J. Math. Cryptol. **8** (2014), 31 – 70
DOI 10.1515/jmc-2012-0010

Persistent asymmetric password-based key exchange

Shaoquan Jiang

Communicated by Kaoru Kurosawa

Abstract. Asymmetric password based key exchange is a key exchange protocol where a client and a server share a low entropic password while the server additionally owns a high entropic secret with respect to a public key. There are simple solutions for this, e.g., [18] and its improvement in [7]. In the present paper, we consider a new threat to this type of protocol: if a server's high entropic secret gets compromised (e.g., due to cryptanalysis or a poor management), the adversary might *quickly* break lots of passwords and cause uncountable damage. In this case, one should not expect the protocol to be secure against an off-line dictionary attack since, otherwise, the protocol is in fact a secure password-only key exchange by making the server high entropic secret public. Of course a password-only key exchange does not suffer from this threat as the server does not have a high entropic secret at all. However, known password-only key exchange protocols are not very efficient (note: we only consider protocols without random oracles). This motivates us to study an efficient and secure asymmetric password key exchange that avoids the new threat. In this paper, we first provide a formal model for the new threat, where essentially we require that the active adversary can break ℓ passwords in $\alpha\ell|\mathcal{D}|$ steps (for $\alpha < 1/2$) only with a probability negligibly close to $\exp(-\beta\ell)$ for some $\beta > 0$, where $\mathcal{D}$ is a password dictionary. Then, we construct a framework of asymmetric password based key exchange. We prove that our protocol is secure in the regular model where server high entropic key is never compromised and that it prevents the new threat. To do this, we introduce a new technique by abstracting a probabilistic experiment from the main proof and providing a neat analysis of it.

Keywords. Cryptographic protocol, password-based key exchange, provable security, projective hash function, dictionary attack.

2010 Mathematics Subject Classification. 68M12, 94A60, 94A62.

This work is supported by NSFC (No. 60973161), Fundamental Research Funds for the Central Universities (No. ZYGX2010X015) and National 973 Program of China (No. 2013CB834203).

1 Introduction

Key exchange (KE) is one of the most important issues in secure communication. It helps two communicants to securely establish a common session key, with which the subsequent communication can be protected. In the literature, there are two types of key exchange. In type one, two parties own high entropic secrets. This type has been extensively studied in the literature; see a very partial list [2, 8, 11, 26]. Type two is password authenticated key exchange, in which it is assumed that the two parties share a human-memorable (low entropy) password. The major threat for this type of key exchange is an off-line dictionary attack. In this case, an adversary can catch a function value of the password (say, $F(pw)$). Since the password space is small, he can find the matching password through an exhaustive search. See [1] for an example. In the literature, two classes of password key exchange protocols are studied. In the first class, two parties only own a common password. This class is studied extensively in the literature. In the second class, the client and server share a password while the server additionally owns a high entropic private key of a public key. In this class, there are simple solutions [7, 18]. In the present paper, we consider a new threat to this class of protocols: when the server high entropic secret is compromised, the attacker might quickly break lots of passwords and cause uncountable damage. When the above threat occurs, it is desired that the pace an attacker breaks passwords is very slow. Under this, the server management will have enough time to realize and defend the attack. Unfortunately, previous protocols (e.g., [7, 18]) are not secure against this. Of course, since we intend to prevent the adversary from breaking a lot of passwords, the problem is meaningful only if the system has a lot of clients where Google, Yahoo and MSN are good examples.

1.1 Motivation

In the above new threat, the problem is meaningful only if the server high entropic secret is compromised while the server password table is safe. This could happen in the following scenarios.

1. *Temporary access to server.* In our daily life, it is not surprising that we temporarily leave our computer system on and unlocked (e.g., for a coffee break). If a server management does this, an attacker (also a user) could take the chance to copy the temporary internal state of the server (which does not need the access code) but he can not copy the password file as it further requires the access code. In this setting, we show that the attacker could manage to break the server key. To be concrete, we assume the server has public/private key pair $(A = g^a, a)$ where g is a generator of a prime group G with $|G| = q$. Suppose the server has a

key exchange protocol that includes the following identification from server (S) to client (C) as a subroutine:

(1) $S \to C$: $X = g^x$;

(2) $C \to S$: $b \leftarrow \mathbb{Z}_q$;

(3) $S \to C$: $y = ab + x$;

(4) C: accept if and only if $g^y = A^b X$.

Now the adversary can run the following attack. He initiates a key exchange with server in his own name (as a user). After he receives X, he does not reply b immediately. Instead, he copies the server internal state, which must include x as it will be later used to compute y. After this, he sends b to S and receives y. From (y, b, x), the attacker can easily compute a.

2. *Cryptanalysis.* An adversary could conduct cryptanalysis directly on the server high entropic key. One might think such a success is unlikely as such a computation resource can be used to directly break the server. However, this is not true always. Although the security of the server high entropic key is usually guaranteed by its underlying security assumption, this holds only under the condition that the server public/pair key is chosen properly. For example, a secure RSA system requires that the prime factors p and q should be of nearly equal size and that $p - 1$, $q - 1$ should contain large prime factors and that the encryption key e should not be small. Conceivably, more restrictions will be imposed with the development of cryptanalysis. However, we can not expect industry engineers always to be aware of such updates. One miss of this or an old system that violates a new restriction could give a cryptanalyst a good chance to break it. In addition, more computation resource does not imply an easy break-in to the server. For the case of a key exchange protocol, one can set a threshold on the number of the consecutive failures, beyond which the user will be denied.

3. *Fault analysis.* Boneh, Demillo and Lipton [6] studied the fault analysis and showed that this might allow an attacker to efficiently break the secret key of a cryptographic system. They mentioned three types of faults and two of them are applicable to our setting.

(a) *Latent fault.* Latent faults are hardware or software bugs that are difficult to catch (e.g., Intel's floating point division bug). Crypto library using such a bugged unit is likely to produce incorrect values.

(b) *Transient fault.* Transient faults are random hardware glitches that cause the processor to miscalculate. They might be caused due to power glitches, high temperature, static electricity and more. Under such faults, a bit in the register could be flipped.

Usually, faulty errors are rare to happen. However, here "rare" only refers to the small frequency of occurrences. Over the long course of the server's life time, the probability that the faulty error occurs *once* could be very high. Also once it happens, the consequence could be very serious. Further, if an attacker has the chance to stay close to the server, he could make transient faults occur by increasing temperature or inducing static electricity.

To see the threat of fault analysis, we introduce the attack in [6] (improved by Arjen Lenstra) on RSA that is implemented using the Chinese Remainder Theorem (i.e., RSA-CRT). This attack breaks RSA using only one faulty error on a single bit. Consider an RSA signature system $(N = pq, e, d)$, where e is the verification key and d is the signing key. To sign message m, compute $x = h(m)$ with a hash function h, then compute $S_1 = x^d \bmod p$ and $S_2 = x^d \bmod q$ and finally use CRT to merge S_1, S_2 to obtain $S = x^d \bmod N$. Assume that computation of S_1 is faulty such that $S_1 = (x^d + 2^i) \bmod p$ for some $i < |p|$ (i.e., one bit faulty error occurs). Notice that $S^e = S_1^e \neq x \bmod p$ while $S^e = S_2^e \bmod q = x$. It follows that $\gcd(x - S^e, N) = q$. This factors N that is based on (m, e, S) only.

1.2 Related work

The server key leakage problem does not occur in the password-only key exchange protocol since in this setting the server does not own a high entropic secret key at all. Hence, an asymmetric password key exchange against this threat is meaningful only if we have a construction that is more efficient than the known password-only protocols. Password-only key exchange was first studied by Bellovin and Merritt [4] and further studied in [5, 21, 28]. The first provably secure solution is due to Bellare et al. [3] but security holds in the random oracle model which is not our main focus. The first key exchange without random oracles is due to Goldreich and Lindell [14]. But it is very inefficient. The first reasonably efficient solution without random oracles is the KOY protocol [23] which has 16 exponentiations for a client and 15 exponentiations for a server. This protocol was abstracted into a framework by Gennaro and Lindell [13] and improved by Gennaro [12] (the contribution of the latter is to remove the signature), where each party costs 12 exponentiations when realizing their scheme by building a hashing proof system over N-residuosity-based CCA2 encryption [13] (note: although there are faster encryptions [20, 27], it is not clear how to build a hash proof system over this type of encryption). Jiang and Gong [22] (recently abstracted into a framework by [16]) constructed an efficient protocol. When using encryption in [20], both schemes cost 6 exponentiations for a client and 6 exponentiations for a server. Katz and Vaikuntanathan [25] constructed a one-round password-only key exchange which is the most appealing feature. But it is not efficient because in their currently best realization each party needs 12 exponentiations and one simulation-sound ZK proof.

The asymmetric password based technique was initiated by Gong [15]. Halevi and Krawczyk [17] (also full version [18]) proposed a very efficient asymmetric password based key exchange, which essentially let the client use a CCA2 secure encryption to encrypt the password information. Using encryption [20], this protocol only needs about two exponentiations for the client and one exponentiation for the server. It was later extended by Boyarsky [7] for security in the multi-user setting. However, neither of the two protocols can prevent the new threat as the password is encrypted under a server public key and can be easily decrypted if its private key is leaked.

1.3 Contribution

We first provide a formal model for the above server key leakage problem. We essentially require that an adversary can break ℓ passwords in $\alpha\ell|\mathcal{D}|$ steps (for $\alpha < 1/2$) only with probability negligibly close to $\exp(-\beta\ell)$ for some $\beta > 0$, where $\mathcal{D}$ is the password dictionary and has a size independent of security parameter. Under this assertion, the adversary can not quickly break lots of passwords. An asymmetric password key exchange protocol with this property is said to be *persistent*. Then, we construct a framework of asymmetric password based key exchange. Our construction is based on a tag-based projective hash family that is modified from the projective hash family (PHF) of Cramer–Shoup. We show that our framework is secure in the multi-user setting of [7] (under a different formalization, where our contribution is a new quantification on the authentication failure). Our proof does not rely on the random oracles. We also prove that our framework is persistent, where our main technical novelty is a probabilistic experiment and we provide a neat analysis for this experiment. Our persistency holds in the random oracle model. It is open to construct a protocol whose security and persistency both hold without random oracles. We instantiate our framework with a concrete tag-PHF. Our realization only costs 5 exponentiations for the client and two exponentiations for the server, which is significantly more efficient than all known password-only schemes. The efficiency of password-only protocols is surveyed in the previous subsection. A comparison between these protocols and ours is summarized in Table 1, where the costs are computed under each protocol's currently best realization (e.g., public key encryption in [16, 22] uses [20] that only has two exponentiations for encryption cost). In this table, the password exponentiation g^{π} in [16, 22, 23] and our work is assumed to store in the server (but not the client as he can not memorize this long secret). We can tell that our protocol

Schemes	ClientCost (exp)	ServerCost (exp)	Framework	MutualAuth
[22]	6	6	No	Yes
[16]	6	6	Yes	Yes
[25]	≥ 12	≥ 12	Yes	No
[12]	12	12	Yes	No
[23]	16	15	No	No
ours	5	2	Yes	Yes

Table 1. Comparison between our protocol and password-only protocols.

is significantly more efficient than known password-only protocols although the price is to let the server hold a high entropic secret.

Notions. $x \leftarrow S$ samples x from S randomly; $A|B$ means concatenating A with B. We use negl : $\mathbb{N} \rightarrow \mathbb{R}$ to denote a *negligible* function: for any polynomial $p(x)$, $\lim_{n\rightarrow\infty} \text{negl}(n)p(n) = 0$. For two functions f, g from $\mathbb{N}$ to $\mathbb{R}$, write $f(n) \approx g(n)$ if $f(n) - g(n)$ is negligible. The probability distance of two random variables A, B over set Ω is defined as

$$\mathsf{dist}[A, B] = \frac{1}{2}\sum_{v\in\Omega}|\Pr[A = v] - \Pr[B = v]|.$$

We say that random variables A, B are statistically close if $\mathsf{dist}[A, B]$ is negligible. For $a \in \mathbb{N}$, define $[a] = \{1, \ldots, a\}$. PPT means probabilistic polynomial time.

2 Security model

2.1 Model when server high entropic key is not compromised

In this section, we introduce a security model for asymmetric password key exchange, which is slightly modified from the password-only setting of Bellare et al. [3]. There are n clients $C_1, \ldots, C_n$ and one server S, where S has a public key Θ (known to all C_i) and a private key θ. It also shares a password π_i with C_i. The server high entropic key θ is assumed uncorrupted.

- $\mathcal{D}$ is a password dictionary. For simplicity, let $\mathcal{D} = \{1, \ldots, N\}$ with a uniform distribution.
- $\Pi_U^{\ell_U}$ is the ℓ_Uth protocol instance in party U, where U is either a client i or a server S.

- Flow_i or msg_i is the ith message in the protocol execution.
- $\mathsf{sid}_U^{\ell_U}$ is the session identifier of $\Pi_U^{\ell_U}$, where U is either a client i or server S. Intuitively, two jointly executing instances have identical sid.
- $\mathsf{sk}_U^{\ell_U}$ is the session key defined by instance $\Pi_U^{\ell_U}$.
- $\mathsf{pid}_U^{\ell_U}$ is the party, with which $\Pi_U^{\ell_U}$ presumably interacts.
- $\mathsf{stat}_U^{\ell_U}$ is the internal state of $\Pi_U^{\ell_U}$ (not including the long term secret).
- $\mathsf{Client}(\Pi_U^{\ell_U})$. We know that for any $\Pi_U^{\ell_U}$, either U or $\mathsf{pid}_U^{\ell_U}$ (but not both) is a client. Hence, it is well-defined if we use $\mathsf{Client}(\Pi_U^{\ell_U})$ to denote this client.

Partnering. $\Pi_U^{\ell_U}$ and $\Pi_V^{\ell_V}$ are *partnered* if

(1) $\mathsf{pid}_U^{\ell_U} = V$ and $\mathsf{pid}_V^{\ell_V} = U$;

(2) $\mathsf{sid}_U^{\ell_U} = \mathsf{sid}_V^{\ell_V}$.

Adversarial model. The capability of adversary $\mathcal{A}$ is defined as follows. He fully controls the network. He can inject, modify, block messages. He can also request any session key. Formally, his behaviors are modeled as access to the following oracles.

$\mathsf{Execute}(i, \ell_i, S, \ell_S)$. When this oracle is called, a protocol execution between $\Pi_i^{\ell_i}$ and $\Pi_S^{\ell_S}$ takes place. Finally, a complete transcript is returned. This oracle call models an eavesdropping attack.

$\mathsf{Send}(d, U, \ell_U, M)$. Upon this query, M is sent to $\Pi_U^{\ell_U}$ as msg_d. This query models active attacks.

$\mathsf{Reveal}(U, \ell_U)$. Upon this query, session key $\mathsf{sk}_U^{\ell_U}$ (if any) is returned. This models a session key loss attack.

$\mathsf{Corrupt}(i)$. Upon this query, the password π_i of C_i as well as his session states $\{\mathsf{stat}_i^{\ell_i}\}_{\ell_i}$ are given to the adversary. After this, he is no longer active. This models a break-in or insider attack.

$\mathsf{Test}(U, \ell_U)$. This query is a security test for session key $\mathsf{sk}_U^{\ell_U}$. The adversary is allowed to query it only once. The queried session must have been successfully completed. Throughout the game, U and $\mathsf{pid}_U^{\ell_U}$ should not be corrupted; $\Pi_U^{\ell_U}$ and its partnered instance (if any) should not be issued a Reveal query. When Test oracle is called, it flips a fair coin b. If $b = 1$, then $\mathsf{sk}_U^{\ell_U}$ is provided to the adversary; otherwise, a random number from the session key space is provided.

The adversary then tries to output a guess bit b'. If $b' = b$, he will be informed $\mathsf{Success}$; otherwise, Fail.

We now define the protocol security, which considers three properties: correctness, authentication and secrecy.

Correctness. If two partners accept, they derive the same session key.

