Information and Computation 285 (2022) 104866

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

A new framework for deniable secure key exchange

Shaoquan Jiang [a,*,1], Yeow Meng Chee [b], San Ling [c], Huaxiong Wang [c], Chaoping Xing [d,2]

[a] University of Windsor, Canada
[b] National University of Singapore, Singapore
[c] Nanyang Technological University, Singapore
[d] Shanghai Jiao Tong University, China

ARTICLE INFO

Article history:
Received 15 February 2010
Received in revised form 6 January 2022
Accepted 9 January 2022
Available online 19 January 2022

Keywords:
Key exchange
Deniable authentication
Projective hash family

ABSTRACT

A deniable secure key exchange protocol allows two parties to agree on a common secret while achieving two seemingly contradictory functionalities: authentication and deniability. The former requires each party to confirm the identity of the other while the latter requires any attacker (e.g., participant or eavesdropper) be unable to prove to a third party an honest party's participation. Designing an efficient secure key exchange with deniability is a challenging problem. In this paper, we first formalize the deniability model by requiring information theoretic deniability with an eavesdropping attack. The information theoretic deniability has the advantage that it can hold forever without any computational assumption. An eavesdropping attack (Di Raimondo et al., CCS'06) allows an attacker to apply eavesdropped transcripts into an active attack session. This gives an attacker more power to make the victim undeniable as he does not know the randomness of the transcript. We then propose an efficient, provably deniable secure framework of key exchange. Our deniability holds non-adaptively in the eavesdropping model. However, if we consider a model without an eavesdropping attack (which is practical in many scenarios), then our framework is proven *adaptively* deniable. This is important since no previous key exchange protocols can satisfy our adaptive and information theoretical deniability. We give a concrete realization for our framework that is more efficient than SKEME (Krawczyk, NDSS'96).

1. Introduction

Key exchange is a procedure that allows two parties to share a secret, with which the subsequent communication is protected. The first protocol was proposed by Diffie-Hellman [18]. Since then, it has been extensively studied in the literature. With the popularity of computer networks, many businesses and transactions have moved onto the Internet. Accordingly, people's security concerns have gone beyond the basic confidentiality. Privacy, such as identity or location anonymity, deniability and non-repudiation, is now being seriously considered. In this paper, we study the deniability issue. Deniability

* Corresponding author.
E-mail addresses: jiangshq@uwindsor.ca, shaoquan.jiang@gmail.com (S. Jiang).

[1] This work was mainly done when S. Jiang was visiting NTU.

[2] The research of Chaoping Xing is supported in part by the National Natural Science Foundation of China under Grant 12031011 and the National Key Research and Development Program 2021YFE0109900.

https://doi.org/10.1016/j.ic.2022.104866
0890-5401/

intuitively means that when one party has participated in a conversation, he can later deny the fact of communication against any malicious adversary (e.g., the other communicant or an *eavesdropper*). That is to say, nobody can form any convincing evidence regarding his participation in the communication. This allows him to avoid any trace that might lead him to undesirable consequences (e.g., a bad reputation, a lawful responsibility or annoying disturbance). The importance of deniability has now been realized by many authors (e.g. the Internet draft [25]). Deniable authentication can be achieved by setting up a common message authentication key between two communicants. However, a physical key setup is not always practical and possible. A natural solution seemingly is to let parties run a key exchange protocol. In this case, it requires the key exchange protocol to be deniable. Note that for a key exchange protocol to be secure, each party must be authenticated. That is, each participant should confirm that his partner indeed sits at the other side jointly running the protocol with him. Hence, a deniable secure key exchange protocol must satisfy two seemingly contradictory properties: deniability and authentication.

1.1. Related work

Deniable authentication was first considered by Dolev et al. [20] and formally investigated by Dwork et al. [21], in which deniability was formulated in the zero knowledge sense [24]: whatever computable through an interaction can be computed by an adversary alone. In terms of deniability, when an adversary presents a proof about a party's participation in a communication procedure, the latter can deny it since the adversary can generate the proof alone. Deniability under this formulation has attracted a considerable amount of attention [22,21,30,37]. Deniable key exchange was informally studied by Mao and Paterson [36]. The formal treatment was first due to Di Raimondo et al. [17]. They presented a non-adaptive deniability model with eavesdropping attacks, where an eavesdropped message is provided to the adversary through an auxiliary input. They showed that SKEME [32] is deniable if the underlying encryption scheme is strongly plaintext aware (i.e., PA2) and IND-CPA secure. A PA2 secure encryption can be realized by Cramer-Shoup [11] (where the PA2 property was proven by Dent [14]) or Kurosawa-Desmedt [35] (where the PA2 property was proven by Jiang and Wang [29]). Partial deniability was also studied in the literature. Dwork et al. [21] formulated the partial deniable authentication as follows. The sender agrees having authenticated something to some receiver but he doe not agree that the authenticated message is what the adversary claims. They realized this formulation under a timing assumption. For the key exchange problem, Di Raimondo et al. [17] defined the partial deniability in a related manner: an honest party admits having run the key exchange with somebody but he does not agree that this 'somebody' is the adversary. They showed that a signature based key exchange protocol SIGMA [33] is *partially* deniable. As mentioned in [17], whether the partial deniability is satisfactory still depends on the specific application. In this paper, we only consider the deniability in the zero knowledge sense (instead of partial deniability discussed here). Dodis et al. [19] considered an on-line deniable authentication under the generalized UC model [6,9]. Their model is very strong so that an adaptive deniable authentication in the PKI model is impossible. They also formulated a weak form of deniable key exchange termed key exchange with incriminating abort (KEIA). Intuitively, KEIA guarantees deniability as long as the protocol terminates successfully; once a shared key is established, deniability is guaranteed. However, the deniability under this formulation is not always satisfactory. For instance, Unger and Goldberg [42,43] adopted this model and constructed several protocols. Their protocol XZDH starts with the first message:

$\mathrm{I} \rightarrow R$: msg_1, $\mathrm{Sig}(PK_I, SK_I, msg_2)$,

where msg_i does not involve any long-term secret and $\mathrm{Sig}(PK_I, SK_I, msg_2)$ is the signature on msg_2 using I's long-term secret SK_I with respect to public key PK_I. Intuitively, with this message, the sender is not deniable. Actually, in our understanding, their provable deniability seems more related to the partial deniability discussed above (especially because msg_2 is random). Similar issues in terms of ring signatures occur in other protocols in [42,43]. A random oracle [5] is an efficient tool for practical constructions. However, the deniability under the random oracle model is not guaranteed [39]. Non-programmable random oracle (that enables deniability) based key exchange protocols were proposed in [28,45]. In this paper, we only consider a deniable secure key exchange without a random oracle. One may also wish to apply a deniable message transmission authenticator [15,16] to a deniable key exchange protocol in the authenticated model [2]. However, the resulting key exchange protocol will be inefficient. Also the deniable authenticators [15,16] assume a timing assumption when considering the deniability in the concurrent model. It turns out the resulting key change protocol also needs this assumption. Jiang [27] constructed a deniable key exchange from a timed encryption which in turn is based on the timing assumption. In this paper, we will study schemes without this assumption. Deniability was also studied in the quantum setting [1,40], where the deniability was formulated in the spirit of deniable encryption [7,38,8].

1.2. Our work

In this paper, we study the deniable security of a key exchange protocol, where the secrecy model is from Bellare-Rogaway [4] and the deniability model is revised from Di Raimondo-Gennaro-Krawczyk [17]. We require the deniability to hold information theoretically while [17] only requires computational deniability. The information theoretical deniability has the advantage that it can hold forever without any computational assumption. Similar to [17], we require the deniability to capture an eavesdropping attack by allowing an adversary to access some properly distributed protocol transcripts. In our (deniability or secrecy) model, all the events are arbitrarily scheduled by the adversary. Especially, the adversary can concurrently and adaptively schedule his attacks. If an adversary makes all the party corruptions at the beginning, then

his attack is *non-adaptive*; otherwise, it is *adaptive*. We stress that here the non-adaptivity is only with respect to the corruption strategy. This has no effect on his adaptive scheduling on other attacks (e.g., an impersonation attack). We construct a new efficient deniable secure key exchange framework using an extractable hash proof system. We show that our framework is non-adaptively deniable with an eavesdropping attack and adaptively secret and authenticated. We notice that the eavesdropping attack is meaningful only if the adversary has an evidence that the eavesdropping transcript is indeed obtained by eavesdropping (instead of by simulation). When such an evidence is not available, it is still important to consider the model without an eavesdropping attack. If this is the case, we show that our framework is adaptively deniable. This is important since no previous key exchange protocols can satisfy our adaptive and information theoretic deniability. We realize our framework using a hash proof system in [35,12]. Each party in the realization needs 6 exponentiations and is more efficient than SKEME, where the latter needs 8 exponentiations.

2. Preliminaries

Notations. For a set S, $x \leftarrow S$ samples x from S randomly; $A|B$ is a concatenation of A with B. $negl : \mathbb{N} \to \mathbb{R}$ is a **negligible** function: for any polynomial $p(n)$, $\lim_{n\to\infty} negl(n)p(n) = 0$. Statistical distance of random variables A, B over a finite alphabet Ω is defined as $\mathsf{dist}[A, B] = \frac{1}{2}\sum_{v\in\Omega} |\Pr[A = v] - \Pr[B = v]|$. PPT stands for probabilistic polynomial time. In this paper, κ denotes the security parameter. For a number a, $|a|$ denotes its bit-length.

2.1. Diffie-Hellman assumptions

Let $p = 2q + 1$ be a safe prime (thus q a prime). $\mathbb{G}$ is the subgroup of $\mathbb{Z}_p^*$ of order q. Let g be a random generator of $\mathbb{G}$.

Decisional Diffie-Hellman Assumption. Let $\mathsf{DH} = (g, g^x, g^y, g^{xy})$, $\mathsf{Rand} = (g, g^x, g^y, g^z)$, for $x, y, z \leftarrow \mathbb{Z}_q$. The *decisional Diffie-Hellman (DDH) assumption* is to say DH and Rand are indistinguishable.

Extended Diffie-Hellman Knowledge (DHK*) Assumption. Diffie-Hellman Knowledge (DHK) assumption [3,13] essentially states that given $g, A \in \mathbb{G}$, any adversary, who produces (g^t, A^t) must know t. The extended DHK assumption (denoted by DHK* assumption) requires that the awareness of t holds even if the attacker can access to additional oracles. We now formally present DHK* assumption and describe DHK assumption as a special case of DHK*.

For any PPT adversary $\mathcal{H}$, there exists a PPT algorithm (called *extractor*) $\mathcal{H}^*$ such that the following experiment outputs 1 with only a negligible probability.

> Challenger $\mathbb{T}$ takes $a \leftarrow \mathbb{Z}_q$ and computes $A = g^a$. He samples $r, r^* \leftarrow \{0,1\}^*$ as a random tape for $\mathcal{H}$ and $\mathcal{H}^*$, respectively. Let $\aleph = \{\}$. $\mathbb{T}$ runs $\mathcal{H}$ with input (p, g, A, r) and $\mathcal{H}^*$ with input (p, g, A, r, r^*). The experiment rules are as follows.
>
> i. *$\mathcal{H}$ can issue any query (B, C) to $\mathcal{H}^*$ who will return some $b \in \mathbb{Z}_q$. If $B^a = C$ but $B \neq g^b$, then the experiment terminates with output 1; otherwise, it continues.*
> ii. *$\mathcal{H}$ can issue any sampling query to $\mathbb{T}$. Upon this, $\mathbb{T}$ takes $u_1, u_2 \leftarrow \mathbb{Z}_q$ and returns $U = g^{u_1}A^{u_2}$ to both $\mathcal{H}$ and $\mathcal{H}^*$. Finally, $\mathbb{T}$ updates $\aleph = \aleph \cup \{(u_1, u_2, U)\}$.*
> iii. *$\mathcal{H}$ can issue any reverse-sampling query U to $\mathbb{T}$. Upon this, if $(u_1, u_2, U) \in \aleph$ for some (u_1, u_2), $\mathbb{T}$ provides (u_1, u_2, U) to $\mathcal{H}$ and $\mathcal{H}^*$; otherwise, he ignores the query.*
>
> If the experiment does not terminate when $\mathcal{H}$ halts, it terminates with 0.

Diffie-Hellman Knowledge (DHK) Assumption. DHK assumption is a special case of DHK*, where $\mathcal{H}$ does not make a sampling query or a reverse-sampling query.

Remark. Compared with the DHK assumption, although $\mathcal{H}$ in a DHK* game has an extra access to two types of queries, DHK* does not seem stronger than DHK, since in items (ii)(iii) in DHK*, $\mathcal{H}$ does not get more information than purely random elements in $\mathbb{Z}_p$ or in $\mathbb{Z}_q^2$. This certainly does not help the attacker create (g^x, A^x) with an unknown x. However, these unimportant queries are useful later in proving the adaptive deniability of our protocol.

2.2. Simulatable random variable

We introduce a notion of *simulatable random variable*. Let Z be a random variable over a finite set V. For $\ell \in \mathbb{N}$, let $\Phi : \{0,1\}^\ell \to V$ be an efficiently computable deterministic function and $\Phi^* : V \to \{0,1\}^\ell$ be an efficiently computable probabilistic function. Then, we say that Z is simulatable by (Φ, Φ^*), if $\mathsf{dist}[Z, \Phi(U_\ell)] = negl(\kappa)$ for $U_\ell \leftarrow \{0,1\}^\ell$ and $\mathsf{dist}[\Phi^*(z), U_\ell(z)] = negl(\kappa)$ for any $z \in V$, where $U_\ell(z)$ is uniformly distributed over $\left\{u \mid \Phi(u) = z, u \in \{0,1\}^\ell\right\}$. Fact 1 below shows that $\Phi^*(Z)$ and U_ℓ are statistically close (see Appendix F for proof). Hence, we can sample Z using $Z = \Phi(U_\ell)$, and recover the randomness of Z by computing $\Phi^*(Z)$.

Fact 1. If Z is simulatable by (Φ, Φ^*), then $\Phi^*(Z)$ and U_ℓ are statistically close.

We now present two useful simulatable random variables. Let $q, p = 2q+1$ be two large primes and $\mathbb{G}$ be the subgroup of $\mathbb{Z}_p^*$ of order q. Note that $\mathbb{G}$ in fact is the set of all quadratic residues in $\mathbb{Z}_p^*$. We show that $u \leftarrow \mathbb{Z}_q$ and $u \leftarrow \mathbb{G}$ are both simulatable.

$u \leftarrow \mathbb{Z}_q$ is simulatable.

Algorithm Φ. To simulate $u \leftarrow \mathbb{Z}_q$, simply take $v \leftarrow \{0,1\}^{2|q|}$ and output $v \pmod q$. Denote this output by $(v)_q$. It is statistically close to uniform by Fact 2.

Fact 2. $\sum_{u \in \mathbb{Z}_q} |\Pr[(v)_q = u] - 1/q| \le 1/q$.

Algorithm $\Phi^*(a)$. For any $a \in \mathbb{Z}_q$, we need to sample $v \leftarrow \{a, a+q, \cdots, a+q\cdot(B_a-1)\}$ for $B_a = 1 + \lfloor \frac{2^{2|q|}-1-a}{q} \rfloor$ (B_a is the maximum i such that $a + q\cdot(i-1) < 2^{2|q|}$). To sample $\Phi^*(a)$, take $b \leftarrow \{0,1\}^{2|q|}$ and output $a + q\cdot(b)_{B_a}$. Notice $2^{|q|} \le B_a < 2^{|q|+1}$ and thus $|B_a| = |q|$. Similar to Fact 2, $\sum_{u=0}^{B_a-1} |\Pr[(b)_{B_a} = u] - 1/B_a| \le 1/q$.

$u \leftarrow \mathbb{G}$ is simulatable.

We first note that a simple one-one and onto mapping between $\mathbb{G}$ and $\mathbb{Z}_q$ exists (see [10]). Denote it by $|\cdot|_p : \mathbb{G} \to \mathbb{Z}_q$. Hence, sampling $u \leftarrow \mathbb{G}$ can be done by sampling $a \leftarrow \mathbb{Z}_q$ and outputting $u = |a|_p^*$, where $|a|_p^*$ is the reverse mapping of $|\cdot|_p$ (i.e., $|a|_p^*$ is defined as u that satisfies $|u|_p = a$).

Since $\mathbb{G}$ and $\mathbb{Z}_q$ are one-one correspondent, when the context is clear, we assume $u \leftarrow \mathbb{Z}_p$ and for $u \leftarrow \mathbb{G}$ are simulatable by the same (Φ, Φ^*) above. This algorithm is more efficient than Dent [14], where the latter requires $|q| \cdot \Omega(1)$ random bits while we only need $2|q|$ random bits.

2.3. Hash proof system

We introduce the hash proof system by Cramer and Shoup [12]. To cater our use, we make some changes. The projective hash family (PHF) is presented only with a hinge to a hard subset membership problem. The ϵ-universal$_2$ projective hash family (PHF) is replaced by the computational universal$_2$ PHF [26]. A new notion of α-extractability/extractability is proposed for a hard subset membership problem, in order to prove the deniability for our protocol.

(a) **Hard subset membership problem.** A hard subset membership problem essentially is a problem, in which one can efficiently sample a hard instance. More formally, a subset membership problem $\mathcal{I}$ is a collection $\{\mathcal{I}_n\}_{n\in\mathbb{N}}$, where $\mathcal{I}_n$ is a probability distribution for a random variable Λ_n that is efficiently sampleable by a polynomial time algorithm as follows.

- Generate a finite non-empty set $X_n, L_n \subseteq \{0,1\}^{poly(n)}$ s.t. $L_n \subset X_n$, and a distribution $D(L_n)$ over L_n and a distribution $D(X_n \backslash L_n)$ over $X_n \backslash L_n$.
- Generate a witness set $W_n \subseteq \{0,1\}^{poly(n)}$ and a NP-relation $R_n \subseteq X_n \times W_n$ such that $x \in L_n$ if and only if there exists $w \in W_n$ s.t. $(x,w) \in R_n$. There exists a polynomial time algorithm that samples x according to $D(L_n)$ and outputs a witness $w \in W_n$ s.t. $(x,w) \in R_n$. We use $x \overset{w}{\leftarrow} D(L_n)$ to denote this procedure and will omit w when there is no confusion. Further, there exists a polynomial time algorithm that samples x according to $D(X_n \backslash L_n)$.

Finally denote $\Lambda_n = \langle X_n, L_n, W_n, R_n, D(L_n), D(X_n \backslash L_n)\rangle$. $\mathcal{I} = \{\mathcal{I}_n\}_n$ is called a **hard subset membership problem** if $x \leftarrow D(L_n)$ and $y \leftarrow D(X_n \backslash L_n)$ are indistinguishable for $\langle X_n, L_n, W_n, R_n, D(L_n), D(X_n \backslash L_n)\rangle \leftarrow \mathcal{I}_n$.

(b) **α-extractability and extractability.**

α-extractability. Now we introduce a new notion of α-extractability for $\mathcal{I}$. A hard subset membership problem $\mathcal{I} = \{\mathcal{I}_n\}_n$ is called α-extractable if for any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{A}^*$ such that the following experiment terminates with 1 only negligibly.

Challenger $\mathbb{T}$ takes $\Lambda = \langle X, L, W, R, D(L), D(X\backslash L)\rangle \leftarrow \mathcal{I}_n$. Let $desc(\Lambda)$ be the description of Λ; $\alpha : \mathcal{K} \to S$ is a deterministic function with sets $\mathcal{K}$ and S specified using $desc(\Lambda)$. $\mathbb{T}$ samples $r, r^* \leftarrow \{0,1\}^*$ as a random tape for $\mathcal{A}$ and $\mathcal{A}^*$, respectively. He runs $\mathcal{A}$ with $(desc(\Lambda), desc(\alpha), r)$ and $\mathcal{A}^*$ with $(desc(\Lambda), desc(\alpha), r, r^*)$, where $desc(\alpha)$ is the description of α. Let $\Upsilon = \{\}$. The experiment is described as follows.

