Quantum-Safe Round-Optimal Password Authentication for Mobile Devices

Zengpeng Li, Ding Wang, and Eduardo Morais

Abstract—Password authentication is the dominant form of access control for the Web and mobile devices, and its practicality and ubiquity is unlikely to be replaced by any other authentication approaches in the foreseeable future. To guarantee the security of data communication and mitigate the problem of password-cracking, a *Password Authenticated Key Exchange* (PAKE) system can be deployed between two peer participants. The main drawback of traditional PAKE is that passwords are exposed in plaintext when the remote server is compromised. To overcome this limitation, it is recommended by industry standards (such as SRP family RFC 5054, RFC6628, RFC7914, OPAQUE, *etc*) to use *asymmetric*-PAKE protocols, which enable the server to store a hash of the user's password with a random salt, providing guarantees that the user's password is never transmitted in plain-text to the server when login. However, most of the existing *asymmetric*-PAKE protocols either are based on traditional hash functions under random oracles, or depend on non-quantum-secure hardness assumptions and become insecure in the quantum era. To bridge the gap between *asymmetric*-PAKE and quantum-security, in this article, we resort to *smooth projective hash functions* (SPHF) and *commitment*-based *password-hashing schemes* (PHS) over lattice-based cryptography, and construct an asymmetric PAKE protocol secure against quantum attacks. Our construction eliminates the costly non-interactive zero-knowledge (NIZK) method, bypasses assumptions of the random oracle model, and achieves quantum resistance. We also show that our asymmetric-PAKE protocol can achieve balanced security and efficiency under the Bellare-Pointcheval-Rogaway (BPR) model. Finally, we develop a prototype implementation of our instantiation and use it to evaluate its performance in realistic settings.

Index Terms—Post quantum security, password authentication, asymmetric password authenticated key exchange, smooth projective hash function, password hashing scheme, lattice-based cryptography

1 Introduction

Passwords have various advantages of being memorable, avoiding complex and computation-consuming public key infrastructure (PKI) for managing certificates and dedicated hardware for storing secret keys. *Password hashing scheme* (PHS), also known as password-based key derivation functions (PBKDFs), e.g., PBKDF1, PBKDF2, Bcrypt, Scrypt, and M3lcrypt *etc* [1], [2], are generally used to derive one or more secret keys from a secret value such as a master key, a password, or a passphrase. As an integral part of the authentication mechanism, PHS could hash the password and store them at the server database with the username and other important information together, so that the server can check against the client's registered password when the client tries to log in. But the security of most of existing PBKDFs builds on hashing algorithms under the random oracle model (ROM), while limitations of ROM have been well discussed in [3].

On the other hand, passwords are the default authentication approach and are unlikely to be replaced in the near future [4], [5], [6], [7]. To prevent unauthorized access and to mitigate the problem of compromised password database, various approaches are proposed for constructing authentication schemes, such as [8], [9], [10], [11]. In essence, these schemes are variants of the password-authenticated key exchange (PAKE in short) protocol, as shown in Fig. 1. In a nutshell, PAKE can be achieved if the server and the client, who hold the same password, prove to each other that they know the password without disclosing any useful information about it, but they also establish a shared secret session key at the end, assuring security against (offline) password-guessing attacks. For the convenience of the reader, this setting is abbreviated to *symmetric*-PAKE in this work. Considerable efforts (e.g., [12], [13]) have been devoted to designing secure and efficient symmetric-PAKE protocols.

However, in reality, *asymmetric*-PAKE (*a.k.a.*, augmented or verifier-based PAKE) protocols first proposed by Bellovin and Merrit [14], [15] are more widely accepted and used by most of existing client-to-server Internet or Internet-of-Things authentication schemes. The main reason is that asymmetric-PAKE can limit the impact of password leakage. In particular, asymmetric-PAKE allows a server to keep a salted-hash the password, which guarantees that the adversary has to use offline dictionary attack to recover it from the salted-hash if the server is compromised. In other words, as the server only stores the knowledge of the

- *Zengpeng Li is with the College of Cyber Science, Nankai University, Tianjin 300350, China, and with The State Key Laboratory of Integrated Services Networks, Xidian University, and also with the College of Computer Science and Technology, Qingdao University, Qingdao 266701, China. E-mail: zengpengli@hotmail.com.*
- *Ding Wang is with the College of Cyber Science, Nankai University, Tianjin 300350, China, and also with the Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin 300350, China. E-mail: wangding@nankai.edu.cn; wangding@pku.edu.cn.*
- *Eduardo Morais is with the Input Output Hong Kong (IOHK) Ltd., Hong Kong. E-mail: eduardo.morais@gmail.com.*

Manuscript received 24 Mar. 2020; revised 16 Sept. 2020; accepted 28 Oct. 2020. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding authors: Ding Wang.)
Digital Object Identifier no. 10.1109/TDSC.2020.3040776

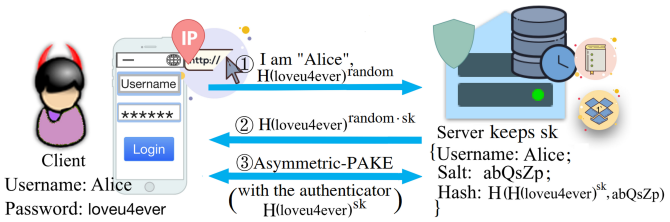

Fig. 1. An exemplary overview of asymmetric authentication. For better illustration, the client registers with a password and her username, then the server stores them in the salted hash. Next, when the client authenticates with her username and randomized password as shown in flow ①, where random stands for a random nonce and $H(\cdot)$ a hash function, the server then exponentiates the received randomized password with its secret key sk and send it back to the user, as shown in flow ②. Finally, the client logins if the authenticator $H(loveu4ever)^{sk}$ passed, and the client establishes the session key with the server using the password authenticated key exchange mechanism, as shown in flow ③.

randomized password with the corresponding random salt, which impedes the attacker but enables the user to take action upon server leakage.