Authentication. If some $\Pi_U^{\ell_U}$, with U and $\mathsf{pid}_U^{\ell_U}$ both uncorrupted, has been successfully completed while it does not have a *unique* partnered instance, then we say authentication is *broken* and denote this event by $\mathsf{Non\text{-}Auth}$. Further, we use $\mathsf{Non\text{-}Auth}_i$ to denote event $\mathsf{Non\text{-}Auth}$ such that $\mathsf{Client}(\Pi_U^{\ell_U}) = P_i$. Note that since the password dictionary $\mathcal{D}$ is small, one can always break the authentication by guessing a password π_i of P_i and impersonating P_i to S (through Send queries). So if Q_i denotes the number of Send queries in which client is P_i, then trivially, $\mathsf{Non\text{-}Auth}_i$ can be achieved with probability $\frac{Q_i}{|\mathcal{D}|}$. Authentication property is to require that this is the best possible success. Formally, the protocol is *authenticated* if

$$\Pr(\mathsf{Non\text{-}Auth}_i) \leq \frac{Q_i}{|\mathcal{D}|} + \mathrm{negl}(\lambda), \quad i = 1, \ldots, n. \tag{2.1}$$

Secrecy. Denote the adversary success in a Test query by Succ. Let $\mathsf{Non\text{-}Auth} = \vee_{i=1}^n \mathsf{Non\text{-}Auth}_i$. We only consider succ under $\neg\mathsf{Non\text{-}Auth}$ as $\mathsf{Non\text{-}Auth}_i$ is already measured by authentication. Then, a protocol is *secrecy* if

$$\Pr[\mathsf{Succ}(\mathcal{A})|\neg\mathsf{Non\text{-}Auth}] < 1/2 + \mathrm{negl}(\lambda).$$

The security definition is summarized as follows.

Definition 1. Let Q_i be the number of $\mathsf{Send}(U, \ell_U, \cdot)$ queries such that $P_i = \mathsf{Client}(\Pi_U^{\ell_U})$. Then, an asymmetric password-based key exchange protocol is *secure* if it satisfies

- Correctness.
- Authentication.

$$\Pr[\mathsf{Non\text{-}Auth}_i] \leq \frac{Q_i}{|\mathcal{D}|} + \mathrm{negl}(\lambda), \quad i = 1, \ldots, n.$$

- Secrecy.

$$\Pr[\mathsf{Succ}(\mathcal{A}) \mid \neg\mathsf{Non\text{-}Auth}] < 1/2 + \mathrm{negl}(\lambda).$$

Remark. (1) Note that in our model the server S is never corrupted; otherwise, nothing can be protected as S knows all passwords of clients.

(2) Let Q be the number of Send queries. Then, $Q = \sum_{i=1}^{n} Q_i$. Hence, our authentication property implies that $\Pr[\mathsf{Non\text{-}Auth}] < \frac{Q}{|\mathcal{D}|} + \mathrm{negl}(\lambda)$. This further indicates that $\Pr[\mathsf{Succ}(\mathcal{A})] < 1/2 + \frac{Q}{2|\mathcal{D}|} + \mathrm{negl}(\lambda)$, which is the security definition [3] for the password-only key exchange setting.

(3) We do not use $\Pr[\mathsf{Non\text{-}Auth}] < \frac{Q}{|\mathcal{D}|} + \mathrm{negl}(\lambda)$ as our authentication definition due to the attack by Boyarsky [7] against [18]. His idea is to eavesdrop a communication transcript *tr* between C_j and S. Then he corrupts C_i and obtains π_i. Next, he uses *tr* to conduct the execution between C_i and S from which π_j will be compromised. In term of our model, he queries $\mathsf{Execute}(j, \ell_j, S, \ell_S)$ oracle once and can break π_j when $Q_j = 0$ (although Q_i could be large). Under our definition, this will not occur.

2.2 Model when server high entropic key is compromised

Motivation. We now formalize the security where the server key θ gets compromised. This is possible due to cryptanalysis or a poor management. In the introduction, we have outlined the possibilities based on hardware fault or temporary server data for the protocol execution. Under such an attack, we can not hope that the protocol is secure against an off-line dictionary attack as otherwise the protocol is in fact a secure password-only protocol (by making the server secret public). We thus are only interested in a weaker guarantee: the adversary should not be able to break lots of passwords quickly. Under this, the server manager can have enough time to realize and defend the attack. Previous protocols [7,17,18] do not prevent this threat as they essentially encrypt a password under public key Θ.

It is desired that if an attacker intends to break ℓ passwords, he has to do so using an dictionary attack individually on each password and with average costs $\ell|\mathcal{D}|/2$ dictionary guesses. That is, if any adversary runs $T < \alpha\ell|\mathcal{D}|$ steps, then he should not be able to break ℓ passwords, where one step is essentially the cost of one dictionary guess and will be defined when the protocol description is available. Qualitatively, it is desired that his success probability is bounded by $\exp(-\beta\ell) + \mathrm{negl}(\lambda)$ for some $\beta > 0$. Also note that since ℓ does not necessarily depend on the security parameter λ, we can not simply require the above adversarial success probability be $\mathrm{negl}(\lambda)$. We notice that it is hard to tell whether an adversary has broken a password π_i or not. Hence, we can not directly use this definition. However, if this occurs, it should be easy for him to successfully impersonate client i, in which case $\mathsf{Non\text{-}Auth}_i$ occurs. Hence, we instead define the adversary success as the occurrence of $\mathsf{Non\text{-}Auth}_i$ for at least ℓ different i. Finally, we define the adversary capability. Since persistency only considers an attack that occurs under a very rare event and lasts only in a short time, oracle queries other than Send are immaterial.

Formal definition. Our formal definition of persistency is presented as follows, where event $\mathsf{Non\text{-}Auth}_i$ is the same as in the definition of authentication property.

Definition 2. Let $\ell \in \mathbb{N}$ and $\alpha < 1/2$. Assume that Ξ is an asymmetric password-based key exchange protocol. Then Ξ is *persistent* if for any PPT adversary $\mathcal{A}$ that is given system parameter param and server high entropic private/public key pair (θ, Θ) and runs $T < \ell\alpha|\mathcal{D}|$ steps with access to Send oracles, the probability that $\mathsf{Non\text{-}Auth}_i$ for ℓ different i occur is upper bounded by $\exp(-\beta\ell) + \mathrm{negl}(\lambda)$ for some $\beta > 0$, where a basic step is specified in a concrete protocol.

Remark. Persistency is to capture the guarantee that it is impossible for an adversary to recover a lot of passwords in a short time. So the definition is only meant for large ℓ. Even though, we do not choose to define the persistency like this: for any $\ell > \lambda$, the adversary succeeds only negligibly. This is so because under such a definition, it is not clear for a fixed ℓ whether the choice of λ (in order to guarantee a certain persistency probability) heavily depends on ℓ or not. Under our definition, this dependence does not exist as the contribution from ℓ is exactly $\exp(-\beta\ell)$ for a constant $\beta > 0$. Our definition also shows that the probability impact from ℓ approaches zero exponentially. That is, slowly increasing ℓ will render the adversary success probability approaching zero fast. This impact is missing if we only require the adversary success probability to be negligible.

3 Tag-based hash proof system

In this section, we introduce a *tag-based hash proof system*, revised from the original hash proof system [10] (in fact the brief introduction in [13] suffices) by adding a tag. Special forms of hash proof system are used by [12, 13, 16, 24, 25] to construct password-only key exchange protocols.

3.1 Subset membership problem

A hard subset membership problem is a problem that one can efficiently sample a hard instance. Formally, *a subset membership problem* $\mathcal{I}$ is a collection $\{\mathcal{I}_n\}_{n\in\mathbb{N}}$, where $\mathcal{I}_n$ is a distribution for a random variable Λ_n defined as follows:

- Generate a *finite* non-empty set $X_n, L_n \subseteq \{0,1\}^{\mathrm{poly}(n)}$ such that $L_n \subset X_n$, and distribution $D(L_n)$ over L_n and distribution $D(X_n \backslash L_n)$ over X_n.

- Generate a witness set $W_n \subseteq \{0,1\}^{\mathrm{poly}(n)}$ and an NP-relation $R_n \subseteq X_n \times W_n$ such that $x \in L_n$ if and only if there exists $w \in W_n$ such that $(x, w) \in R_n$. Here L_n can be sampled in polynomial time according to distribution $D(L_n)$ which outputs $x \in L_n$ and its witness w. We use

$$x \overset{w}{\leftarrow} D(L_n)$$

to denote this procedure, and omit w if w is not our interest. Further, $x \leftarrow D(X_n \backslash L_n)$ can be also sampled in polynomial time.

Finally let

$$\Lambda_n = \langle X_n, L_n, W_n, R_n, D(L_n), D(X_n \backslash L_n)\rangle.$$

We say that $\mathcal{I} = \{\mathcal{I}_n\}_{n\in\mathbb{N}}$ is a *hard subset membership problem* if for $\langle X_n, L_n, W_n, R_n, D(L_n), D(X_n \backslash L_n)\rangle \leftarrow \mathcal{I}_n$, x and y are indistinguishable for $y \leftarrow D(X_n \backslash L_n)$, $x \leftarrow D(L_n)$.

3.2 Tag-based projective hash function

Let $\Lambda = \langle X, L, W, R, D(L), D(X\backslash L)\rangle$ be sampled from a hard subset membership problem $\mathcal{I}_n$. Consider a tuple $\Psi = \langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha\rangle$, where $G, S, \mathcal{K}$ are finite non-empty sets, $\mathcal{H} = \{H_k(\cdot,\cdot) \mid k \in \mathcal{K}\}$ is a set of functions from $\{0,1\}^* \times X$ to G and $\alpha : \mathcal{K} \to S$ is a deterministic function and $\alpha(k)$ is called a *projection*. $\mathcal{K}$ is called a *key space*, $k \in \mathcal{K}$ is called the *hash key*; S is called the *projection space* for α. We call Ψ a *tag-based projective hash function* (tag-PHF) for Λ if for any $x \in L$ and tag $z \in \{0,1\}^*$, $H_k(z, x)$ is uniquely determined by $\alpha(k), z, x$. It is called an *efficient* tag-PHF if $\alpha(k)$ and $H_k(z, x)$ are both polynomially computable from (k, x, z) and if $H_k(z, x)$ also is polynomially computable from $x, w, \alpha(k), z$ where $(x, w) \in R$. In this paper, by tag-PHF, we mean an efficient tag-PHF. For simplicity, we also directly use $H_k(\cdot,\cdot)$ to represent the underlying tag-PHF.

The following notion of *computational universal*$_2$ is slightly revised from [19], which in turn is extended from the notion of *universal*$_2$ [10] by relaxing the statistical indistinguishability to the computational indistinguishability.

Definition 3. Let $\Lambda = \langle X, L, W, R, D(L), D(X\backslash L)\rangle \leftarrow \mathcal{I}_n$, where $\{\mathcal{I}_n\}_n$ is a hard subset membership problem. Assume that $\Psi = \langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha\rangle$ is a tag-based projective hash function for Λ. We say that Ψ is *computational universal*$_2$ if for any PPT $\mathcal{A}$ with access to the oracles below, $\Pr(b' = b) \approx 1/2$. Initially, $\mathcal{A}$ is given $(\Psi, \alpha(k))$ for $k \leftarrow \mathcal{K}$.

$\mathsf{Eval}_1(x, z)$. Upon query (x, z), check if $x \in L$ (maybe in exponential time). If yes, return $H_k(z, x)$; otherwise $\perp$.

$\mathsf{Eval}_2(x_1, z_1)$. This oracle can be queried once and $x_1 \in X \backslash L$. Upon this, return $H_k(z_1, x_1)$.

$\mathsf{Test}(x_2, z_2)$. This is the test oracle and can be queried once. It requires that $(z_2, x_2) \neq (z_1, x_1)$ and $x_2 \in X \backslash L$. Upon this, take $b \leftarrow \{0, 1\}$, $K_1 \leftarrow G$ and compute $K_0 = H_k(z_2, x_2)$. Finally return K_b.

Finally, $\mathcal{A}$ outputs bit b' and succeeds if $b' = b$.

3.3 A useful lemma

Consider a hard subset membership problem $\{\mathcal{I}_\lambda\}_\lambda$. Assume that $\Psi = \langle \mathcal{H}, \mathcal{K}, X, L, G, S, \alpha \rangle$ is a tag-based PHF for Λ, where

$$\Lambda = \langle X, L, W, R, D(L), D(X \backslash L) \rangle \leftarrow \mathcal{I}_\lambda \quad \text{and} \quad G = \{0, 1\}^{2\lambda}.$$

Here Ψ has a description desc(Ψ). Let $\mathsf{MAC}_K : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ be a message authentication code with secret key $K \in \{0, 1\}^\lambda$.

Consider an experiment EXP involving an adversary $\mathcal{A}$ who can *adaptively* access to the following two oracles. Initially, let $k \leftarrow \mathcal{K}$ and provide $(\alpha(k)$, desc$(\Psi))$ to $\mathcal{A}$. Let $\Theta = \{\}$ and a challenge bit $c \leftarrow \{0, 1\}$.

$\mathsf{Chal}(z)$. Upon this, take $x \stackrel{w}{\leftarrow} D(L)$, compute $(a_0, s_0) = H_k(z, x)$, $(a_1, s_1) \leftarrow \{0, 1\}^{2\lambda}$, return (x, a_c, s_c) and update $\Theta = \Theta \cup \{(z, x, a_c, s_c)\}$.

$\mathsf{Comp}(z, x, \sigma, m)$. Upon this, if $(z, x, a', s') \in \Theta$ for some a', s', let $a = a'$, $s = s'$; otherwise, let $(a, s) = H_k(z, x)$. If $\sigma = \mathsf{MAC}_a(m)$, return (a, s); otherwise $\perp$.

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

The above experiment is to distinguish many $\{H_k(z, x)\}_{x \in L}$ from random, given access to many values $\{H_k(z, x)\}_{x \in X}$ (x not necessary from L) provided that the query issuer can prove that he has some knowledge about $H_k(z, x)$. The following lemma states that the adversary only has a negligible advantage in this task. The proof idea is outlined as follows.

Consider a hybrid experiment EXP^ℓ where the reply to the first $\ell - 1$ Chal queries is (a_1, s_1) while the reply to the remaining Chal queries is (a_0, s_0). By a hybrid argument, it suffices to show that for all ℓ, adversary advantages in EXP^ℓ and $\mathsf{EXP}^{\ell-1}$ are negligibly close. We revise EXP^ℓ to $\widehat{\mathsf{EXP}}^\ell$ such that in the ℓth Chal query, $x \leftarrow D(X \backslash L)$ in the latter (instead of $x \leftarrow D(L)$ in the former). This revision does not change adversary advantage from the hardness of $\mathcal{I}_\lambda$. So we only need to show that adversary advantages in $\widehat{\mathsf{EXP}}^{\ell}$ and $\widehat{\mathsf{EXP}}^{\ell-1}$ are negligibly close. This can be reduced to the computational universal$_2$ property of the hash proof system. To do this, we simulate $\widehat{\mathsf{EXP}}^{\ell}$ under the help of computational universal$_2$ challenger. The ith $\mathsf{Chal}(z)$ query for $i \neq \ell$ can be handled easily as we know the witness w for x. The ℓth such a query can be handled using a test query to computation universal$_2$ (CU$_2$) challenger and so it is easy too. $\mathsf{Comp}(z, x, \sigma, m)$ for $x \in L$ is easy under the evaluation help of his CU$_2$ challenger. $\mathsf{Comp}(z, x, \sigma, m)$ for $x \notin L$ can be rejected simply as $H_k(z, x)$ is indistinguishable from random in view of the attacker (recall only one evaluation query for $x \notin L$ to CU$_2$ has been used now and so σ is valid only negligibly, unless MAC is forgeable). When challenge bit b in CU$_2$ challenger is 0, then the simulated game is $\widehat{\mathsf{EXP}}^{\ell-1}$; $\widehat{\mathsf{EXP}}^{\ell}$ otherwise. Hence, distinguishability between $\widehat{\mathsf{EXP}}^{\ell}$ and $\widehat{\mathsf{EXP}}^{\ell-1}$ implies breaking CU$_2$. The detailed proof is put in Appendix A.

Lemma 1. *$\{\mathcal{I}_\lambda\}_\lambda$ is a hard subset membership problem, Ψ is computational universal$_2$ and* MAC *is existentially unforgeable. Then* $\Pr[c' = c] = 1/2 + \mathrm{negl}(\lambda)$.

4 Red ball experiment

We consider an experiment: there are n boxes, where box i contains a_i identical balls except that one ball is colored red (located at any position with equal probability) and the rest of them are colored white. Algorithm $\mathcal{A}$ adaptively draws t balls from these boxes. Each time it chooses a box and then draws a ball randomly from it without replacement. Let $\ell \in \{1, \ldots, n\}$. Let $\Theta^{\mathcal{A}}_{t,n,\ell}(a_1, \ldots, a_n)$ denote the probability that $\mathcal{A}$ draws t balls from these n boxes such that ℓ balls are red. Let $\Theta_{t,n,\ell}(a_1, \ldots, a_n) = \max_{\mathcal{A}} \Theta^{\mathcal{A}}_{t,n,\ell}(a_1, \ldots, a_n)$. It is easy to see that $\Theta_{t,n,\ell}(a_1, \ldots, a_n)$ is symmetric on $(a_1, \ldots, a_n)$. We can fully characterize it as in the following lemma.