- *$\mathcal{A}$ can issue query $x \in X$ to $\mathcal{A}^*$. $\mathcal{A}^*$ responds with some $w \in W \cup \{\perp\}$ to $\mathcal{A}$. If $x \in L$ but $(x,w) \notin R$, then the experiment terminates with 1; otherwise, it continues.*
- *$\mathcal{A}$ can issue α-query to $\mathbb{T}$. Upon this, $\mathbb{T}$ takes $k \leftarrow \mathcal{K}$ and provides $\alpha(k)$ to both $\mathcal{A}$ and $\mathcal{A}^*$. He then updates $\Upsilon = \Upsilon \cup \{(k, \alpha(k))\}$.*
- *$\mathcal{A}$ can issue α-reverse-query D to $\mathbb{T}$. Upon this, if $(k, D) \in \Upsilon$ for some k, then $\mathbb{T}$ provides (k, D) to both $\mathcal{A}$ and $\mathcal{A}^*$; otherwise, he ignores the query.*

If $\mathcal{A}$ halts before the experiment terminates, the experiment outputs 0.

Extractability. We also consider the *plain extractability* (or simply *extractability*) without involving α function. The notion of *extractability* is defined by removing α-queries and α-reverse-queries from the α-extractability. Of course, under this change, the input $desc(\alpha)$ to $\mathcal{A}$ and $\mathcal{A}^*$ is no longer needed.

(c) **Projective hash functions.** Let $\Lambda = \langle X, L, W, R, D(L), D(X\backslash L)\rangle$ be sampled from a subset membership problem $\mathcal{I}_n$. Consider $\langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha\rangle$ that is described by $desc(\Lambda)$ and $\lambda \leftarrow \{0,1\}^n$, where $G, S, \mathcal{K}$ are finite, non-empty sets, $\mathcal{H} = \{H_k \mid k \in \mathcal{K}\}$ is a set of hash functions from X to G and $\alpha : \mathcal{K} \to S$ is a deterministic function. Here $\mathcal{K}$ is called a *key space*, $k \in \mathcal{K}$ is called a *projection key*; S is called the *projection space* of α. We call $\langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha\rangle$ a projective hash family (PHF) for Λ, if $H_k(x)$ for $x \in L$, is uniquely determined by $\alpha(k)$ and x. We call it an efficient PHF, if $\alpha(k)$ and $H_k(x)$ are polynomially computable from (k, x) and $H_k(x)$ also is polynomially computable from $(x, w, \alpha(k))$ for $(x, w) \in R$. In this paper, by PHF, we mean an efficient PHF.

Definition 1. Let $\mathcal{I} = \{I_n\}_n$ be a hard subset membership problem. Take

$$\Lambda = \langle X, L, W, R, D(L), D(X\backslash L)\rangle \leftarrow \mathcal{I}_n.$$

Assume $\Pi = \langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha\rangle$ is a PHF for Λ with $desc(\Pi) = (\lambda, desc(\Lambda))$ for $\lambda \leftarrow \{0,1\}^n$. Then, Π is said **computational universal$_2$** if any PPT $\mathcal{A}$ only has a negligible advantage in the following game. For $k \leftarrow \mathcal{K}$, run $\mathcal{A}$ with input $(\lambda, desc(\Lambda), \alpha(k))$.

- $\mathcal{A}$ can adaptively issue an **Evalu** query with any $x \in X$, where **Evalu** oracle first checks if $x \in L$ (maybe in exponential time). If yes, it returns $H_k(x)$; otherwise, it returns $\perp$.
- Throughout the game, $\mathcal{A}$ can come up with two distinct $x_1, x_2 \in X\backslash L$ (unnecessarily at the same time or in the same order). For query x_1, he receives $H_k(x_1)$; for query x_2, he receives K_b, where $b \leftarrow \{0,1\}$, $K_0 = H_k(x_2)$ and $K_1 \leftarrow \mathcal{K}$.

At the end of game, $\mathcal{A}$ outputs a guess bit b' for b. He succeeds if $b' = b$.

Remark. In the above definition, after queries x_1 and x_2, $\mathcal{A}$ can still query any x to **Evalu** oracle. There is no restriction on x (especially, $x = x_1$ or x_2 is allowed). But since $x_1, x_2 \notin L$, queries x_1, x_2 will be answered with $\perp$. We notice that in our definition, x_1 is generated by attacker, while x_1 in [26] is sampled randomly by challenger. However, we find that after this change, the result [26, Lemma 6.3] still holds except that a target collision hashing h should be modified to a collision resistent hashing. Hence, later we still use their lemma without a proof.

(d) **F-indistinguishability for a computational universal$_2$ PHF.**

Now we prove a property (called *F-indistinguishability*) for a computational universal$_2$ PHF. This property essentially asserts that the adversary can not distinguish a set of projective hash values from uniformly random values, even if he can adaptively request to evaluate $H_k(x)$ for any x as long as he knows partial information about $H_k(x)$.

Definition 2. Let $\Lambda = \langle X, L, W, R, D(L), D(X\backslash L)\rangle \leftarrow \mathcal{I}_\kappa$ where $\{\mathcal{I}_\kappa\}_\kappa$ is a hard subset membership problem. Assume $\Pi = \langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha\rangle$ is a projective hash family for Λ with $G = \{0,1\}^{2\kappa}$ and $desc(\Pi) = (\lambda, desc(\Lambda))$. Let F be a message authentication code from $\{0,1\}^*$ to $\{0,1\}^\kappa$ with a key space $\{0,1\}^\kappa$. We say that Π is **F-indistinguishable** if any PPT adversary $\mathcal{A}$ only has a negligible advantage in the following game. For $k \leftarrow \mathcal{K}$ and $c \leftarrow \{0,1\}$, challenger $\mathcal{CH}$ runs $\mathcal{A}$ with $(\alpha(k), \lambda, desc(\Lambda))$ and answers his adaptive queries as follows. Let $\Theta = \{\}$.

- **Challenge Query.** Upon this, $\mathcal{CH}$ takes $x \xleftarrow{w} D(L)$, sets $(a_0, s_0) = H_k(x)$, $(a_1, s_1) \leftarrow \{0,1\}^{2\kappa}$ and then returns (x, a_c, s_c) to $\mathcal{A}$. He updates $\Theta = \Theta \cup \{(x, a_c, s_c)\}$.
- **Compute Query** $\xi = (x, \sigma, m)$. Upon this, if $(x, a', s') \in \Theta$ for some a', s', let $a = a', s = s'$; otherwise, let $(a, s) = H_k(x)$. If $\sigma = F_a(m)$, $\mathcal{CH}$ returns (a, s); he returns $\perp$ otherwise.

At the end of the game, $\mathcal{A}$ outputs a guess bit c' for c. He is successful if $c' = c$.

The following lemma asserts that any computational universal$_2$ HPF is F-indistinguishable (see Appendix E for a proof).

Lemma 1. *Let $\{\mathcal{I}_\kappa\}_\kappa$ be a hard subset membership problem, Π be computational universal$_2$ for Λ, F be an existentially unforgeable message authentication code. Then, Π is F-indistinguishable. Further, if we use E to denote an event in Compute Query (x, σ, m), where σ is valid but $x \notin L$, then E for a given $c \in \{0,1\}$ occurs only with a negligible probability.*

3. Security model

In this section, we introduce the security model of a key exchange protocol, including a secrecy model from Bellare-Rogaway [4] and a deniability model revised from Di Raimondo-Gennaro-Krawczyk [17]. Assume there are n parties

$P_1, \cdots, P_n$. P_i and P_j might jointly execute a key exchange protocol Ξ to agree on a common secret key (called *a session key*). We will use the following notations.

- $\Pi_i^{\ell_i}$. P_i can execute many copies of Ξ (with possibly different parties). Each copy is called an *instance* or a *session*. We use $\Pi_i^{\ell_i}$ to represent the ℓ_ith instance in P_i.
- $Flow_i$. This is the ith message flow in protocol Ξ.
- $\mathsf{sid}_i^{\ell_i}$. This is the session identifier of $\Pi_i^{\ell_i}$. It will be specified when analyzing the protocol security. Supposedly, two communicating instances should share the same session identifier.
- $\mathsf{pid}_i^{\ell_i}$. It is the party that $\Pi_i^{\ell_i}$ presumably interacts with.
- $\mathsf{stat}_i^{\ell_i}$. This is the internal state of $\Pi_i^{\ell_i}$ and will be updated after each activation of $\Pi_i^{\ell_i}$. The internal state does not include the long term secret of P_i.
- $sk_i^{\ell_i}$. This is the session key defined by $\Pi_i^{\ell_i}$ after a successful execution of Ξ.
- *initiator* and *responder*. If $\Pi_i^{\ell_i}$ sends out the first message $Flow_1$, we say that P_i is an *initiator*; if $\Pi_i^{\ell_i}$ is a receiver of $Flow_1$, we say that P_i is a *responder*.

Partnering. Sessions $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ are partnered if (1) $\mathsf{pid}_i^{\ell_i} = P_j$ and $\mathsf{pid}_j^{\ell_j} = P_i$; (2) $\mathsf{sid}_i^{\ell_i} = \mathsf{sid}_j^{\ell_j}$. Intuitively, two sessions are partnered if they are jointly executing Ξ.

Adversarial Model. Now we present a formal adversary model. Essentially, we would like to capture the concern that the adversary can fully control the network and launch a concurrent attack. In particular, he can inject, modify, block and delete messages at will. He can also corrupt some users and obtain their secret keys and internal states. He is also able to collect some session keys. Finally, Ξ is secure if the session key of any adversely chosen instance remains computationally random, where of course the adversary is assumed not to compromise this session key in an obvious way (e.g., through a party corruption or a session key request).

The security model is presented in terms of a game between a challenger $\mathbb{T}$ and an attacker $\mathcal{A}$. $\mathbb{T}$ maintains a set of oracles that represent events during protocol executions. Adversarial capabilities are modeled as a sequence of adaptive queries to these oracles. The arbitrary choices of queries also imply the full concurrency of the attack. That is, $\mathcal{A}$ could simultaneously keep a polynomial number of instances running through a proper sequence of queries and can choose to activate any of them at any time. The protocol is initialized with a function I. Initially, $\mathbb{T}$ executes $I(1^\kappa)$ to sample the system parameters params and for each P_i, generates a public key PK_i and a private key SK_i. He provides params, $\{PK_i\}_i$ to $\mathcal{A}$ and maintains oracles as follows.

Send(d, i, ℓ_i, M). When this oracle is called, message M is sent to instance $\Pi_i^{\ell_i}$ as $Flow_d$. By default, when $d = 0$, it is assumed that this query is to start a new instance as an *initiator* in P_i. In this case, $M =$ "ke : j" is a key exchange invocation request with $\mathsf{pid}_i^{\ell_i} = j$. When $d = 1$, this query is to start a new instance at P_i as a *responder*. The fact that $\mathcal{A}$ triggers the oracle with $d = 0$ or 1 essentially means that the key exchange events for honest parties are completely scheduled by $\mathcal{A}$. Upon a Send query, the oracle follows the specification of Ξ to process M.

Reveal(i, ℓ_i). Upon this, it outputs $sk_i^{\ell_i}$ if $sk_i^{\ell_i}$ is defined; otherwise, it outputs $\perp$. This oracle call reflects a session key loss attack.

Corrupt(i). Upon this query, P_i is corrupted. His secret key SK_i and internal states are available to $\mathcal{A}$. Further, P_i is no longer active and his future action will be taken by $\mathcal{A}$.

Test(i, ℓ_i). This is a security test. $\mathcal{A}$ can query it only once. The queried session must have successfully completed and should not be *compromised* (see below for the definition). Upon this, the oracle flips a coin b and provides α_b to $\mathcal{A}$, where $\alpha_0 = sk_i^{\ell_i}$ and $\alpha_1 \leftarrow \mathcal{K}$ and $\mathcal{K}$ is the space of $sk_i^{\ell_i}$. $\mathcal{A}$ then tries to output a bit b'. He is informed of **success** if $b' = b$; otherwise, **fail**.

$\Pi_i^{\ell_i}$ is said **compromised** if a **Reveal** query was issued to $\Pi_i^{\ell_i}$ or its partnered session, or if P_i or $\mathsf{pid}_i^{\ell_i}$ was corrupted.

The choice of a test session is arbitrary as long as it is not compromised. Especially, it could be one session involving two honest parties. In this case, the test is to examine the session key security under a passive attack. Note that it is important not to compromise the test session; otherwise, the adversary can easily learn the test session key and win the test. Such a success does not reflect any flaw of the protocol.

Now we are ready to define the protocol security. It contains four conditions: correctness, secrecy, authentication and deniability.

Correctness. If $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ successfully complete and are partnered, then $sk_i^{\ell_i} = sk_j^{\ell_j}$.

Secrecy. Let $\mathsf{Succ}(\mathcal{A})$ denote the success of $\mathcal{A}$ in the Test query. The secrecy property requires that $\mathcal{A}$ should not be able to correctly guess b with probability significantly better than 1/2. That is, $\Pr[\mathsf{Succ}(\mathcal{A})] < \frac{1}{2} + negl(\kappa)$.

Authentication. Essentially, the authentication is to require that when $\Pi_i^{\ell_i}$ successfully completes, $\mathsf{pid}_i^{\ell_i}$ indeed attended the joint execution. Formally, consider event Non-Auth on the test session $\Pi_i^{\ell_i}$: either $\Pi_i^{\ell_i}$ does not have a partner session or its partner session is not unique. Then Ξ is said to be *authenticated* if Pr[Non-Auth($\mathcal{A}$)] is negligible. Note in [4], the authentication is defined with respect to any $\Pi_i^{\ell_i}$ (not just a test session as here). This has no essential difference from ours since the security definition requires both secrecy and authentication. Thus, if a successfully completed session does not have a partner session, then choosing it as the test session will result in breaking the authentication. This version of definition is from [28] and can simplify the proof of the authentication.

Deniability. Deniability [21] intuitively states that the adversary's view in the interaction can be simulated by himself (i.e., without an interaction), where the view of an entity consists of his random tape and the data received externally. Applying to our key exchange model, this means that for any adversary $\mathcal{A}$ with access to oracles above, there exists a simulator $\mathcal{S}$ that can simulate the view of $\mathcal{A}$ without access to these oracles. To check if the view of $\mathcal{A}$ has been simulated, we quantify it through the statistical distance between the real view of $\mathcal{A}$ and the simulated view, and require it to be negligible. However, a few concerns arise.

1. The same as $\mathcal{A}$, simulator $\mathcal{S}$ will be provided with input (params, $\{PK_i\}_i$). Evidently, it is necessary that $\mathcal{S}$ would simulate based on this public information. But be careful! $\mathcal{S}$ can simulate a new params$'$, $\{PK_i', SK_i'\}_i$, with which, the view of $\mathcal{A}$ can be perfectly simulated (by invoking $\mathcal{A}$). This certainly is not a deniable simulation. To avoid this, the variable for computing the statistical distance that quantifies the deniability, should include the (simulated) view of $\mathcal{A}$ jointly with (params, $\{PK_i\}_i$). Under this, the simulated view of $\mathcal{A}$ (by $\mathcal{S}$) uses the *provided* (params, $\{PK_i\}_i$) (from $\mathbb{T}$), as the real view of $\mathcal{A}$ does it.
2. We want to capture an eavesdropping attack in [17], where an adversary can eavesdrop the transcripts *aux* between honest parties, the randomness of which is unknown to him. Then, he can use these transcripts to attack some party's deniability. As the attacker does not know the randomness in these transcripts, the victim might be provably undeniable. To capture this concern, we provide *aux* as an auxiliary input to the attacker. Similar to the case above for (params, $\{PK_i\}_i$), we must prevent $\mathcal{S}$ from simulating the view of $\mathcal{A}$ using his own *aux*$'$ (where $\mathcal{S}$ knows the randomness in *aux*$'$). Toward this, the variables in computing the statistical distance for the deniability need to include the provided *aux*, jointly with the view of $\mathcal{A}$ and (params, $\{PK_i\}_i$). We emphasize that the model of an eavesdropping attack is meaningful only if an attacker has an external evidence that *aux* is obtained by eavesdropping (instead of by simulation). Such an evidence might be a witness or the server cache. In this paper, we do not consider how such evidences are obtained and simply assume they already exist.
3. Initially, SK_i is not provided to $\mathcal{S}$ (the same as for $\mathcal{A}$). But $\mathcal{A}$ might later issue a **Corrupt** query. To answer this query, $\mathcal{S}$ should be granted to issue **Corrupt** query as well. However, we must be careful. $\mathcal{S}$ could corrupt all parties, under which the real view of $\mathcal{A}$ can be easily simulated. This of course is not a deniable simulation. To prevent this, the variable in computing the statistical distance for deniability is expanded to include the corrupted parties $\mathcal{C}$ (recorded by $\mathbb{T}$), besides the (simulated) view of $\mathcal{A}$ and (params, $\{PK_i\}_i, aux$). Under this, the set of corrupted parties in the simulated view of $\mathcal{A}$ must equal to that recorded by $\mathbb{T}$, as this is the case for the real view of $\mathcal{A}$. Thus, $\mathcal{S}$ can only corrupt parties that $\mathcal{A}$ does.
4. Finally, as the Test oracle is only used to test the session key secrecy, there is no need to include it in the deniability model.

With the above discussion, we can now formally define the deniability. This is achieved through the following two games.

Game Γ^{rea}. $\mathbb{T}$ prepares params and $\{PK_i, SK_i\}_{i=1}^n$, *aux*, where *aux* is a collection of complete transcripts between honest parties. Then, he runs $\mathcal{A}$ with (params, $\{PK_i\}_i, aux$). $\mathbb{T}$ maintains **Send**, **Reveal** and **Corrupt** oracles. $\mathcal{A}$ can adaptively query to these oracles.

Game Γ^{sim}. $\mathbb{T}$ prepares params, $\{PK_i, SK_i\}_{i=1}^n$, *aux*, and runs $\mathcal{S}$ with (params, $\{PK_i\}_i, aux$). In addition, he maintains **Corrupt** oracle (but not **Send** and **Reveal** oracles). $\mathcal{S}$ can adaptively query to **Corrupt** oracle. Finally, $\mathcal{S}$ generates an output $\mathsf{out}(\mathcal{S}, \Gamma^{sim})$.

Let the *view* of an entity consist of his random tape and all the data received externally. Let $\mathsf{view}(\mathcal{A}, \Gamma^{rea})$ denote the view of $\mathcal{A}$ in game Γ^{rea}. By allowing $\mathcal{A}$ to adaptively query the oracles, we essentially allow many protocol instances concurrently running. Under this formulation, the deniability in our model has the full concurrency. Finally, protocol Ξ is said **deniable** if for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that

$$\mathsf{dist}[\chi|\mathcal{C}_0|\mathsf{view}(\mathcal{A}, \Gamma^{rea}), \chi|\mathcal{C}_1|\mathsf{out}(\mathcal{S}, \Gamma^{sim})] = negl(\kappa), \tag{1}$$

where $\chi = (\mathsf{params}, \{PK_i\}_i, aux)$ and $\mathcal{C}_0$ (resp. $\mathcal{C}_1$) is the set of corrupted parties by $\mathcal{A}$ in Γ^{rea} (resp. $\mathcal{S}$ in Γ^{sim}).

If Eq. (1) holds when all party corruptions are made before the game starts, then Ξ is said non-adaptively deniable. It should be noted that this non-adaptivity only corresponds to the corruption strategy. In Γ^{rea}, $\mathcal{A}$ can still adaptively query to **Send** and **Reveal** oracles. If Eq. (1) holds when *aux* is removed, then Ξ is (adaptively) deniable without an eavesdropping attack.

We summarize the security definition into the following.

Definition 3. A key exchange protocol Ξ is said *deniable secure* if for any PPT adversary $\mathcal{A}$, the following holds:

- **Correctness.**
- **Secrecy.** $\Pr[Succ(\mathcal{A})] \leq \frac{1}{2} + negl(\kappa)$.
- **Authentication.** $\Pr[Non\text{-}Auth(\mathcal{A})] = negl(\kappa)$.
- **Deniability.** For any PPT adversary $\mathcal{A}$ in Γ^{rea}, there exists a PPT simulator $\mathcal{S}$ in Γ^{sim} such that Eq. (1) holds.