There have been a number of asymmetic-PAKE protocols proposed and formally analyzed, such as [16], [17], [18], [19], [20] in the standard model, and [4], [21], [22], [23], [24] in the random oracle model. More concretely, Gentry *et al.* [16] proposed an ideal functionality in the Universal Composable (UC) framework [16] with a generic transformation given by the Ω-method. Benhamouda and Pointcheval [17] with the follow-up works [18], [19], [20] proposed the first asymmetric construction in the standard model with game-based security analysis. However, no lattice-based asymmetric-PAKE construction in the standard model has been proposed so far. To our knowledge, there are some lattice-based symmetric-PAKE constructions [25], [26], [27], [28] based on *smooth projective hash function* (SPHF in short), which have been proposed following the methodology of Gennaro-Lindell (i.e., GL-SPHF) [13] and Katz-Vaikuntanathan (i.e., KV-SPHF) [29], respectively. The crux is that the transformation over lattices from symmetric-PAKE to asymmetric-PAKE is non-trivial, because there is no concrete solution of lattice-based password-hashing available to assist the transformation. While Benhamouda and Pointcheval [17] introduced the construction of password hashing based on random oracles, and Kiefer and Manulis [18] built a password hashing scheme via Pedersen commitment [30], those schemes are vulnerable to quantum computer attacks. The main goal of our work is to answer the following question:

> Is it possible to develop a practical round-optimal asymmetric PAKE over lattices without random oracles while remaining secure under the quantum era?

We answer the above question in the affirmative. We give a detailed description of the concrete results and contributions in Section 1.1, where we also present the high-level ideas and techniques used throughout this work.

1.1 Our Results and Techniques

Before detailing our results and techniques, we first point out two important observations. On the one hand, conventional password-hashing solutions (e.g., PBKDFs) have the ability to validate the hashed password. But the issue is that these solutions make it impossible to disclose the password in plaintext to the server, but malicious participants still have the ability to launch an offline dictionary attack against passwords, as is the case of a password-verifier table [31]. Furthermore, in order to login into the server and guarantee the security of the communication, the client first authenticates himself, then the client and the server cooperate to establish a session key. Once the client has registered successfully, we assume that the client has already acquired the associated password-protected *credential* which identifies only the legitimate client for login. In this case, as for example in [19], a zero-knowledge (or non-interactive zero-knowledge, NIZK) proof is introduced by the client to convince the server that he has knowledge of the *credential* for every login attempt. However, the server is not requested to show to the client that he knows the right *credential* for the registered password.

On the other hand, to our knowledge, the common reference string (CRS) is clearly weaker than a PKI assumption, and the first practical symmetric-PAKE solution in the CRS model was designed by Katz, Ostrovsky, and Yung (or KOY) [12], which is supported by the security against chosen-ciphertext attack (CCA) [32] of the labeled Cramer-Shoup [33]. Subsequently, Gennaro and Lindell [13], [34] generalized the KOY protocol using generic building blocks, i.e., SPHF and CCA-secure encryption schemes [35], instead of specific number-theoretic assumptions. Furthermore, an important property of SPHF is that each party does not request to show the correct password that he knows to others. We also note that there are some results, e.g., [17], [18], which argue they can achieve an asymmetric-PAKE from a SPHF-based symmetric-PAKE by using a PHS. However, no concrete construction over lattices has ever been provided.

Contributions. The above two observations prompt us to design *an efficient password-authentication over lattices without random oracles, which consists of password registration, password authentication, and asymmetric*-PAKE . To bridge the gap, we summarize our key achievements as follows:

- *Password hashing over lattices in the standard model*. The security of most of the existing PHSs (e.g., PBKDFs) is demonstrated from several mathematically complex hashing algorithms under the ROM model, which represents a problem on how to design post-quantum PHSs in the standard model because, in the quantum world, we have to prove security against the quantum ROM. Inherited from the advantages of the PBKDFs, the discrete logarithm-based PHSs [17], [18] and Pedersen commitment-based PHSs [18], [36] are proposed sequentially. Further, to achieve quantum resistance, Nguyen *et al.* [37] provided a PHS and a *zero-knowledge password policy check* solution via the commitment over lattices to replace the functionality of PHS, which also assists the server in checking the policy of the hashed password. But the construction of [37] relies on the complexity of a random permutation. Thus, the original definition of PHS in [17] is adopted in our solution, but we instantiate a new commitment-based PHS by following the Pedersen-style PHS methodology of [18], [21], using the CDGLW commitment of Cabarcas *et al.* [38] as a building block in the lattice setting. The primary benefit of our instantiated PHS

via CDGLW commitment is to equip the authntication phase and the asymmetric-PAKE phase.

– *Password authentication via* PHS *over lattices.* Essentially, to bypass the random oracle and maintain the security in the coming quantum era, a quantum-safe registration-authentication protocol via PHS over lattices is proposed in order to transform the symmetric-PAKE into the asymmetric-PAKE. Here, a client registers with her corresponding username uid and hashed password $\mathsf{rpw} = \mathsf{PreHash}(\bar{s}, \mathsf{uid}\|\mathsf{pw})$ along with a pre-salt $\bar{s}$, and the server stores her received uid and a hashed value $\bar{y} \leftarrow \mathsf{PHash}((\bar{s}, s), \mathsf{rpw})$ with a random salt s. When the client attempts to authenticate himself with the server, the corresponding uid and rpw are transferred to the server who then validates whether $\mathsf{rpw} = \mathsf{rpw}'$ and $y = y'$ for the same uid. Once the validation succeeds, the server will send a *token* $\mathsf{tk}_{\mathsf{msk}} \leftarrow \mathsf{Auth}_{msk}(data)$ to the client using her master secret key msk, where Auth is either a message authentication code (MAC in short) or a one-time signature.

– *Round-optimal asymmetric-PAKE over lattices.* The SPHF scheme ensures that both parties (acting as the prover and the verifier, respectively) end up generating the same strong session key if both parties know the same authenticated factors (e.g., password or biometric template), and independent session keys otherwise. The main advantage of the SPHF for the PAKE is that the verifier does not require to do further verification, because SPHF is in essence a designated-verifier zero-knowledge. Hence, this permits for obtaining the round-optimal (i.e., two synchronous flows) PAKE. Therefore, walking along the research line of [17], the challenge to obtain the round-optimal asymmetric-PAKE protocol can be addressed by integrating the password authentication via PHS into the one-round SPHF-based symmetric-PAKE protocol in the lattice setting, which can be accomplished without introducing an extra round. Finally, following the Bellare-Pointcheval-Rogaway (BPR) game-based model enhanced by Benhamouda and Pointcheval [17], we formally prove that our asymmetric-PAKE is secure. Asymmetric-PAKE under the BPR model is achieved.