Lemma 2. *If* $1 \leq a_1 \leq a_2 \leq \cdots \leq a_n$, $0 \leq \ell \leq n, t \geq 0$, *then*

$$\Theta_{t,n,\ell}(a_1, \ldots, a_n) = \Pr\Big[\sum_{i=1}^{\ell} x_i \leq t : x_i \leftarrow [a_i]\Big]. \tag{4.1}$$

We now outline the proof idea; details are in Appendix B. Use Left and Right to denote the left- and right-hand side of (4.1), respectively. First of all, there is an algorithm $\mathcal{A}_0$ achieving Right and so Left $\geq$ Right. $\mathcal{A}_0$ simply draws the ball from box 1 until the red ball is picked. Then, it turns to box 2 using the same strategy, then box 3, etc. Let the red ball in box i be obtained by using x_i picks and then $x_i \leftarrow [a_i]$. Since it succeeds if and only if $x_1 + \cdots + x_\ell \leq t$, Right is achieved.

It remains to show that Left $\le$ Right. This is done by induction. Case $\ell = 0$ or $t = 0$ is trivial. Generally, if the box id of the first pick by $\mathcal{A}$ is j, we have

$$\begin{aligned}\Theta_{t,n,\ell}(a_1,\ldots,a_n) &= a_j^{-1}\cdot\Theta_{t-1,n,\ell-1}(a_1,\ldots,a_{j-1},0,a_{j+1},\ldots,a_n)\\ &\qquad + (1-a_j^{-1})\Theta_{t-1,n,\ell}(a_1,\ldots,a_{j-1},a_j-1,a_{j+1},\ldots,a_n)\\ &= a_j^{-1}\cdot\Theta_{t-1,n-1,\ell-1}(a_1,\ldots,a_{j-1},a_{j+1},\ldots,a_n)\\ &\qquad + (1-a_j^{-1})\Theta_{t-1,n,\ell}(a_1,\ldots,a_{j-1},a_j-1,a_{j+1},\ldots,a_n).\end{aligned}$$

By induction, $\Theta_{t-1,n-1,\ell-1}()$ and $\Theta_{t-1,n,\ell}()$ can be bounded using the right-hand side of (4.1). Substituting these into the above equation, simplifying will give Right. The main effort in Appendix B is to handle the tedious details and show that the sum is in fact equal to the neat result

$$\Pr\Big[\sum_{i=1}^{\ell} x_i \le t : x_i \leftarrow [a_i]\Big].$$

Theorem 1. *If $t < \alpha\ell a$ and $\alpha < 0.5$, then*

$$\Theta_{t,n,\ell}(a,\ldots,a) < \exp\big(-2(0.5-\alpha)^2\ell\big).$$

Proof. By Lemma 2,

$$\begin{aligned}\Theta_{t,n,\ell}(a,\ldots,a) &= \Pr[x_1+\cdots+x_\ell \le t]\\ &= \Pr\Big[\frac{\sum_{i=1}^{\ell} x_i}{\ell} - \frac{a}{2} \le -\Big(\frac{a}{2}-\frac{t}{\ell}\Big)\Big]\\ &\overset{*}{\le} \exp(-2\delta^2\ell/a^2), \qquad \delta = \frac{a}{2}-\frac{t}{\ell} > (0.5-\alpha)a\\ &\le \exp\big(-2(0.5-\alpha)^2\ell\big),\end{aligned}$$

where inequality $*$ holds since $\mathbf{E}[x_i] = \frac{a}{2}$ and the Hoeffding inequality. □

5 Our PAKE framework

We now introduce our client-server password key exchange framework. Let $\mathcal{I} = \{\mathcal{I}_\lambda\}_\lambda$ be a hard subset membership problem. Sample

$$\Lambda = (X, L, W, R, D(L), D(X\backslash L)) \leftarrow \mathcal{I}_\lambda.$$

Assume that $\Psi = (\mathcal{H}, \mathcal{K}, X, L, G, S, \alpha)$ is a tag-based projective hash family for Λ, where $G = \{0,1\}^{2\lambda}$. Note that usually G is not of the form $\{0,1\}^\nu$ and

$C_i(\pi_i)$		$S(\pi_i, \theta)$
$x \overset{w}{\leftarrow} D(L)$, $y = \mathsf{T}(\pi_i, x)$ $(k_0, k_1) = H_\theta(i, x)$ using w $\tau_0 = \mathsf{MAC}_{k_0}(C_i\|S\|y)$	$\xrightarrow{C_i \mid y \mid \tau_0}$	$x = \mathsf{T}^*(\pi_i, y)$, $\zeta \leftarrow \{0,1\}^\lambda$ $(k'_0, k'_1) = H_\theta(i, x)$ using θ $\tau_0 \overset{?}{=} \mathsf{MAC}_{k'_0}(C_i\|S\|y)$ $\omega = C_i\|S\|y\|\zeta$, $\tau_1 = \mathsf{MAC}_{k'_0}(\omega\|1)$
$\omega = C_i\|S\|y\|\zeta$ $\tau_1 \overset{?}{=} \mathsf{MAC}_{k_0}(\omega\|1)$ $\tau_2 = \mathsf{MAC}_{k_0}(\omega\|2)$ output $\mathsf{sk} = k_1$	$\xleftarrow{S \mid \tau_1 \mid \zeta}$	
	$\xrightarrow{\tau_2}$	$\tau_2 \overset{?}{=} \mathsf{MAC}_{k'_0}(\omega\|2)$, output $\mathsf{sk} = k'_1$

Figure 1. Password key exchange framework HPS-PAKE (details in the bodytext).

we can obtain this by further going through a key derivation function (e.g., [9]) $\mathsf{KDF} : G' \to \{0,1\}^\nu$, where KDF has the property that when $x \leftarrow G'$, $\mathsf{KDF}(x)$ is statistically close to uniform in $\{0,1\}^\nu$. Let $\mathcal{D} = \{1, \ldots, N\}$ be the set of all possible passwords with uniform distribution. We say $\mathsf{T}, \mathsf{T}^* : \mathcal{D} \times X \to X$ are a *regular transformation pair* if they are efficiently computable and also satisfy the following.

R-1 For any $\pi \in \mathcal{D}$, $\mathsf{T}^*(\pi, \mathsf{T}(\pi, x)) = x$, for all $x \in X$, i.e., $\mathsf{T}^*(\pi, \cdot)$ is the inverse function of $\mathsf{T}(\pi, \cdot)$.

R-2 For any $y \in X$, there is at most one $\pi \in \mathcal{D}$ such that $\mathsf{T}^*(\pi, y) \in L$.

$\mathsf{MAC}_k : \{0,1\}^* \to \{0,1\}^\lambda$ is a secure message authentication code. The system setup is as follows. For the server S, take $\theta \leftarrow \mathcal{K}$ and compute $\Theta = \alpha(\theta)$. Then, θ will be the private key for S and Θ will be its public key. For each client C_i, take $\pi_i \leftarrow \mathcal{D}$ as the password for C_i, shared with S. The key exchange protocol between S and C_i is as follows (also see Figure 1).

(1) C_i takes $x \overset{w}{\leftarrow} D(L)$, computes $(k_0, k_1) = H_\theta(i, x)$ using (w, x, Θ), $y = \mathsf{T}(\pi_i, x)$ and $\tau_0 = \mathsf{MAC}_{k_0}(C_i|S|y)$. Finally, he sends $C_i|y|\tau_0$ to S.

(2) Upon $C_i|y|\tau_0$, S de-transforms $x = \mathsf{T}^*(\pi_i, y)$ and computes $(k'_0, k'_1) = H_\theta(i, x)$ using (θ, x). It then verifies if

$$\tau_0 \overset{?}{=} \mathsf{MAC}_{k'_0}(C_i|S|y).$$

If no, reject; otherwise, it takes $\zeta \leftarrow \{0,1\}^\lambda$ and computes $\tau_1 = \mathsf{MAC}_{k'_0}(\omega|1)$ for $\omega = C_i|S|y|\zeta$. Finally, it sends $S|\tau_1|\zeta$ to C_i.

(3) Upon $S|\tau_1|\zeta$, C_i verifies if

$$\tau_1 \stackrel{?}{=} \mathsf{MAC}_{k_0}(\omega|1)$$

for $\omega = C_i|S|y|\zeta$. If no, reject; otherwise, he sends $\tau_2 = \mathsf{MAC}_{k_0}(\omega|2)$ to S and outputs session key $\mathsf{sk} = k_1$.

(4) Upon τ_2, S verifies if

$$\tau_2 \stackrel{?}{=} \mathsf{MAC}_{k_0'}(\omega|2).$$

If no, reject; otherwise, output session key $\mathsf{sk} = k_1'$.

Remark. We outline how some attacks are prevented.

1. *Impersonation attack.* If the attacker impersonates C_i to generate and send $\mathsf{msg}_1 = C_i|y|\tau_0$ to S, then since he does not know π_i, $\mathsf{T}^*(\pi_i, y) \in L$ holds with probability $1/|\mathcal{D}|$ only. When $x := \mathsf{T}^*(\pi_i, y) \notin L$, τ_0 will be rejected since $(k_0, k_1) = H_\theta(i, x)$ appears random to the attacker.

2. *Insider attack* [7]. When a malicious C_j eavesdrops a transcript $tr = C_i|y|\tau_0|S|\tau_1|\zeta|\tau_2$ between C_i and S, then he executes the protocol with S in the name of himself but using tr as a help. Toward this, he might send $\mathsf{msg}_1 = C_j|y|\tau_0^*$ to S and hope to receive a response from the latter. τ_0^* is acceptable only if $\tau_0^* = \mathsf{MAC}_{k_0^*}(C_j|S|y)$, where $(k_0^*, k_1^*) := H_\theta(j, x^*)$ for $x^* = \mathsf{T}^*(\pi_j, y)$. The only useful information is τ_0 which is computed using $(k_0, k_1) := H_\theta(i, x)$ for $x = \mathsf{T}^*(\pi_i, y)$. However, no matter $\pi_j = \pi_i$ or not, we have that $(i, x^*) \neq (j, x)$ as $i \neq j$ (this is the main reason we use tag-HPS instead of HPS in this paper). This allows us to claim that k_0^* and k_0 are computationally independent. If $x \notin L$, this is automatically true by the computational universal$_2$ definition. In our protocol, even if $x \leftarrow D(L)$, this computational independency still holds; otherwise, one can simply reduce to break the hardness of L. Thus, S will always reject τ_0^*. Since this rejection occurs without considering the value of π_i, it follows that the candidate space of π in view of the adversary does not reduce.

3. *Session key secrecy.* The session key $\mathsf{sk} = k_1$ is computed by $(k_0, k_1) = H_\theta(i, x)$. Client C_i can compute this since he knows the witness w of $x \in L$ and server S can compute this since it knows π_i (for recovering x from y) and θ for (k_0, k_1). Any outsider can not compute (k_0, k_1) since given x and Θ, $H_\theta(i, x)$ is indistinguishable from random (Lemma 1).

5.1 A concrete example

In the following, we present a concrete tag-based HPS toward realizing our protocol framework by slightly revising HPS in [10, 27].

Hard subset membership problem. Let $q, p = 2q + 1$ be large primes. Let $\mathbb{G}$ be the prime subgroup of order q in $\mathbb{Z}_p^*$. Take $g_1, g_2 \leftarrow \mathbb{G}$. Define $X = \{(g_1^{w_1}, g_2^{w_2}) \mid w_1, w_2 \in \mathbb{Z}_q\}$ and $L = \{(g_1^w, g_2^w) \mid w \in \mathbb{Z}_q\}$. Notice that $(g_1^w, g_2^w) \in L$ has the witness w. By Decisional Diffie–Hellman assumption, L and X are indistinguishable.

Tag-based projective hash function H_θ. Let a hash key $\theta = (a_1, a_2, b_1, b_2) \in \mathbb{Z}_q^4$ and its projection $\Theta = \alpha(\theta) = (\Theta_1, \Theta_2) = (g_1^{a_1} g_2^{a_2}, g_1^{b_1} g_2^{b_2})$. Let $\mathsf{KDF} : \mathbb{G} \to \{0,1\}^{2\lambda}$ be a secure key derivation function (that is, for $x \leftarrow \mathbb{G}$ and $u \leftarrow \{0,1\}^{2\lambda}$, $\mathsf{KDF}(x)$ and u are statistically close). Assume that $h_\phi : \{0,1\}^* \to \{0,1\}^\lambda$ is a collision-resistant hashing indexed by $\phi \leftarrow \{0,1\}^\lambda$. For $(x_1, x_2) \in X$ and tag i, define the projective hash $H_\theta(i, (x_1, x_2)) = \mathsf{KDF}(x_1^{a_1+b_1\eta} x_2^{a_2+b_2\eta})$, where $\eta = h_\phi(i, x_1, x_2)$. If $(x_1, x_2) = (g_1^w, g_2^w)$, then

$$\begin{aligned} H_\theta(i, x_1, x_2) &= \mathsf{KDF}(x_1^{a_1+b_1\eta} x_2^{a_2+b_2\eta}) \quad \text{(using } \theta) \\ &= \mathsf{KDF}((\Theta_1 \Theta_2^\eta)^w) \quad \text{(using witness } w). \end{aligned}$$

So H_θ is a projective hash function.

The difference of the above HPS from original HPS [10,27] is that originally h_ϕ does not get input tag i and that KDF is not used there. These changes are minor. With almost the same proof as in [19, Lemma 6.3], we can show the following result.

Lemma 3. *If h_ϕ is collision-resistant, then H_θ must be computational universal$_2$.*

In order to realize our framework, we need to further specify T and T^*.

Regular transformation pair $(\mathsf{T}, \mathsf{T}^*)$. For $\pi \in \mathcal{D}$, $(x_1, x_2) \in X$, let $(y_1, y_2) = \mathsf{T}(\pi, (x_1, x_2)) = (x_1, x_2 g_2^\pi)$ and $\mathsf{T}^*(\pi, (y_1, y_2)) = (y_1, y_2 g_2^{-\pi})$. Evidently, property R-1 is satisfied. Property R-2 is satisfied as long as there are no $\pi_1, \pi_2 \in \mathcal{D}$ such that $\pi_1 \equiv \pi_2 \pmod{q}$. To satisfy this, it suffices to take $\mathcal{D} = \{1, \ldots, N\}$ for $N < q$. However, in order to show the persistency, we actually take $N = 2^{\lambda/3}$.

Efficiency of the protocol realization. Besides π_i (in C_i and S) and θ (in S) are securely stored, assume $g_2^{\pi_i}$ is securely stored in S (but not in C_i as he can not memorize this long secret). The cost of MAC is negligible. So the cost of the realized protocol is dominated by $x \overset{w}{\leftarrow} D(L)$ and $H_\theta(i, x)$. Specifically, the client's cost is dominated by 5 exponentiations for computing $(g_2^{\pi_i}, g_1^w, g_2^{w+\pi_i})$ and $\mathsf{KDF}((\Theta_1 \Theta_2^\eta)^w)$. The server's cost is dominated by two exponentiations for computing $\mathsf{KDF}(y_1^{a_1+b_1\eta} y_2'^{a_2+b_2\eta})$. We do not account the cost for verifying $(y_1, y_2) \in \mathbb{G}^2$, as in Section 8 we show that this can be waived with only a negligible price. Note that a password-only key exchange can also solve the server key leak problem. However, our protocol is much more efficient than such protocols; see Table 1 in Section 1.

6 Security

Now we prove the security of our protocol. Before this, we define the session id in the protocol as $\mathsf{sid}_U^{\ell_U} = C_i|S|y|\zeta$, where U is the client i or server S. Since the password π_i and the high entropic secret key θ are both fixed after the system initiation, $H_\theta(i,x)$ is determined for given $C_i|S|y$. Hence, two partnered parties must have the same session key. It remains to show the authentication and secrecy. They are showed together. The proof idea is as follows; details are put in Appendix C.