Remark. Information theoretical deniability can hold forever without any computational assumption. With the time going on, any computational assumption can become insecure. This occurs especially when the assumption reaches its expiration. If the indistinguishability of deniable simulation relies on a computational assumption, then the deniability could be broken after its expiration. That is to say, information theoretical deniability is advantageous over computational one. But information theoretical deniability is not easy to achieve. In fact, to our knowledge, no existing schemes are information theoretically deniable.

Remark. As discussed in the introduction, the deniability has been studied in the literature. It is necessary to discuss the relations between our model and theirs.

1. The deniability in [17] requires that the session key can be simulated and (upon query) provided to an attacker. This requirement is captured in our model because we have a **Reveal** oracle, from which the adversary can obtain the session key of any successfully completed session. We stress that this is easy for general key exchange model (without deniability), as the simulator usually knows almost every long-term secret. But it is not easy for deniability model as a simulator does not know any honest party's long-term secret.
2. Our definition of deniability is different from [17] in the following.
   (a). We consider an auxiliary input *aux* (representing an eavesdropping attack) that is a list of properly distributed complete transcripts while *aux* in [17] can be an arbitrary string and especially can be partial transcripts. But their deniability theorems only consider a list of (complete or partial) transcripts (which is not an arbitrary string). The partial transcript *aux* is still strictly stronger than our complete transcript *aux*. In fact, under a non-adaptive corruption, our protocol in this paper is deniable in the complete transcript model but undeniable in the partial transcript model; see Remark (6) in Section 4 for details. Our restriction to a complete transcript is reasonable as a protocol execution can be finished in seconds. The scenario that an eavesdropper can not obtain a complete transcript is rare (at least it is hard for an attacker to find an external evidence to convince a judge that he has only obtained a partial transcript).
   (b). Our model requires information theoretic deniability, while [17] only requires computational deniability. Information theoretical deniability has the advantage that it can hold forever without any computational assumption.
3. In the *deniable authentication* model [15,16], Di Raimondo et al. considered the forward deniability: if the sender reveals his secret key at some moment, then the execution before this moment remains deniable. That is, the simulation for the deniability remains indistinguishable even if the distinguisher is provided with the sender's secret key. Especially, the receiver is deniable even if the sender changes his mind at some moment and tries to break the receiver's deniability. Di Raimondo et al. noticed that a computational deniability based on a computational zero-knowledge (ZK) proof is unlikely to have the forward deniability. Thus, they formalized the forward deniability as the information theoretical indistinguishability between the simulated and real adversary views. Carrying it to the key exchange, this is the deniability *without an eavesdropping attack* in our model. However, one might wish to consider the forward deniability in its basic form: the simulated adversary view is *computationally* indistinguishable from the real one even if the distinguisher is additionally given all uncorrupted parties' secret keys. We call this *computational forward deniability*. It might be interesting to make clear how much gap exists between the computational forward deniability and the information theoretical deniability. Toward this, one might wish to consider a non-adaptive or adaptive model, with or without an eraser, with or without an eavesdropping attack, etc. It is clear that in any case the information theoretical deniability implies the computational forward deniability, as an information theoretical distinguisher can obtain uncorrupted parties' long-term secrets himself. However, we are unclear whether the other direction is also true. Although it is unlikely, finding counter examples do not seem easy.
4. Dodis et al. [19] defined on-line deniability under a generalized universal compositional (GUC) model. The main feature of this model is that the distinguisher (an online Judge, which is the environment in the GUC model) can interact with an adversary in order to decide whether the protocol execution is real or simulated. A protocol is deniable if the ideal process adversary can simulate the execution with only the knowledge of the real adversary. The deniability under this model is very strong. Especially, it guarantees the deniability when the protocol runs concurrently within a large network that may share some state information (e.g., public keys or parameters) with this protocol. However, it is very strong: the adaptive deniable authentication in the PKI model provably does not exist; SKEME [17] and pRO-KE [28] are not deniable even in the non-adaptive model. The protocol proposed in this paper suffers from the same attack. Our model only guarantees a concurrent self-composable deniability (instead of in the GUC model) and hence seems weaker. However, we require information theoretical deniability while they require computational deniability. Further, the (adaptive or non-adaptive) information theoretical deniability is impossible in their model. Indeed, without loss of generality, assume that $Flow_2$ is the first message that needs the secret key of its sender. Then, a judge can send

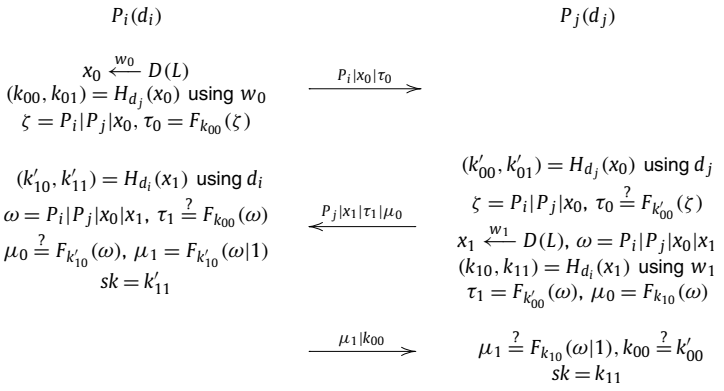

Fig. 1. Our Deniable Key Exchange Protocol HPS-KE (details in the bodytext).

$Flow_1$ to one party (through instructing the adversary to deliver the message). After receiving $Flow_2$, he just outputs $Flow_2$, his random tape and public keys. Since $Flow_2$ needs the secret key of its sender, the ideal process output is not information theoretically consistent (due to the fake $Flow_2$) while the real process output is. The reason that $\mathcal{S}$ above can not compute a real $Flow_2$ is that $\mathcal{S}$ is in fact jointly executing the protocol with the judge exactly as two honest parties in the real world. But he need to do this without any honest party's secret which is of course impossible as $Flow_2$ needs the sender's secret. In our deniability model, when $\mathcal{S}$ interacts with $\mathcal{A}$ that sends $Flow_1$, he might manage to compute $Flow_2$ without a secret. Using our protocol as example, this can be roughly justified as follows. Essentially, $\mathcal{S}$ invokes $\mathcal{A}$ with a random tape r_A and answers his oracle queries. If $Flow_1$ is sent by adversary $\mathcal{A}$ in **Send** query, then $\mathcal{S}$ can provide r_A to an extractor to extract a certain witness of $Flow_1$, with which, $Flow_2$ can be computed without any secret. Under the GUC model, $\mathcal{S}$ can not invoke $\mathcal{A}$ as $\mathcal{A}$'s attack is scheduled by the Judge, while GUC requires the security holds for any Judge (when $\mathcal{S}$ is given); see further details in remark (5) after the protocol description in Section 4. Thus, our model is not completely weaker than the GUC model.

4. Our framework for key exchange

We now present our protocol. Let $\mathcal{I} = \{\mathcal{I}_\kappa\}_\kappa$ be a hard subset membership problem. Take $\Lambda = \langle X, L, W, R, D(L), D(X\backslash L)\rangle \leftarrow \mathcal{I}_\kappa$. Assume $\wp = \langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha\rangle$ is a projective hash family for Λ, described by a random string $\lambda \leftarrow \{0,1\}^\kappa$ and $desc(\Lambda)$, where $G = \{0,1\}^{2\kappa}$. The system parameter params is defined as params $= (\lambda, desc(\Lambda))$. For each user P_i, take $d_i \leftarrow \mathcal{K}$ and compute $D_i = \alpha(d_i)$. P_i's public key is D_i and his private key is d_i. Our framework for key exchange HPS-KE between P_i and P_j is defined as follows (also see Fig. 1).

1. P_i takes $x_0 \xleftarrow{w_0} D(L)$ and computes $(k_{00}, k_{01}) = H_{d_j}(x_0)$ using the tuple (x_0, w_0, D_j). Let $\zeta = P_i|P_j|x_0$. He derives $\tau_0 = F_{k_{00}}(\zeta)$ and sends $P_i|x_0|\tau_0$ to P_j. Intuitively, τ_0 is a proof that P_i knows w_0.
2. Upon $P_i|x_0|\tau_0$, P_j computes $(k'_{00}, k'_{01}) = H_{d_j}(x_0)$ using (d_j, x_0). Let $\zeta = P_i|P_j|x_0$. He verifies $\tau_0 \stackrel{?}{=} F_{k'_{00}}(\zeta)$. If no, he rejects; otherwise, he does the following. He takes $x_1 \xleftarrow{w_1} D(L)$ and computes $(k_{10}, k_{11}) = H_{d_i}(x_1)$ using (x_1, w_1, D_i). Let $\omega = P_i|P_j|x_0|x_1$. He computes $\tau_1 = F_{k'_{00}}(\omega), \mu_0 = F_{k_{10}}(\omega)$. Finally, he sends $P_j|x_1|\tau_1|\mu_0$ to P_i.
3. Upon $P_j|x_1|\tau_1|\mu_0$, P_i computes $(k'_{10}, k'_{11}) = H_{d_i}(x_1)$ using (d_i, x_1), verifies if $\tau_1 \stackrel{?}{=} F_{k_{00}}(\omega)$ and $\mu_0 \stackrel{?}{=} F_{k'_{10}}(\omega)$ for $\omega = P_i|P_j|x_0|x_1$. If no, he rejects; otherwise, he computes $\mu_1 = F_{k'_{10}}(\omega|1)$ and sends $\mu_1|k_{00}$ to P_j. Finally, he defines $sk = k'_{11}$.
4. Upon $\mu_1|k_{00}$, if $\mu_1 = F_{k_{10}}(\omega|1)$ and $k_{00} = k'_{00}$, P_j defines $sk = k_{11}$; otherwise, he rejects.

In the protocol, we assume that $x_0, x_1 \in X$ have been checked by the respective recipient. It usually will incur a cost. In Section 7, we will see that for a class of X this check can be avoided.

Remark. Some comments are necessary.

(1) One may think of defining $sk = k_{01} \oplus k_{11}$. However, k_{01} under this definition can be obtained by an adversary $\mathcal{A}$. Indeed, let $sk = k_{01} \oplus k_{11}$ be defined with respect to a transcript $P_i|x_0|\tau_0|P_j|x_1|\tau_1|\mu_0|\mu_1|k_{00}$. Seeing this transcript, $\mathcal{A}$ can first corrupt another party P_z and then send $P_z|x_0|\tau'_0$ to P_j (say, precisely $\Pi_j^{\ell^*}$), where τ'_0 is normally computed using k_{00}. After receiving P_j's reply, $\mathcal{A}$ uses d_z to normally compute μ'_1. Let $sk_j^{\ell^*} = k_{01} \oplus k^*_{11}$. Finally, $\mathcal{A}$ asks to reveal $sk_j^{\ell^*}$. Since $\mathcal{A}$ can compute k^*_{11} using d_z, k_{01} is derived.

(2) Releasing k_{00} in $Flow_3$ is important. This provides deniability against an eavesdropping attack. Indeed, if k_{00} is removed from $Flow_3$, a (non-adaptive) deniability attacker can do the following. Given $aux = P_i|x_0|\tau_0|P_j|x_1|\tau_1|\mu_0|\mu_1$, the attacker in Γ^{rea} can issue a query **Send**$(1, j, \ell_j, P_i|x_0|\tau_0)$ and receive message $P_j|x_1'|\tau_1'|\mu_0'$ in Γ^{rea}. To be deniable, the simulator in Γ^{sim} should be able to generate this reply too. However, he is unable to do this. Indeed, he can not define $x_1' = x_1$ as this is different from the view of an attacker in Γ^{rea} (where x_1' is uniformly random in L). On the other hand, the simulator can not set x_1' different from x_1, as he does not know d_j and can not compute k_{00} and hence can not generate a valid τ_1'. So the deniability simulation fails. Besides k_{00}, we will see that τ_0 and μ_0 are also used to maintain the deniability. Essentially, they serve as a proof that the generating party knows the witness w_0 or w_1.

(3) A careful reader might think of substituting ω in computing τ_1 with $\zeta|1$ to avoid the simulation problem for $x_1' \neq x_1$ in item (2) (where we remove k_{00} from $Flow_3$), as in this case the simulator can simply define $\tau_1' = \tau_1$. However, this change will break the authentication property. Indeed, when honest P_i sends $P_i|x_0|\tau_0$ to honest P_j, P_j will reply with $P_j|x_1|\tau_1|\mu_0$. In this case, $\mathcal{A}$ can change the message of P_j to $P_j|x_1'|\tau_1|\mu_0'$, where x_1' is sampled normally with a known witness w_1' and hence he can compute a valid μ_0'. It turns out that P_i will accept $\mathcal{A}$'s message while there does not exist a partner session in P_j for P_i (because partnered sessions must see the same x_1 in order to compute the same k_{11}, no matter how session identifier is defined)!

(4) Since the release of k_{00} does not compromise the protocol security, one may think to also release k_{10}. But this will not work since in this case $\mathcal{A}$ can derive k_{11} too. Indeed, assume $\mathcal{A}$ eavesdrops $P_i|x_0|\tau_0|P_j|x_1|\tau_1|\mu_0|k_{10}|k_{00}$. Later, when P_i sends $Flow_1 = P_i|x_0'|\tau_0'$ to a corrupted party P_z, $\mathcal{A}$ can reply with $P_z|x_1|\tau_1'|\mu_0'$ normally using d_z and k_{10}. P_i then will accept. Since sk for the new instance in P_i is still k_{11}, $\mathcal{A}$ can obtain it by **Reveal** query and so the secrecy of the eavesdropped session is broken.

(5) Although later we will see that this framework is deniable in our model, it is not non-adaptively deniable under PKI in the GUC model of Dodis et al. [19] (see the end of the last section for a brief introduction of their model) since Judge can compute $Flow_1$ without d_i and then request the adversary (either $\mathcal{A}$ in Γ^{rea} or $\mathcal{S}$ in Γ^{sim}) to send it to P_j and ask him to forward $Flow_2$ back. Judge then can decide whether $Flow_2$ is simulated by checking the validity of τ_1, as no one except Judge and P_j can compute τ_1 (especially $\mathcal{S}$ can not compute it). Notice that since $Flow_2$ forwarded by $\mathcal{A}$ in Γ^{rea} has a valid τ_1, Judge can realize that he is interacting with $\mathcal{A}$, not $\mathcal{S}$. This attack is in the non-adaptive model as no corruption is made at all. In our model, there is no online judge and $Flow_1$ is generated by $\mathcal{A}$. To simulate the view of $\mathcal{A}$, $\mathcal{S}$ in Γ^{sim} can run $\mathcal{A}$ with a known random tape r_A. When $\mathcal{A}$ sends out $Flow_1$, $\mathcal{S}$ can run an extractor of L (with r_A as part of his input) to extract the witness w_0 of x_0 in $Flow_1$. With this w_0, $\mathcal{S}$ can easily generate $Flow_2$.

The above attack separates GUC and our model in the sense of deniability. One might wonder why our protocol with online judge has the simulation difficulty while it does not under our model. To see this, notice that our model says that for any $\mathcal{A}$ there exists $\mathcal{S}$. That is, $\mathcal{S}$ can depend on $\mathcal{A}$. In the GUC model, Judge can instruct a dummy real adversary how to attack and is the actual adversary. However, a deniability simulator $\mathcal{S}$ can not run Judge because GUC security model states that "for any $\mathcal{A}$, there exists simulator $\mathcal{S}$ such that for *any* Judge (environment),....". That is, a valid simulator should work for any Judge. Now if $\mathcal{S}$ runs a specific Judge, then the description of $\mathcal{S}$ should include the description of this Judge. Since this Judge can be arbitrary, $\mathcal{S}$ can not be described using a valid Turing machine.

(6) Our protocol is not deniable under the partial transcript eavesdropping attack. In the partial transcript eavesdropping model, it is possible that $aux = Flow_1|Flow_2$. Then, an attacker can send $Flow_1$ in aux to P_j as a new session, who then will generate $Flow_2'$. Since $Flow_2$ and $Flow_2'$ are correlated with the same secret key k_{00} (unknown to attacker), P_j is undeniable. Our deniability model requires that aux is the complete transcript and hence k_{00} (in $Flow_3$) is included in aux. Under this, $Flow_2'$ is simulatable and so the above attack does not work.

5. Security analysis

In the following, we analyze deniability and secrecy. We define the session id for two partnered sessions $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ to be $\mathsf{sid}_i^{\ell_i} = \mathsf{sid}_j^{\ell_j} = P_i|P_j|x_0|x_1$. We define the session state for the protocol execution as follows. For any **Send**(d, i, ℓ_i, M) query, if $\Pi_i^{\ell_i}$ rejects in its reply, then we define $\mathsf{stat}_i^{\ell_i} = \perp$ as in this case the instance terminates and nothing needs to keep. If a **Send** oracle does not reject, then $\mathsf{stat}_i^{\ell_i}$ can be updated according to different cases. Specifically, after **Send**$(0, i, \ell_i,$ "ke: j") query, define $\mathsf{stat}_i^{\ell_i} = P_i|P_j|x_0|k_{00}$; after **Send**$(1, j, \ell_j, P_i|x_0|\tau_0)$ query, define $\mathsf{stat}_j^{\ell_j} = P_i|P_j|x_0|x_1|k_{00}'|k_{10}|k_{11}$; after **Send**$(2, i, \ell_i, Flow_2)$ query, define $\mathsf{stat}_i^{\ell_i} = P_i|P_j|k_{11}'$; after **Send**$(3, j, \ell_j, Flow_3)$ query, define $\mathsf{stat}_j^{\ell_j} = P_i|P_j|k_{11}$.

5.1. Non-adaptive deniability with an eavesdropping attack

For the non-adaptive deniability with an eavesdropping attack, $\mathcal{S}$ receives $\{D_i\}_{i\notin\mathcal{C}}, \{d_i\}_{i\in\mathcal{C}}$ (for a corrupt set $\mathcal{C}$) and an auxiliary input aux. In Γ^{rea}, $\mathcal{A}$ can adaptively query **Send** and **Reveal** oracles and obtain the replies. Thus, $\mathcal{S}$ in Γ^{sim} must also be able to simulate such oracle replies. The main difficulty is that $\mathcal{S}$ does not know d_t for $t \notin \mathcal{C}$ and hence can not compute τ_1 in **Send**$(1, t, \ell_t, P_i|x_0|\tau_0)$ oracle or μ_1 in **Send**$(2, t, \ell_t, P_j|x_1|\tau_1|\mu_0)$ oracle. To resolve the problem, we design

$\mathcal{S}$ to consist of two algorithms: an extractability adversary $\mathcal{E}$ for L and its extractor $\mathcal{E}^*$. The actual oracle simulation is actually done by $\mathcal{E}$. Specifically, $\mathcal{E}$ invokes $\mathcal{A}$ and tries to answer his oracle queries. Whenever $\mathcal{E}$ needs to compute $H_{d_t}(x)$, he requests $\mathcal{E}^*$ to extract the witness w for $x \in L$, with which, $H_{d_t}(x)$ can be easily computed. But there is a syntax issue here. On the one hand, as a simulator for $\mathcal{A}$, $\mathcal{E}$ needs to know $\{D_i\}_{i\notin\mathcal{C}}, \{d_i\}_{i\in\mathcal{C}}$, aux and $(\lambda, desc(\Lambda))$. On the other hand, as an extractability adversary for L, he can only get an input $desc(\Lambda)$ and random tape r_E. To resolve this issue, we encode $\{D_i\}_{i\notin\mathcal{C}}, \{d_i\}_{i\in\mathcal{C}}, aux, \lambda$ into a pseudorandom string and feed to $\mathcal{E}$ as part of its random tape. Further, since $\mathcal{E}^*$ will receive r_E as part of his input, this also resolves the similar issue for $\mathcal{E}^*$. Now we slightly detail the idea for $\mathcal{E}$ to simulate **Send** and **Reveal** oracles without $\{d_t\}_{t\notin\mathcal{C}}$. This is outlined as follows.