1.2 Paper Organization

In Section 2, we review related notions along with the definitions of PHS and SPHF. In Section 3, we revisit the BPR model. In Section 4, we modify the system framework so that it can integrate the commitment-based PHS into the SPHF-based symmetric-PAKE in order to obtain round-optimal asymmetric-PAKE protocol. In Section 5, we instantiate our asymmetric-PAKE over lattices by instantiating the PHS described in Section 4.1 and instantiating the SPHF construction in Section 5.2, respectively. Afterwards, in Section 6 we present the results of our experiments. Finally, in Section 7 we conclude the results.

2 Preliminaries

Notations. Lower-case bold $\mathbf{x}$ letter and upper-case bold letter $\mathbf{A}$ denote vectors and matrices, respectively. Further, $\Lambda = \{\mathbf{Bs} \,|\, \mathbf{s} \in \mathbb{Z}^n\}$ denotes an m-dimensional lattice with the basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$ for $m \geq n\lceil \log q \rceil$, and the determinant of Λ is $det(\Lambda) = \sqrt{det(\mathbf{B}^T\mathbf{B})}$. A probability proportional $\exp(-\pi \frac{(x-\mu)^2}{\alpha \cdot q})$ is used to denote a discrete Gaussian distribution with a width parameter $\alpha \cdot q$ and a centre parameter μ. Meanwhile, the standard deviation of a continuous Gaussian is $\sigma = \alpha q/\sqrt{2\pi}$, whose width parameter is given by $\alpha \cdot q$. Finally, λ denotes the security parameter.

2.1 Lattices, LWE, and LWR

Definition 2.1 ([39], [40]). *If $\chi = \chi(\lambda)$ over the integers is a distribution ensemble and it satisfies $\Pr[x \xleftarrow{R} \chi; |x| \geq B] \leq 2^{-\Omega(n)}$, then we have $|\chi| \leq B$ and B is called B-bounded.*

Definition 2.2 (Decision-LWE$_{n,q,\chi,m}$ [39], [40]). *An independent sample $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times 1}$ could be obtained in two distinct approaches. Using the first approach, $(\mathbf{A}, \mathbf{b})$ is obtained by computing $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \pmod q$, whose distribution is $\{(\mathbf{A}, \mathbf{b}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}, \mathbf{s} \leftarrow \mathbb{Z}_q^{n \times 1}, \mathbf{e} \leftarrow \chi^{m \times 1}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \pmod q\}$. The second approach is identical to the first one except that $\mathbf{b}$ is sampled from a uniform distribution, then distribution is denoted as $\{(\mathbf{A}, \mathbf{b}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}, \mathbf{b} \leftarrow \mathbb{Z}_q^{m \times 1}\}$. Thus, the two distinct distributions are computationally indistinguishable.*

Definition 2.3 (Lattice rounding, adopted from [41]). *Let q, m, and n be integers with $m \geq n$, and let $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$ be the lattice basis. Then, the rounding algorithm F is deterministic for the "rounding" function $\lfloor \cdot \rceil_{\Lambda(\mathbf{B})}^F : \mathbb{Z}_q \rightarrow \mathbb{Z}_p$ that enables to map $\mathbf{x} \rightarrow \mathbf{u}$ s.t., $F(\mathbf{x}) = \mathbf{B} \cdot \mathbf{u}$.*

Definition 2.4 (Learning with Rounding, LWR *[41]) *Regarding a λ and integers n, m, p, and q, the LWR problems state that the two distributions $(\mathbf{A}, \lfloor \mathbf{A} \cdot \mathbf{s} \rfloor_p)$ and $(\mathbf{A}, \lfloor \mathbf{u} \rfloor_p)$ are computationally indistinguishable for $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and $\mathbf{u} \leftarrow \mathbb{Z}_q^m$.*

Lemma 2.5 (Trapdoors for LWR, *adopted from [41]). *Given any integers $n \geq 1$, $q \geq 2$, and sufficiently large $m = O(n \log q)$, $\mathsf{GenTrap}(1^n, 1^m, q)$ is named as an efficient randomized algorithm that enables to generate a parity-check matrix $\mathbf{A}$ along with an associated "trapdoor" $\mathbf{T_A}$ such that the distribution of $\mathbf{A}$ is $negl(\lambda)$-close to uniform. Additionally, another efficient algorithm $\mathsf{LWRInvert}(\mathbf{T_A}, \mathbf{A}, \mathbf{c})$ that enables to generate $\mathbf{s}$ and $\mathbf{e}$ with overwhelming probability over all random choices, for $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \pmod{2q}$, where $\mathbf{s} \in \mathbb{Z}_{2q}^n$ is arbitrary and either $\|\mathbf{e}\| < q/O(\sqrt{n \log q})$ or $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \alpha q, 0}$ for $1/\alpha \geq \sqrt{n \log q} \cdot w(\sqrt{\log n})$.*

Modulus Switching. As discussed in [41], for any positive integers p, q, the modulus switching function $\lfloor \cdot \rceil\ q \rightarrow p$ is denoted as: $\lfloor x \rceil_{p \rightarrow q} = \lfloor (p/q) \cdot x \rceil \pmod q$. It is easy to show that for any $x \in \mathbb{Z}_q$ and $p < q \in \mathbb{N}$, $x' = \lfloor \lfloor x \rceil_{q \rightarrow p} \rceil_{p \rightarrow q}$ is an element near to x, i.e., $|x' - x \pmod q| \leq \lfloor q/(2p) \rceil$. When $\lfloor \cdot \rceil\ q \rightarrow p$ is used to an element $x \in \mathbb{Z}_q$ or a vector $\mathbf{x} \in \mathbb{Z}_q^k$, the procedure is applied to each coefficient individually.