Idea of authentication and secrecy. The authentication property is to bound $\mathsf{Non\text{-}Auth}_i$ for each i and the secrecy property is to bound the adversary success in Test query. Both events lie in the adversary view. Our strategy is to revise the adversary-challenger game Γ_0 into Γ_1, Γ_2 such that neighboring games have indistinguishable adversary views and then consider Γ_2 for these two events. Γ_1 differs from Γ_0 only at (k_0, k_1) in $\mathsf{Send}(0, \cdot)$ oracle, where Γ_1 takes $(k_0, k_1) \leftarrow \{0,1\}^{2\lambda}$ while Γ_0 takes $(k_0, k_1) = H_k(i, x)$. Indistinguishability in adversary views between Γ_1 and Γ_0 is based on the negligible advantage of Experiment EXP in Lemma 1. Indeed, (k_0, k_1) can be set as (a_c, s_c) in Chal query and τ_i can be verified in Comp query. So the simulation when $c = 0$ is Γ_0 while it is Γ_1 when $c = 1$. Indistinguishability follows from Lemma 1. Then Γ_2 differs from Γ_1 at x in $\mathsf{Send}(0, \cdot)$, where Γ_2 takes $x \leftarrow X$ while Γ_1 takes $x \leftarrow L$. This revision does not change the adversary view due to the hardness of the subset membership problem. It remains to analyze Γ_2. Toward this, we first show some properties for Γ_2. Firstly, for each $\Pi_i^{\ell_i}$ that accepts $S|\zeta^*|\tau_1^*$, it must have a unique partner $\Pi_S^{\ell_s}$. This is true; otherwise, this implies that the adversary can forge a valid τ_1^* under MAC key k_0^* (set by $\Pi_i^{\ell_i}$), violating the unforgeability of MAC. Secondly, if $\Pi_S^{\ell_s}$ accepts τ_2^* and $\mathsf{Flow}_1 = C_i|y^*|\tau_0^*$ was from some $\Pi_i^{\ell_i}$, then $\Pi_S^{\ell_s}$ must have a unique partner $\Pi_i^{\ell_i}$. The reason for this is similar to the first property. We now consider the secrecy in Γ_2 conditional on $\neg\mathsf{Non\text{-}Auth}_i$ for any i. Since no $\mathsf{Non\text{-}Auth}$ event, "$\mathsf{Flow}_1 = C_i|y^*|\tau_0^*$ was from some $\Pi_i^{\ell_i}$" required in the second property above is always satisfied. So any accepting instance (especially, test instance) has the unique partner, which also has this instance as its unique partner. That is, accepting instances can be uniquely paired with partnership. Especially, (k_0, k_1) is only used in two partnered instances. If one of them is the test instance, then both instances can not be compromised by the restriction of Test oracle and hence the session key k_1 of the test instance is independent of the adversary view. So the adversary success probability in Test query is $1/2$, conditional on $\neg\mathsf{Non\text{-}Auth}_i$ for any i. Further, by the above two properties, we know that $\mathsf{Non\text{-}Auth}_i$ occurs only at $\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$ oracle, which has two cases. In Case 1, $\mathsf{T}^*(\pi_i', y) \notin L$ for any password π_i' in candidate space $\mathcal{D}_i$ of π_i. In this case, k_0 is independent of the adversary view due to computation universal$_2$ of $H_k(i, x)$ and hence τ_0 can be rejected without leaking anything about π_i (because each candidate of π_i will give the same result: reject). In Case 2, $x \in L$ for some $\pi_i' \in \mathcal{D}_i$. In this case, if $\pi_i \neq \pi_i'$, then the decision is reject and no information beyond $\pi_i \in \mathcal{D}_i = \mathcal{D}_i \backslash \{\pi_i'\}$ is leaked and hence after this π_i is uniform in the updated $\mathcal{D}_i$; otherwise $\pi_i = \pi_i'$ which has the probability $1/|\mathcal{D}_i|$. As a summary, the first $\mathsf{Send}(S, \cdot, C_i|y|\tau_0)$ with y not generated by C_i is accepted with probability $1/|\mathcal{D}|$. Similarly, the second query is accepted with probability $\frac{|\mathcal{D}|-1}{|\mathcal{D}|} \cdot \frac{1}{|\mathcal{D}|-1} = \frac{1}{|\mathcal{D}|}$, etc. Since there are at most Q_i queries with client C_i, $\mathsf{Non\text{-}Auth}_i$ occurs with probability at most

$$Q_i \cdot \frac{1}{|\mathcal{D}|} = \frac{Q_i}{|\mathcal{D}|}.$$

Hence, the theorem follows.

Theorem 2. *Let $\mathcal{I} = \{\mathcal{I}_\lambda\}_\lambda$ be a hard subset membership problem. Assume that* $\mathsf{MAC} : \{0,1\}^* \to \{0,1\}^\lambda$ *is an existentially unforgeable message authentication code. Assume that Ψ is computational universal$_2$ for $\mathcal{I}$. Then* $\mathsf{HPS\text{-}PAKE}$ *is secure.*

7 Persistency

In this section, we prove our protocol's persistency. We need the following notion.

Definition 4. Let $\mathcal{H} = \{H_\theta\}_{\theta \in \mathcal{K}}$ with $H_\theta : \{0,1\}^* \times X \to \{0,1\}^{2\lambda}$ be a tag-PHF. We say that $\mathcal{H}$ is *locally unique* with respect to a deterministic function $F : \mathcal{D} \times X \to X$ if for any PPT adversary $\mathcal{A}$,

$$\Pr\Big[\lfloor H_\theta(z, F(\pi_1, y))\rfloor_\lambda = \lfloor H_\theta(z, F(\pi_2, y))\rfloor_\lambda \text{ for some } \pi_1 \neq \pi_2 : \\ \theta \leftarrow \mathcal{K}, (z, y) \leftarrow \mathcal{A}(\theta, \mathrm{desc}(\mathcal{H}))\Big],$$

is negligible, where $\lfloor x\rfloor_\lambda$ is the least λ-bit of x and the probability is over the randomness of $\mathcal{A}, \mathcal{H}, \theta$.

Remark. A tag-PHF is always defined with respect to a hard subset membership problem Λ. So $\mathcal{H}$ inherits the randomness in sampling Λ. The local uniqueness of $\mathcal{H}$ essentially requires that for any adversarially chosen (z, y), each λ-bit string s corresponds to at most only one password π such that $\lfloor H_\theta(z, F(\pi, y))\rfloor_\lambda = s$.

Persistency idea of our protocol. We want to show that for an adversary who has an access to Send oracles and MAC oracle (maintained by a challenger), within $\alpha\ell|\mathcal{D}|$ basic steps, $\mathsf{Non\text{-}Auth}_i$ occurs to ℓ different i only with exponentially small probability. Toward this, we first modify $\mathsf{Send}(0,\cdot)$ oracle so that $x \leftarrow D(X)$ instead of $x \leftarrow D(L)$. This modification does not change the adversary success as X and L are indistinguishable due to the hardness of L in X. Then, y can be further modified as $y \leftarrow D(X)$ (instead of $y = \mathsf{T}(\pi_i, x)$ for $x \leftarrow D(X)$) as these two generations are identically distributed. After this, y no longer carries information of π_i. We also modify Send oracle such that k_1, sk are not computed. This is no problem since they are not used to compute the oracle output. After this treatment, the only place to use password π_i is to compute k_0 and in turn k_0 will be only used to compute MACs τ_0, τ_1, τ_2. We then present a simulation of challenger by splitting it to two entities $(\mathcal{C}_1, \mathcal{C}_2)$. Here $\mathcal{C}_1$ holds the password assignment $\{\pi_i\}$ for all clients and $\mathcal{C}_2$ does not have $\{\pi_i\}$ and will maintain Send oracle and MAC oracle using $(\Theta, \theta, \mathrm{desc}(\mathcal{H}))$. We present a technique in simulating MAC oracle such that τ_i can be computed without password assignment $\{\pi_j\}$ and instead $\mathcal{C}_2$ only needs to ask $\mathcal{C}_1$ with (i, π) whether $\pi_i = \pi$. Our simulation has the property that any Auth_i implies the successful password verification query (i, π). Hence, the adversary success in ℓ different Auth_i implies ℓ successful password verifications at $\mathcal{C}_1$. On the other hand, the password verification process implies a red ball game: (i, π) is a pick of π in box i and it hits π_i if $\pi_i = \pi$. As there are at most $\ell\alpha|\mathcal{D}|$ picks, by Theorem 1, it hits ℓ different passwords with probability exponentially small. The detailed proof is in the following theorem.

Theorem 3. *Let* $\mathsf{MAC} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ *be a random oracle and* $\mathcal{H} = \{H_\theta\}$ *be locally unique with respect to* T^*. *Then,* $\mathsf{HPS\text{-}PAKE}$ *is persistent, where one* MAC *evaluation is a basic step.*

Proof. We use PRS_ℓ to denote the event that $\mathsf{Non\text{-}Auth}_i$ occurs to ℓ different i, when adversary $\mathcal{A}$ is given $(\theta, \Theta, \mathrm{desc}(\mathcal{H}))$ and access to Send oracles and random oracle MAC. We regard the interaction between $\mathcal{A}$ and the challenger (who maintains Send oracles and MAC oracle) as a game. Denote the game, where Send oracle is maintained according to the specification, by Γ_0. Then, we need to bound the probability $\Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_0))$.

Game Γ_1. We revise Γ_0 to Γ_1 such that if for $\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$, there exist $a \in \{0,1\}^\lambda$ and more than one π such that $\lfloor H_\theta(i, \mathsf{T}^*(\pi, y))\rfloor_\lambda = a$, then we announce the success of $\mathcal{A}$. Otherwise, for any $\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$ query from $\mathcal{A}$ and any $a \in \{0,1\}^\lambda$, there is at most one such π and hence we can define $\mathsf{pw}_{\theta,i,y}(a) = \pi$ in case of existence and $\mathsf{pw}_{\theta,i,y}(a) = \mathrm{nil}$ otherwise. By local uniqueness of H_θ, we have the following result.

Lemma 4. $\Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_0)) = \Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_1)) + \text{negl}(\kappa)$.

For simplicity, from now on, we assume that $\mathcal{A}$ never succeeds due to multiple π event above and hence for any $\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$ query, $\mathsf{pw}_{\theta,i,y}(a)$ is always well-defined.

Game Γ_2. We modify Γ_1 to Γ_2 such that $\mathsf{Send}(0, \cdot)$ oracle takes $x \leftarrow D(X)$ (instead of $D(L)$). To be consistent, the only change in maintaining oracles in Γ_1 is to evaluate $H_\theta(z, x)$ using θ (instead of using the witness of x) in $\mathsf{Send}(0, \cdot)$ oracle. By the hardness of L in X, adversary views View in Γ_1 and Γ_2 are negligibly close, where an adversary view is defined as his random tape and all the data received from the challenger. As PRS_ℓ is deterministic in the adversary view, we have the following result.

Lemma 5. $\Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_1)) = \Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_2)) + \text{negl}(\kappa)$.

Game Γ_3. We modify Γ_2 to Γ_3 such that the challenger is split into two parties $(\mathcal{C}_1, \mathcal{C}_2)$, where $\mathcal{C}_1$ holds the password assignment $\{\pi_i\}$ of all clients, and $\mathcal{C}_2$ holds $\langle \Theta, \text{desc}(\mathcal{H}), \theta \rangle$. In addition, $\mathcal{C}_2$ maintains Send oracles and MAC oracle. $\mathsf{Send}(0, \cdot)$ oracle directly takes $y \leftarrow D(X)$ without computing x. To maintain Send oracles, $\mathcal{C}_2$ can request $\mathcal{C}_1$ to compute $k_0 = H_\theta(i, \mathsf{T}^*(\pi_i, y)$ with a description of function $H_\theta(i, \mathsf{T}^*(\cdot, y))$ (i.e., $(i, y, \theta, \text{desc}(\mathcal{H}), \text{desc}(\mathsf{T}^*))$) as the query input. Send oracles never compute k_1 and sk. The remaining description for Send oracle and MAC oracle is normal.

Lemma 6. $\Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_2)) = \Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_3))$.

Proof. Taking $y \leftarrow D(X)$ has an identical distribution as taking $x \leftarrow D(X)$ and computing $y = \mathsf{T}(\pi_i, x)$ because $\mathsf{T}(\pi_i, \cdot)$ is a permutation of X. Hence, the revised $\mathsf{Send}(0, \cdot)$ does not change the adversary view. Notice that k_0 computed in the alternative in Γ_3 is identical to that in Γ_2. Further, k_1 and sk are never used in generating an oracle output. Hence, each send oracle output in Γ_3 has a perfect distribution as Γ_2. Finally, PRS_ℓ is deterministic in adversary view. Hence the lemma follows. $\square$

Game Γ_4. We modify Γ_3 to Γ_4 with the following changes. $\mathcal{C}_2$ never asks $\mathcal{C}_1$ to compute k_0 but he will request $\mathcal{C}_1$ with any pair (i, π) to verify whether $\pi_i = \pi$. In addition, he also maintains a candidate space $\mathcal{D}_i$ of π_i for each i, where initially $\mathcal{D}_i = \mathcal{D}$. In Send oracle with client C_i, if $|\mathcal{D}_i| = 1$, $\mathcal{C}_2$ can process normally using the only π_i in $\mathcal{D}_i$ to compute k_0; if $|\mathcal{D}_i| > 1$, whenever it requires to compute $\mathsf{MAC}(k_0, C_i|S|y|*)$, he defines a virtual symbol $\text{undef-}k_0(i, y)$ to represent $k_0 =$

$\lfloor H_\theta(i, \mathsf{T}^*(\pi_i, y))\rfloor_\lambda$ (unknown) and issues a MAC query MAC(undef-$k_0(i, y)$, $C_i|S|y|*$). Since in Γ_3, π_i is only used to compute k_0 and k_0 is only used to compute the MAC function, Γ_4 is well-defined if MAC oracle is changed to be compatible with the above query, which is done as follows.

MAC(key, m). Here key is the MAC key. The oracle maintains a list $\mathcal{L}$ of records $(x, \mathsf{MAC}(x))$. If (key, m) was queried before, then reply with the existing record; otherwise, process in two cases.

(i) key $=$ undef-$k_0(i, y)$. This query is from $\mathcal{C}_2$ and $m = C_i|S|y|*$. Take mac $\leftarrow \{0, 1\}^\lambda$, add (key, m, mac) into $\mathcal{L}$ and return mac.

(ii) key is a concrete number. If $m = C_i|S|y|*$ and $|\mathcal{D}_i| > 1$, check

$$\pi_i \stackrel{?}{=} \mathsf{pw}_{\theta,i,y}(\mathsf{key})$$

by querying $\mathcal{C}_1$. In case of yes, update $\mathcal{D}_i = \{\mathsf{pw}_{\theta,i,y}(\mathsf{key})\}$; in case of no, update $\mathcal{D}_i = \mathcal{D}_i \backslash \{\mathsf{pw}_{\theta,i,y}(\mathsf{key})\}$.

If it is the first MAC query where $|\mathcal{D}_i| = 1$, update each undef-$k_0(i, y')$ in $\mathcal{L}$ with its concrete value $\lfloor H_\theta(i, \mathsf{T}^*(\pi_i, y'))\rfloor_\lambda$.

In any case (including case $m \neq C_i|S|y|*$), if (key, m, mac) was recorded in $\mathcal{L}$ for some mac, then return mac; otherwise, take mac $\leftarrow \{0, 1\}^\lambda$ and add (key, m, mac) into $\mathcal{L}$ and return mac.

To analyze Γ_4, we first prove the following claim.

Claim 1. *After each* MAC *query, if we take $\{\pi_j\}$ as any $\{\pi'_j\} \in \prod_j \mathcal{D}_j$ (hence realize every symbol* undef-$k_0(i, y)$ *in $\mathcal{L}$), then $\mathcal{L}$ is consistent: there does not exist two distinct records* (key, m, mac$_1$) *and* (key, m, mac$_2$) *in $\mathcal{L}$.*

Proof. Otherwise, after some MAC query and when realizing $\{\pi_j\} = \{\pi'_j\} \in \prod_j \mathcal{D}_j$, there exist (key, m, mac$_1$) and (key, m, mac$_2$) in $\mathcal{L}$ for mac$_1 \neq$ mac$_2$. Since for each query (key, m), MAC oracle will first check whether there exists an existing record, it follows that before realizing $\{\pi_j\} = \{\pi'_j\}$, the two (unordered) records must be (undef-$k_0(i, y)$, $C_i|S|y|A$, mac$_1$) and (key, $C_i|S|y|A$, mac$_2$). This implies that key $= \lfloor H_\theta(i, \mathsf{T}^*(\pi'_i, y))\rfloor_\lambda$ and so $\mathsf{pw}_{\theta,i,y}(\mathsf{key}) = \pi'_i \in \mathcal{D}_i$. There are two cases:

1. The record with mac$_1$ was recorded in $\mathcal{L}$ earlier than that with mac$_2$. In this case, when processing query (key, m), either update $\mathcal{D}_i = \{\mathsf{pw}_{\theta,i,y}(\mathsf{key})\}$ and make undef-$k_0(i, y)$ concrete, or update $\mathcal{D}_i = \mathcal{D}_i \backslash \{\mathsf{pw}_{\theta,i,y}(\mathsf{key})\}$. For the former, the oracle reply will be mac$_1$; for the latter, π'_j is removed from $\mathcal{D}_i$. Both cases are impossible.