Notice that d_j for $j \notin \mathcal{C}$ is only used in computing $(k_{u0}, k_{u1}) = H_{d_j}(x_u)$ in **Send** oracle for $u = 0, 1$. Specifically, when $u = 0$, $\mathcal{E}$ needs to verify τ_0 and compute τ_1 in **Send**$(1, j, \ell_j, P_i|x_0|\tau_0)$ oracle. If k_{00} appears in aux, then the computation is easy. Otherwise, if $x_0 \in L$, then $H_{d_j}(x_0)$ can be computed (without d_j) using the witness w_0 of x_0. $\mathcal{E}$ can obtain w_0 by sending x_0 to $\mathcal{E}^*$ as the extraction query. However, the simulation is valid only if $\mathcal{E}^*$ can always extract the witness w_0 when $x_0 \in L$. But this is not always true. In our design of $\mathcal{E}$, if $\mathcal{E}^*$ fails to extract w_0, then $\mathcal{E}$ simply rejects τ_0. Of course, this decision might be wrong. We denote this event by Bad. We will show through a sequence of game technique that $P(\mathsf{Bad})$ is negligible. Intuitively, if $x_0 \notin L$, then τ_0 is provably invalid by reducing to the F-indistinguishability of H and hence τ_0 can be safely rejected. The case $u = 1$ is similar. Finally, simulating the **Reveal** oracle is easy as in this case the session key has already been defined through some **Send** oracles. This completes our deniability idea. We now state the deniability theorem; see Appendix B for a formal proof.

Theorem 1. *Assume $\mathcal{I} = \{\mathcal{I}_\kappa\}_\kappa$ is an extractable hard subset membership problem. Let $\Lambda = \langle X, L, W, R, D(L), D(X\backslash L)\rangle \leftarrow \mathcal{I}_\kappa$. Let $\wp = \langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha\rangle$ be a computational universal$_2$ PHF for Λ, where $G = \{0,1\}^{2\kappa}$ and $desc(\wp) = (\lambda, desc(\Lambda))$ for $\lambda \leftarrow \{0,1\}^\kappa$. Let F be a pseudorandom function family from $\{0,1\}^*$ to $\{0,1\}^\kappa$ with key space $\{0,1\}^\kappa$. Assume $x \leftarrow D(X)$ is simulatable by (Φ_0, Φ_0^*) and $D \leftarrow S$ is simulatable by (Φ_1, Φ_1^*) and $d \leftarrow \mathcal{K}$ is simulatable by (Φ_2, Φ_2^*). Then, HPS-KE is non-adaptively deniable with an eavesdropping attack.*

5.2. Adaptive deniability without an eavesdropping attack

We outline the idea of the adaptive deniability without an eavesdropping attack. Similar to the non-adaptive deniability above, we still design a deniability simulator $\mathcal{S}$ to consist of two algorithms: α-extractability adversary $\mathcal{E}$ and its extractor $\mathcal{E}^*$. As an eavesdropping attack is removed, the task to pack aux into the random tape of $\mathcal{E}$ is no longer needed. The input to $\mathcal{S}$ is $\mathcal{PK}, \lambda, desc(\Lambda)$. Now $\mathcal{E}$ can receive $(\lambda, desc(\Lambda))$ as his input and r_E as his random tape. Further, $\mathcal{E}$ can obtain $PK_1, \cdots, PK_n$ through α-queries. In the oracle simulation, $\mathcal{E}$ can obtain d_i through an α-reverse-query (if $\mathcal{A}$ corrupts P_i). For **Send** oracles, $\mathcal{E}$ can still use the strategy in the non-adaptive case. Mainly, he needs to verify τ_0 and compute τ_1 in **Send**$(1, j, \ell_j, P_i|x_0|\tau_0)$ oracle and to verify μ_0 and compute μ_1 in **Send**$(2, i, \ell_i, P_j|x_1|\tau_1|\mu_0)$ oracle. W.O.L.G., consider **Send**$(1, \cdot)$ oracle only. In this case, $\mathcal{E}$ can send x_0 to $\mathcal{E}^*$ and in turn receive the witness w_0 of x_0. Now since $\mathcal{E}$ is a legal extractability adversary, $\mathcal{E}^*$ fails to extract a witness w_0 for $x_0 \in L$ only with a negligible probability (which we ignore). Thus, an invalid witness w_0 (returned by $\mathcal{E}^*$) implies that $x_0 \notin L$. On the other hand, when $x_0 \notin L$, τ_0 is valid if and only if an event E occurs in a F-indistinguishability game, which is negligible. Thus, $\mathcal{S}$ can safely reject τ_0 whenever w_0 from $\mathcal{E}^*$ is invalid. Therefore, **Send**$(1, \cdots)$ query can be smoothly handled. **Send**$(2, \cdot)$ can be answered similarly. This completes our simulation idea. We state the result formally in the following theorem (see a detailed proof in Appendix C).

Theorem 2. *Let $\mathcal{I} = \{\mathcal{I}_\kappa\}_\kappa$ be an α-extractable hard subset membership problem and $\Lambda = \langle X, L, W, R, D(L), D(X\backslash L)\rangle \leftarrow \mathcal{I}_\kappa$. Let $\wp = \langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha\rangle$ be a computational universal$_2$ PHF for Λ with $G = \{0,1\}^{2\kappa}$. Assume F is a pseudorandom function from $\{0,1\}^*$ to $\{0,1\}^\kappa$, with key space $\{0,1\}^\kappa$. Then, HPS-KE is adaptively deniable without an eavesdropping attack.*

5.3. On the possibility for adaptive deniability with an eavesdropping attack

We have proved that our framework is non-adaptively deniable with an eavesdropping attack and adaptively deniable without an eavesdropping attack. An immediate question is whether our protocol is adaptively deniable with an eavesdropping attack. Especially, one may think that the proof for the adaptive deniability without an eavesdropping attack can be extended to this setting. However, this seems difficult. Essentially, this is because our strategy for the deniability theorems uses the extractability assumption of L to extract the witness w_0 in x_0 (or w_1 in x_1), in order for the simulator to simulate **Send** oracles without an uncorrupted party's private key. To make use of this extractability, we have to construct an adversary $\mathcal{E}$ in the extractability game who sends x_0 (or x_1) to an extractor $\mathcal{E}^*$ to extract w_0 (or w_1). Since $\mathcal{E}$ is internally interacting with $\mathcal{A}$, he especially needs to know the input of Γ^{rea}, which is $\lambda, desc(\Lambda), \{PK_i\}_i$ and aux. However, as an extractability adversary, the input of $\mathcal{E}$ is $(\lambda, desc(\Lambda))$ and a random tape r_E. In addition, $\mathcal{E}$ can obtain $PK_1, \cdots, PK_n$ through a sequence of α-queries. However, providing aux to $\mathcal{E}$ is not easy. If $\mathcal{A}$ is non-adaptive, then $\mathcal{S}$ can encode aux into a pseudorandom binary string and provide to $\mathcal{E}$ as a part of his random tape r_E. Since $\mathcal{A}$ has a non-adaptive corruption, then this encoded pseudorandom string remains pseudorandom throughout the simulation. For adaptive corruption, the situation is different. If party i is corrupted later, then the encoded aux is no longer pseudorandom. Specifically, as in Theorem 2, at the

beginning, *aux* is compressed into a list of $x_0|x_1|\mu_0|\mu_1|k_{00}$ and then encoded into pseudorandom strings and feeded to $\mathcal{E}$ as part of his random tape. Assume $x_0|x_1|\mu_0|\mu_1|k_{00}$ corresponds to the record between P_i, P_j. If later $\mathcal{A}$ corrupts P_i, then d_i will be provided to $\mathcal{E}$ (as the α-reverse query). However, given d_i, $x_0|x_1|\mu_0|\mu_1|k_{00}$ is no longer pseudorandom as μ_0 is efficiently computable from $d_0, x_0|x_1|P_i|P_j$. It follows that $\mathcal{E}$ is no longer a legal α-extractability adversary. Consequently, the performance of $\mathcal{E}^*$ can not be guaranteed. That is to say, we can not use our old strategy. It is very interesting if we can find a different proof strategy (maybe under a new assumption) so that our protocol is adaptively deniable with the eavesdropping attack.

5.4. Secrecy

Now we outline the secrecy idea of HPS-KE. If the test session is Π_z^ℓ, we need to show that sk_z^ℓ is indistinguishable from a random number. We only consider the case where Π_z^ℓ is an initiator (the responder case is similar). Assume $\mathsf{pid}_z^\ell = P_{z^*}$. In the nutshell, our proof relies on the F-indistinguishability of the projective hash family $\wp$. With this property, we can replace (k_{00}, k_{01}) in *any* $\Pi_z^{\ell_z}$ with $\mathsf{pid}_z^{\ell_z} = P_{z^*}$ by a random number and replace (k_{10}, k_{11}) in *any* $\Pi_{z^*}^{\ell_z^*}$ with $\mathsf{pid}_{z^*}^{\ell_z^*} = P_z$ (i.e., the party under the test) by a random number.

Then, we show that the test session Π_z^ℓ has a unique partner session $\Pi_{z^*}^{\ell^*}$ (so they have the same view on $P_z|P_{z^*}|x_0|x_1$). The idea of this uniqueness is as follows. Firstly, there is at most one such $P_{z^*}^{\ell^*}$ since otherwise two instances in P_{z^*} sample the same x_1(negligible!). Secondly, such $\Pi_{z^*}^{\ell^*}$ exists; otherwise, as the attacker knows neither w_0 (in x_0) nor d_i, he can not create a valid τ_1 and hence Π_z^ℓ will reject $Flow_2$ and can not be chosen as a test session. So we can assume that Π_z^ℓ has the same view on $k_{00}^*|k_{01}^*|k_{10}^*|k_{11}^*|x_0^*|x_1^*$ as his unique partner $\Pi_{z^*}^{\ell^*}$. To prove the theorem, it suffices to prove that no instance other than Π_z^ℓ and $\Pi_{z^*}^{\ell^*}$ defines a session key as k_{11}^*. Since k_{11}^* is modified as uniformly random, any $\Pi_i^{\ell_i}$ defines a session key k_{11}^* only if (1) $i = z$ and $\Pi_i^{\ell_i}$ (initiator) accepts $Flow_2$ that contains x_1^*, or (2) $\Pi_i^{\ell_i}$ (responder) samples $x_1 = x_1^*$ in $Flow_2$. (2) is impossible since it samples the same x_1^* as $\Pi_{z^*}^{\ell^*}$ (negligible!). (1) is impossible since k_{10}^* is random and hence μ_0 in $Flow_2$ will be rejected by $\Pi_z^{\ell_z}$. As a summery, k_{11}^* is a session key only for Π_z^ℓ and $\Pi_z^{\ell^*}$ and hence cannot be revealed. So an adversary can not distinguish it. The detailed proof is put in Appendix D.

Theorem 3. *Let F be pseudorandom from $\{0,1\}^*$ to $\{0,1\}^\kappa$, $\mathcal{I}$ be a hard subset membership problem and $\wp$ be computational universal$_2$ PHF. Then, HPS-KE satisfies the adaptive secrecy.*

5.5. Authentication

Theorem 4. *Let F be pseudorandom from $\{0,1\}^*$ to $\{0,1\}^\kappa$, $\mathcal{I}$ be a hard subset membership problem and $\wp$ be computational universal$_2$ PHF. Then,* $\Pr[\text{Non-Auth}(\mathcal{A})]$ *is negligible.*

Proof. We regard the secrecy game Γ_0 as an experiment. It outputs 1 if Non-Auth occurs. We summarize the proven results in proofs of Lemmas in Theorem 3. $\mathsf{view}(\mathcal{A}, \Gamma_i)$ and $\mathsf{view}(\mathcal{A}, \Gamma_{i+1})$ for $i = 2, 3$ are negligibly close and thus Non-Auth events in each pair of them are negligibly close too. Non-Auth differs in Γ_1 and Γ_2 only if it occurs after $\langle P_z$ or P_{z^*} is corrupted or Π_z^ℓ is revealed$\rangle$ and before the Test query. However, Non-Auth is defined only for the Test session (see the authentication model in Section 3). So it can only occur after the test session is chosen. So $\Pr[\text{Non-Auth}(\Gamma_1)] = \Pr[\text{Non-Auth}(\Gamma_2)]$. Further, by Lemma 2 (Eq. (A.1)), $\Pr[\text{Non-Auth}(\mathcal{A}, \Gamma_0)]$ and $\Pr[\text{Non-Auth}(\mathcal{A}, \Gamma_1)]$ are either both negligible or both non-negligible. Finally, by Lemma 12, $\Pr[\text{Non-Auth}(\mathcal{A}, \Gamma_4)]$ is negligible. Therefore, $\Pr[\text{Non-Auth}(\mathcal{A}, \Gamma_0)] = negl(\kappa)$. □

6. Concrete schemes

We can realize our key exchange protocol using a hash proof system [35,12].

- **Description of $\mathcal{I}_k$.** Sample a large prime $p = 2q + 1$ where q is also a large prime. Let $\mathbb{G}$ be the prime group of $\mathbb{Z}_p^*$ of order q. Take $g_1, g_2 \leftarrow \mathbb{G}$. The set $X = \{(g_1^{r_1}, g_2^{r_2}) \mid r_1, r_2 \in \mathbb{Z}_q\}$. Language L is defined as $L = \{(g_1^r, g_2^r) \mid r \in \mathbb{Z}_q\}$. $D(L)$ is defined as taking $r \leftarrow \mathbb{Z}_q$ and outputting (g_1^r, g_2^r). Similarly define $D(X\backslash L)$. $\mathcal{I}$ is a hard subset membership problem by the DDH assumption in $\mathbb{G}$.
- **Description of $\wp$.** Let $S = \mathbb{G}$ and $G = \{0,1\}^{2\kappa}$. Let key space $\mathcal{K} = \{(a_1, a_2, b_1, b_2) \mid a_1, a_2, b_1, b_2 \in \mathbb{Z}_q\}$. $D = \alpha(d) = (D_1, D_2) = (g_1^{a_1} g_2^{a_2}, g_1^{b_1} g_2^{b_2})$ for $d = (a_1, a_2, b_1, b_2) \in \mathcal{K}$. Let h_λ be a collision resistant hash function, indexed by $\lambda \leftarrow \{0,1\}^\kappa$. For $(u_1, u_2) \in X$, define $H_d(u_1, u_2) = \mathsf{KDF}(u_1^{a_1 + b_1\tau} u_2^{a_2 + b_2\tau})$, where $\tau = h_\lambda(u_1, u_2)$. If $(u_1, u_2) = (g_1^r, g_2^r)$, then

$$\begin{aligned} H_d(u_1, u_2) &= \mathsf{KDF}(u_1^{a_1 + b_1\tau} u_2^{a_2 + b_2\tau}) \\ &= \mathsf{KDF}((g_1^{a_1 + b_1\tau} g_2^{a_2 + b_2\tau})^r) = \mathsf{KDF}((D_1 D_2^\tau)^r), \end{aligned}$$

where KDF is a key derivation function (e.g., the least half bits of the input) and is not used in the original HPS [35,12]. This is a projective hash family for $\mathcal{I}$. Further, it is computational universal$_2$ [26, Lemma 6.3]. $desc(\Lambda) = (p, g_1, g_2)$. Based on the DHK assumption, $\mathcal{I}$ is an extractable hard subset membership problem. $desc(\wp) = (\lambda, p, g_1, g_2)$. Besides, D_1, D_2 and $x \leftarrow \mathbb{Z}_q$ are simulatable (see Dent [14] or a more efficient algorithm in Section 2.2).

Denote a HPS-KE protocol using the above HPS by DHK-KE. We have

Corollary 1. *Assume that h_λ is collision-resistant and F is pseudorandom. Under* DDH *and* DHK *assumptions, DHK-KE is non-adaptively deniable with an eavesdropping attack, adaptive secrecy and authenticated.*

Notice that the α-extractability assumption for HPS in DHK-KE is actually the DHK* assumption. We therefore have the following result.

Corollary 2. *Assume h_λ is collision-resistant and F is pseudorandom. Then, under* DDH *and* DHK* *assumptions, DHK-KE is adaptively deniable without an eavesdropping attack, adaptive secrecy and authenticated.*

Remark. We can realize our framework using HPS [26] from a n-linear assumption. Let $g_1, \cdots, g_n, h$ be generators of prime group $\mathbb{G}$ of order q in $\mathbb{Z}_p$ for $p = 2q + 1$. Then a n-linear assumption is to say that $(g_1^{r_1}, \cdots, g_n^{r_n}, h^{r_1+\cdots+r_n})$ is indistinguishable from $(g_1^{r_1}, \cdots, g_n^{r_n}, K)$ for $K \leftarrow \mathbb{G}, r_i \leftarrow \mathbb{Z}_q$. Define $X = \{(g_1^{r_1}, \cdots, g_n^{r_n}, h^r) \mid r_i, r \in \mathbb{Z}_q\}$. By the n-linear assumption, a language L for X can be defined as a subset of X where $r = r_1 + \cdots + r_n$. To guarantee this HPS has an extraction property, we need to introduce a new assumption, called a **n-linear knowledge assumption**, which essentially states that, given $(g_1, \cdots, g_n, h)$, if an adversary comes up with $(g_1^{r_1}, \cdots, g_n^{r_n}, h^{r_1+\cdots+r_n})$, then there exists an extractor to extract $(r_1, \cdots, r_n)$. When $n = 1$, this is the DHK assumption. It is easy to prove that a n-linear knowledge assumption implies a $(n-1)$-linear knowledge assumption. The formal definition can be obtained similarly as for the DHK assumption. Similar to DHK*, we can also define the **n-linear* knowledge assumption**. Again, a n-linear* knowledge assumption implies a $(n-1)$-linear* knowledge assumption. Based on these assumptions, we can realize our framework with a n-linear assumption based HPS [26]. The resulting protocol is non-adaptively deniable with an eavesdropping attack if the n-linear knowledge assumption is assumed; it is adaptively deniable without an eavesdropping attack if a n-linear* knowledge assumption is assumed. Details are omitted.

7. Improving DHK-KE: how to avoid checking group membership

A valid membership of prime group $\mathbb{G}$ of order q in $\mathbb{Z}_p^*$ is important to maintain the security for many protocols [31,34]. To check whether $a \in \mathbb{G}$, we usually verify if $a^q = 1 \pmod p$. This adds a price of one exponentiation to the protocol for the verification of each element and significantly degrades the protocol efficiency. Now we show how to avoid this using a price of just one squaring. The idea is simple: whenever we need to compute and send an element $X = g^x$ for a *known* x, we instead first compute $x' = x/2$, compute $X' = g^{x'}, X = X'^2 \pmod p$, process normally for the remaining computation but finally send X' (instead of X). Receiving X', the receiver first computes $X = X'^2 \pmod p$ and then proceeds normally. Now we use DHK-KE as an example to show how this works.

The system setup is as in DHK-KE. Let $(2^{-1})_q := 2^{-1} \pmod q = (q+1)/2$. P_i has public key $D_i = (D_{i1}, D_{i2}) = (g_1^{a_{i1}} g_2^{a_{i2}}, g_1^{b_{i1}} g_2^{b_{i2}})$ and private key $d_i = (a_{i1}, a_{i2}, b_{i2}, b_{i2})$. System parameter params $= (g_1, g_2, \lambda, p)$. Then the key exchange between P_i and P_j is as follows.

1. P_i takes $x \leftarrow \mathbb{Z}_q$, computes
   $x' = x \cdot (2^{-1})_q,\ c_1' = g_1^{x'},\ c_2' = g_2^{x'},\ c_1 = {c_1'}^2,\ c_2 = {c_2'}^2,\ \zeta = P_i|P_j|c_1|c_2$
   $\sigma_0 = h_\lambda(\zeta),\ (k_{00}, k_{01}) = \mathsf{KDF}(D_{j1}^x D_{j2}^{\sigma_0 x}),\ \tau_0 = F_{k_{00}}(\zeta)$.
   Finally he sends to P_j message $P_i|c_1'|c_2'|\tau_0$. Essentially, instead of sending $P_i|c_1|c_2|\tau_0$, P_i sends $P_i|\sqrt{c_1}|\sqrt{c_2}|\tau_0$ to P_j.
2. Receiving $P_i|c_1'|c_2'|\tau_0$, P_j computes $c_1 = {c_1'}^2, c_2 = {c_2'}^2, \zeta = P_i|P_j|c_1|c_2$, $\sigma_0 = H_\lambda(\zeta)$ and $(k_{00}', k_{01}') = \mathsf{KDF}(c_1^{a_{j1}+\sigma_0 b_{j1}} c_2^{a_{j2}+\sigma_0 b_{j2}})$. Verify whether $\tau_0 = F_{k_{00}'}(\zeta)$. If valid, he takes $y \leftarrow \mathbb{Z}_q$, computes

$$y' = y \cdot (2^{-1})_q, \quad f_1' = g_1^{y'}, \quad f_2' = g_2^{y'}, \quad f_1 = {f_1'}^2, \quad f_2 = {f_2'}^2,$$
$$\omega = P_i|P_j|f_1|f_2|c_1|c_2, \quad \sigma_1 = h_\lambda(\omega|1),$$
$$(k_{10}, k_{11}) = \mathsf{KDF}(D_{i1}^y D_{i2}^{\sigma_1 y}), \quad \tau_1 = F_{k_{00}'}(\omega|1), \quad \mu_0 = F_{k_{10}}(\omega).$$

   Finally, send $P_j|f_1'|f_2'|\tau_1|\mu_0$ to P_i. Essentially, P_j sends to P_i message $P_j|\sqrt{f_1}|\sqrt{f_2}|\tau_1|\mu_0$.

