguish it from a random value without knowing $sk^{\mathsf{t}}$ or $\mathsf{rk}$. If $\mathcal{A}_2$ is attempting to break the PR-LEK-security, it cannot learn neither $sk^{\mathsf{t}}$ (due to the privacy of $\mathsf{x}_2$) nor $\mathsf{rk}$. Note that by setting the pair of public keys $(pk^t, \mathsf{rpk})$ as the target two honestly registered keys, the $\bar{\mathcal{O}}(\cdot)$ oracle can be perfectly emulated using the underlying CorruptReveal$(\cdot)$ oracle. Due to page limitations, we drop the details here. $\square$

Note that $\mathsf{KRF_{new}}$ and $\mathsf{NIKE}$ also meet *Simulatability*, and the corresponding simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ can be constructed as follows:

```
$\mathcal{S}_1(\mathsf{epk}, \mathsf{rpk}, -, \mathsf{y})$:
    $(pk^{\mathsf{t}}, \sigma) \leftarrow \mathsf{y}$
    return $pk \overset{def}{=} pk^{\mathsf{t}}$
```

```
$\mathcal{S}_2(pk, \mathsf{epk}, \mathsf{ek}, \mathsf{rpk}, \mathsf{rk}, -)$:
    $pk^{\mathsf{t}} \leftarrow pk$, $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{ek}, pk^{\mathsf{t}})$
    $\mathsf{w} \leftarrow \mathsf{NIKE.Key}(\mathsf{rk}, pk^{\mathsf{t}})$
return $(\mathsf{y} \overset{def}{=} (pk^{\mathsf{t}}, \sigma), \mathsf{w})$
```

**Comparisons Between the Two Constructions.** We compared our new proposal $\mathcal{P}_{\text{new}}$ with the SIG(2KEM+DH) construction by instantiating it using the most efficient factoring-based NIKE [10] that was proved secure in the RO model. To make them comparable, we instantiate the required KEM in the generic SIG(2KEM+DH) construction using the ElGamal encryption after applying a FO-transformation [13,14], thus a ciphertext includes at least 2 group elements, and each call of the encapsulation (resp., decapsulation) algorithm costs at least 2 (resp., 3) modular exponentiations. The comparison details are shown in Table 14. On both of the communication and computation overheads, our proposal is more efficient.

**Table 14.** Comparisons Between $\mathcal{P}_{\text{new}}$ and SIG(2KEM+DH)

| Scheme | Security Model | Group elements sent. per party | Exp. per party | Compiler Used |
|---|---|---|---|---|
| SIG(2KEM+DH) | CK-PFS | 3 | 7 | SIG($\cdot$) |
| $\mathcal{P}_{\text{new}}$ | CK-PFS | 1 | 4 | Our modular construction |

†† We do not distinguish an Exponentiation (Exp.) in a DH group from an Exp. in an RSA group. As both schemes involve signature generating and validating, we omit them in the comparisons.

The comparisons supported the usability of our framework well, namely, it is not only a generalization of the existing works, but also a useful tool to construct efficient protocols in different models due to its simplicity.

## 6.2 A Construction Secure in All the Considered Models

Cremers [8] pointed out that the original CK [7], CK$^+$ [18] and eCK [19] models are not comparable, by showing a protocol can be secure in one model and yet insecure in other models. One of the reasons behind is that these models used matching notions in different ways. They defined four types of session relations:

– $s \approx_A s' \overset{\text{def}}{=} s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor} \wedge s_{sent} = s'_{recv} \wedge s_{recv} = s'_{sent}$;
– $s \approx_B s' \overset{\text{def}}{=} s \approx_A s' \wedge (s_{role} \neq s'_{role} \vee s_{actor} = s_{peer})$;
– $s \approx_C s' \overset{\text{def}}{=} s \approx_A s' \wedge (s_{role} \neq s'_{role})$;
– $s \approx_D s' \overset{\text{def}}{=} s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor} \wedge s_{id} = s'_{id}$.

The original CK, CK$^+$ and eCK models used $\approx_D$, $\approx_A$ and $\approx_C$, respectively. Two sessions matched in one model are not necessarily matched in another model, thus trivial success may occur. However, we have unified the way to define matching sessions in these models, i.e., the $\approx_A$ type. As we are considering the security in the context of ORKE, which is role-symmetric (i.e., the messages of each role are identical up to their order), the $\approx_A$ type definition is our preference. In [8], it is also pointed out that "role-symmetric protocols with key type $\approx_B$ or $\approx_C$ do not satisfy CK$^+$ security." The key type is defined as: a protocol has key type $\approx_{T|T\in\{A,B,C,D\}}$, if for all completed sessions $s$ and $s'$, $\mathsf{kdf(s)} = \mathsf{kdf(s')} \Leftrightarrow \mathsf{s} \approx_{\mathsf{T}} \mathsf{s'}$, where $\mathsf{kdf}$ is an abstraction of the key derivation function of this protocol. Technically, our modular construction adopted a $\approx_A$ type key derivation function. Our result does not contradict the results in [8].

Hence, even if we have a preconception that the CK, CK$^+$ and eCK models are incomparable, it is not precluded that a protocol can be secure in two or more models. Therefore, it is an nature question that does there exist a protocol that is secure in all of the CK, CK$^+$, eCK, CK-PFS, CK$^+$-PFS and eCK-PFS models we considered. The answer is yes based on Theorem 1:

**Corollary 2.** *If KE is passively secure and KRF is fully secure, i.e., meets PE, PR-LEK and PR-LX simultaneously, the modular construction illustrated in* Fig. 6 *is secure in the CK, CK-PFS, CK$^+$, CK$^+$-PFS, eCK and eCK-PFS models at the same time.*

Combining Theorem 1 and 7, the protocol $\mathcal{P}_{\mathrm{BJS}}$ illustrated in Fig. 17 (constructed from the basic idea of BJS [3]) is simultaneously secure in these models.

## Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work was supported in part by National Natural Science Foundation of China (Nos. 61772520, 61802392 and 61972094).

## References

1. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: EUROCRYPT'00. pp. 139–155 (2000). https://doi.org/10.1007/3-540-45539-6_11
2. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: CRYPTO'93. pp. 232–249 (1993). https://doi.org/10.1007/3-540-48329-2_21
3. Bergsma, F., Jager, T., Schwenk, J.: One-Round Key Exchange with Strong Security: An Efficient and Generic Construction in the Standard Model. In: PKC'15. pp. 477–494 (2015). https://doi.org/10.1007/978-3-662-46447-2_21
4. Boyd, C., Cliff, Y., Nieto, J.M.G., Paterson, K.G.: Efficient One-Round Key Exchange in the Standard Model. In: ACISP'08. pp. 69–83 (2008). https://doi.org/10.1007/978-3-540-70500-0_6
5. Boyd, C., Cliff, Y., Nieto, J.M.G., Paterson, K.G.: One-round key exchange in the standard model. IJACT **1**(3), 181–199 (2009). https://doi.org/10.1504/IJACT.2009.023466
6. Boyd, C., Nieto, J.G.: On Forward Secrecy in One-Round Key Exchange. In: IMACC'11. pp. 451–468 (2011). https://doi.org/10.1007/978-3-642-25516-8_27
7. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: EUROCRYPT'01. pp. 453–474 (2001). https://doi.org/10.1007/3-540-44987-6_28
8. Cremers, C.: Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In: ASIA CCS'11. pp. 80–91 (2011). https://doi.org/10.1145/1966913.1966925
9. Cremers, C.J.F., Feltz, M.: Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal. In: ESORICS'12. pp. 734–751 (2012). https://doi.org/10.1007/978-3-642-33167-1_42
10. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-Interactive Key Exchange. In: PKC'13. pp. 254–271. https://doi.org/10.1007/978-3-642-36362-7_17
11. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly Secure Authenticated Key Exchange from Factoring, Codes, and Lattices. In: PKC'12. pp. 467–484. https://doi.org/10.1007/978-3-642-30057-8_28
12. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In: ASIA CCS '13. pp. 83–94 (2013). https://doi.org/10.1145/2484313.2484323
13. Fujisaki, E., Okamoto, T.: How to Enhance the Security of Public-Key Encryption at Minimum Cost. In: PKC '99. pp. 53–68 (1999). https://doi.org/10.1007/3-540-49162-7_5

14. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: CRYPTO '99. pp. 537–554 (1999). https://doi.org/10.1007/3-540-48405-1_34
15. Just, M., Vaudenay, S.: Authenticated Multi-Party Key Agreement. In: ASIACRYPT'96. pp. 36–49 (1996). https://doi.org/10.1007/BFb0034833
16. Kim, M., Fujioka, A., Ustaoglu, B.: Strongly Secure Authenticated Key Exchange without NAXOS' Approach. In: IWSEC'09. pp. 174–191 (2009). https://doi.org/10.1007/978-3-642-04846-3_12
17. Krawczyk, H.: SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. In: CRYPTO'03. pp. 400–425 (2003). https://doi.org/10.1007/978-3-540-45146-4_24
18. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: CRYPTO'05. pp. 546–566 (2005). https://doi.org/10.1007/11535218_33
19. LaMacchia, B.A., Lauter, K.E., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: ProvSec'07. pp. 1–16 (2007). https://doi.org/10.1007/978-3-540-75670-5_1
20. Law, L., Menezes, A., Qu, M., Solinas, J.A., Vanstone, S.A.: An Efficient Protocol for Authenticated Key Agreement. Des. Codes Cryptogr. **28**(2), 119–134 (2003)
21. Okamoto, T.: Authenticated Key Exchange and Key Encapsulation in the Standard Model. In: ASIACRYPT'07. pp. 474–484 (2007). https://doi.org/10.1007/978-3-540-76900-2_29
22. Peikert, C.: Lattice Cryptography for the Internet. In: PQCrypto'14. pp. 197–219 (2014). https://doi.org/10.1007/978-3-319-11659-4_12
23. Strangio, M.A.: On the Resilience of Key Agreement Protocols to Key Compromise Impersonation. In: EuroPKI'06. pp. 233–247 (2006). https://doi.org/10.1007/11774716_19
24. Xue, H., Lu, X., Li, B., Liang, B., He, J.: Understanding and Constructing AKE via Double-Key Key Encapsulation Mechanism. In: ASIACRYPT'18. pp. 158–189 (2018). https://doi.org/10.1007/978-3-030-03329-3_6
25. Yang, Z.: Efficient eCK-Secure Authenticated Key Exchange Protocols in the Standard Model. In: ICICS'13. pp. 185–193 (2013). https://doi.org/10.1007/978-3-319-02726-5_14
26. Yang, Z., Chen, Y., Luo, S.: Two-Message Key Exchange with Strong Security from Ideal Lattices. In: CT-RSA'18. pp. 98–115 (2018). https://doi.org/10.1007/978-3-319-76953-0_6
27. Yoneyama, K.: One-Round Authenticated Key Exchange with Strong Forward Secrecy in the Standard Model against Constrained Adversary. In: IWSEC 2012. pp. 69–86 (2012). https://doi.org/10.1007/978-3-642-34117-5_5