2. The record with mac_1 was recorded in $\mathcal{L}$ later than that with mac_2. From the specification, when processing (key, m) query, either update $\mathcal{D}_i = \mathcal{D}_i \backslash \{\pi'\}$ or $|\mathcal{D}_i| = 1$ after query. The former is impossible by our assumption. By our specification of Send oracles, when $|\mathcal{D}_i| = 1$, no MAC query of the form $(\mathsf{undef}\text{-}k_0(i, y), *)$ will be issued, a contradiction.

Hence, the two cases never occur and $\mathcal{L}$ is consistent with any assignment $\{\pi_i\} \in \prod_i \mathcal{D}_i$. □

Lemma 7. $\Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_3)) = \Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_4))$.

Proof. We show that the adversary view in Γ_4 is identical to that in Γ_3. The adversary view consists of its local randomness, messages from Send oracle and MAC. The adversary view at Send oracle between Γ_3 and Γ_4 differs only in the MAC replies. However, by Claim 1 above, $\mathcal{L}$ in MAC oracle can be consistently explained as the record list of MAC oracle when $\{\pi_i\}$ takes any value in $\prod_i \mathcal{D}_i$, which of course includes the true password assignment in $\mathcal{C}_1$ as each removed password from $\mathcal{D}_i$ is confirmed not π_i by the equality test. As the MAC value mac in each record (x, mac) is taken uniformly random in $\{0, 1\}^\lambda$, $\mathcal{L}$ in MAC oracle of Γ_4 for the true password assignment is in fact identically distributed as that in Γ_3. This proves the equivalence of the adversary view in these two games. Finally, as PRS is deterministic in the adversary view, the lemma follows. □

Bounding $\Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_4))$. We first characterize event $\mathsf{Non\text{-}Auth}_i$. It can occur only in $\mathsf{Send}(2, \cdot)$ and $\mathsf{Send}(3, \cdot)$ as follows. Let $\mathsf{sid}_i^{\ell_i} = \mathsf{sid}_s^{\ell_s} = C_i|S|y|\zeta$.

(a) $\mathsf{Non\text{-}Auth}_i$ occurs in $\mathsf{Send}(2, i, \ell_i, S|y|\tau_1|\zeta)$ query only if τ_1 is accepted while S does not hold a session with $\mathsf{sid}_s^{\ell_s} = C_i|S|y|\zeta$ and hence has never issued a MAC query $(\mathsf{key}, C_i|S|y|\zeta)$, where key is either the concrete $k_0(i, y)$ or its virtual symbol $\mathsf{undef}\text{-}k_0(i, y)$. If $\mathcal{L}$ does not have a record for this $(\mathsf{key}, C_i|S|y|\zeta|1)$ before verifying τ_1, then τ_1 is accepted with probability at most $2^{-\lambda}$ (ignored); otherwise, the corresponding MAC query must be issued by $\mathcal{A}$ where key is the concrete $k_0(i, y)$, which means that the test

$$\pi_i \stackrel{?}{=} \mathsf{pw}_{\theta,i,y}(\mathsf{key})$$

issued during processing this MAC query is successful.

(b) $\mathsf{Non\text{-}Auth}_i$ occurs in $\mathsf{Send}(3, S, \ell_S, \tau_2)$ with client C_i. So τ_2 is accepted (e.g., by session $\mathsf{sid}_s^{\ell_s} = C_i|S|y|\zeta$) while C_i does not have a session $\mathsf{sid}_i^{\ell_i} = C_i|S|y|\zeta$, which implies that $\mathcal{C}_2$ did not issue a MAC query $(\mathsf{key}, C_i|S|y|\zeta|2)$, where key is the concrete $k_0(i, y)$ or its virtual symbol $\mathsf{undef}\text{-}k_0(i, y)$. If $\mathcal{L}$ does not have a record for this ($\mathsf{key}, C_i|S|y|\zeta|2$) before verifying τ_2, then τ_2 is accepted with probability at most $2^{-\lambda}$ (ignored); otherwise, the corresponding MAC query must be issued by $\mathcal{A}$ where key is the concrete $k_0(i, y)$, which means that test

$$\pi_i \stackrel{?}{=} \mathsf{pw}_{\theta,i,y}(\mathsf{key})$$

issued during processing this MAC query is successful.

From cases (a), (b), we can see that Auth_i implies a successful verification of π_i at $\mathcal{C}_1$. Denote the probability of the successful verification of ℓ different π_i by p_ℓ. Then $\Pr(\mathsf{PRS}_\ell(\mathcal{A}, \Gamma_4)) \le p_\ell$. We note that the password equality test between $\mathcal{C}_1$ and $\mathcal{C}_2$ is exactly a red ball experiment: π_i is red ball and $\mathsf{pw}_{\theta,i,y}(\mathsf{key})$ is a pick from box i. Notice that $\{\pi_i\}$ initially is completely uniformly random in $\mathcal{D}^n$. Each pick either hits the red ball π_i or eliminates one white ball $\mathsf{pw}_{\theta,i,y}(a)$ from box i. To be successful, ℓ red balls should be hit. One pick in the induced red ball game implies one MAC query. As $\mathcal{A}$ makes at most $\alpha\ell|\mathcal{D}|$ queries, the number of picks by $\mathcal{C}_2$ is bounded by it. By Theorem 1, within $T < \alpha\ell|\mathcal{D}|$ picks, ℓ red balls are selected with probability at most by $\exp(-2\ell(0.5-\alpha)^2)$.

Summarizing the above bounding on p_ℓ and Lemmas 4–7, we conclude the proof of Theorem 3. □

8 Analysis of our concrete protocol

In Section 5.1, we present an HPS for our framework HPS-PAKE. Call the realized protocol HPS_{cs}-PAKE. In this section, we analyze it.

Security. As H_θ is computationally universal$_2$, security is implied by Theorem 2.

Persistency. By Theorem 3, the persistency holds if $H_\theta(z, x)$ is *locally unique*, which is seen in the following lemma.

Lemma 8. *If h_ϕ is a random oracle and* $\mathsf{KDF} : \mathbb{G} \to \{0,1\}^{2\lambda}$ *is a statistically secure key derivation function. Then, $H_\theta()$ is locally unique with respect to* T^*.

Proof. Since b_2 is uniform over $\mathbb{Z}_q$, we ignore the probability $b_2 = 0$. Let (z^*, x_1^*, x_2^*) be the output of $\mathcal{A}$. For any distinct $\pi_1, \pi_2 \in [N]$, let

$$A = H_\theta(z^*, \mathsf{T}^*(\pi_1, x_1^*, x_2^*)) = x_1^{*a_1} x_2^{*a_2} g_2^{-\pi_1 a_2} \cdot (x_1^{*b_1} x_2^{*b_2} g_2^{-b_2\pi_1})^{\tau_1},$$
$$B = x_1^{*a_1} x_2^{*a_2} g_2^{-\pi_2 a_2} \cdot (x_1^{*b_1} x_2^{*b_2} g_2^{-b_2\pi_2})^{\tau_2},$$

where $\tau_1 = h_\phi(z^*, x_1^*, x_2^* g_2^{-\pi_1})$ and $\tau_2 = h_\phi(z^*, x_1^*, x_2^* g_2^{-\pi_2})$. As $q > N$,

$$(x_1^{*b_1} x_2^{*b_2} g_2^{-b_2\pi_1})/(x_1^{*b_1} x_2^{*b_2} g_2^{-b_2\pi_2}) = g_2^{b_2(\pi_2-\pi_1)}$$

has an order of q. Further notice that τ_1 and τ_2 are independent and uniformly random (in $\mathbb{Z}_q$). It follows that either B or A is uniformly distributed over $\mathbb{G}$. Assume B has an order of q. From the independence between τ_1 and τ_2, B is uniformly random over $\mathbb{G}$ for a fixed A. Hence, $\lfloor \mathsf{KDF}(B) \rfloor_\lambda = \lfloor \mathsf{KDF}(A) \rfloor_\lambda$ with probability $2^{-\lambda}$ only. As $N = 2^{\lambda/3}$, the existence of a pair (π_1, π_2) such that the equality holds, only has a probability at most $2^{-\lambda/3}$, negligible. □

Avoid a verification of $y \in \mathbb{G}^2$. Our efficiency claim in Section 5.1 does not include the cost for the verification of $(y_1, y_2) \in \mathbb{G}^2$ by S which needs one more exponentiation. This cost can be avoided by a slight modification. In msg_1, instead of sending $y = (g_1^w, g_2^{w+\pi_i})$, client i computes

$$y^* := (y_1^*, y_2^*) := (g_1^{w/2}, g_2^{(w+\pi)/2}),$$

sets $y = (y_1^{*2}, y^{*2}_2)$ and replaces y in the original msg_1 message by y^*. The remaining specification for the client is unchanged. Correspondingly, the server computation is as follows. It first recovers $y = (y^{*2}_1, y^{*2}_2)$ from y^* when receiving msg_1 and the remaining specification for the server is unchanged. Denote the modified protocol by HPS^*_{cs}-PAKE. The cost for the client and the server each increases by 2 squarings, which is tiny. In addition, the security of HPS_{cs}-PAKE implies the security of the modified protocol HPS^*_{cs}-PAKE. The proof uses the fact that for $y \in \mathbb{G}$, $\sqrt{y} = y^{(q+1)/2}$. So the attacker for HPS_{cs}-PAKE can easily simulate the environment for an attacker in HPS^*_{cs}-PAKE and the reduction follows. Details are omitted here.

A Proof of Lemma 1

Use EXP_c to denote EXP when the challenge bit is c. It suffices to show that $\Pr[\mathcal{A}(\mathsf{EXP}_0) = 1] \approx \Pr[\mathcal{A}(\mathsf{EXP}_1) = 1]$. Let EXP_0^ℓ denote the variant of EXP_0, where the first ℓ Chal queries are answered as in EXP_1 while the remaining such queries are answered as in EXP_0. Let the number of Chal queries be bounded by N. Then, $\mathsf{EXP}_0^0 = \mathsf{EXP}_0$ and $\mathsf{EXP}_0^N = \mathsf{EXP}_1$. If the lemma is violated by $\mathcal{A}$, then there exists ℓ such that $|\Pr[\mathcal{A}(\mathsf{EXP}_0^{\ell-1}) = 1] - \Pr[\mathcal{A}(\mathsf{EXP}_0^\ell) = 1]|$ is non-negligible. Let $\widehat{\mathsf{EXP}}_0^i, i = \ell-1, \ell$ be the variant of EXP_0^i such that in the ℓth Chal query, $x \leftarrow D(X \backslash L)$ (instead of $x \leftarrow D(L)$), where correspondingly $H_k(z, x)$ is computed using k. By reducing to the hardness of $\mathcal{I}$, we have

$$\Pr[\mathcal{A}(\mathsf{EXP}_0^i) = 1] \approx \Pr[\mathcal{A}(\widehat{\mathsf{EXP}}_0^i) = 1].$$

Hence, $\Pr[\mathcal{A}(\widehat{\mathsf{EXP}}_0^{\ell-1}) = 1] - \Pr[\mathcal{A}(\widehat{\mathsf{EXP}}_0^\ell) = 1]$ is non-negligible. We build an adversary $\mathcal{D}$ that uses $\mathcal{A}$ to break computationally universal$_2$ of Ψ. Upon

$(\alpha(k), \text{desc}(\Psi))$, $\mathcal{D}$ invokes $\mathcal{A}$ with it and simulates $\widehat{\mathsf{EXP}}_0^{\ell}$ as follows. He defines c to be the hidden bit in his challenge key K_c (parsed as (a_c^*, s_c^*) in this proof).

(i) $\mathsf{Chal}(z)$. Assume it is the ith Chal query. If $i \neq \ell$, $\mathcal{D}$ simulates normally as in $\widehat{\mathsf{EXP}}_0^{\ell}$ except that $(a_0, s_0) = H_k(z, x)$ is computed using witness w. If $i = \ell$, he takes $x^* \leftarrow D(X \backslash L)$ and sets (z, x^*) as his test query (z_2, x_2). When receiving K_c, he defines $(a_c^*, s_c^*) = K_c$ and then processes normally.

(ii) $\mathsf{Comp}(z, x, \sigma, m)$. If $(z, x, a', s') \in \Theta$ for some a', s', he proceeds normally; otherwise, he queries $\mathsf{Eval}_1(z, x)$ query to his challenger and in turn receives (a, s). If $(a, s) = \perp$ (hence $x \notin L$), he outputs $\perp$; otherwise, he proceeds normally.

At the end of game, $\mathcal{D}$ outputs whatever $\mathcal{A}$ does.

Denote the simulated game of $\mathcal{D}$ with bit c by $\overline{\mathsf{EXP}}_0^{\ell-c}$. Then $\overline{\mathsf{EXP}}_0^{\ell-c}$ is identical to $\widehat{\mathsf{EXP}}_0^{\ell-c}$, except when $x \notin L$ in Comp query. In this case, the challenger of $\mathcal{D}$ returns $(a, s) = \perp$ and $\mathcal{D}$ will output $\perp$ while in $\widehat{\mathsf{EXP}}_0^{\ell-c}$, σ will be verified using a in $(a, s) = H_k(x)$ and (if valid) (a, s) is returned. Hence, an inconsistency between the two games occurs only if the following event occurs to some $\mathsf{Comp}(z, x, \sigma, m)$ query in $\overline{\mathsf{EXP}}_0^{\ell-c}$: $(z, x, *, *) \notin \Theta$ and $x \notin L$ but $\sigma = \mathsf{MAC}_a(m)$. Denote this event by E. We have that

$$\left|\Pr[\mathcal{A}(\widehat{\mathsf{EXP}}_0^{\ell-c}) = 1] - \Pr[\mathcal{A}(\overline{\mathsf{EXP}}_0^{\ell-c}) = 1]\right| \leq \Pr[\mathsf{E}(\overline{\mathsf{EXP}}_0^{\ell-c})].$$

We claim that $\Pr[\mathsf{E}(\overline{\mathsf{EXP}}_0^{\ell-c})] = \text{negl}(\lambda)$ with $c = 0, 1$; otherwise, computational universal$_2$ of Ψ can be broken by adversary $\mathcal{D}'$ as follows. Without loss of generality, assume that $\Pr[\mathsf{E}(\overline{\mathsf{EXP}}_0^{\ell})]$ is non-negligible. Upon receiving $(\Psi, \alpha(k))$, $\mathcal{D}'$ simulates $\overline{\mathsf{EXP}}_0^{\ell}$ by playing the role of $\mathcal{D}$ and the challenger of $\mathcal{D}$, except the evaluation of $H_k(z, x)$ is done under his own challenger's help. Specifically, the ith $\mathsf{Chal}(z)$ query for $i \neq \ell$ is answered by himself using witness w; for the ℓth $\mathsf{Chal}(z)$ query, he takes $x^* \leftarrow X \backslash L$ and issues $\mathsf{Eval}_2(z, x^*)$ query to evaluate $H_k(z, x^*)$; for a $\mathsf{Comp}(z, x, \sigma, m)$ query, he issues $\mathsf{Eval}_1(z, x)$ to his own challenger and in turn he will receive $(a, s) = \perp$ if $x \notin L$; $H_k(z, x)$ otherwise. In case of the former, he records (z, x) into a list $\mathcal{L}$ and rejects normally (as in $\overline{\mathsf{EXP}}_0^{\ell-c}$); in case of the latter, $\mathcal{D}'$ answers the query using the received $H_k(z, x)$ normally. The remaining simulation is normal. This simulation is perfectly consistent with $\overline{\mathsf{EXP}}_0^{\ell-c}$ for both cases $c = 0$ and 1. At the end of game, if $c = 1$, $\mathcal{D}'$ outputs 0/1 randomly; otherwise, he takes (z^*, y^*) randomly from $\mathcal{L}$ and issues $\mathsf{Test}(z^*, y^*)$ query. In turn he will receive (a_b^*, s_b^*), where $(a_0^*, s_0^*) = H_k(z^*, y^*)$ or $(a_1, s_1) \leftarrow \{0, 1\}^{2\lambda}$. Then he reviews all the Comp queries in $\mathcal{L}$ with forms (z^*, y^*, σ, m) for any σ, m and denotes event $\sigma = \mathsf{MAC}_{a_b^*}(m)$ by inc. In case of inc, $\mathcal{D}'$ outputs 0; otherwise he outputs 1. Note that if $b = 1$, then inc occurs to y^* negligibly by ungorgeability of MAC. If $b = 0$, then inc event is E event in $\overline{\mathsf{EXP}}_0^{\ell}$ that occurs to (z^*, y^*). Since any E event must occur to some (z, x) in $\mathcal{L}$, inc occurs in $\mathcal{D}$'s algorithm for $b = 0$ with probability at least $\Pr[\mathsf{E}(\overline{\mathsf{EXP}}_0^{\ell})]/|\mathcal{L}|$, non-negligible. The non-negligible gap of the two cases implies non-negligible advantage of $\mathcal{D}'$, a contradiction. Hence,

$$\Pr[\mathcal{A}(\overline{\mathsf{EXP}}_0^{\ell}) = 1] - \Pr[\mathcal{A}(\overline{\mathsf{EXP}}_0^{\ell-1}) = 1]$$

is non-negligible, which is the success advantage of $\mathcal{D}$, a contradiction.