3. Upon $P_j|f_1'|f_2'|\tau_1|\mu_0$, P_i computes

$$f_1 = f_1'^2, \quad f_2 = f_2'^2, \quad \omega = P_i|P_j|f_1|f_2|c_1|c_2,$$
$$\sigma_1 = h_\lambda(\omega|1), \quad (k'_{10}, k'_{11}) = \mathsf{KDF}(f_1^{a_{i1}+\sigma_1 b_{i1}} f_2^{a_{i2}+\sigma_1 b_{i2}}),$$

verifies whether $\tau_1 = F_{k_{00}}(\omega|1)$ and $\mu_0 = F_{k'_{10}}(\omega)$. If both are valid, send $\mu_1 = F_{k'_{10}}(\omega|1)$ and k'_{00} to P_j and define session key $sk = k'_{11}$.
4. Verify whether $\mu_1 = F_{k_{10}}(\omega|1)$. If valid, define session key $sk = k_{11}$; otherwise, reject.

Denote the modified protocol DHK-KE′. Notice that for any $z = 2z_1 + z_0$ for $z_0 \in \{0,1\}$, $z \times (2^{-1})_q = z_1 + z_0 \cdot \frac{q+1}{2}$. So in contrast to DHK-KE, each party in DHK-KE′ uses the cost of 4 squaring to avoid 2 group membership verifications. The following theorem states that they are equivalent in all security properties.

Theorem 5. *Let h_λ be collision resistent and F be pseudorandom. Then under* DHK *(resp.* DHK*) and* DDH *assumptions,* DHK-KE′ *is non-deniable (resp. adaptive deniable) and adaptively secret and authenticated.*

Proof. For any $\mathcal{A}'$ against DHK-KE′, we construct an attacker $\mathcal{A}$ against DHK-KE with the same success probability. Note for any $X \in \mathbb{G}$, $\sqrt{X} = X^{(q+1)/2}$. The strategy of $\mathcal{A}$ is to forward the query from $\mathcal{A}'$ to his own challenger and relay the reply from the latter back to $\mathcal{A}'$, except

- In **Send**$(0, i, \ell_i, \cdot)$ oracle, $P_i|c_1|c_2|\tau_0$ is changed to $P_i|\sqrt{c_1}|\sqrt{c_2}|\tau_0$ before forwarding to $\mathcal{A}'$.
- A similar change is made to **Send**$(1, \cdot)$ oracle.

Since the view of $\mathcal{A}'$ is identical to the real execution, $\mathcal{A}'$'s breaking deniability or secrecy or authentication in DHK-KE′ implies $\mathcal{A}$'s breaking the respective property in DHK-KE. By Theorems 1–4, the conclusion follows. □

Efficiency. Each party in DHK-KE′ protocol needs 6 exponentiations, 4 squarings, 4 F values, 2 hashes and 2 KDFs. Computation of KDF, F and hashes can be negligible. For instance, take the least 2κ bits for KDF; use SHA-1 for hash; use a collision-resistant hashing followed by super pseudorandom permutation (e.g., use AES) for F or alternatively use GGM [23] with pseudorandom generator [41] or simply RFC 4402 [44]. So DHK-KE′ needs essentially 6 exps for each party. In contrast, SKEME using Kurosawa-Desmedt [35], if using the technique here to avoid a group membership check, has 8 exps for each party, $2F$, 2 MACs, 2 hashes and 8 squarings.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank the anonymous referee for valuable comments that improve the deniability definition and our understanding on the deniability and help us simplify the proofs of Theorems 1 and 2. We also thank Hugo Krawczyk for confirming the pseudorandomness of the authentication function for the deniability of SKEME [17,32].

Appendix A. Abortion lemma

Let Exp be an experiment with output $o \in \{0,1\}$. Assume that $\mathsf{E}_1, \cdots, \mathsf{E}_n$ are n *disjoint* events that might occur in Exp and $\Pr[\vee_{i=1}^n \mathsf{E}_i] = 1$.

Let Exp* be a variant of Exp: take $i \leftarrow \{1, \cdots, n\}$ and then run Exp. If E_j occurs in Exp for some $j \neq i$, abort with output 0, 1 randomly; otherwise, normally execute Exp and finally outputs whatever it does.

Let X be a binary random variable defined in Exp. Define a random variable X^* in Exp* as follows. If X is well-defined before the abortion of Exp in Exp*, $X^* \stackrel{def}{=} X$; otherwise, $X^* \stackrel{def}{=} 0$.

Lemma 2. *Assume that X is a binary random variable in* Exp *and o(*Exp**) (resp. o(*Exp*)) is the output of* Exp* *(resp.* Exp*). Then,*

$$\Pr[X^* = 1] \leq \Pr[X = 1] \leq n \Pr[X^* = 1], \tag{A.1}$$
$$\Pr[o(\mathsf{Exp}) = 1] - 1/2 = n(\Pr[o(\mathsf{Exp}^*) = 1] - 1/2). \tag{A.2}$$

Proof. Since $\Pr[\vee_k \mathsf{E}_k] = 1$ and $\mathsf{E}_1, \cdots, \mathsf{E}_n$ are disjoint, each instance of Exp will have exactly one E_j occurring. If $i = j$, then $X^* = X$ as Exp in Exp* never aborts in this case. Thus, $\Pr[X^* = 1 \wedge i = j] = \Pr[X = 1 \wedge i = j]$. Hence,

$$\begin{aligned}\Pr[X^* = 1] &= \Pr[X^* = 1 \wedge i = j] + \Pr[X^* = 1 \wedge i \neq j] \\ &= \Pr[X = 1 \wedge i = j] + \Pr[X^* = 1 \wedge i \neq j]. \end{aligned} \tag{A.3}$$

By definition of X^*, $X^* = 1$ when $i \neq j$, only if X is realized with 1 before E_j occurs (denoted by event Good). Thus,

$$Eq.\ (A.3) = \Pr[X = 1 \wedge i = j] + \Pr[X = 1 \wedge i \neq j \wedge \mathsf{Good}]. \tag{A.4}$$

Thus, Eq. (A.4) implies that $\Pr[X^* = 1] \leq \Pr[X = 1]$. In addition, notice that i and Exp are independent. So i is independent of (j, X). Thus, $\Pr[i = j] = \frac{1}{n}$ and $\Pr[X = 1 \wedge i = j] = \frac{1}{n}\Pr[X = 1]$. Hence, Eq. (A.4) is lower bounded by $\frac{1}{n}\Pr[X = 1]$. This gives Eq. (A.1).

Since Eq. (A.3) does not depend on how X^* is defined after the abortion and also i is independent of (j, Exp),

$$\Pr[o(\mathsf{Exp}^*) = 1] = \Pr[o(\mathsf{Exp}) = 1]\Pr[i = j] + \Pr[o(\mathsf{Exp}^*) = 1 \wedge i \neq j]. \tag{A.5}$$

Since $\Pr[i = j] = \frac{1}{n}$ and $\Pr[o(\mathsf{Exp}^*) = 1 | i \neq j] = 1/2$, Eq. (A.2) follows. □

Appendix B. Proof of Theorem 1

Proof. Let $\mathcal{A}$ be a non-adaptive deniability adversary for a corrupt set $\mathcal{C}$. Then, $\mathcal{S}$ (in Γ^{sim}) needs to simulate the view of $\mathcal{A}$ (in Γ^{rea}). Initially, $\mathbb{T}$ will prepare $\Lambda, \wp$ and $\{(D_i, d_i)\}_{i=1}^n$ with $desc(\wp) = (\lambda, desc(\Lambda))$ for $\lambda \leftarrow \{0,1\}^\kappa$. Let $\mathcal{PK} = (D_1, \cdots, D_n)$ and $\mathcal{SK} = (d_1, \cdots, d_n)$. Let $\mathcal{SK}_\mathcal{C}$ be the set of secret keys of parties in $\mathcal{C}$. Then, $\mathbb{T}$ will invoke $\mathcal{S}$ with $\lambda, desc(\Lambda), \mathcal{PK}, \mathcal{SK}_\mathcal{C}$, auxiliary input aux. In our construction, $\mathcal{S}$ consists of an extractability adversary $\mathcal{E}$ (to be specified) and the corresponding extractor $\mathcal{E}^*$. Note that $\mathcal{E}^*$ is well defined by the extractability assumption once $\mathcal{E}$ is specified. Hence, the code of $\mathcal{S}$ consists of $\mathcal{E}, \mathcal{E}^*$ and how $\mathcal{E}, \mathcal{E}^*$ will be used. We will delay the specification of $\mathcal{E}$ and first specify how $\mathcal{E}$ and $\mathcal{E}^*$ will be used. Toward this, we denote the random tape of $\mathcal{S}$ as three parts r_A, r'_E, r_{E^*}. The code for $\mathcal{S}$ to use $\mathcal{E}, \mathcal{E}^*$ is as follows.

i. W.O.L.G., assume each honest pair (P_i, P_j) has μ transcripts in aux (the case that aux contains a different number of transcripts for each pair (P_i, P_j) can be covered in this setting because $\mathcal{A}$ or $\mathcal{S}$ can just ignore the extra transcripts). So $\mathcal{S}$ can encode aux into a string of $(n - |\mathcal{C}|)(n - |\mathcal{C}| - 1)$ portions (denoted by aux^*), where the portion (i, j) is a concatenation of the μ segments of $x_0|x_1|\mu_0|\mu_1|k_{00}$. Here identities P_i and P_j are not included in the coding as they are implicitly implied by the portion id (i, j). Also τ_0, τ_1 are not included in the coding as they are redundant (given k_{00}). By our theorem assumption, aux^* can be further coded as $\overline{aux^*}$, where the portion (i, j) consists of μ segments of $(\Phi_0^*(x_0)|\Phi_0^*(x_1)|\mu_0|\mu_1|k_{00})$. Further, by our theorem assumption, D_i, d_i can be coded as $\Phi_1^*(D_i)$ and $\Phi_2^*(d_i)$. Let the coding result for $\mathcal{PK}_{\overline{\mathcal{C}}}$ (res. $\mathcal{SK}_\mathcal{C}$) be $\overline{\mathcal{PK}}_{\overline{\mathcal{C}}}$ (resp. $\overline{\mathcal{SK}}_\mathcal{C}$). So the input to $\mathcal{S}$ by $\mathbb{T}$ can be re-formatted as $(\lambda, desc(\Lambda), \overline{aux^*}, \overline{\mathcal{PK}}_{\overline{\mathcal{C}}}, \overline{\mathcal{SK}}_\mathcal{C})$. $\mathcal{S}$ then define $r_E = (\lambda, \overline{aux^*}, \overline{\mathcal{PK}}_{\overline{\mathcal{C}}}, \overline{\mathcal{SK}}_\mathcal{C}, r_A, r'_E)$. Finally, he invokes $\mathcal{E}$ with input $desc(\Lambda)$ and random tape r_E, and invokes $\mathcal{E}^*$ with input $desc(\Lambda)$, r_E and random tape r_{E^*}.

ii. In the specification of $\mathcal{E}$ later, $\mathcal{E}$ will run $\mathcal{A}$ with input $\lambda, desc(\Lambda), \mathcal{PK}, \mathcal{SK}_\mathcal{C}$ and aux (decoded from part of his r_E) and random tape r_A. Then, $\mathcal{E}$ will interact externally with $\mathcal{E}^*$ and internally with $\mathcal{A}$, until the termination of $\mathcal{E}$. During this process, $\mathcal{S}$ only *observes* the messages exchanged between $\mathcal{E}$ and $\mathcal{A}$. At the end of the simulation, $\mathcal{S}$ will output the view of $\mathcal{A}$ (including r_A, the initial input and messages between $\mathcal{E}$ and $\mathcal{A}$).

Now it remains to specify $\mathcal{E}$. Continuing with item (ii) above, we only need to specify how $\mathcal{E}$ maintains the oracles for $\mathcal{A}$. The details are as follows.

Send$(0, i, \ell_i, \cdot)$. In this case, d_i is not required. $\mathcal{E}$ generates $Flow_1$ normally.

Send$(1, j, \ell_j, P_i|x_0|\tau_0)$. In this case, if x_0 occurs in some $Flow_1$ of aux with receiver P_j, then $\mathcal{E}$ takes the corresponding k_{00} from aux and proceeds normally; otherwise, he sends x_0 as an extraction query to $\mathcal{E}^*$. In turn, he will receive w_0. If w_0 is the witness for $x_0 \in L$, then he computes (k'_{00}, k'_{10}) using w_0 and proceeds normally; otherwise, he rejects.

Send$(2, i, \ell_i, P_j|x_1|\tau_1|\mu_0)$. $\mathcal{E}$ uses k_{00} from **Send**$(0, i, \ell_i, \cdot)$ to verify τ_1 normally. Then, he verifies μ_0 and computes μ_1, similar to the case for (τ_0, τ_1) in **Send**$(1, \cdot)$ oracle above.

Send$(3, j, \ell_j, \mu_1|k'_{00})$. In this case, $(k_{00}, k_{01}, k_{10}, k_{11})$ has been defined by **Send**$(1, j, \ell_j)$ oracle. $\mathcal{E}$ proceeds normally.

Reveal(t, ℓ_t) oracle. $\mathcal{E}$ proceeds normally as **Send** oracles above have computed k_{11} when $\Pi_t^{\ell_t}$ successfully completes.

Finally, $\mathcal{S}$ outputs the view of $\mathcal{A}$. Since $\mathcal{A}$'s view consists of the input, random tape r_A and the messages between $\mathcal{A}$ and $\mathcal{E}$, it follows that $\mathcal{S}$ can see and collect it. This completes the description of $\mathcal{S}$ and Γ^{sim}. Information theoretical deniability requires to show that $(\lambda, desc(\Lambda))|\mathcal{PK}|aux|\mathcal{C}|\mathsf{view}(\mathcal{A}, \Gamma^{sim})$ is statistically close to $(\lambda, desc(\Lambda))|\mathcal{PK}|aux|\mathcal{C}|\mathsf{view}(\mathcal{A}, \Gamma^{rea})$. In our code of $\mathcal{E}$, $(\lambda, desc(\Lambda))|\mathcal{PK}|aux|\mathcal{C}|$ is already in the view of $\mathcal{A}$. Hence, it suffices to show that the view of $\mathcal{A}$ in Γ^{sim} is statistically close to that in Γ^{rea}. From the code of $\mathcal{E}$ (in $\mathcal{S}$), the views of $\mathcal{A}$ in Γ^{sim} and Γ^{rea} have the following differences.

- In **Send**$(1, j, \ell_j, P_i|x_0|\tau_0)$ oracle, if x_0 is not in some $Flow_1$ in aux with receiver *exactly* P_j, $\mathcal{E}$ will request $\mathcal{E}^*$ to extract the witness w_0 of x_0 (no matter it exists or not). If $w_0 \neq \perp$, k_{00}, k_{01} is computed and τ_0 is verified; otherwise, τ_0 is rejected. However, in Γ^{rea}, (k_{00}, k_{01}) is computed by evaluating $H_{d_j}(x_0)$ using d_j. Denote the event that the decision of $\mathcal{E}$ in verifying τ_0 is wrong by Bad_1. We remark that for the case where x_0 is in $Flow_1$ of aux with receiver P_j, the verification of $\mathcal{E}$ using k_{00} from aux is correct, as this k_{00} is from $H_{d_j}(x_0)$.
- In **Send**$(2, i, \ell_i, P_j|x_1|\tau_1|\mu_0)$ oracle, when x_1 is not in some $Flow_1$ in aux (as $\tilde{x}_0$) with receiver *exactly* P_i, $\mathcal{E}$ will request $\mathcal{E}^*$ to extract the witness w_1 of x_1 (no matter it exists or not). If $w_1 \neq \perp$, k_{10}, k_{11} is computed and μ_0 is verified; otherwise, μ_0 is rejected. However, in Γ^{rea}, (k_{10}, k_{11}) is computed by evaluating $H_{d_i}(x_1)$ using d_i. Denote the event that the decision of $\mathcal{E}$ in verifying μ_0 is wrong by Bad_2. As in the previous case, we remark that for the case where x_1 is in $Flow_1$ of aux (as $\tilde{x}_0$) with receiver P_i, the verification of $\mathcal{E}$ using $k_{10} = \tilde{k}_{00}$ from aux is correct, as this $\tilde{k}_{00}$ is from $H_{d_i}(\tilde{x}_0)$.

Define $\mathsf{Bad} = \mathsf{Bad}_1 \vee \mathsf{Bad}_2$. To prove the theorem, we only need to show that $\Pr[\mathsf{Bad}(\Gamma^{sim})] = negl(\kappa)$, $v = 1, 2$. This is done by a sequence of game technique. Let $\Gamma_0 = \Gamma^{sim}$.

Game Γ_1**.** We modify $\Gamma_0 = \Gamma^{sim}$ to Γ_1 s.t. $(k_{u0}, k_{u1}) = H_{d_t}(x_u)$ in generating aux is re-defined (by $\mathbb{T}$) as $(k_{u0}, k_{u1}) \leftarrow \{0,1\}^{2\kappa}$. $\mathbb{T}$ records (t, x_u, k_{u0}, k_{u1}) into a list $\mathcal{L}$ (initially empty).

Lemma 3. $\Pr[\mathsf{Bad}(\Gamma_0)] = \Pr[\mathsf{Bad}(\Gamma_1)] + negl(\kappa)$.

Proof. Let $n' = n - |\mathcal{C}|$. W.O.L.G., assume $d_1, \cdots, d_{n'}$ are the uncorrupted secret keys. Define Γ_1^z to be a variant of Γ_1 such that when $t \leq z$, defining $(k_{u0}, k_{u1}) \leftarrow \{0,1\}^{2\kappa}$ (as in Γ_1) for $(t, x_u, k_{u0}, k_{u1}) \in \mathcal{L}$ while for $t > z$, defining it as in Γ_0. It is clear that $\Gamma_1^{n'} = \Gamma_1$ and $\Gamma_1^0 = \Gamma_0$. So if the lemma is violated by adversary $\mathcal{A}$, there exists z such that $\Pr[\mathsf{Bad}(\Gamma_1^z)] - \Pr[\mathsf{Bad}(\Gamma_1^{z-1})]$ is non-negligible. We construct an adversary $\mathcal{D}$ to break the F-indistinguishability of $\wp$. Given $pk = (\alpha(k), desc(\wp))$, $\mathcal{D}$ plays the role of $\mathbb{T}$ to prepare the setup of Γ_1^z, except that $D_z = \alpha(k)$ ($d_z = k$ unknown) and that (x_u, k_{u0}, k_{u1}) w.r.t. d_z (toward computing aux) is obtained through his Challenge Query. Then, he runs $\mathcal{S}$ to simulate Γ_1^z for $\mathcal{A}$. Note since the code of $\mathcal{S}$ (including $\mathcal{E}, \mathcal{E}^*$) does not use d_z, the simulation is normal. Note if the challenge bit c for $\mathcal{D}$ is 0, then the simulation of $\mathcal{D}$ is exactly Γ_1^{z-1}; otherwise, it is Γ_1^z.