2.2 Password Hashing Scheme

Below we consider a password hashing scheme for a list of Q pairs of matching client-server,

$$\mathsf{PHS} = (\mathsf{Setup}, \mathsf{PSalt}, \mathsf{PreHash}, \mathsf{PHash}, \mathsf{PreSalt}).$$

Benhamouda and Pointcheval [17] proposed the first construction of PHS in 2013. The PHS allows to have a verification algorithm that is based on salting techniques over the hash of the client password. Next, the PHS turned out to be an important building block, together with a SPHF scheme, to construct the asymmetric-PAKE in the BPR framework, but their symmetric-PAKE cannot prevent quantum computer attacks. In order to achieve quantum-safe asymmetric-PAKE, we modify the framework of Kiefer and Manulis [36] by re-defining the PHS as follows.

Definition 2.6 (Adopted from [17]). *A* PHS *must contain the following six polynomial algorithms:*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda, \ell)$ *is a probabilistic algorithm that generates the set of parameters* param*, containing the description of random salt spaces* $\mathbb{S}_P$ *and* $\mathbb{S}_H$ *by taking as input the security parameter* λ *and the bit-length* $\ell \leq \lambda$ *of the password. Below,* param *is the default input for the following algorithms, we omit it in order to have a cleaner notation.*
- $s \leftarrow \mathsf{PSalt}(1^\lambda)$ *is a probabilistic algorithm that generates a salt* $s \in \mathbb{S}_H$ *by taking as input* 1^λ*.*
- $\bar{s} \leftarrow \mathsf{PreSalt}(1^\lambda)$ *is a probabilistic algorithm that generates a* pre-hash *salt* $\bar{s}$ *in set* $\mathbb{S}_P$ *by taking as input* 1^λ*.*
- $\bar{y} \leftarrow \mathsf{PreHash}(\bar{s}, \mathsf{pw})$ *is a deterministic algorithm, and outputs a* pre-hash *value* $\bar{y}$ *by taking as input a pre-salt* $\bar{s}$*, and a password* $\mathsf{pw} \in \{0,1\}^\ell$*.*
- $y \leftarrow \mathsf{PHash}(s, \bar{s}, \bar{y})$ *is a deterministic algorithm. It outputs a hash value* y *by taking as input two salts* s *and* $\bar{s}$*, and a pre-hash value* $\bar{y}$ *instead of a password* pw*.*

Meanwhile, the PHS satisfies the following security properties, including *password hiding*, *salt indistinguishability*, *pre-image resistance*, *second pre-image resistance*, *pre-hash entropy preservation* and *entropy preservation*. Notably, [18] introduced a new property called *password-hiding* to ensure that the hashed password should not leak any knowledge about the real password, and we keep this property in our new PHS.

- *Password hiding*. Informally, no adversary enables to tell the hashed real password from a random string. Formally, for the public parameter $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, and all PPT algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, if $\mathcal{A}_1(\mathsf{param})$ outputs two equal-length passwords pw_0 and pw_1, then $\mathcal{A}_2(y)$ outputs a bit b' by taking as input a $y \leftarrow \mathsf{PHash}(\mathsf{param}_{\mathsf{ph}}, \bar{y}, \bar{s}, s)$, where $\bar{s} \leftarrow \mathsf{PreSalt}(\mathsf{param})$, $s \leftarrow \mathsf{PSalt}(\mathsf{param})$, and $\bar{y} \leftarrow \mathsf{PreHash}(\bar{s}, \mathsf{pw}_b)$ for a random bit $b \in \{0,1\}$, then we have $|\Pr[b = b'] - \frac{1}{2}| \leq negl(\lambda)$, for a negligible function $negl(\cdot)$ with the λ.
- *Salt indistinguishability*.For any choice of parameters such that $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, where $\bar{s} \leftarrow \mathsf{PreSalt}(\mathsf{param})$ and $s \leftarrow \mathsf{PSalt}(\mathsf{param})$, we have that $\bar{s}$ and s are indistinguishable.
- *One-wayness* (or pre-image resistance) is the property to show it is hard for a hash function to invert. Thus, for all PPT algorithms $\mathcal{A}$ running in time at most t, we have

$$\Pr[(i, \bar{y}) \leftarrow \mathcal{A}^{\mathsf{PHash}_{\bar{y}}(\cdot), \mathsf{Verify}(\cdot)}(\mathsf{param}); \mathsf{Verify}(i, \bar{y}) = 1] \leq \frac{\alpha t}{2^\theta t_{\mathsf{PreHash}}} + negl(\lambda),$$

for small $\alpha \in \{0,1\}$, t_{PreHash} being the running time of PreHash, and θ denotes the min-entropy of the password distribution $\mathcal{D}$, with $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$.

- *Second pre-image resistance* implies it is computationally infeasible to find any alternative input that enables to generate the same output as that of a specified input. Thus, for any password $\mathsf{pw} \in \{0,1\}^\ell$ and any unbounded $\mathcal{A}$

$$\mathrm{Adv}(\mathcal{A}) := \Pr\left[\begin{array}{c} \mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda, \ell); \\ \bar{s} \leftarrow \mathsf{PreSalt}(\mathsf{param}); \\ s \leftarrow \mathsf{PSalt}(\mathsf{param}); \\ \bar{y} \leftarrow \mathsf{PreHash}(\mathsf{pw}, \bar{s}); \\ \bar{y}' \leftarrow \mathcal{A}(\mathsf{param}, \bar{y}, s) : \\ \bar{y} \neq \bar{y}' \bigwedge \\ y \leftarrow \mathsf{PHash}(s, \mathsf{pw}) = \bar{y}' \end{array}\right] \leq negl(\lambda).$$

  there exists a negligible function $negl(\cdot)$ in λ.
- *Pre-hash entropy preservation.* Regarding any PPT adversary $\mathcal{A}$ who could guess the outputs of the pre-hash (and hash) value is less than $2^{-\theta}$. Thus, there exists a $negl(\cdot)$ in λ such that

$$\mathrm{Adv}(\mathcal{A}) := \Pr\left[\begin{array}{c} \mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda, \ell); \\ (\bar{s}, \bar{y}) \leftarrow \mathcal{A}(\mathsf{param}); \\ \mathsf{pw} \leftarrow \mathcal{D} : \\ s \in \mathbb{S}_H \bigwedge \\ \bar{y} = \mathsf{PreHash}(\bar{s}, \mathsf{pw}) \end{array}\right] \leq 2^{-\theta} + negl(\lambda).$$