B Proof of Lemma 2

Use Left and Right to denote the left- and right-hand side of (4.1), respectively. First of all, we show that Left $\geq$ Right by presenting an algorithm $\mathcal{A}_0$ achieving Right. Here $\mathcal{A}_0$ simply draws the ball from box 1 until the red ball is picked. Then, it turns to box 2 using the same strategy, then box 3, etc. If it draws a red ball from box ℓ before t picks are used up, it succeeds; otherwise, it fails. Let the red ball in box i be obtained by using x_i picks. Then, it is simple to verify that $x_i \leftarrow [a_i]$. Hence, the success probability of $\mathcal{A}_0$ is exactly the right-hand side of (4.1).

It remains to show that Left $\leq$ Right. When $\ell = 0$, the conclusion holds trivially since both sides are 1. Assume $\ell \geq 1$. When $n = 1$, the two sides of (4.1) equal $\min\{t/a_1, 1\}$ for the (only) case $\ell = 1$. For $n \geq 2$ and $\ell \geq 1$, we use induction on t. Note that $\Theta_{t,n,\ell}(a_1, \ldots, a_n)$ can always be achieved by a deterministic algorithm by computing the maximum success probability over the randomness of $\mathcal{A}$. Hence, we assume that $\mathcal{A}$ is deterministic. When $t = 0$, the two sides of (4.1) are zero. The conclusion holds trivially. When $t = 1$, assume that the box id of the first pick by $\mathcal{A}$ is j. Then

$$\begin{aligned}
\Theta_{1,n,\ell}(a_1, \ldots, a_n) &= a_j^{-1} \cdot \Theta_{0,n,\ell-1}(a_1, \ldots, a_{j-1}, 0, a_{j+1}, \ldots, a_n) \\
&\quad + (1 - a_j^{-1})\Theta_{0,n,\ell}(a_1, \ldots, a_{j-1}, a_j - 1, a_{j+1}, \ldots, a_n) \\
&= a_j^{-1} \cdot \Theta_{0,n-1,\ell-1}(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n) \\
&\quad + (1 - a_j^{-1})\Theta_{0,n,\ell}(a_1, \ldots, a_{j-1}, a_j - 1, a_{j+1}, \ldots, a_n).
\end{aligned}$$

If $\ell = 1$, then this gives $\Theta_{1,n,\ell}(a_1, \ldots, a_n) = a_j^{-1} \leq a_1^{-1} =$ Right. Hence, Left $\leq$ Right. If $\ell \geq 2$, then $\Theta_{1,n,\ell}(a_1, \ldots, a_n) = 0$ as

$$\begin{aligned}
&\Theta_{0,n-1,\ell-1}(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n) = 0, \\
&\Theta_{0,n,\ell}(a_1, \ldots, a_{j-1}, a_j - 1, a_{j+1}, \ldots, a_n) = 0.
\end{aligned}$$

In addition, since $x_1 + \cdots + x_\ell \geq \ell > 1$, we have Right $= 0$. Hence, Left $=$ Right.

Now assume Left $\le$ Right for $t-1$ draws, which implies Left $=$ Right for $t-1$ draws since Left $\ge$ Right is proven at the beginning. We consider t ($t \ge 2$). Assume that the first box chosen by $\mathcal{A}$ is j. Then,

$$\begin{aligned}\Theta_{t,n,\ell}(a_1,\ldots,a_n) &= a_j^{-1}\cdot\Theta_{t-1,n,\ell-1}(a_1,\ldots,a_{j-1},0,a_{j+1},\ldots,a_n)\\ &\quad+(1-a_j^{-1})\Theta_{t-1,n,\ell}(a_1,\ldots,a_{j-1},a_j-1,a_{j+1},\ldots,a_n)\\ &= a_j^{-1}\cdot\Theta_{t-1,n-1,\ell-1}(a_1,\ldots,a_{j-1},a_{j+1},\ldots,a_n)\\ &\quad+(1-a_j^{-1})\Theta_{t-1,n,\ell}(a_1,\ldots,a_{j-1},a_j-1,a_{j+1},\ldots,a_n)\end{aligned}$$

There are two cases.

Case $a_j = 1$. Then,

$$\Theta_{t,n,\ell}(a_1,\ldots,a_n) = \Theta_{t-1,n-1,\ell-1}(a_1,\ldots,a_{j-1},a_{j+1},\ldots,a_n).$$

Let $a_1^*,\ldots,a_{\ell-1}^*$ be the $\ell-1$ smallest numbers in $\{a_1,\ldots,a_n\}\backslash\{a_j\}$. By induction, we have

$$\begin{aligned}&\Theta_{t-1,n-1,\ell-1}(a_1,\ldots,a_{j-1},a_{j+1},\ldots,a_n)\\ &\quad= \Pr\big[x_1^*+\cdots+x_{\ell-1}^*\le t-1 : x_i^*\leftarrow[a_i^*]\big].\end{aligned}$$

If $j>\ell$, then $a_1=\cdots=a_\ell=1$ as $a_1\le a_2\le\cdots\le a_n$ and $a_j=1$. Hence, $(a_1^*,\ldots,a_{\ell-1}^*)$ equals $(a_1,\ldots,a_{\ell-1})$. Therefore,

$$\Pr\Big[\sum_{i=1}^{\ell-1}x_i^*\le t-1 : x_i^*\leftarrow[a_i^*]\Big] = \Pr\Big[\sum_{i=1}^{\ell-1}x_i\le t-1 : x_i\leftarrow[a_i]\Big].$$

Since $a_\ell=1$, it follows that $x_\ell=1$ always holds when $x_\ell\leftarrow[a_\ell]$. So

$$\Pr\Big[\sum_{i=1}^{\ell-1}x_i\le t-1 : x_i\leftarrow[a_i]\Big] = \Pr\Big[\sum_{i=1}^{\ell}x_i\le t : x_i\leftarrow[a_i]\Big].$$

The induction holds in this case.

If $j\le\ell$, then $\{a_1^*,\ldots,a_{\ell-1}^*\}=\{a_1,\ldots,a_{j-1},a_{j+1},\ldots,a_\ell\}$. Hence,

$$\begin{aligned}\Pr\Big[\sum_{i=1}^{\ell-1}x_i^*\le t-1 : x_i^*\leftarrow[a_i^*]\Big] &= \Pr\Big[\sum_{1\le i\le\ell,i\ne j}x_i\le t-1 : x_i\leftarrow[a_i]\Big]\\ &= \Pr\Big[\sum_{i=1}^{\ell}x_i\le t : x_i\leftarrow[a_i]\Big],\end{aligned}$$

where the last equality holds since $a_j = 1$ and hence $x_j = 1$ holds always. Hence, the induction holds in this case too.

Case $a_j > 1$ and $j > \ell$. In this case, $\{a_1, \dots, a_{\ell-1}\}$ are the $\ell - 1$ smallest numbers in $\{a_1, \dots, a_n\} \backslash \{a_j\}$. By induction assumption on $t - 1$, we have

$$\begin{aligned} &a_j^{-1} \cdot \Theta_{t-1,n-1,\ell-1}(a_1, \dots, a_{j-1}, a_{j+1}, \dots, a_n) \\ &\quad = a_j^{-1} \cdot \Pr\Big[\sum_{i=1}^{\ell-1} x_i \le t-1 : x_i \leftarrow [a_i]\Big]. \end{aligned}$$

In addition, if $a_j > a_\ell$, then $\{a_1, \dots, a_\ell\}$ are the ℓ smallest numbers in $\{a_1, \dots, a_{j-1}, a_j - 1, a_{j+1}, \dots, a_n\}$. Hence,

$$\begin{aligned} &(1 - a_j^{-1})\Theta_{t-1,n,\ell}(a_1, \dots, a_{j-1}, a_j - 1, a_{j+1}, \dots, a_n) \\ &\quad = (1 - a_j^{-1}) \cdot \Pr\Big[\sum_{i=1}^{\ell} x_i \le t-1 : x_i \leftarrow [a_i]\Big]. \end{aligned}$$

Therefore, in (4.1), we have that Right − Left equals

$$\Pr\Big[\sum_{i=1}^{\ell} x_i = t\Big] + a_j^{-1} \cdot \Pr\Big[\sum_{i=1}^{\ell} x_i \le t-1\Big] - a_j^{-1} \cdot \Pr\Big[\sum_{i=1}^{\ell-1} x_i \le t-1\Big].$$

We need to show that Right − Left ≥ 0. We split event $\sum_{i=1}^{\ell-1} x_i \le t-1$ into two sub-events

$$A : t-1 \ge \sum_{i=1}^{\ell-1} x_i \ge t - a_\ell \quad \text{and} \quad B : \sum_{i=1}^{\ell-1} x_i \le t - 1 - a_\ell.$$

Note in case of event A, there exists $1 \le x_\ell^* \le a_\ell$ such that $x_\ell^* + \sum_{i=1}^{\ell-1} x_i = t$. Hence, as $\ell < j$,

$$\begin{aligned} \Pr\Big[\sum_{i=1}^{\ell} x_i = t\Big] - \Pr[A] &\ge \Pr\Big[\sum_{i=1}^{\ell} x_i = t \wedge x_\ell = x_\ell^*\Big] - a_j^{-1} \Pr[A] \\ &= a_\ell^{-1} \Pr[A] - a_j^{-1} \Pr[A] \ge 0. \end{aligned}$$

In case of event B, since $x_\ell \le a_\ell$ always holds,

$$a_j^{-1} \Pr[B] \le a_j^{-1} \Pr\Big[\sum_{i=1}^{\ell} x_i \le t-1\Big].$$

Hence, Right $\ge$ Left holds in this case.

If $a_j \le a_\ell$, then $a_j = a_\ell$ since by assumption $a_j \ge a_\ell$ for $j > \ell$ holds always. In this case, $\{a_1, \ldots, a_{\ell-1}, a_\ell - 1\}$ are the ℓ smallest numbers among $\{a_1, \ldots, a_{j-1}, a_j - 1, a_{j+1}, \ldots, a_n\}$. Hence,

$$
\begin{aligned}
&(1 - a_j^{-1})\Theta_{t-1,n,\ell}(a_1, \ldots, a_{j-1}, a_j - 1, a_{j+1}, \ldots, a_n) \\
&\quad = (1 - a_\ell^{-1}) \cdot \Pr\Big[x_\ell^* + \sum_{i=1}^{\ell-1} x_i \le t - 1 : x_i \leftarrow [a_i], x_\ell^* \leftarrow [a_\ell - 1]\Big] \\
&\quad = (1 - a_\ell^{-1}) \sum_{u=1}^{a_\ell - 1} \Pr\Big[x_\ell^* + \sum_{i=1}^{\ell-1} x_i \le t - 1 \wedge x_\ell^* = u : \\
&\qquad\qquad\qquad\qquad\qquad x_i \leftarrow [a_i], x_\ell^* \leftarrow [a_\ell - 1]\Big] \\
&\quad = a_\ell^{-1} \sum_{u=1}^{a_\ell - 1} \Pr\Big[u + 1 + \sum_{i=1}^{\ell-1} x_i \le t : x_i \leftarrow [a_i], i < \ell\Big] \\
&\quad = \sum_{u=1}^{a_\ell - 1} \Pr\Big[\sum_{i=1}^{\ell} x_i \le t \wedge x_\ell = u + 1 : x_i \leftarrow [a_i], i \le \ell\Big] \\
&\quad = \Pr\Big[\sum_{i=1}^{\ell} x_i \le t \wedge x_\ell > 1 : x_i \leftarrow [a_i]\Big].
\end{aligned}
$$

Further,

$$
\begin{aligned}
&a_j^{-1} \cdot \Theta_{t-1,n-1,\ell-1}(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n) \\
&\quad = a_\ell^{-1} \cdot \Pr\Big[\sum_{i=1}^{\ell-1} x_i \le t - 1 : x_i \leftarrow [a_i]\Big] \\
&\quad = \Pr\Big[\sum_{i=1}^{\ell} x_i \le t \wedge x_\ell = 1 : x_i \leftarrow [a_i]\Big].
\end{aligned}
$$

Combining the above two equations, we have that in this case Left = Right.

Case $a_j > 1$ and $j \le \ell$. In this case, $\{a_1, \ldots, a_\ell\} \backslash \{a_j\}$ are the $\ell - 1$ smallest numbers among $\{a_1, \ldots, a_n\} \backslash \{a_j\}$. By induction assumption on $t - 1$, we have

$$
\begin{aligned}
&a_j^{-1} \cdot \Theta_{t-1,n-1,\ell-1}(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n) \\
&\quad = a_j^{-1} \cdot \Pr\Big[\sum_{1 \le i \le \ell, i \ne j} x_i \le t - 1 : x_i \leftarrow [a_i]\Big]
\end{aligned}
$$

$$= \Pr\Big[\sum_{1\le i\le \ell} x_i \le t \wedge x_j = 1 : x_i \leftarrow [a_i]\Big].$$

Note that $\{a_1, \ldots, a_{j-1}, a_j - 1, a_{j+1}, \ldots, a_\ell\}$ are the ℓ smallest numbers in $\{a_1, \ldots, a_{j-1}, a_j - 1, a_{j+1}, \ldots, a_n\}$. Hence,

$$\begin{aligned}
&(1 - a_j^{-1})\Theta_{t-1,n,\ell}(a_1, \ldots, a_{j-1}, a_j - 1, a_{j+1}, \ldots, a_n)\\
&\quad = (1 - a_j^{-1}) \Pr\Big[x_j^* + \sum_{i=1, i\neq j}^{\ell} x_i \le t - 1 : x_i \leftarrow [a_i], x_j^* \leftarrow [a_j - 1]\Big]\\
&\quad = (1 - a_j^{-1}) \sum_{u=1}^{a_\ell - 1} \Pr\Big[x_j^* + \sum_{i=1, i\neq j}^{\ell} x_i \le t - 1 \wedge x_j^* = u :\\
&\qquad\qquad x_i \leftarrow [a_i], i \neq j, x_j^* \leftarrow [a_j - 1]\Big]\\
&\quad = a_j^{-1} \sum_{u=1}^{a_\ell - 1} \Pr\Big[u + 1 + \sum_{i=1, i\neq j}^{\ell} x_i \le t : x_i \leftarrow [a_i], i \neq j\Big]\\
&\quad = \sum_{u=1}^{a_\ell - 1} \Pr\Big[\sum_{i=1}^{\ell} x_i \le t \wedge x_j = u + 1 : x_i \leftarrow [a_i]\Big]\\
&\quad = \Pr\Big[\sum_{i=1}^{\ell} x_i \le t \wedge x_j > 1 : x_i \leftarrow [a_i]\Big].
\end{aligned}$$

Combining the above two equations, we conclude the result in this case. As a summary, the induction holds for all cases. This completes the proof of Lemma 2.