We call **Send**$(1, z, \ell_z, \cdot|x_0|\tau_0)$ (or **Send**$(2, z, \ell'_z, \cdot|x_1| \cdot |\mu_0)$) query *irregular*, if x_0 (or x_1) was not in $Flow_1$ in aux with receiver P_z and further w_0 (or w_1) from $\mathcal{E}^*$ is $\perp$. $\mathcal{D}$ picks a random irregular query (say, **Send**$(1, z, \ell_z, P_i|x_0|\tau_0)$) in the simulation and sends $(x_0, \tau_0, P_i|P_z|x_0)$ to his challenger for a Compute query. In turn, he will receive (a, s). If $(a, s) \neq \perp$, $\mathcal{D}$ outputs 1; otherwise, he outputs 0. Note since $(a, s) = \perp$ if and only if τ_0 is invalid, it follows that $\mathcal{D}$ outputs 1 if and only if the picked irregular query is a Bad event. If the total number of irregular queries is μ, the picked irregular query is Bad with probability $\Pr[\mathsf{Bad}(\Gamma_1^{z-1+c})]/\mu$, which is also the probability that $\mathcal{D}$ outputs 1. Thus, $\mathcal{D}$ has a non-negligible advantage. This contradicts Lemma 1. □

Game Γ_2**.** We modify Γ_1 to Γ_2 such that x_0, x_1 in aux is taken as $x_0, x_1 \leftarrow D(X)$ (instead of from $D(L)$). By reduction to the hardness of $\mathcal{I}$ (where a distinguisher plays the role of $\mathbb{T}$ to pick $d_1, \cdots, d_n$ in order to verify a Bad event), we immediately have

Lemma 4. $\Pr[\mathsf{Bad}(\Gamma_1)] = \Pr[\mathsf{Bad}(\Gamma_2)] + negl(\kappa)$.

Game Γ_3**.** We modify Γ_2 to Γ_3 such that for each $P_i|P_j|x_0|x_1|\mu_0|\mu_1|k_{00}$ in aux, $\mathbb{T}$ computes μ_0, μ_1 using a random function $R_{k_{10}}()$ (instead of $F_{k_{10}}()$), where k_{10} is extracted from $(j, x_1, k_{10}, k_{11}) \in \mathcal{L}$. Since this k_{10} is uniformly random in $\{0,1\}^\kappa$ and is only used by $\mathbb{T}$ to compute μ_0, μ_1 in aux, the following immediately follows from the pseudorandomness of F.

Lemma 5. $\Pr[\mathsf{Bad}(\Gamma_2)] = \Pr[\mathsf{Bad}(\Gamma_3)] + negl(\kappa)$.

It is left to bound $\Pr[\mathsf{Bad}(\Gamma_3)]$. To do this, we separate Bad event into two sub events. If $x_0 \in L$ (or $x_1 \in L$) in a Bad event, we denote this event by Bad'; otherwise, we denote it by Bad''. It suffices to show that both Bad' and Bad'' are negligible.

Lemma 6. $\Pr[\mathsf{Bad}'(\Gamma_3)] = negl(\kappa)$.

Proof. Since $x_u \leftarrow D(X), \mu_0, \mu_1 \leftarrow \{0,1\}^\kappa$ and $(k_{u0}, k_{u1}) \leftarrow \{0,1\}^{2\lambda}$ in aux, it follows that $\overline{aux^*}$ in Γ_3 is statistically close to a uniformly random binary string. Further, $\Phi_1^*(D_i), \Phi_2(d_i)$ are uniformly random binary strings. Thus, $r_E = r_A|r'_E|\overline{aux^*}|\overline{\mathcal{PK}_{\overline{\mathcal{C}}}}|\overline{\mathcal{SK}_{\mathcal{C}}}|\lambda$ is statistically close to uniform. Hence, under the extractability assumption for L, $\mathcal{E}^*$ can not extract a valid witness w_u when $\mathcal{E}$ queries $x_u \in L$, is negligible. The conclusion follows. □

In the following lemma, we further analyze Bad'' event.

Lemma 7. $\Pr[\mathsf{Bad}''(\Gamma_3)] = negl(\kappa)$.

Proof. Our strategy is to reduce $P(\mathsf{Bad}'')$ to event E in F-indistinguishability game with challenge bit $c = 0$. Specifically, if the conclusion is violated by adversary $\mathcal{A}$, then an adversary $\mathcal{B}$ that causes a non-negligible E event in F-indistinguishability can be constructed as follows. Given $pk = (\alpha(k), desc(\wp))$, $\mathcal{B}$ takes $j^* \leftarrow \{1, \cdots, n\}\backslash\mathcal{C}$ and plays as the deniability challenger $\mathbb{T}$ to generate $\{D_i, d_i\}_{i=1}^n$ and aux, except that $(D_{j^*}, d_{j^*}) = (\alpha(k), k(unknown))$. Then, he invokes $\mathcal{S}$ with $desc(\wp)$, $\{D_i\}_{i\notin\mathcal{C}}$, $\{d_i\}_{i\in\mathcal{C}}$ and aux. At the end of simulation, $\mathcal{B}$ checks whether an event E occurs w.r.t. d_{j^*}. Toward this, $\mathcal{B}$ can ask Compute query to his challenger for help. Specifically, for each **Send**$(1, j^*, \ell_{j^*}, P_i|x_0|\tau_0)$ such that x_0 not in aux with receiver P_{j^*} and that $\mathcal{E}$ does not receive a valid witness w_0 for x_0, $\mathcal{B}$ sends a Compute Query $(x_0, \tau_0, P_i|P_{j^*}|x_0)$ to his challenger. In turn, he will receive (a, s). In this case, if $(a, s) = \perp$, then τ_0 must be invalid; otherwise, τ_0 is valid. Since $\mathcal{E}$ fails to extract a witness for $x_0 \in L$ is negligible (by Lemma 6 and ignore it), it follows that $x_0 \notin L$. Thus, an E event occurs if and only if $(a, s) \neq \perp$. That is, $\mathcal{B}$ can detect event E w.r.t. x_0 from **Send**$(1, \cdot)$ query. Similar, he can detect E w.r.t. x_1 from **Send**$(2, \cdot)$ query.

When Bad'' occurs, it must occur w.r.t. some j. As $P(j^* = j) = 1/(n - |\mathcal{C}|)$, it follows $P(\mathsf{Bad}'') \leq (n - |\mathcal{C}|)P(\mathsf{E}) = negl(\kappa)$, contradicting the non-negligibility of $P(\mathsf{Bad}'')$. □

Finishing Theorem 1. From Lemmas 6 and 7, $P(\mathsf{Bad}(\Gamma_3)) = negl(\kappa)$. Further, from Lemmas 3–5, $\Pr[\mathsf{Bad}(\Gamma_0)] = \Pr[\mathsf{Bad}(\Gamma^{sim})] = negl(\kappa)$. Since $\mathsf{view}(\mathcal{A}, \Gamma^{sim})$ and $\mathsf{view}(\mathcal{A}, \Gamma^{rea})$ differs only Bad occurs in Γ^{sim}, the theorem follows. □

Appendix C. Proof of Theorem 2

Proof. Let $\mathcal{A}$ be an adversary against the adaptive deniability without aux. We need to construct a simulator $\mathcal{S}$ in Γ^{sim} who generates an output out such that Eq. (1) holds. In Γ^{sim}, $\mathbb{T}$ normally generates Λ, λ and $\{(D_i, d_i)\}_{i=1}^n$ with $\lambda \leftarrow \{0,1\}^\kappa$. Let $\mathcal{PK} = (D_1, \cdots, D_n)$ and $\mathcal{SK} = (d_1, \cdots, d_n)$. $\mathbb{T}$ then runs $\mathcal{S}$ with $\mathcal{PK}, \lambda, desc(\Lambda)$. In our simulator design, $\mathcal{S}$ will consist of two algorithms: an α-extractability adversary $\mathcal{E}$ and its extractor $\mathcal{E}^*$ (by the α-extractability assumption). It remains how to specify $\mathcal{E}$ and how $\mathcal{S}$ will use $\mathcal{E}$ and $\mathcal{E}^*$. Toward this, $\mathcal{S}$ simply takes $r_E, r_{E^*} \leftarrow \{0,1\}^*$ and invokes $\mathcal{E}$ with random tape r_E and input $(desc(\Lambda), \lambda)$ (by our definition it describes (Λ, α)), and invokes $\mathcal{E}^*$ with r_E, r_{E^*} and input $(\lambda, desc(\Lambda))$. Then, $\mathcal{S}$ will play the α-extractability challenger for $\mathcal{E}$ and $\mathcal{E}^*$. Inside $\mathcal{S}$, $\mathcal{E}$ will first use α-queries to obtain $PK_1, \cdots, PK_n$ from $\mathcal{S}$. Then, $\mathcal{E}$ will internally invoke $\mathcal{A}$ with $\mathcal{PK}, \lambda, desc(\Lambda)$ to simulate Γ^{rea}, and externally interact with $\mathcal{E}^*$. For simplicity, we assume that if the witness w extracted by $\mathcal{E}^*$ for query x is not $\perp$, then it is valid (this is for simplicity only, as whether $(x, w) \in L$ can be easily verified and so an invalid w can be redefined to $\perp$). Finally, the output of $\mathcal{S}$ is the view of $\mathcal{A}$ in the simulated Γ^{rea} of $\mathcal{E}$. It remains to specify how $\mathcal{E}$ maintains the **Send**, **Reveal** and **Corrupt** oracles for $\mathcal{A}$. This is detailed as follows.

Send$(0, i, \ell_i, \cdot)$. In this case, no d_i is required and so $\mathcal{E}$ generates $Flow_1$ normally.

Send$(1, j, \ell_j, P_i|x_0|\tau_0)$. In this case, $\mathcal{E}$ sends x_0 as an extraction query to $\mathcal{E}^*$ and in turn receives w_0 from the latter. If $w_0 = \perp$, $\mathcal{S}$ rejects; otherwise, by our convention, w_0 is a valid witness for $x_0 \in L$ and so $\mathcal{E}$ normally processes this query with w_0. The simulation output differs from Γ^{rea} only if the rejection is wrong when $w_0 = \perp$ (denote by event Bad_1).

Send$(2, i, \ell_i, P_j|x_1|\tau_1|\mu_0)$. Similar to **Send**$(1, \cdot)$, $\mathcal{E}$ sends x_1 as an extraction query to $\mathcal{E}^*$ and in turn receives w_1 from the latter. If $w_1 = \perp$, $\mathcal{E}$ rejects; otherwise, by our convention, w_1 is a valid witness for $x_1 \in L$ and so $\mathcal{E}$ normally processes this query with w_1. The simulation output differs from Γ^{rea} only if the rejection is wrong when $w_1 = \perp$ (denote by event Bad_2).

Send$(3, j, \ell_j, \mu_1|k'_{00})$ and **Reveal**(t, ℓ_t). Note that $(k_{10}, k_{11}, k_{00}, k_{01})$ has been computed before these queries and hence the reply is normal.

Corrupt(i). In this case, $\mathcal{E}$ issues an α-reverse query for PK_i to his challenger $\mathcal{S}$. Then, $\mathcal{S}$ issues **Corrupt**(i) query to $\mathbb{T}$. After receiving d_i, $\mathcal{S}$ provides $\mathcal{E}$ with d_i as his answer to the α-reverse query for PK_i. $\mathcal{E}$ then provides d_i and the internal states of P_i to $\mathcal{A}$, where internal states are well-defined by **Send** oracles.

This completes the description of $\mathcal{S}$ (including $\mathcal{E}, \mathcal{E}^*$) and Γ^{sim}. At the end of simulation, $\mathcal{S}$ outputs $\mathsf{view}(\mathcal{A}, \Gamma^{sim})$ in the simulation of $\mathcal{E}$. From the code of $\mathcal{S}$, $\mathsf{view}(\mathcal{A}, \Gamma^{sim})$ differs from $\mathsf{view}(\mathcal{A}, \Gamma^{rea})$ only if $\mathsf{Bad} = \mathsf{Bad}_1 \vee \mathsf{Bad}_2$ occurs. This has two cases:

(a) $\mathcal{E}^*$ fails to extract w_0 in **Send**$(1, \cdot)$ when $x_0 \in L$ (resp. w_1 in **Send**$(2, \cdot)$ when $x_1 \in L$);
(b) $x_0 \notin L$ but τ_0 is valid in **Send**$(1, j, \ell_j, P_i|x_0|\tau_0)$ query, or,
$x_1 \notin L$ but μ_0 is valid in **Send**$(2, i, \ell_i, P_j|x_1|\tau_1|\mu_0)$ query.

From our description of $\mathcal{E}$, the α-extractability game involving $(\mathcal{E}, \mathcal{E}^*)$ and their challenger $\mathcal{S}$ is property defined. Hence, by α-extractability assumption for L, case (a) occurs negligibly. By the following claim, case (b) occurs negligibly too. Thus, our theorem follows. □

Claim. *If $\wp$ is computational universal$_2$, then case (b) occurs negligibly.*

Proof. If the claim is violated, we construct an adversary $\mathcal{D}$ to break computational universal$_2$ property of $\wp$. Upon $pk = (\alpha(k), \lambda, desc(\Lambda))$, $\mathcal{D}$ does as follows. He takes $J \leftarrow \{1, \cdots, n\}$ and plays the roles of $\mathbb{T}$ and $\mathcal{S}$ to normally simulate Γ^{sim}, except that D_J is defined as $\alpha(k)$ in pk and d_J is unknown. The simulation of $\mathcal{D}$ is perfect prior to the corruption of P_J: an uncorrupted d_J is not used in the simulation of $\mathcal{S}$ (especially $\mathcal{E}$). If P_J is corrupted by $\mathcal{A}$, then $\mathcal{D}$ aborts with failure. When $\mathcal{D}$ does not abort, then after the simulation, he can check whether Bad occurs to the public key D_J by first finding out the set $A = \{x \mid x \notin L, x \text{ is sent to } P_J\}$ (by querying every x with receiver P_J to Evalu oracle where $\perp$ is returned if and only if $x \notin L$) and then taking $x^* \leftarrow A$ as $\mathcal{D}$'s challenge. In turn, he will receive $(a, s) \in \{0,1\}^{2\kappa}$, which is either $H_k(x^*)$ or uniformly random. Then he uses a to verify whether Bad event occurs. If it occurs, he outputs 1; otherwise, he outputs 0. Note if a is uniformly random, then Bad occurs negligibly by pseudorandomness of F. Hence, $\mathcal{D}$ has an advantage $\Pr[\mathbf{Bad}(\Gamma^{sim})]/(n\nu) - negl(\kappa)$, non-negligible, where ν is the upper bound of $|A|$. This contradicts the computational universal$_2$ property of $\mathcal{I}$. □

Appendix D. Proof of Theorem 3

Proof. We use Γ_0 to denote the secrecy game. We need to show that for any PPT adversary $\mathcal{A}$, $\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_0)] = 1/2 + negl(\kappa)$. Toward this, we modify Γ_0 into a sequence of games $\Gamma_1, \cdots, \Gamma_4$, build a relation between $\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_i)]$ and $\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_{i+1})]$ and finally bound $\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_4)]$. Let $\mathbb{T}$ be the challenger in Γ_0.

Game Γ_1. Let L be an upper bound on the total number of instances in a particular party. We modify Γ_0 to Γ_1: initially, $\mathbb{T}$ takes $z \leftarrow \{1, \cdots, n\}, \ell \leftarrow \{1, \cdots, L\}, z^* \leftarrow \{1, \cdots, n\} \backslash \{z\}$; then he simulates Γ_0 with $\mathcal{A}$ until $\mathbf{Test}(v, \ell_v)$. In this case, if $v = z, \ell_v = \ell$ and $\mathsf{pid}_z^\ell = P_{z^*}$, then $\mathbb{T}$ simulates Γ_0 normally; otherwise, it terminates and sets Succ = 0 or 1 randomly. In the remaining proof, z, ℓ, z^* *always mean the variables defined here*. By Lemma 2 (Eq. (A.2)), we have

Lemma 8. $\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_1)] = \frac{1}{2} + \frac{1}{L(n-1)n} \cdot \left\{\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_0)] - \frac{1}{2}\right\}$.

Game Γ_2. We modify Γ_1 to Γ_2 such that if P_z or P_{z^*} is corrupted or instance Π_z^ℓ is revealed, then $\mathbb{T}$ aborts with output 0, 1 randomly. Note when one of these conditions occurs, by definition of a test session, it is impossible for $\mathbf{Test}(v, \ell_v)$ to satisfy $v = z, \ell_v = \ell, \mathsf{pid}_z^\ell = P_{z^*}$. Therefore,

Lemma 9. $\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_1)] = \Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_2)]$.

Game Γ_3. We define Γ_3 to be Γ_2 except that for $u = 0, 1$, $(k_{u0}, k_{u1}) = H_{d_{z^*}}(x_u)$ by P_z is replaced by $(k_{u0}, k_{u1}) \leftarrow \{0,1\}^{2\kappa}$. Details follow, where the main effort is to make this change consistent. Initially, let $\Theta = \{\}$. All the oracles are simulated as in Γ_2, except for the following.

i. In $\mathbf{Send}(0, z, \ell_z, \text{"ke: } z^*\text{"})$ query, he proceeds normally, except that he defines $(k_{00}, k_{01}) \leftarrow \{0,1\}^{2\kappa}$ (instead of taking $(k_{00}, k_{01}) = H_{d_{z^*}}(x_0)$). Finally, he adds (x_0, k_{00}, k_{01}) into Θ.
Later $\mathbf{Send}(2, z, \ell_z, P_j|*)$ query is processed normally (including case $j = z^*$) with (k_{00}, k_{01}).
ii. In $\mathbf{Send}(1, z^*, \ell_{z^*}, P_i|x_0|\tau_0)$ query, he first checks if $(x_0, a, s) \in \Theta$ for some (a, s). If yes, he defines $(k'_{00}, k'_{01}) = (a, s)$; otherwise, he computes $(k'_{00}, k'_{01}) = H_{d_{z^*}}(x_0)$ normally. The remaining computation for this query is normal.
iii. In $\mathbf{Send}(1, z, \ell_z, P_i|x_0|\tau_0)$ query. If $i \neq z^*$, it is processed normally; otherwise, $\mathbb{T}$ still proceeds normally, except $(k_{10}, k_{11}) \leftarrow \{0,1\}^{2\kappa}$ (instead of $(k_{10}, k_{11}) = H_{d_{z^*}}(x_1)$) and finally he adds (x_1, k_{10}, k_{11}) into Θ.
iv. In $\mathbf{Send}(2, z^*, \ell_{z^*}, P_j|x_1|\tau_1|\mu_0)$ query, he first checks if $(x_1, a, s) \in \Theta$ for some (a, s). If yes, he defines $(k'_{10}, k'_{11}) = (a, s)$; otherwise, he computes $(k'_{10}, k'_{11}) = H_{d_{z^*}}(x_1)$ normally. The remaining computation for this query is normal.

The above simulation maintains the property: each $(x, a, s) \in \Theta$ is recorded in the execution of some Π_z^ℓ and also if (a, s) is a replacement of $H_{d_{z^*}}(x)$, then $H_{d_{z^*}}(x)$ in the whole simulation (i.e., all **Send** oracles) is replaced by (a, s). It is also worth noting that if each $(x, a, s) \in \Theta$ happens to satisfy $(a, s) = H_{d_{z^*}}(x)$, then Γ_3 is exactly Γ_2.