- *Entropy preservation.* There exists a $negl(\lambda)$ in λ such that for any unbounded adversary $\mathcal{A}$

$$\mathrm{Adv}(\mathcal{A}) := \Pr\left[\begin{array}{c} \mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda, \ell); \\ (s, y) \leftarrow \mathcal{A}(\mathsf{param}); \\ \mathsf{pw} \leftarrow \mathcal{D} : \\ (\bar{s} \in \mathbb{S}_H) \bigwedge s \in \mathbb{S}_H \\ (\bigwedge y = \mathsf{PHash}(s, \mathsf{pw})) \end{array}\right] = 2^{-\theta} + negl(\lambda).$$

2.3 Smooth Projective Hash Functions

The methodology of SPHF families [35] is computationally distinguishable regarding a random element in an underlying NP language $\mathcal{L} \subseteq \mathcal{X}$ and a random element in a domain $\mathcal{X} \setminus \mathcal{L}$.

Approximate Word-Independent SPHF. It means the projection key ph is independent of the word (or wd) and the property of *adaptive smoothness* holds even if wd is selected adaptively after seeing ph. The property of approximate word-independent SPHF is used to achieve a one-round PAKE. More precisely, an SPHF over $\mathcal{L}(\subseteq \mathcal{X})$ is captured by four probabilistic polynomial time (PPT) algorithms.

- Hashing key generation $hk \leftarrow \mathsf{HashKG}(\mathcal{L})$. It outputs a "private" hashing key hk by taking as input $\mathcal{L}$.
- Projection key generation $ph \leftarrow \mathsf{ProjKG}(hk, \mathcal{L}, \mathsf{wd})$. It outputs a "public" ph by taking as input a hk, and a $\mathsf{wd} \in \mathcal{L}$.
- Hashing algorithm $h \leftarrow \mathsf{Hash}(hk, \mathcal{L}, \mathsf{wd})$. It outputs a hash value $h \in \{0,1\}^v$ for a positive integer $v = \Omega(\lambda)$ by taking as input hk and $\mathsf{wd} \in \mathcal{L}$.

– Projection hashing algorithm $p \leftarrow \mathsf{Proj}(ph, \mathcal{L}, \mathsf{wits})$. It outputs a projective hashing value $p \in \{0,1\}^v$ by taking as input $\mathcal{L}$, ph, and a witness wits.

Importantly, the SPHFs should satisfy the properties of (approximate) correctness and smoothness:

– *Approximate correctness*. The main feature of SPHF is that the two outputs of $\mathsf{Hash}(hk, \mathsf{wd})$ and $\mathsf{Proj}(ph, \mathsf{wd}, w)$ are (statistically) indistinguishable for a $\mathsf{wd} \in \mathcal{L}$ with an associated wits. Formally, the approximate correctness (i.e., ε-correct) property holds if we have the negligible probability $\Pr[\mathsf{HD}(\mathsf{Hash}(hk, \mathcal{L}, \mathsf{wd}), \mathsf{Proj}(hk, \mathcal{L}; \mathsf{wits})) > \varepsilon \cdot v] = negl(\lambda)$.
– *Smoothness*. If the distribution $(ph, h \leftarrow \mathsf{Hash}(hk, \mathcal{L}, \mathsf{wd}))$ and the distribution $(ph, h \leftarrow \{0,1\}^v)$ have a negligible statistical distance in λ for $hk \leftarrow \mathsf{HashKG}(\mathcal{L})$ and $ph \leftarrow \mathsf{ProjKG}(hk, \mathcal{L}, \mathsf{wd})$, then the smoothness property holds. We remark that *smoothness* implies statistically indistinguishability.

Here, we must stress that, to be more precise, SPHF implies that the approximate SPHF is 0-correct. However, a 0-correct scheme can not be achieved in lattice-based setting.

Approximate Universal b-PHF. Approximate word-independent universal bit-projective hash function (in short b-PHF) is regarded as a warm-up, which can be used to build an approximate word-independent SPHF. In a nutshell, b-PHF only supports that the targeted hashing value is a bit (i.e., $v = 1$). Meanwhile, the properties of approximate correctness and smoothness are modified as follows.

– *Approximate correctness*. b-PHF is ε-correct, if there exists

$$\Pr[\mathsf{Hash}(hk, \mathcal{L}, \mathsf{wd}) = \mathsf{Proj}(ph, \mathcal{L}, w)] \geq 1 - \varepsilon,$$

for any word $\mathsf{wd} \in \mathcal{L}$, where the probability is taken over the choice of $hk \leftarrow \mathsf{HashKG}(\mathcal{L})$.
– *Universality*. b-PHF is ε-universal if for any $\mathsf{wd} \in \mathcal{X} \setminus \mathcal{L}$ and any ph, $|2 \cdot \Pr[\mathsf{Hash}(hk, \mathcal{L}, \mathsf{wd}) = 1 | ph = \mathsf{ProjKG}(hk, \mathcal{L}, \mathsf{wd})] - 1| \leq \varepsilon$, where the probability is taken over the random coins of $\mathsf{Hash}(\cdot)$ and the choice of $hk \leftarrow \mathsf{HashKG}(1^\lambda)$. The b-PHF is regarded as statistically universal if it is $negl(\lambda)$-universal. If an approximate b-PHF is $negl(\lambda)$-correct then we name it as b-PHF.

Transformation From b-PHF *to* SPHF. The work of Katz and Vaikuntanathan [25] implicitly pointed out the generic transformation from a ε-correct $negl(\lambda)$-universal b-PHF to an approximate $(\varepsilon + \varepsilon')$-correct SPHF (for any positive constant ε') and then into an SPHF. However, the resulting (approximate) SPHF is not word-independent. To obtain PAKE over LWE with polynomial modulus, the restriction of word-independent is overcomed by Benhamouda *et al.* [27], who gave a transformation from an ε-universal word-independent b-PHF to a word-independent SPHF, by amplifying the two important properties of smoothness or universality (and assuming $1 - \varepsilon \geq 1/poly(\lambda)$).

3 Bellare-Pointcheval-Rogaway Model

Below, we review the security model of Bellare-Pointcheval-Rogaway that will be used in our security analysis.