C Proof of Theorem 2

We modify the security game (denoted by Γ^{rea}) into games Γ_0 ($= \Gamma^{\mathrm{rea}}$), Γ_1, Γ_2 such that any adversary view (hence events $\mathsf{Non\text{-}Auth}_i$ or Succ as they are in the adversary view) between each neighboring pair is negligibly close. For simplicity, we regard $\mathsf{Execute}$ query as a result of four Send queries (i.e., $\mathsf{Send}(d, \cdot)$, $d = 0, 1, 2, 3$) and later will remove its effect on $\mathsf{Non\text{-}Auth}_i$ and Succ by analyzing these special Send queries. For simplicity, we assume the Normal condition: sampling $x \leftarrow D(L)$ never repeats the same x (as it is negligible; otherwise, $\mathcal{I}$ is not hard: given challenge x, sample $y \leftarrow D(L)$. Then $x = y$ for $x \leftarrow D(L)$ holds non-negligibly while $x \neq y$ always holds for $x \leftarrow D(X\backslash L)$).

Game Γ_1. We modify Γ_0 to Γ_1 with the following differences. $\mathsf{Send}(0, i, \ell_i, \mathrm{null})$ oracle defines $(k_0, k_1) \leftarrow \{0, 1\}^{2\lambda}$ (instead of $(k_0, k_1) = H_\theta(i, x)$). Γ_1 maintains a list $\mathcal{Q}$ of record (i, y, k_0, k_1). For consistency, $\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$ is handled as follows. First check if $(i, y, u_0, u_1) \in \mathcal{Q}$ for some (u_0, u_1). If no, process normally using θ; otherwise, define $(k_0', k_1') = (u_0, u_1)$ and proceed normally.

Lemma 9. $\mathsf{View}(\mathcal{A}, \Gamma_0) \approx \mathsf{View}(\mathcal{A}, \Gamma_1)$.

Proof. If the views of $\mathcal{A}$ are distinguished by $\mathcal{D}$, we construct adversary $\mathcal{B}$ to violate Lemma 1. Upon desc$(\Psi), \Theta = \alpha(\theta)$, $\mathcal{B}$ simulates Γ_0 as follows. Let $\mathcal{Q} = \{\}$.

$\mathsf{Send}(0, i, \ell_i, \text{null})$. Upon this, $\mathcal{B}$ queries $\mathsf{Chal}(i)$ and in turn receives (x, a_c, s_c). He defines $(k_0, k_1) = (a_c, s_c)$ and normally finishes the simulation in this query. Finally, he defines $\mathsf{state}_i^{\ell_i} = C_i|S|y|k_0|k_1$ and updates $\mathcal{Q} = \mathcal{Q} \cup \{(i, y, k_0, k_1)\}$. Note in this case, the challenger of $\mathcal{B}$ will update his list $\Omega = \Omega \cup \{(i, x, k_0, k_1)\}$.

$\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$. Upon this, he computes $x = \mathsf{T}^*(\pi_i, y)$ and issues $\mathsf{Comp}(i, x, \tau_0, C_i|S|y)$. In turn, he will receive (a, s). If $(a, s) = \perp$, he rejects; otherwise, he defines $(k_0', k_1') = (a, s)$ and finishes the remaining simulation in this query normally. In the later case, he also updates $\mathsf{stat}_S^{\ell_S} = C_i|S|y|\zeta|k_0'|k_1'$. Note that if x was generated in $\mathsf{Send}(0, i, \cdot)$, then $(i, x, a_c, s_c) \in \Omega$. In this case, the simulation is consistent with Γ_c: if $\tau_0 = \mathsf{MAC}_{a_c}(C_i|S|y)$, then Comp oracle returns $(a, s) = (a_c, s_c)$; otherwise, it returns $(a, s) = \perp$ (and $\mathcal{B}$ will correctly reject τ_0). If x is not generated in $\mathsf{Send}(0, i, \cdot)$ (note it could be generated by client $i' \neq i$), then $(i, x, *, *) \notin \Omega$ and hence τ_0 will be verified by the challenger of $\mathcal{B}$ using $(k_0, k_1) = H_\theta(i, x)$ computed using θ. In this case, $(a, s) = \perp$ if τ_0 is invalid; $(a, s) = (k_0, k_1)$ otherwise. Hence, in any case, the simulation in this query is consistent with Γ_c.

$\mathsf{Send}(2, i, \ell_i, S|\zeta|\tau_1)$. Upon this case, use $\mathsf{stat}_i^{\ell_i}$ to simulate normally. Finally, if τ_1 is accepted, update $\mathsf{stat}_i^{\ell_i} = C_i|S|k_1$.

$\mathsf{Send}(3, S, \ell_S, \tau_2)$. Upon this case, use $\mathsf{stat}_S^{\ell_S}$ to simulate normally. Finally, if τ_2 is accepted, update $\mathsf{stat}_S^{\ell_S} = C_i|S|k_1'$.

$\mathsf{Reveal}(U, \ell_U)$ and $\mathsf{Test}(U, \ell_U)$. This occurs only when $\Pi_U^{\ell_U}$ is successfully completed. In this case, $\mathsf{sk}_U^{\ell_U}$ is well defined in $\mathsf{stat}_i^{\ell_i}$ above. So the simulation is normal.

$\mathsf{Corrupt}(i)$. As seen above, $\mathsf{stat}_i^{\ell_i}$ is well defined and π_i is known. Hence, the simulation is normal.

From the description of $\mathcal{B}$, we can see that when challenge bit $c = 0$, the simulated game by $\mathcal{B}$ is Γ_0; otherwise, it is Γ_1. Hence, the distinguishability between Γ_0 and Γ_1 leads to violate Lemma 1. □

Game Γ_2. We modify Γ_1 to Γ_2 as follows. In oracle $\mathsf{Send}(0, i, \ell_i, \text{null})$, take $x \leftarrow X$ (instead of $x \leftarrow L$). Note that since w is not used in the simulation of Γ_1, no further change is required toward the consistency with this modification. By simply reducing to hardness of L, we have the following result.

Lemma 10. $\mathsf{View}(\mathcal{A}, \Gamma_1) \approx \mathsf{View}(\mathcal{A}, \Gamma_2)$.

We analyze Γ_2. Recall that, in $\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$, when $(i, y, *, *) \notin \mathcal{Q}$, we define $(k'_0, k'_1) = H_\theta(i, x)$ and verify τ_0 with k'_0. Consider a Bad event in this query: $(i, y, *, *) \notin \mathcal{Q}$ and $\mathsf{T}^*(\pi_i, y) \notin L$ but τ_0 is valid.

Lemma 11. $\Pr[\mathsf{Bad}(\Gamma_2)] = \text{negl}(\lambda)$.

Proof. Let us assume that the lemma is not true. Let *an irregular query* be a $\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$ query where $(i, y, *, *) \notin \mathcal{Q}$ and $\mathsf{T}^*(\pi_i, y) \notin L$. Let the number of irregular queries be bounded by ν. Use Bad_i to represent the event: the ith irregular query is the *first* Bad event. Note that when Bad occurs, there exists a unique Bad_i event.

We now construct an adversary $\mathcal{A}'$ to break the computational universal$_2$ property of Ψ. Upon $\text{desc}(\Psi), \Theta$, $\mathcal{A}'$ takes $t \leftarrow \{1, \ldots, \nu\}$ and initializes π_i for each C_i and simulates Γ_2, except when he needs to use θ, which is one of the following scenarios (especially note that (k_0, k_1) in $\mathsf{Send}(0, \cdot)$ is taken randomly in $\{0, 1\}^{2\lambda}$ without using θ). (1) S is corrupted and θ should be given to $\mathcal{A}$. This will not occur since we assume S is uncorrupted. (2) In $\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$, $\mathcal{A}'$ will use θ to compute (k'_0, k'_1) *in case of* $(i, y, *, *) \notin \mathcal{Q}$. In this case, $\mathcal{A}'$ can compute $x = \mathsf{T}^*(\pi_i, y)$ and query his Eval_1 oracle to compute $H_\theta(i, x)$. When $x \in L$, he will receive $H_\theta(i, x)$; when $x \notin L$, he will receive $\perp$. For the former case, he proceeds normally; for the latter case, it is an irregular query. If this is the jth irregular query for $j < t$, then he rejects τ_0; if it is the tth irregular query, he issues (i, x) as a Test query, in turn he will receive (a_c, s_c) for challenge bit c. If $\tau_0 = \mathsf{MAC}_{a_c}(C_i|S|y)$, he outputs 0; otherwise 1. First of all, when $c = 1$, a_c is independent of the adversary view prior to the current query, by unforgeability of MAC, $\tau_0 = \mathsf{MAC}_{a_1}(C_i|S|y)$ holds negligibly only. We ignore this tiny probability. When $c = 0$ and t is correct, the adversary view till the current query is identical to his view in Γ_2. In this case, the validity of τ_0 is a Bad_t event, in which $\mathcal{A}'$ must output 0. Since Bad_t event implies that τ_0 is valid and that upon such an event the simulation by $\mathcal{A}'$ prior to the tth irregular query is identical to Γ_2 (even without considering the output of $\mathcal{A}'$ in the case $c = 0$ with an incorrect t), we always have that

$$\left|\Pr[\mathcal{A}'^{\mathsf{Eval}_1(0,\cdot)} = 0] - \Pr[\mathcal{A}'^{\mathsf{Eval}_1(1,\cdot)} = 0]\right| \geq \Pr[\mathsf{Bad}_t(\Gamma_2)] - \mathrm{negl}(\lambda)$$
$$\geq \frac{\Pr[\mathsf{Bad}(\Gamma_2)]}{\nu} - \mathrm{negl}(\lambda)$$

(where $\Pr[\mathsf{Bad}_t(\Gamma_2)] = \Pr[\mathsf{Bad}(\Gamma_2)]/\nu$ as t is uniformly random), non-negligible, a contradiction. □

For simplicity, we now assume that Bad event never occurs.

Lemma 12. *If initiator $\Pi_i^{\ell_i}$ accepts $\mathsf{msg}_2 = S|\zeta^*|\tau_1^*$, it has a unique partner $\Pi_S^{\ell_S}$.*

Proof. Note that $\mathsf{sid}_i^{\ell_i} = C_i|S|y^*|\zeta^*$. Since S will not sample the same ζ^* twice (ignore the negligible probability), it follows that the number of partnered instance $\Pi_S^{\ell_S}$ for $\Pi_i^{\ell_i}$ is at most one. It suffices to prove the existence. If it does not exist, we show MAC is forgeable. Assume that $\mathsf{stat}_i^{\ell_i}$ after sending msg_1 is $C_i|S|y^*|k_0^*|k_1^*$. Then, reviewing the definitions of oracles in Γ_2, *besides computing* $\mathsf{MAC}_{k_0^*}()$ *function*, k_0^* (and its identical copy $k_0^{*'}$) will be used only in the following scenarios *before* $\Pi_i^{\ell_i}$ *verifies* msg_2: k_0^* is revealed due to the corruption of C_i (note S is uncorrupted), which is impossible since a corrupted party is controlled by $\mathcal{A}$ and so $\mathsf{Send}(2, i, \ell_i, \mathsf{msg}_2)$ query would not have occurred. Hence, prior to verifying msg_2 by $\Pi_i^{\ell_i}$, Γ_2 uses k_0^* only for evaluating $\mathsf{MAC}_{k_0^*}()$. To reduce to the unforgeability of MAC, it suffices to show that prior to verifying msg_2 in $\Pi_i^{\ell_i}$, the simulator never evaluates and outputs $\mathsf{MAC}_{k_0^*}()$ with input $C_i|S|y^*|\zeta^*|1$. Otherwise, since τ_0, τ_1, τ_2 have different input formats, this evaluation must be done by S in some $\mathsf{Send}(1, S, \ell'_S, \cdot)$, which already implies that $\Pi_s^{\ell'_s}$ is partnered with C_i, contradicting our assumption. Thus, the validity of τ_1^* implies breaking the unforgeability of MAC. □

Lemma 13. *Let* $\mathsf{pid}_S^{\ell_S}$ $(:= C_i)$ *be uncorrupted. If* $(i, y^*, \cdot, \cdot) \in \mathcal{Q}$ *in* $\mathsf{Send}(1, S, \ell_S, C_i|y^*|\tau_0^*)$ *oracle and* τ_2^* *is accepted in* $\mathsf{Send}(3, S, \ell_S, \tau_2^*)$, *then* $\Pi_S^{\ell_S^*}$ *has a unique partner* $\Pi_i^{\ell_i}$.

Proof. The number of partners of $\Pi_S^{\ell_S}$ is at most one, due to Normal condition on x. It suffices to prove the existence. Assume this is not true. By assumption, in $\mathsf{Send}(1, S, \ell_S^*, C_i|y^*|\tau_0^*)$, it holds that $(i, y^*, k_0^*, k_1^*) \in \mathcal{Q}$ for some k_0^*, k_1^* and it also holds that $\tau_0^* = \mathsf{MAC}_{k_0^*}(C_i|S|y^*)$ (otherwise, τ_0^* in msg_1 was rejected and it would be impossible for $\Pi_S^{\ell_S}$ to verify and accept τ_2^*). Hence, the fact that (i, y^*, k_0^*, k_1^*) was recorded in $\mathcal{Q}$ implies that $\Pi_i^{\ell_i}$ for some ℓ_i must have sampled $x = \mathsf{T}^*(\pi_i, y^*)$. By Normal condition, $\Pi_i^{\ell_i}$ is the only instance that samples this value. Since $\Pi_i^{\ell_i}$ is not partnered with $\Pi_S^{\ell_S}$, $\Pi_i^{\ell_i}$ does not compute $\mathsf{MAC}_{k_0^*}()$ with input $C_i|S|y^*|\zeta^*|2$, where ζ^* is generated by $\Pi_S^{\ell_S}$. As in the previous lemma, k_0^* is only used in evaluating $\mathsf{MAC}_{k_0^*}()$. To prove the lemma, it suffices to show that the simulator never evaluates and outputs $\mathsf{MAC}_{k_0^*}()$ with input $C_i|S|y^*|\zeta^*|2$. Otherwise, it must be done by some instance $\Pi_i^{\ell_i'}$ in C_i in generating msg_3 (recall inputs for τ_0, τ_1, τ_2 have different formats). Hence, since $C_i|S|y^*$ implies that $\Pi_i^{\ell_i'}$ samples x as $\mathsf{T}^*(\pi_i, y^*)$, it follows that $\ell_i' = \ell_i$, contradicting that $\Pi_i^{\ell_i}$ is not partnered with $\Pi_S^{\ell_S}$. Hence, if $\Pi_i^{\ell_i}$ does not exist, then a forgery for MAC is obtained, contradicting the MAC security. □

Lemma 14. $\Pr[\mathsf{Succ}(\mathcal{A}) \mid \neg\mathsf{Non\text{-}Auth}] = 1/2$ *in* Γ_2.

Proof. Let $\Pi_U^{\ell_U^*}$ be the test instance and $\mathsf{pid}_U^{\ell_U^*} = V$. Let $\mathsf{sid}_U^{\ell_U^*} = C_J|S|y^*|\zeta^*$. Then, $\{U, V\} = \{J, S\}$. If $U = J$, then $V = S$ and (by Lemma 12) there is the unique partnered $\Pi_S^{\ell_S^*}$ for $\Pi_J^{\ell_J^*}$. If $U = S$, then $V = J$.

In this case, if a partnered $\Pi_J^{\ell_J^*}$ in C_J for $\Pi_S^{\ell_S^*}$ does not exist, then $\Pi_S^{\ell_S^*}$'s accepting τ_2^* implies $\mathsf{Non\text{-}Auth}_J$ event. Hence, under $\neg\mathsf{Non\text{-}Auth}$ event, there is a partnered $\Pi_J^{\ell_J^*}$ for $\Pi_S^{\ell_S^*}$ and by Normal condition it is unique. So in any case, conditional on $\neg\mathsf{Non\text{-}Auth}$, there is a uniquely partnered $\Pi_V^{\ell_V^*}$ for $\Pi_U^{\ell_U^*}$. Let (k_0^*, k_1^*) be the uniformly random keys defined in $\mathsf{Send}(0, J, \ell_J^*, \text{nil})$ to replace $H_\theta(J, x^*)$ where $x^* = \mathsf{T}^*(\pi_J, y^*)$. Let $b \in \{0,1\}, \alpha_1 \in \{0,1\}^\lambda$ be the random number in Test oracle. We notice that in Γ_2, $\mathsf{sk}_U^{\ell_U^*} = k_1^*$ is taken uniformly random from $\{0,1\}^\lambda$. Let $\alpha_0 = \mathsf{sk}_U^{\ell_U^*}$. Let the randomness in the whole game for Γ_2, except k_1^*, b, α_1, be denoted by r. Use $\mathsf{view}_t(\mathcal{A})$ to denote the adversary view after the tth query. Then to prove the lemma, it suffices to show that $\mathsf{view}_t(\mathcal{A})$ for each t is deterministic in r, α_b. We actually also show that $\{\mathsf{stat}_i^{\ell_i}\}_{(i,\ell_i)\neq(J,\ell_J^*),(S,\ell_S^*)}$ is deterministic in r, α_b. Initially, $\mathsf{view}_0(\mathcal{A})$ only consists of public parameters and the conclusion holds. Assume it holds for $t-1$ queries. Consider query t.