Lemma 10. *If $\wp$ is computational universal$_2$, then*

$$\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_3)] = \Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_2)] + negl(\kappa).$$

Proof. If the conclusion is violated by adversary $\mathcal{A}$, then an attacker $\mathcal{D}$ can violate the F-indistinguishability of $\wp$ (Lemma 1) as follows. Upon receiving $desc(\Lambda), \lambda$ and a public key D, $\mathcal{D}$ simulates Γ_3 with $\mathcal{A}$ against it. He defines $D_{z^*} = D$ (d_{z^*} unknown) and defines $(D_i, d_i) = (\alpha(k_i), k_i)$ for $i \neq z^*$ normally by taking $k_i \leftarrow \mathcal{K}$. He then provides $\{D_i\}_{i=1}^n, desc(\Lambda), \lambda$ to $\mathcal{A}$ and maintains the oracles normally except for the following. Note Θ is maintained by $\mathcal{D}$'s challenger.

i. In $\mathbf{Send}(0, z, \ell_z, \text{"ke: } z^*\text{"})$ query, he proceeds normally, except that $\mathcal{D}$ obtains (x_0, k_{00}, k_{01}) through a Challenge query. In this query, if challenge bit c for $\mathcal{D}$ is 0, the simulation is perfect as in Γ_2; otherwise, it is according to Γ_3.

ii. In **Send**$(1, z^*, \ell_{z^*}, P_i|x_0|\tau_0)$ query, he issues $(x_0, \tau_0, P_i|P_j|x_0)$ as a Compute query. In turn, he will receive (a, s). If $(a, s) = \perp$, he rejects; otherwise, he defines $(k'_{00}, k'_{01}) = (a, s)$ and proceeds normally. By the code of the Compute oracle, τ_0 is invalid if and only if $(a, s) = \perp$. Thus, the simulation for this query is perfectly consistent with Γ_{2+c}.
iii. In **Send**$(1, z, \ell_z, P_i|x_0|\tau_0)$ query. If $i \neq z^*$, $\mathbb{T}$ proceeds normally; otherwise, it is still normal except for the following: $\mathbb{T}$ obtains (x_1, k_{10}, k_{11}) through a Challenge query. The simulation is consistent with Γ_2 if $c = 0$, and it is perfectly consistent with Γ_3 if $c = 1$.
iv. In **Send**$(2, z^*, \ell_{z^*}, P_j|x_1|\tau_1|\mu_0)$ query, he issues $(x_1, \tau_1, P_{z^*}|P_j|x_0|x_1)$ as a Compute query and receives (a, s). If $(a, s) = \perp$, he rejects; otherwise, he defines $(k'_{10}, k'_{11}) = (a, s)$ and proceeds normally. The simulation for this query is consistent with Γ_{2+c}.

This completes the description of $\mathcal{D}$. At the end of the simulation, when $\mathcal{A}$ succeeds, $\mathcal{D}$ outputs 0; otherwise, he outputs 1. From the description of $\mathcal{D}$, we know that when $c = 0$, the simulated game is Γ_2; otherwise, it is Γ_3. Hence, a non-negligible success gap of $\mathcal{A}$ between Γ_2 and Γ_3 implies a non-negligible advantage of $\mathcal{D}$, contradicting Lemma 1. □

Game Γ_4. We define Γ_4 to be Γ_3 except that for $u = 0, 1$, $(k_{u0}, k_{u1}) = H_{d_z}(x_u)$ by P_{z^*} is replaced by $(k_{u0}, k_{u1}) \leftarrow \{0,1\}^{2\kappa}$. Details follow, where the main effort is to make this change consistent. Initially, let $\Theta' = \{\}$. The oracles are unchanged as in Γ_3, except for the following.

i. In **Send**$(0, z^*, \ell_{z^*}, \text{“ke: } z\text{”})$ query, he proceeds normally, except that he defines $(k_{00}, k_{01}) \leftarrow \{0,1\}^{2\kappa}$ (instead of taking $(k_{00}, k_{01}) = H_{d_z}(x_0)$). Finally, he adds (x_0, k_{00}, k_{01}) into Θ'.
Later **Send**$(2, z^*, \ell_{z^*}, P_j|*)$ query is processed normally (including case $j = z$) with (k_{00}, k_{01}).
ii. In **Send**$(1, z, \ell_z, P_i|x_0|\tau_0)$ query, he first checks if $(x_0, a, s) \in \Theta'$ for some (a, s). If this is true, he defines $(k'_{00}, k'_{01}) = (a, s)$; otherwise, he computes $(k'_{00}, k'_{01}) = H_{d_z}(x_0)$ normally. The remaining computation for this query is normal.
iii. In **Send**$(1, z^*, \ell_{z^*}, P_j|x_0|\tau_0)$ query, if $j \neq z$, it proceeds normally; otherwise, $\mathbb{T}$ still proceeds normally, except $(k_{10}, k_{11}) \leftarrow \{0,1\}^{2\kappa}$ (instead of $(k_{10}, k_{11}) = H_{d_z}(x_1)$). Finally, he adds (x_1, k_{10}, k_{11}) into Θ'.
iv. In **Send**$(2, z, \ell_z, P_j|x_1|\tau_1|\mu_0)$ query, he first checks if $(x_1, a, s) \in \Theta'$ for some (a, s). If yes, he defines $(k'_{10}, k'_{11}) = (a, s)$; otherwise, he computes $(k'_{10}, k'_{11}) = H_{d_z}(x_1)$ normally. The remaining computation for this query is normal.

Similar to Lemma 10, we have that

Lemma 11. *If $\wp$ is computational universal$_2$,*

$$\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_4)] = \Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_3)] + negl(\kappa).$$

Now we will analyze Γ_4. We remind that Π^{ℓ}_z is the test session (since Γ_1). We first show

Lemma 12. *There exists a unique partnered instance $\Pi^{\ell^*}_{z^*}$ for Π^{ℓ}_z.*

Proof. We prove the conclusion in two cases. In this proof, we assume condition Normal : $x \leftarrow D(L)$ never repeats the same x, which is violated only negligibly by the hardness of L. (Otherwise, given a challenge y, a distinguisher samples $x \leftarrow D(L)$ and claims $y \in L$ if and only if $x = y$. When $y \in L$, $x = y$ occurs non-negligibly; when $y \notin L$, $x = y$ occurs with probability 0.)
Π^{ℓ}_z is an initiator. Let the message transcript in the view of Π^{ℓ}_z be $P_z|\bar{x}_0|\bar{\tau}_0|P_{z^*}|\bar{x}_1|\bar{\tau}_1|\bar{\mu}_0|\bar{\mu}_1|\bar{k}_{00}$. Then $\mathsf{sid}^{\ell}_z = P_z|P_{z^*}|\bar{x}_0|\bar{x}_1$. We need to show that there is a unique instance $\Pi^{\ell^*}_{z^*}$ such that $\mathsf{sid}^{\ell^*}_{z^*} = \mathsf{sid}^{\ell}_z$. First of all, there does not exist two such instances. Otherwise, two instances in P_{z^*} sample the same $\bar{x}_1$, violating the Normal condition! Hence, it suffices to prove the existence of ℓ^*. We claim that if ℓ^* does not exist, then $\bar{\tau}_1$ will be rejected by Π^{ℓ}_z. Reviewing the definitions of all oracles, we can see that $\bar{k}_{00}$ (or its identical copy $\bar{k}'_{00}$), defined in Π^{ℓ}_z w.r.t. $(\bar{x}_0, D_{z^*})$, will be used only in the following settings (this observation holds similarly for k_{00} defined in any general $\Pi^{\ell_i}_i$):
(1) **Send**$(0, z, \ell, \text{“}ke : z^*\text{”})$ oracle computes $\bar{\tau}_0$ and **Send**$(2, z, \ell, P_{z^*}|\bar{x}_1|\bar{\tau}_1|\bar{\mu}_0)$ verifies $\bar{\tau}_1$ and (if $\bar{\tau}_1, \bar{\mu}_0$ valid) puts $\bar{k}_{00}$ in $Flow_3$.
(2) **Send**$(1, z^*, \ell_{z^*}, P_t|\tilde{x}_0|\tilde{\tau}_0)$ oracle verifies $\tilde{\tau}_0$ and (if valid) generates $\tilde{\tau}_1$; also **Send**$(3, z^*, \ell_{z^*}, \tilde{\mu}_1|\tilde{k}_{00})$ oracle verifies $\bar{k}_{00} = \tilde{k}_{00}$.
(3) **Send**$(2, z^*, \ell'_{z^*}, P_u|x'_1|\tau'_1|\mu'_0)$ for $x'_1 = \bar{x}_0$ uses $\bar{k}_{00}$ to verify μ'_0 and (if valid) generates μ'_1;
(4) P_z (resp. P_{z^*}) is corrupted such that $\bar{k}_{00}$ in item (1) (resp. item (2)) as part of its session state is provided to $\mathcal{A}$, where session states are defined at the beginning of Section 5.

Now we analyze these cases. First, item (4) is impossible as Π^{ℓ}_z is a test session with $\mathsf{pid}^{\ell}_z = P_{z^*}$ by our definition starting in Γ_1. Hence, we focus on items (1)(2)(3). We claim that event E: **equality in verifying $\bar{k}_{00} = \tilde{k}_{00}$ in item (2) holds before Π^{ℓ}_z in item (1) verifies $\bar{\tau}_1$**, is negligible. Indeed, if E occurs, then prior to its occurrence, $\bar{k}_{00}$ is only used to evaluate $F_{\bar{k}_{00}}()$.

So non-negligible event E can be reduced to break F (by checking $F_{\bar{k}_{00}}(M) \stackrel{?}{=} \tau$, where τ is the oracle reply for an arbitrary query M).

Therefore, ignoring event E, we know that before verifying $\bar{\tau}_1$ in item (1), the only use of $\bar{k}_{00}$ is to evaluate $F_{\bar{k}_{00}}()$ in item (1)(2)(3). We now prove that if ℓ^* does not exist, then $\bar{\tau}_1$ in item (1) is invalid. By the security of F, it suffices to show that the simulator never evaluates $F_{\bar{k}_{00}}()$ with this input. In fact, the non-existence of ℓ^* implies that $\tilde{x}_1$ from **Send**$(1, z^*, \ell_{z^*}, P_t|x_0|\tilde{\tau}_0)$ must satisfy $P_t|\tilde{x}_1 \neq P_z|\bar{x}_1$. So the input for $\tilde{\tau}_1$ (generated by $\Pi_{z^*}^{\ell_{z^*}}$) is different from that for $\bar{\tau}_1$. Further, the input for $\tilde{\tau}_0$ in item (1) and the input for $\tilde{\tau}_0$ in item (2) have formats different from that for $\bar{\tau}_1$. Finally, the input format for μ_1', μ_0' in item (3) is $P_{z^*}|P_u|\cdot$, different from $P_z|P_{z^*}|\cdot$ for $\bar{\tau}_1$. Hence, the security of F implies that $\bar{\tau}_1$ will be rejected. That is, the non-existence of ℓ^* implies that Π_z^ℓ will not accept, contradicting that Π_z^ℓ is a test session. Thus, Π_z^ℓ must have a partner session and the case for an initiator Π_z^ℓ is proved.

Π_z^ℓ **is a responder.** This case is similar to the case Π_z^ℓ is an initiator. We omit it here. □

Lemma 13. *If Π_z^ℓ is an initiator, assume that the transcript in his view is $P_z|\bar{x}_0|\bar{\tau}_0|P_{z^*}|\bar{x}_1|\bar{\tau}_1|\bar{\mu}_0|\bar{\mu}_1|\bar{k}_{00}$. Let $\Pi_u^{\ell_u}$ be any successfully completed instance such that $\mathsf{sid}_u^{\ell_u} = P_a|P_b|\hat{x}_0|\hat{x}_1$ s.t. $P_z|\bar{x}_1 = P_b|\hat{x}_0$ or $P_a|\hat{x}_1$, then $\Pi_u^{\ell_u} = \Pi_z^\ell$ or $\Pi_{z^*}^{\ell^*}$ (where $\Pi_{z^*}^{\ell^*}$ is the unique partner of Π_z^ℓ defined in Lemma 12).*

Proof. Assume $\Pi_u^{\ell_u} \neq \Pi_{z^*}^{\ell^*}$. Let the message transcript in the view of $\Pi_u^{\ell_u}$ be $P_a|\hat{x}_0|\hat{\tau}_0|P_b|\hat{x}_1|\hat{\tau}_1|\hat{\mu}_0|\hat{\mu}_1|\hat{k}_{00}$. No matter $P_z|\bar{x}_1 = P_b|\hat{x}_0$ or $P_a|\hat{x}_1$, P_z is the recipient of $\bar{x}_1$. If $P_u \neq P_z$, then $\Pi_u^{\ell_u}$ generates $\bar{x}_1$. As Π_z^ℓ is an initiator, its unique partner $\Pi_{z^*}^{\ell^*}$ must also have generated $\bar{x}_1$. Hence, by Normal condition, $\Pi_u^{\ell_u} = \Pi_{z^*}^{\ell^*}$, contradiction! Hence, we can assume $z = u$ and $\Pi_u^{\ell_u}$ is the recipient of $\bar{x}_1$. So it suffices to show $\ell_z = \ell$.

By Normal condition, $\Pi_{z^*}^{\ell^*}$ is the unique instance that samples $x \leftarrow D(L)$ with $x = \bar{x}_1$. Thus, from the definitions of oracles, $\bar{k}_{10}$ (with respect to $\bar{x}_1$ and P_z) will be only used in settings:

(1) **Send**$(1, z, \ell_z', P_i|x_0'|\tau_0')$ oracle with $x_0' = \bar{x}_1$, verifies τ_0' and (if valid) generates τ_1' and later **Send**$(3, z, \ell_z', \mu_1|\tilde{k}_{00})$ verifies $\tilde{k}_{00} \stackrel{?}{=} \bar{k}_{10}$;
(2) **Send**$(1, z^*, \ell^*, P_z|\bar{x}_0|\bar{\tau}_0)$ oracle computes $\bar{\mu}_0$; **Send**$(3, z^*, \ell^*, \bar{\mu}_1'|\bar{k}_{00}')$ oracle verifies $\bar{\mu}_1' \stackrel{?}{=} \bar{\mu}_1$;
(3) **Send**$(2, z, \ell_z'', P_j|\bar{x}_1|\tau_1''|\mu_0'')$ verifies μ_0'' and computes μ_1'';
(4) P_z or P_{z^*} are corrupted and $\bar{k}_{10}$ as session state in items (1)(2)(3) will be revealed.

Item (4) is impossible as Π_z^ℓ is a test session with $\mathsf{pid}_z^\ell = P_{z^*}$. We focus on items (1)(2)(3). From the description of items (1)(2)(3), prior to accepting τ_0' in item (1) for some $\Pi_z^{\ell_z'}$, the only use of $\bar{k}_{10}$ is to evaluate $F_{\bar{k}_{10}}()$. However, the input for τ_0' has a different format from that computed in item (2)(3). Hence, the validity of τ_0' implies breaking F. Thus, we can remove item (1). So $\bar{k}_{10}$ is only used in items (2)(3). Assume x_0 in $\Pi_z^{\ell_z''}$ in item (3) is x_0''. Then, by Normal condition, $x_0'' \neq \bar{x}_0$ if $\ell_z'' \neq \ell$. Hence, before verifying μ_0'' in $\Pi_z^{\ell''}$ for $\ell_z'' \neq \ell$, the simulator never evaluates $F_{\bar{k}_{10}}()$ with input $P_z|P_j|x_0''|\bar{x}_1$. Hence, acceptance of μ_0'' implies breaking F. Therefore, the validity of μ_0'' in item (3) implies $\ell_z'' = \ell$.

Since $\Pi_z^{\ell_z}$, as a recipient of $\bar{x}_1$, must have used $\bar{k}_{10}$, he must use it in either item (1) or item (3). Since it has successfully completed, only item (3) is possible (item (1) has been removed above). Thus, $\Pi_z^{\ell_z}$ must have accepted μ_0''. So $\ell_z = \ell$. This completes the proof of the lemma. □

Similar to Lemma 13, we have the following.

Lemma 14. *If $\Pi_{z^*}^{\ell^*}$ is an initiator, assume that the transcript in his view is $P_{z^*}|\bar{x}_0|\bar{\tau}_0|P_z|\bar{x}_1|\bar{\tau}_1|\bar{\mu}_0|\bar{\mu}_1|\bar{k}_{00}$. Let $\Pi_u^{\ell_u}$ be any accepted instance such that $\mathsf{sid}_u^{\ell_u} = P_a|P_b|\hat{x}_0|\hat{x}_1$ s.t. $P_{z^*}|\bar{x}_1 = P_b|\hat{x}_0$ or $P_a|\hat{x}_1$, then $\Pi_u^{\ell_u} = \Pi_z^\ell$ or $\Pi_{z^*}^{\ell^*}$.*

Now we are ready to evaluate $\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_4)]$.

Lemma 15. $\Pr[\mathbf{Succ}(\mathcal{A}, \Gamma_4)] = 1/2$.

Proof. Assume Π_z^ℓ is an initiator (the responder case is similar). Let $\alpha_0 = sk_z^\ell = \bar{k}_{11}$ and $\alpha_1 \leftarrow \{0,1\}^\kappa$. Let r_{ch} be all the randomness in challenger $\mathbb{T}$ in Γ_4, excluding $\alpha_0|\alpha_1|b$. Let view_t be the adversary view **before** the tth query Q_t. We first claim that for any t, the response to query Q_t is deterministic in $(\mathsf{view}_t, r_{ch}, \alpha_b)$ and that $\mathsf{stat}_i^{\ell_i}$ for $\Pi_i^{\ell_i} \neq \Pi_z^\ell, \Pi_{z^*}^{\ell^*}$ is deterministic in $(\mathsf{view}_t, r_{ch}, \alpha_b)$. Note Q_t is also determined by view_t. We now verify the claim for each oracle.

Send oracles. k_{11} is never used in **Send** oracle to generate an output. Further, (α_1, b) are never used in **Send** oracles. Hence, the first part of the claim remains valid after the query. We now consider the second part. For any initiator $\Pi_i^{\ell_i}$

with $\mathsf{pid}_i^{\ell_i} = P_j$, $\mathsf{stat}_i^{\ell_i}$ is determined by $P_i|P_j|x_0|x_1|k_{00}|k_{01}|k_{10}|k_{11}$. As $P_i|P_j|x_0|x_1$ is either in view_t or computed from r_{ch}, we consider $k_{00}|k_{01}|k_{10}|k_{11}$.

If $j \neq z^*, z$, then $k_{00}|k_{01} = H_{d_j}(x_0)$, where notice that d_j is determined by r_{ch}. If $j = z^*$, then $k_{00}|k_{01}$ is either sampled from $\{0,1\}^{2\kappa}$ using randomness r_{ch} or equals $H_{d_j}(x_0)$. So in any case $k_{00}|k_{01}$ are determined by r_{ch} and view_t. The case $j = z$ is similar.

If $i \neq z, z^*$, then $k_{10}|k_{11} = H_{d_i}(x_1)$. For $i = z$, if $x_1 \neq \bar{x}_1$, then (k_{10}, k_{11}) must be determined by view_t and r_{ch}. Further, if $x_1 = \bar{x}_1$, then $\Pi_z^{\ell_z}$ will put $\bar{k}_{11}$ in the $\mathsf{stat}_z^{\ell_z}$ only if μ_0 is accepted. However, by Lemma 13, this occurs only if $\ell_z = \ell$ (contradicting $\Pi_i^{\ell_i} \neq \Pi_z^{\ell}$). For $i = z^*$, $k_{10}k_{11}$ is $H_{d_{z^*}}(x_1)$ or its replacement. The randomness is independent of (α_0, α_1, b), as $\alpha_0 = \bar{k}_{11}$ is from the replacement of $H_{d_z}(*)$. Further $i \neq z$ implies that this **Send** oracle call is not for the test session and hence α_1, b is independent in this oracle.

So we can safely conclude that $\mathsf{stat}_z^{\ell_z}$ for $\ell_z \neq \ell$ must be determined by r_{ch} and view_t.

Test(z, ℓ). The oracle output is α_b. Hence, the claim is valid.

Corrupt(i). The oracle output is d_i and internal states of P_i. As d_i is determined by r_{ch} and $\mathsf{stat}_i^{\ell_i}$ for $i \neq z, z^*$ is determined by view_t and r_{ch}. Hence, the claim remains valid.

Reveal(i, ℓ_i). By **Test** restriction, $\Pi_i^{\ell_i} \neq \Pi_z^{\ell}, \Pi_{z^*}^{\ell^*}$. As $sk_i^{\ell_i}$ is in the $\mathsf{stat}_i^{\ell_i}$, the claim remains valid from the analysis in **Send** oracles.