Users and Passwords. Clients are denoted by $C \in \mathcal{C}$ and servers are denoted by $S \in \mathcal{S}$. Each client C holds a password pw_C, while each server S holds a random salt $\mathsf{PHS.PSalt}(\mathsf{param}) \rightarrow s_S$ and a hash value $y_S = \mathsf{PHS.PHash}(s_S, \mathsf{pw}_S)$ of some password pw_S, received from the client. Without loss of generality, we suppose that each client C is paired with a server S, and that for each matching pair (C, S), $\mathsf{pw}_C = \mathsf{pw}_S$. For the sake of simplicity, each client or server appears exactly in one pair. We point out that this is not a restriction of the model, and it is possible to execute between a client and a non-matching server.

The BPR model is abbreviated from Bellare, Pointcheval, and Rogaway [42], which is adopted in our work following the prior works [43], [44], [45]. In this model, each participant enables to perform the session key generation protocol with different partners multiple times (possibly concurrently). In addition, the BPR model permits the participants to instantiate unlimited instances but each one is only used once. The adversary is enabled to oracle access different instantiated instances of the participants; furthermore, each instantiated instance requires to maintain (local) state that is updated in the process of the experiment.

Partnering. Partnering implies that two instances are partnered if transcripts of them are matched. Let $U_C, U_S \in U$, then instances $\Pi^i_{U_C}$ and $\Pi^j_{U_S}$ are partnered if: (1) $\mathsf{sid}^i_U = \mathsf{sid}^i_{U_C} \neq \mathrm{NULL}$; and (2) $\mathsf{pid}^i_{U_C} = U_C$ and $\mathsf{pid}^j_{U_S} = U_S$.

Adversarial Model. For any protocol execution, an initialization phase is necessary to establish public parameters. The protocol participant set is $\mathsf{U} = \mathcal{C} \cup \mathcal{S}$, then the adversary enables to fully control the external network, which implies that he (1) is free to manipulate messages (e.g., block, inject, modify, and delete *etc*); (2) adaptively requests any session keys. Finally, we model how the adversary $\mathcal{A}$ can create several concurrent instances $U \in \mathsf{U}$ and has the ability to interact with instantiated instances via the defined oracle queries.

– $\mathsf{Execute}(U_C, i, U_S, j)$ captures a passive attack, where the attacker eavesdrops on honest executions between the instance $\Pi^i_{U_C}$ and the instance $\Pi^j_{U_S}$. Finally, it outputs the protocol transcript (i.e., the complete ordered messages) that enables to exchange between instances.
– $\mathsf{Send}(U^i_C, U^j_S, M)$ captures an active attack, where the adversary could arbitrarily operate (e.g., intercept, modify, create, replay or forward) a message to $\Pi^j_{U_S}$ (e.g., U^j_S) in the name of $\Pi^i_{U_C}$ (e.g., U^i_C). To initiate a session between the both parties (i.e., the client and the server), a Start command is sent in the name of a $\Pi^j_{U_S}$ to a $\Pi^i_{U_C}$. The output of this oracle is the transcript message generated by U^j_S after receiving M from the adversary.
– $\mathsf{Corrupt}(S)$ captures the case of the server corruption, and the output is the salt s_S and the hash value y_S.
– $\mathsf{Test}(U_C, i)$ is allowed to query many times by the adversary and outputs a random bit b in the real-or-random (i.e., ROR) flavor.[1] Due to the clients and the

1. Notably, in the Find-then-Guess setting, $\mathsf{Test}(U_C, i)$ oracle is only allowed to query once, and $\mathsf{Reveal}()$ oracle is not useful anymore [21].

servers are paired in the client-server setting, the oracle $\mathsf{Test}(U_C, i)$ then answers as follows.

- The output is $\perp$, if either on session key has been computed by the server instance of U_S^i or the password hash of a partnered server instance U_S^i has been corrupted.
- The output is the same as for the previous query, if the server instance U_S^i queried a $\mathsf{Test}(U_C, i)$ oracle.
- The output is the same as for the previous partner's query, if the server instance's partner U_C^i (if U_S^i has any) queried a $\mathsf{Test}(U_C, i)$ oracle and they are matching client-server. On the contrary, the output is a random session key, if U_C^i and U_S^i are not matching.

If $b = 1$, then it implies that the adversary obtains a session key skey_U^i. Otherwise, the adversary obtains a random session key. Finally, the adversary attempts to guess a random bit b'. If $b = b'$ then the adversary wins the game with the advantage $\mathrm{Adv}_{\mathcal{A},\Pi}(\lambda) = 2\Pr[b = b'] - 1$.

Remark 3.1. Benhamouda and Pointcheval [17] ignored the case of client corruption queries and did not deal with *forward-secrecy*. Notably, Hwang *et al.* [46], Jutla and Roy [20], Jarecki *et al.* [22], Bradley *et al.* [47] deal with forward secrecy by defining the client corruption oracle $\mathsf{Corrupt}(C)$. The client corruption query allows to show that all the previous keys remain secure even in case of the password-corruption, and the server corruption query allows to express the quality of the password hashing scheme.

- $\mathsf{Corrupt}(C)$. This oracle is used to model the client corruption, and it outputs the corrupted password pw_C. The aim of *forward secrecy* is that even knowing a password should not help to distinguish previous session keys from random keys. As a consequence, for corrupted clients and their matching partners, Test-queries are answered with the actual session keys, but only after the client corruption, since the keys established before should remain indistinguishable from random elements.

Correctness. If the instance $\Pi_{U_C}^i$ and $\Pi_{U_S}^j$ are partnered then there exist $\mathsf{acc}_{U_C}^i = \mathsf{acc}_{U_S}^j = \mathrm{TRUE}$ and $\mathsf{skey}_{U_C}^i = \mathsf{skey}_{U_S}^j$, and two instances both obtain the session key.