$\mathsf{Send}(0, i, \ell_i, \text{null})$. The randomness in sampling x and the randomness for k_0 are from r. Hence, $C_i|y|\tau_0$ is deterministic in $\mathsf{view}_{t-1}(\mathcal{A})$ and r. Note that $\mathsf{stat}_i^{\ell_i} = C_i|S|y|k_0|k_1$. When $(i, \ell_i) \neq (J, \ell_J^*)$, k_1 is determined by r. Hence, the conclusion holds after this query.

$\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$. Oracle first checks if $(i, y, *, *) \in \mathcal{Q}$. If yes, extract k_0 from it and proceed normally. If no, compute $(k_0', k_1') = H_\theta(i, x)$ for $x = \mathsf{T}^*(\pi_i, y)$ and proceed normally. Notice that (i, y, k_0) as part of a record in $\mathcal{Q}$ is computed using the randomness r; ζ is generated using r too. θ is based on the randomness in the initialization of Γ_2 and hence based on r too. So adversary view in this query is deterministic in $\mathsf{view}_{t-1}(\mathcal{A})$ and r. If it outputs msg_2, then

$\mathsf{stat}_S^{\ell_S}$ is updated as $C_i|S|y|k_0|k_1$. As $\Pi_S^{\ell_S^*}$ is the unique partner of $\Pi_i^{\ell_i^*}$, when $(S, \ell_S) \neq (S, \ell_S^*)$, k_1 is computed using r. Hence, the conclusion holds after this query.

$\mathsf{Send}(2, \cdot)$ and $\mathsf{Send}(3, \cdot)$. $\mathsf{Send}(2, \cdot)$ and $\mathsf{Send}(3, \cdot)$ are deterministic in the view of $\mathcal{A}$ before the query and its session state. By the induction, the conclusion holds after this query.

$\mathsf{Reveal}(i, \ell_i)$. This query is $\mathsf{sk}_i^{\ell_i}$. By the restriction on Test definition, we have $\Pi_i^{\ell_i} \neq \Pi_S^{\ell_S^*}, \Pi_J^{\ell_J^*}$ and hence by induction, its internal state is deterministic in $\mathsf{view}_{t-1}(\mathcal{A})$ and r, α_b. Since $\mathsf{sk}_i^{\ell_i}$ is in his internal state, the conclusion holds after this query.

$\mathsf{Corrupt}(i)$. Upon this query π_i as well as $\{\mathsf{stat}_i^{\ell_i}\}_{\ell_i}$ will be available to $\mathcal{A}$. Since $i \neq J, S$ by Test restriction, by induction, the conclusion holds after this query.

$\mathsf{Test}(u, \ell_u^*)$. The reply in this query is α_b. The conclusion holds trivially after this query.

As a summary, our conclusion holds and hence $\mathsf{view}(\mathcal{A})$ is independent of b. □

Lemma 15. $\Pr[\mathsf{Non\text{-}Auth}_i(\mathcal{A}, \Gamma_2)] \leq \frac{Q_i}{|\mathcal{D}|} + \text{negl}(\lambda)$.

Proof. To prove the lemma, we show how to simulate Γ_2 when $\{\pi_i\}_i$ is random while the remaining randomness r of the game is fixed. Let $\mathcal{D}_i$ be the candidate space for π_i after each query. Our simulation has a $\star$-property: after query t, $\mathsf{view}_t(\mathcal{A})$ is unchanged over each $(\pi_1, \ldots, \pi_n) \in \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_n$. Hence, given $\mathsf{view}_t(\mathcal{A})$, $(\pi_1, \ldots, \pi_n)$ is uniformly distributed over $\mathcal{D}_1 \times \cdots \times \mathcal{D}_n$.

Initially, $\mathsf{view}_0(\mathcal{A}) = \langle \text{desc}(\Lambda), \alpha(\theta) \rangle$ which is independent of $(\pi_1, \ldots, \pi_n)$. Hence, $\mathcal{D}_1 = \cdots = \mathcal{D}_n = \mathcal{D}$ and $\star$-property holds. Assume this simulation is done for query $t-1$. Consider query t, which is one of the following.

$\mathsf{Send}(0, i, \ell_i, \text{null})$. Oracle takes $y \leftarrow X$, $(k_0, k_1) \leftarrow \{0,1\}^{2\lambda}$ and computes $\tau_0 = \mathsf{MAC}_{k_0}(C_i|S|y)$. Finally, update $\mathcal{Q} = \mathcal{Q} \cup \{(i, y, k_0, k_1)\}$. The adversary view in this query is $C_i|y|\tau_0$. For any $\{\pi_j\}_{j=1}^n \in \prod_{j=1}^n \mathcal{D}_j$, the adversary view in the current query is identical. By induction assumption, after this query, if $\mathcal{D}_j$, $t = 1, \ldots, n$, remains unchanged, $\star$-property holds. Finally, set $\mathsf{stat}_i^{\ell_i} = C_i|S|y|k_0|k_1$.

$\mathsf{Send}(1, S, \ell_S, C_i|y|\tau_0)$. Upon this, if $(i, y, k_0, k_1) \in \mathcal{Q}$, then (regardless of the concrete value for π_i), the oracle will take (k_0, k_1) from it and finish the remaining simulation in this query normally and keep all $\{\mathcal{D}_t\}$ unchanged. If $(i, y, k_0, k_1) \notin \mathcal{Q}$, oracle will use θ and π_i to verify τ_0, and (if valid) announce the success of $\mathcal{A}$, which has two cases.

(1) τ_0 is valid and $\mathsf{T}^*(\pi_i, y) \in L$. This case occurs only for at most one π_i (denoted by $\pi_i(y)$) by regularity property R-2 of $(\mathsf{T}, \mathsf{T}^*)$.

(2) τ_0 is valid and $\mathsf{T}^*(\pi_i, y) \notin L$. This is a Bad event in Γ_2 (negligible, ignored, see Lemma 11).

Hence, case (1) occurs (hence $\pi_i = \pi_i(y)$) with probability $\leq 1/|\mathcal{D}_i|$ by induction (since, given $\mathsf{view}_{t-1}(\mathcal{A})$, $\{\pi_j\}_j$ is uniform in $\prod_j \mathcal{D}_j$ and especially π_i is uniform in $\mathcal{D}_i$); when case (1) does not occur (i.e., τ_0 invalid), then the adversary view in this query is identical (i.e., reject) for any password setup: take $\pi_i \in \mathcal{D}_i \backslash \{\pi_i(y)\}$ and take $\pi_j \in \mathcal{D}_j$ for all $j \neq i$. Hence, in this case, if $\mathcal{D}_j$ for $j \neq i$ remains unchanged and $\mathcal{D}_i = \mathcal{D}_i \backslash \{\pi_i(y)\}$, $\star$-property holds. Finally, $\mathsf{stat}_S^{\ell_S} = C_i|S|y|k_0|k_1$ is well-defined.

Reveal, Test, Send$(2, \ldots)$, Send$(3, \ldots)$. They are processed only with a session state from Send$(0, \cdot)$ or Send$(1, \cdot)$, which is well-defined as above. Thus the simulation is perfect.

Corrupt(i). In this case, π_i is revealed and hence $\mathcal{D}_i$ is updated to $\{\pi_i\}$. Notice that $\{\mathsf{stat}_i^{\ell_i}\}_{\ell_i}$ are consistent with all $\{\pi_j\}_j \in \prod_j \mathcal{D}_j$ by induction. Thus, if we keep $\mathcal{D}_j$ unchanged for $j \neq i$, then $\star$-property still holds.

Now we consider Non-Auth$_i$ event. It occurs at either some $\Pi_i^{\ell_i}$ or $\Pi_S^{\ell_S}$ with $\mathsf{pid}_S^{\ell_S} = C_i$. By Lemma 12, it is impossible to the former. For the latter, by Lemma 13, it must hold that $(i, y, *, *) \notin \mathcal{Q}$ in Send$(1, S, \ell_S, C_i|y|\tau_0)$ query and hence case (1) (i.e., $\pi_i = \pi_i(y)$) must occur (since case (2) is negligible and ignored). It remains to calculate the probability $\pi_i = \pi_i(y)$ throughout the game. As analyzed above, it has a probability $1/|\mathcal{D}_i|$, conditional on that previous queries with $\mathsf{msg}_1 = C_i|*$ do not have such an event. Hence, as a summery, $\pi_i = \pi_i(y)$ occurs in the ℓth such a Send$(1, S, \cdot, C_i|\cdot|\cdot)$ query with probability

$$\frac{|\mathcal{D}|-1}{|\mathcal{D}|} \cdot \frac{|\mathcal{D}|-2}{|\mathcal{D}|-1} \cdots \frac{1}{|\mathcal{D}|-\ell-1} = \frac{1}{|\mathcal{D}|}.$$

We claim that there are at most Q_i Send$(1, S, \cdot, C_i|y|\cdot)$ queries for fixed C_i such that $(i, y, *, *) \notin \mathcal{Q}$ with Client$(\Pi_S^{\ell_S}) = C_i$. Indeed, although we decomposed Execute at the beginning of the theorem proof into four Send$(d, \cdot)$ queries, this treatment does not invalidate the above statement: in the special Send$(1, S, \ell_S, C_i|y|\tau_0)$ query (decomposed from query Execute(i, ℓ_i, S, ℓ_S)), $(i, y, *, *) \in \mathcal{Q}$ was recorded by $\Pi_i^{\ell_i}$ in Send$(0, i, \ell_i, \mathrm{null})$ (decomposed from the same Execute query). So Non-Auth$_i$ does not occur to such a special Send query. Thus,

$$\Pr[\text{Non-Auth}_i(\mathcal{A}, \Gamma_2)] \leq \frac{Q_i}{|\mathcal{D}|}. \qquad \square$$

We return to the proof of Theorem 2. As $\mathsf{Non\text{-}Auth}_i$ and Succ are in $\mathsf{view}(\mathcal{A})$, each is negligibly close between $\Gamma_0, \Gamma_1, \Gamma_2$. By Lemmas 14 and 15, our theorem follows.

Acknowledgments. The author would like to thank the anonymous referees for valuable comments that significantly improved our paper.

Bibliography

[1] F. Bao, Security analysis of a password authenticated key exchange protocol, in: *Proceedings of ISC 2003* (Bristol 2003), Springer, Berlin (2003), 208–217.

[2] M. Bellare, R. Canetti and H. Krawczyk, A modular approach to the design and analysis of authentication and key exchange protocols, in: *Proceedings of STOC'98* (Dallas 1998), ACM (1998), 419–428.

[3] M. Bellare, D. Pointcheval and P. Rogaway, Authenticated key exchange secure against dictionary attacks, in: *Proceedings of EUROCRYPT 2000* (Bruges 2000), Springer, Berlin (2000), 139–155.

[4] S. Bellovin and M. Merritt, Encrypted key exchange: Password-based protocols secure against dictionary attacks, in: *Proceedings of IEEE Symposium on Research in Security and Privacy* (1992), 72–84.

[5] S. Bellovin and M. Merritt, Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise, in: *Proceedings of CCS 1993* (Fairfax 1993), ACM (1993), 244–250.

[6] D. Boneh, R. A. DeMillo and R. J. Lipton, On the importance of eliminating errors in cryptographic computations, *J. Cryptology* **14** (2001), 101–119.

[7] M. Boyarsky, Public-key cryptography and password protocols: The multi-user case, in: *Proceedings of CCS 1999* (Singapore 1999), ACM (1999), 63–72.

[8] R. Canetti and H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in: *Proceedins of Eurocrypt 2001* (Innsbruck 2001), Springer, Berlin (2001), 453–474.

[9] O. Chevassut, P. A. Fouque, P. Gaudry and D. Pointcheval, Key derivation and randomness extraction, preprint (2005), `http://eprint.iacr.org/2005/061.pdf`.

[10] R. Cramer and V. Shoup, Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption, in: *Proceedings of EUROCRYPT 2002* (Amsterdam 2002), Springer, Berlin (2002), 45–64.

[11] W. Diffie, P. van Oorschot and M. Wiener, Authentication and authenticated key exchanges, *Des. Codes Cryptogra.* **2** (1992), 107–125.

[12] R. Gennaro, Faster and shorter password-authenticated key exchange, in: *Proceedings of TCC 2008* (New York 2008), Springer, Berlin (2008), 589–606.

[13] R. Gennaro and Y. Lindell, A framework for password-based authenticated key exchange, in: *Proceedings of EUROCRYPT 2003* (Warsaw 2003), Springer, Berlin (2003), 524–543.

[14] O. Goldreich and Y. Lindell, Session-key generation using human passwords only, in: *Proceedings of CRYPTO 2001* (Santa Barbara 2001), Springer, Berlin (2001), 408–432.

[15] L. Gong, T. Mark, A. Lomas, R. Needham and J. H. Saltzer, Protecting poorly chosen secrets from guessing attacks, *IEEE J. Sel. Areas Comm.* **11** (1993), 648–656.

[16] A. Groce and J. Katz, A new framework for efficient password-based authenticated key exchange, in: *Proceedings of CCS 2010* (Chicago 2010), ACM (2010), 516–525.

[17] S. Halevi and H. Krawczyk, Public-key cryptography and password protocols, in: *Proceedings of CCS 1998* (San Francisco 1998), ACM (1998), 122–131.

[18] S. Halevi and H. Krawczyk, Public-key cryptography and password protocols, *ACM Trans. Inf. Syst. Secur.* **2** (1999), 230–268.

[19] D. Hofheinz and E. Kiltz, Secure hybrid encryption from weakened key encapsulation, in: *Proceedings of CRYPTO 2007* (Santa Barbara 2007), Springer, Berlin (2007), 553–571.

[20] D. Hofheinz and E. Kiltz, Practical chosen ciphertext secure encryption from factoring, in: *Proceedings of EUROCRYPT 2009* (Cologne 2009), Springer, Berlin (2009), 313–332.

[21] D. P. Jablon, Extended password key exchange protocols immune to dictionary attacks, in: *Proceedings of WETICE 1997* (Cambridge 1997), IEEE Computer Society (1997), 248–255.

[22] S. Jiang and G. Gong, Password based key exchange with mutual authentication, in: *Proceedings of SAC 2004* (Waterloo 2004), Springer, Berlin (2004), 267–279.

[23] J. Katz, R. Ostrovsky and M. Yung, Efficient password-authenticated key exchange using human-memorable passwords, in: *Proceedings of EUROCRYPT 2001* (Innsbruck 2001), Springer, Berlin (2001), 475–494.

[24] J. Katz and V. Vaikuntanathan, Smooth projective hashing and password-based authenticated key exchange from lattices, in: *Proceedings of ASIACRYPT 2009* (Tokyo 2009), Springer, Berlin (2009), 636–652.

[25] J. Katz and V. Vaikuntanathan, Round-optimal password-based authenticated key exchange, in: *Proceedings of TCC 2011* (Providence 2011), Springer, Berlin (2011), 293–310.

[26] H. Krawczyk, SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie–Hellman and its use in the IKE-protocols, in: *Proceedings of CRYPTO 2003* (Santa Barbara 2003), Springer, Berlin (2003), 400–425.

[27] K. Kurosawa and Y. Desmedt, A new paradigm of hybrid encryption scheme, in: *Proceedings of CRYPTO 2004* (Santa Barbara 2004), Springer, Berlin (2004), 426–442.

[28] S. Lucks, Open key exchange: How to defeat dictionary attacks without encrypting public keys, in: *Proceedings of Security Protocols Workshop* (Paris 1997), Springer, Berlin (1997), 79–90.

Received May 14, 2012; revised April 14, 2013; accepted September 29, 2013.

Author information

Shaoquan Jiang, School of Computer Science and Engineering, University of Electronic Science and Technology of China; and Institute of Information Security, Mianyang Normal University, Mianyang 621000, P. R. China.
E-mail: shaoquan.jiang@gmail.com