Finally, since each oracle output is determined by $\mathsf{view}_t, r_{ch}, \alpha_b$ and view_0 is the random tape r_A of $\mathcal{A}$, it follows that view_t is determined by r_A, r_{ch}, α_b. Since α_0 and α_1 are identically distributed, b is independent of (r_A, r_{ch}, α_b). Hence, b is independent of view_t and further independent of b' as b' is determined by view_t for some t. Thus, the conclusion follows. □

Finishing Theorem 3. From Lemmas 8-15, the theorem follows. □

Appendix E. Proof of Lemma 1

Proof. Use $\Re_c$ to denote the F-indistinguishability game with a challenge bit c. We show that $\Pr[\mathcal{A}(\Re_0) = 1] = \Pr[\mathcal{A}(\Re_1) = 1] + negl(\kappa)$. Let $\Re_0^\ell$ denote a variant of $\Re_0$, where the first ℓ `Challenge` queries are answered as in $\Re_1$ while the remaining such queries are answered as in $\Re_0$. Let ♯ of `Challenge` queries be bounded by N. Then, $\Re_0^0 = \Re_0$ and $\Re_0^N = \Re_1$. If the conclusion is wrong, then there exists ℓ such that $|\Pr[\mathcal{A}(\Re_0^{\ell-1}) = 1] - \Pr[\mathcal{A}(\Re_0^{\ell}) = 1]|$ is non-negligible. Let $\hat{\Re}_0^i, i = \ell - 1, \ell$ be the variant of $\Re_0^i$ such that in the ℓth `Challenge` query, $x \leftarrow D(X \backslash L)$ (instead of $x \leftarrow D(L)$), where correspondingly $H_k(x)$ is computed using k. By the hardness of $\mathcal{I}$, $\Pr[\mathcal{A}(\Re_0^i) = 1] = \Pr[\mathcal{A}(\hat{\Re}_0^i) = 1] + negl(\kappa)$. Hence, $|\Pr[\mathcal{A}(\hat{\Re}_0^{\ell-1}) = 1] - \Pr[\mathcal{A}(\hat{\Re}_0^{\ell}) = 1]|$ is non-negligible. However, if this is true, we build an adversary $\mathcal{D}$ to break the computationally universal$_2$ property of Π. Specifically, upon input $pk = (\lambda, desc(\Lambda), \alpha(k))$, $\mathcal{D}$ invokes $\mathcal{A}$ with pk and simulates $\hat{\Re}_0^\ell$ with it as follows. He defines c to his own challenge bit (unknown).

•ith `Challenge` Query. If $i \neq \ell$, $\mathcal{D}$ takes $x \xleftarrow{w} D(L)$, computes $(a_0, s_0) = H_k(x)$ using w and processes the query as in $\hat{\Re}_0^\ell$. If $i = \ell$, he outputs $x^* \leftarrow D(X \backslash L)$ as his x_2 query. In turn, he receives (a_c^*, s_c^*). He then sends (x^*, a_c^*, s_c^*) to $\mathcal{A}$ and updates $\Theta = \Theta \cup \{(x^*, a_c^*, s_c^*)\}$.

•`Compute` Query (x, σ, m). If $(x, a', s') \in \Theta$ for some a', he verifies σ using a'; otherwise, he queries an Evalu query x to his challenger and in turn receives (a, s). If $(a, s) = \perp$ (in this case $x \notin L$) or $\sigma \neq F_a(m)$, he outputs $\perp$; otherwise, he outputs (a, s).

At the end of game, $\mathcal{D}$ outputs whatever $\mathcal{A}$ does. Note in the above simulated computational universal$_2$ game, $\mathcal{D}$ defines $x_2 = x^*$ and does not issue query x_1.

Denote the simulated game of $\mathcal{D}$ with bit c by $\bar{\Re}_0^{\ell-1+c}$. Then $\bar{\Re}_0^{\ell-1+c}$ is identical to $\hat{\Re}_0^{\ell-1+c}$, except in the case of $x \notin L$ in `Compute` query. In this case, if the challenger of $\mathcal{D}$ returns $(a, s) = \perp$, $\mathcal{D}$ will output $\perp$ directly while $\hat{\Re}_0^{\ell-1+c}$ will first verify σ using a in $(a, s) = H_k(x)$ and (if valid) output (a, s). In other words, $\hat{\Re}_0^{\ell-1+c}$ differs from $\bar{\Re}_0^{\ell-1+c}$ only when event E occurs in $\bar{\Re}_0^{\ell-1+c}$. Hence, $|\Pr[\mathcal{A}(\hat{\Re}_0^{\ell-1+c}) = 1] - \Pr[\mathcal{A}(\bar{\Re}_0^{\ell-1+c}) = 1]| \leq \Pr[\mathsf{E}(\bar{\Re}_0^{\ell-1+c})]$. We claim that $\Pr[\mathsf{E}(\bar{\Re}_0^{\ell-1+c})] = negl(\kappa), c = 0, 1$; otherwise, the computational universal$_2$ property of Π can be broken by adversary $\mathcal{D}'$ as follows. W.O.L.G, assume $\Pr[\mathsf{E}(\bar{\Re}_0^{\ell})]$ is non-negligible. Upon receiving pk, $\mathcal{D}'$ simulates $\bar{\Re}_0^{\ell}$ by playing the role of $\mathcal{D}$ and the challenger of $\mathcal{D}$, except the evaluation of $H_k(x)$ is done under his own challenger's help. Specifically, for the ith `Challenge` query for $i \neq \ell$, he can take $x \leftarrow D(L)$ and compute $H_k(x)$ with w himself; for the ℓth `Challenge` query, he takes $x^* \leftarrow D(X \backslash L)$ and asks his challenger to evaluate $H_k(x^*)$ as the first challenge (i.e., x_1 in the definition); upon a `Compute` query (x, σ, m), he issues an Evalu query x to his challenger and in turn receives (a, s), where $(a, s) = \perp$ if $x \notin L$ and $(a, s) = H_k(x)$ otherwise. In case of the former, he records x into $\mathcal{L}$ if $x \neq x^*$. In any case, the remaining simulation is normal. At the end of game, if $c = 1$, he outputs 0/1 randomly; otherwise, he takes y^* randomly from $\mathcal{L}$ and ask y^* as the second challenge (i.e., x_2 in the definition). In turn he will receive (a_b^*, s_b^*), where $(a_0^*, s_0^*) = H_k(y^*)$ or $(a_1, s_1) \leftarrow \{0,1\}^{2\kappa}$. Then he reviews all the `Compute` queries with forms (y^*, σ, m) for any σ, m and denotes event $\sigma = F_{a_b^*}(m)$ by inc. In case of inc, he outputs 0; otherwise, he outputs 1. Note if $b = 1$, then inc occurs to y^* negligibly by unforgeability of F. If $b = 0$, then inc event is E event occurs to y^*. When an E event occurs, it will occur to some x in $\mathcal{L}$. Hence, inc occurs in $\mathcal{D}$'s algorithm for $b = 0$ with

probability at least $\Pr[\mathsf{E}(\bar{\Re}_0^\ell)]/|\mathcal{L}|$, non-negligible. The non-negligible gap of the two cases implies non-negligible advantage of $\mathcal{D}'$, contradiction. Hence, $\Pr[\mathcal{A}(\bar{\Re}_0^\ell) = 1] - \Pr[\mathcal{A}(\bar{\Re}_0^{\ell-1}) = 1]$ is non-negligible. This implies that $\mathcal{D}$ has a non-negligible advantage, contradiction.

The proof that $\Pr(\mathsf{E}(\Re_c)) = negl(\kappa)$ follows almost the same line (without defining x_1) as that for $\Pr[\mathsf{E}(\bar{\Re}_c^\ell)] = negl(\kappa)$. This completes our proof of the lemma. □

Appendix F. Proof of Fact 1

Proof. Let $\mathcal{U}_\ell(z) = \left\{u_\ell \mid \Phi(u_\ell) = z, u_\ell \in \{0,1\}^\ell\right\}$. By definition of (Φ, Φ^*), $\mathsf{dist}[\Phi^*(Z), U_\ell(Z)] = negl(\kappa)$ and $\mathsf{dist}[Z, \Phi(U_\ell)] = negl(\kappa)$. Then,

$$
\begin{aligned}
&\mathsf{dist}[\Phi^*(Z), U_\ell] \\
&\le \mathsf{dist}[\Phi^*(Z), U_\ell(Z)] + \mathsf{dist}[U_\ell(Z), U_\ell] \\
&= negl(\kappa) + \tfrac{1}{2}\textstyle\sum_{u\in\{0,1\}^\ell} \left|\Pr[Z = \Phi(u)] \cdot |\mathcal{U}_\ell(\Phi(u))|^{-1} - 2^{-\ell}\right| \\
&= negl(\kappa) + \tfrac{1}{2}\textstyle\sum_{u\in\{0,1\}^\ell} \left|\left(\Pr[Z = \Phi(u)] - |\mathcal{U}_\ell(\Phi(u))| \cdot 2^{-\ell}\right) \cdot |\mathcal{U}_\ell(\Phi(u))|^{-1}\right| \\
&= negl(\kappa) + \tfrac{1}{2}\textstyle\sum_{z\in\Phi(\{0,1\}^\ell)} \left|\Pr[Z = z] - |\mathcal{U}_\ell(z)| \cdot 2^{-\ell}\right| \\
&\le negl(\kappa) + \tfrac{1}{2}\textstyle\sum_{z\in V} \left|\Pr[Z = z] - |\mathcal{U}_\ell(z)| \cdot 2^{-\ell}\right| \\
&= negl(\kappa).
\end{aligned}
$$

For the first "=", $\Pr[U_\ell(Z) = u] = \Pr[Z = \Phi(u)] \cdot |\mathcal{U}_\ell(\Phi(u))|^{-1}$ holds since $U_\ell(Z) = u$ implies $Z = \Phi(u)$ by definition of $U_\ell(Z)$. The third "=" holds, since $\Phi(u) = z$ for all $u \in \mathcal{U}_\ell(z)$. Notice that $\Phi : \{0,1\}^\ell \to V$ and Z is a variable over V. Hence, the second inequality holds. The last "=" holds since $negl(\kappa) = \mathsf{dist}[Z, \Phi(U_\ell)] = \frac{1}{2}\sum_{z\in V} \left|\Pr[Z = z] - |\mathcal{U}_\ell(z)| \cdot 2^{-\ell}\right|$. □

References

[1] A. Atashpendar, G.V. Policharla, P.B. Rønne, P.Y.A. Ryan, Revisiting deniability in quantum key exchange - via covert communication and entanglement distillation, in: Secure IT Systems - 23rd Nordic Conference, NordSec 2018, in: Lect. Notes Comput. Sci., vol. 11252, Springer, November 2018, pp. 104–120.

[2] M. Bellare, R. Canetti, H. Krawczyk, A modular approach to the design and analysis of authentication and key exchange protocols, in: Proc. 30th Annual ACM Symposium on the Theory of Computing (STOC), ACM, 1998, pp. 419–428.

[3] M. Bellare, A. Palacio, Towards plaintext-aware public-key encryption without random oracles, in: Advances in Cryptology - ASIACRYPT 2004, in: Lect. Notes Comput. Sci., vol. 3329, Springer, December 2004, pp. 48–62.

[4] M. Bellare, P. Rogaway, Entity authentication and key distribution, in: Advances in Cryptology - CRYPTO 1993, in: Lect. Notes Comput. Sci., vol. 773, Springer, August 1993, pp. 232–249.

[5] M. Bellare, P. Rogaway, Random oracle is practical: a paradigm for designing efficient protocols, in: Proc. 1st ACM Conference on Computer and Communications Security (CCS), ACM, 1993, pp. 62–73.

[6] R. Canetti, Universally composable security: a new paradigm for cryptographic protocols, in: Proc. 42nd Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2001, pp. 136–145.

[7] R. Canetti, C. Dwork, M. Naor, R. Ostrovsky, Deniable encryption, in: Advances in Cryptology - CRYPTO 1997, in: Lect. Notes Comput. Sci., vol. 1294, Springer, August 1997, pp. 90–104.

[8] R. Canetti, S. Park, O. Poburinnaya, Fully deniable interactive encryption, in: Advances in Cryptology - CRYPTO 2020, Part I, in: Lect. Notes Comput. Sci., vol. 12170, Springer, August 2020, pp. 807–835.

[9] R. Canetti, T. Rabin, Universal composition with joint state, in: Advances in Cryptology - CRYPTO 2003, in: Lect. Notes Comput. Sci., vol. 2729, Springer, August 2003, pp. 265–281.

[10] O. Chevassut, P. Fouque, P. Gaudry, D. Pointcheval, The twist-augmented technique for key exchange, in: Public Key Cryptography - PKC 2006, in: Lect. Notes Comput. Sci., vol. 3958, Springer, April 2006, pp. 410–426.

[11] R. Cramer, V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, in: Advances in Cryptology - CRYPTO 1998, in: Lect. Notes Comput. Sci., vol. 1462, Springer, August 1998, pp. 13–25.

[12] R. Cramer, V. Shoup, Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption, in: Advances in Cryptology - EUROCRYPT 2002, in: Lect. Notes Comput. Sci., vol. 2232, Springer, May 2002, pp. 45–64.

[13] I. Damgard, Towards practical public key systems secure against chosen ciphertext attacks, in: Advances in Cryptology - CRYPTO 1991, in: Lect. Notes Comput. Sci., vol. 576, Springer, August 1991, pp. 445–456.

[14] A. Dent, The Cramer-Shoup encryption scheme is plaintext aware in the standard model, in: Advances in Cryptology - EUROCRYPT 2006, in: Lect. Notes Comput. Sci., vol. 4004, Springer, May 2006, pp. 289–307.

[15] M. Di Raimondo, R. Gennaro, New approaches for deniable authentication, in: Proc. 12th ACM Conference on Computer and Communications Security (CCS), ACM, 2005, pp. 112–121.

[16] M. Di Raimondo, R. Gennaro, New approaches for deniable authentication, J. Cryptol. 22 (2009) 572–615, https://doi.org/10.1007/s00145-009-9044-3.

[17] M. Di Raimondo, R. Gennaro, H. Krawczyk, Deniable authentication and key exchange, in: Proc. 13th ACM Conference on Computer and Communications Security (CCS), ACM, 2006, pp. 400–409.

[18] W. Diffie, M. Hellman, New directions in cryptography, IEEE Trans. Inf. Theory 22 (1976) 644–654, https://doi.org/10.1109/TIT.1976.1055638.

[19] Y. Dodis, J. Katz, A. Smith, S. Walfish, Composability and on-line deniability of authentication, in: Theory of Cryptography, 6th Theory of Cryptography Conference (TCC), in: Lect. Notes Comput. Sci., vol. 5444, Springer, March 2009, pp. 146–162.

[20] D. Dolev, C. Dwork, M. Naor, Non-malleable cryptography, SIAM J. Comput. 30 (2) (2000) 391–437, https://doi.org/10.1137/S0097539795291562.

[21] C. Dwork, M. Naor, A. Sahai, Concurrent zero-knowledge, in: Proc. 30th Annual ACM Symposium on the Theory of Computing (STOC), ACM, 1998, pp. 409–418.
[22] C. Dwork, A. Sahai, Concurrent zero-knowledge, reducing the need for timing constraints, in: Advances in Cryptology - CRYPTO 1998, in: Lect. Notes Comput. Sci., vol. 1462, Springer, August 1998, pp. 442–457.
[23] O. Goldreich, S. Goldwasser, S. Micali, How to construct random functions, J. ACM 33 (4) (1986) 792–807, https://doi.org/10.1145/6490.6503.
[24] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems, SIAM J. Comput. 18 (1) (1989) 186–208, https://doi.org/10.1137/0218012.
[25] D. Harkins, C. Kaufman, T. Kivinen, S. Kent, R. Perlman, Design rationale for IKEv2, Internet Draft, https://datatracker.ietf.org/doc/html/draft-ietf-ipsec-ikev2-rationale, 2002. (Accessed 4 January 2022).
[26] D. Hofheinz, E. Kiltz, Secure hybrid encryption from weakened key encapsulation, in: Advances in Cryptology - CRYPTO 2007, in: Lect. Notes Comput. Sci., vol. 4622, Springer, August 2007, pp. 553–571.
[27] S. Jiang, Timed encryption with application to deniable key exchange, Theor. Comput. Sci. 560 (2014) 172–189, https://doi.org/10.1016/j.tcs.2014.02.005.
[28] S. Jiang, R. Safavi-Naini, An efficient deniable key exchange protocol, in: Financial Cryptography and Data Security, 12th International Conference, FC 2008, in: Lect. Notes Comput. Sci., vol. 5143, Springer, January 2008, pp. 47–52.
[29] S. Jiang, H. Wang, Plaintext-awareness of hybrid encryption, in: Topics in Cryptology - CT-RSA 2010, in: Lect. Notes Comput. Sci., vol. 5985, Springer, March 2010, pp. 57–72.
[30] J. Katz, Efficient and non-malleable proofs of plaintext knowledge and applications, in: Advances in Cryptology - EUROCRYPT 2003, in: Lect. Notes Comput. Sci., vol. 2656, Springer, May 2003, pp. 211–228.
[31] J. Katz, R. Ostrovsky, M. Yung, Efficient password-authenticated key exchange using human-memorable passwords, in: Advances in Cryptology - EUROCRYPT 2001, in: Lect. Notes Comput. Sci., vol. 2045, Springer, May 2001, pp. 475–494.
[32] H. Krawczyk, SKEME, a versatile secure key exchange mechanism for Internet, in: Proc. 1996 Symposium on Network and Distributed System Security (NDSS), IEEE, 1996, pp. 114–127.
[33] H. Krawczyk, SIGMA: the 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE-protocols, in: Advances in Cryptology - CRYPTO 2003, in: Lect. Notes Comput. Sci., vol. 2729, Springer, August 2003, pp. 400–425.
[34] H. Krawczyk, HMQV: a high-performance secure Diffie-Hellman protocol, in: Advances in Cryptology - CRYPTO 2005, in: Lect. Notes Comput. Sci., vol. 3621, Springer, August 2005, pp. 546–566.
[35] K. Kurosawa, Y. Desmedt, A new paradigm of hybrid encryption scheme, in: Advances in Cryptology - CRYPTO 2004, in: Lect. Notes Comput. Sci., vol. 3152, Springer, August 2004, pp. 426–442.
[36] W. Mao, K. Paterson, On the plausible deniability feature of Internet protocols, manuscript, https://www.isg.rhul.ac.uk/~kp/IKE.ps, 2002. (Accessed 4 January 2022).
[37] M. Naor, Deniable ring authentication, in: Advances in Cryptology - CRYPTO 2002, in: Lect. Notes Comput. Sci., vol. 2442, Springer, August 2002, pp. 481–498.
[38] A. O'Neill, C. Peikert, B. Waters, Bi-deniable public-key encryption, in: Advances in Cryptology - CRYPTO 2011, in: Lect. Notes Comput. Sci., vol. 6841, Springer, August 2011, pp. 525–542.
[39] R. Pass, On the deniability in the common reference string and random oracle model, in: Advances in Cryptology - CRYPTO 2003, in: Lect. Notes Comput. Sci., vol. 2729, Springer, August 2003, pp. 316–337.
[40] W. Shi, Y. Zhou, Y. Yang, Quantum deniable authentication protocol, Quantum Inf. Process. 13 (2014) 1501–1510, https://doi.org/10.1007/s11128-014-0743-9.
[41] R. Steinfeld, J. Pieprzyk, H. Wang, On the provable security of an efficient RSA-based pseudorandom generator, in: Advances in Cryptology - ASIACRYPT 2006, in: Lect. Notes Comput. Sci., vol. 4284, Springer, December 2006, pp. 194–209.
[42] N. Unger, I. Goldberg, Deniable key exchanges for secure messaging, in: Proc. 22nd ACM Conference on Computer and Communications Security (CCS), ACM, 2015, pp. 1211–1223.
[43] N. Unger, I. Goldberg, Improved strongly deniable authenticated key exchanges for secure messaging, Proc. Priv. Enh. Technol. 2018 (1) (2018) 21–66, https://doi.org/10.1515/popets-2018-0003.
[44] N. Williams, A pseudo-random function (PRF) for the Kerberos V generic security service application program interface (GSS-API) mechanism, RFC 4402, https://datatracker.ietf.org/doc/html/rfc4402, 2006. (Accessed 4 January 2022).
[45] A. Yao, Y. Zhao, Deniable Internet key exchange, in: Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, in: Lect. Notes Comput. Sci., vol. 6123, Springer, June 2010, pp. 329–348.