Security. If the asymmetric-PAKE is regarded as secure, then the advantage of any $\mathcal{A}$ in time t is upper-bounded by: (1) if $\mathcal{A}$ does not query the $\mathsf{Test}(U_S, i)$ oracle for the server by using a corrupted password hash, then upper-bounded is $Q(\lambda)/|N| + negl(\lambda)$, where $Q(\lambda)/|N|$ bounds when the adversary launches an *on-line dictionary attack* with one password in per session. (2) Otherwise, the upper-bound is $Q(\lambda)/|N| + \mathrm{Adv}_{\mathcal{B},\mathsf{owf}}(\lambda) + negl(\lambda)$ for the adversary $\mathcal{B}$ in time at most t. The advantage $\mathrm{Adv}_{\mathcal{B},\mathsf{owf}}(\lambda)$ is used to bound the probability when the adversary either launches a *brute-force attack* to compromise a server or to find the hashed password from the server.

Definition 3.2. *For all* PPT *adversaries $\mathcal{A}$ launching at most $Q(\lambda)$ online attacks, if it still holds the advantage $\mathrm{Adv}_{\mathcal{A},\Pi}(\lambda) \leq Q(\lambda)/|N| + \mathrm{Adv}_{\mathcal{B},}\mathsf{PH}(\lambda) + negl(\lambda)$, then the asymmetric-*PAKE *protocol Π is a secure protocol.*

Remark 3.3 ([28], [48]). The advantage of $\mathcal{A}$ is denoted as $Q(\lambda)/|N| + negl(\lambda)$ for all dictionary sizes $|N|$ in the existing uniform-model. Very recently, Li and Wang [28], [48] provided a tight analysis to bound the adversary's advantage as $C' \cdot Q^{s'}(\lambda) + negl(\lambda)$, they consider the password distribution matches the Zipf-distribution, since Wang *et al.* [49], [50] pointed out that the advantages of the adversary are underestimated in the uniform-model. In this paper, to illustrate the features (i.e., pre-image resistance) of the proposed PHS, we follow the min-entropy-based formulation (i.e. $Q(\lambda) \cdot 1/2^{\beta} + \mathrm{Adv}_{\mathcal{B},\mathsf{owf}}(\lambda) + negl(\lambda)$) proposed by [17] to capture the advantage of the adversary, where β denotes the min-entropy of the password distribution. This min-entropy-based approach is stronger than the uniform-model-based formulation $(Q(\lambda) \cdot |D| + \mathrm{Adv}_{\mathcal{B},\mathsf{owf}}(\lambda) + negl(\lambda))$ that bounds the advantage using the dictionary size $|D|$, but this approach is weaker than the Zipf-model-based formulation (i.e., $C' \cdot Q(\lambda)^{s'} + \mathrm{Adv}_{\mathcal{B},\mathsf{owf}}(\lambda) + negl(\lambda)$) for the Zipf parameters C' and s'. It is our future work to figure out a more accurate formulation to capture the advantage. However, we still recommend to use the accurate Zipf-model-based formulation [28], [48], [50] for the asymmetric-PAKE without using PHS.

4 Quantum-Safe Password Authentication With Password-Protected Session Key Establishment Via PHS in the Standard Model

4.1 Password Hashing Scheme

Password-hashing schemes are used to hash the password, such as PBKDF2, Bcrypt and Scrypt [1]. Indeed, the functionalities of the lattice-based PHS are similar to the traditional schemes, but the major benefit of lattice-based PHS is that it can play the role of intermediate component to transform the SPHF-based symmetric-PAKE into the asymmetric-PAKE using specific number-theoretic assumptions in the standard model instead of using several mathematically complex hashing algorithms under the ROM, while remaining secure against quantum attackers. Below we detail our instantiation of the PHS over lattices.

CDGLW Commitment Scheme. There are other additive homomorphic commitments in the literature, here we consider the CDGLW solution proposed by Cabarcas *et al.* [38], which is a good alternative since it is conceptually simple and has the additive homomorphism. We then review the construction of the unconditionally hiding and long-term binding properties.

- $\mathsf{param} \leftarrow \mathsf{ComGen}(1^{\lambda}, k)$. Samples two matrices $\mathbf{A}_1 \leftarrow \mathbb{Z}_q^{m \times n}$ and $\mathbf{A}_2 \leftarrow \mathbb{Z}_q^{m \times k}$ for parameters n, m, $q \in \mathbb{Z}$ and B, $\sigma \in \mathbb{R}^+$ *s.t.*, $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2) \in \mathbb{Z}_q^{m \times (n+k)}$ has trivial kernel. Outputs $\mathsf{param} := (n,\ m,\ q,\ \mathbf{A})$.
- $(\mathbf{c}, \mathbf{s}) \leftarrow \mathsf{Com}(\mathsf{param}, \mathbf{m} \in \mathbb{Z}_q^n; \mathbf{s})$. Samples $\mathbf{s} \leftarrow \mathbb{Z}_q^k$ and $\mathbf{e} \leftarrow \mathcal{D}_{\sigma}^m$, calculates $\mathsf{com} := \mathsf{Com}(\mathsf{param}, \mathbf{m}; \mathbf{e}) = \mathbf{A}_1 \cdot \mathbf{m} + \mathbf{A}_2 \cdot \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$, and outputs $(\mathbf{c}, \mathbf{s})$.
- $\mathsf{Open}(\mathbf{c}, \mathbf{s})$. Returns $\mathbf{m}$ if $\|\mathbf{c} - \mathbf{A}_1\mathbf{m} - \mathbf{A}_2\mathbf{s}\| \leq B$, and $\perp$ if not, where B is denoted as the bound of the noise.

PHS Instantiation via CDGLW Commitment. Below, armed with the CDGLW commitment scheme, we develop a new lattice-based password hashing scheme.

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$ obtains $\mathbf{A}_1$ and $\mathbf{A}_2$ for intergers n, m, and q by executing $(\mathbf{A}_1, \mathbf{T}_{\mathbf{A}_1}) \leftarrow \mathsf{GenTrap}(1^n, 1^m, q)$ and $(\mathbf{A}_2, \mathbf{T}_{\mathbf{A}_2}) \leftarrow \mathsf{GenTrap}(1^n, 1^m, q)$, respectively. Outputs $\mathsf{param} := (n, m, q, \mathbf{A}_1, \mathbf{A}_2)$. It is important to remark that although next algorithms use this parameters, we are not showing them explicitly as input arguments in order to simplify notation.
- $\bar{s} \leftarrow \mathsf{PreSalt}(1^\lambda)$ picks a random vector $\mathbf{s}_p \leftarrow \mathbb{Z}_q^k$ and sets the pre-salt $\bar{s} := \mathbf{s}_p$.
- $s \leftarrow \mathsf{PSalt}(1^\lambda)$ picks a random matrix $\mathbf{U} \in \mathbb{Z}_q^{k \times k}$, then outputs the salt $s := \mathbf{s} = \mathbf{U} \cdot \mathbf{s}_p \leftarrow \mathcal{D}_\sigma^k$.
- $\bar{y} \leftarrow \mathsf{PreHash}(\bar{s}, \mathsf{pw})$. To obtain the *pre-hash value*, the algorithm takes as input the $\mathsf{pw} \in \mathbb{Z}_q^n$, and the *pre-salt* $\bar{s} := \mathbf{s}_p \in \mathbb{Z}_q^k$, then computes and outputs

$$\bar{y} := \mathbf{p} = \left\lfloor (\mathbf{A}_1, \mathbf{A}_2) \cdot \begin{bmatrix} \mathsf{pw} \\ \bar{s} \end{bmatrix} \right\rceil_p \in \mathbb{Z}_p^m.$$

- $y \leftarrow \mathsf{PHash}(\bar{s}, s, \bar{y})$. To obtain the *hash value*, the algorithm takes as input the salt $s := \mathbf{s}$, the pre-hash $\bar{y} := \mathbf{p}$, and the public matrix $\mathbf{A}_3 \in \mathbb{Z}_q^{m \times k}$ by executing $\mathsf{GenTrap}(1^n, 1^m, q)$ again, then it outputs

$$\begin{aligned} y \;:=\; \mathbf{h} &= \mathbf{p} + \lfloor \mathbf{A}_3 \cdot \mathbf{s} \rceil_p \\ &= \left\lfloor (\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3) \cdot \begin{bmatrix} \mathsf{pw} \\ \bar{s} \\ s \end{bmatrix} \right\rceil_p \in \mathbb{Z}_p^m, \end{aligned}$$

where we have the public matrix $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3) \in \mathbb{Z}_q^{m \times (n+2k)}$, $\mathbf{b} := (\mathsf{pw}, \mathbf{s}_p, \mathbf{s})^T \in \mathbb{Z}_q^{n+2k}$ for $\bar{s} := \mathbf{s}_p \leftarrow \mathbb{Z}_q^k$.

Security Analysis. Our proposed PHS inherits the security properties of the commitment scheme and satisfies the following requirements: password hiding, pre-image resistance, second pre-image resistance, pre-hash entropy preservation, and entropy preservation.

- *1). Password hiding* is guaranteed by the *statistical hiding property* of the commitment scheme. Strictly, two distinct passwords pw_0 and pw_1 are assumed to map two corresponding integers π_0 and π_1 (e.g., in $\mathbb{Z}_{95^n}$). Thus, if the password hiding property holds, then the ability of $\mathcal{A}$ to tell the difference between π_0 and π_1 is negligible. In fact, the hash value $y = \mathbf{A} \cdot \mathbf{b} + \mathbf{e} \pmod q$ received by $\mathcal{A}$ that corresponds to a commitment on π. Therefore, breaking the hiding property of PHS can thus be turned into an attack on the hiding of the subjacent commitment.
- *2). Second pre-image resistance* is guaranteed by the property of *computational binding property* for commitment scheme. In particular, if two distinct pre-hash values $\bar{y}_0 \leftarrow \mathsf{PreHash}(\bar{s}, \mathsf{pw}_0)$ and $\bar{y}_1 \leftarrow \mathsf{PreHash}(\bar{s}, \mathsf{pw}_1)$ (i.e., $\bar{y}_0 \neq \bar{y}_1$) could yield the same value $y \leftarrow \mathsf{PHash}(\bar{s}, s, \bar{y}_i)$ for $i = 0$ or 1 and any two distinct pw_0, pw_1, any pre-salt $\bar{s}$, and salt s, then one can use these values (i.e., $\bar{y}$, $\bar{y}'$, and y) to break the property of *computational binding* because $\mathsf{PHash}(\bar{s}, s, \mathsf{pw}_0) = \mathsf{PHash}(\bar{s}, s, \mathsf{pw}_1)$. Thus, this implies that the short integer solution (or SIS) assumption over lattices enables to guarantee the property of the second pre-image resistance.
- *3). Pre-image resistance*. In order to find the pre-image $\bar{y} := \mathbf{Ab} = \mathbf{h}$, (i.e., one accidentally finds a solution to the SIS problem with the associated $\mathbf{A}$), the adversary $\mathcal{A}$ must perform 2^θ invocations of $\mathsf{PreHash}$ by calculating $\mathbf{A} \cdot \mathbf{b}$ for each candidate pw^*. Thus, the pre-image resistance holds since the pre-salt $\bar{s}$ and the salt s are randomly chosen, and the probability of a collision existing on every invocation of $\mathsf{PHash}(\cdot)$ is negligible. The hash value $y := \mathbf{h}$ generated by $\mathsf{PHash}(\cdot)$ is a perfect hiding commitment proved by the hiding property of Cabarcas *et al.* [38] commitment scheme. Further, for any pre-hash value $\bar{y} := \mathbf{Ab}$ generated by $\mathsf{PreHash}(\cdot)$ (taking as input the randomness of matrix $\mathbf{A}$, a pw and a $\bar{s} := \mathbf{s}$), it should satisfy the second pre-image resistance proved by the *computational binding property*.
- *4). Pre-hash entropy and hash entropy preservation*. Below, we consider every pre-hash and pre-salt pair $(\bar{y}, \bar{s})$ sampled by the *pre-hash* entropy adversary, then $\Pr[\bar{y} = \mathbf{Ab}] \leq 2^{-\theta} + negl(\lambda)$, if the pre-salt $\bar{s}$ is hidden, then the min-entropy of $\bar{y}$ generated via the pre-salt $\bar{s}$ is larger than the min-entropy of pw due to its randomness. Further, for every hash and salt pair (y, s) sampled by the *hash* entropy adversary, $\Pr[y = \mathbf{Ab}] \leq 2^{-\theta} + negl(\lambda)$.

4.2 Password Authentication Via Password Hashing

Generic Password Authentication. Below we present in detail each phase of log-on and log-in as discussed in [51], [52]. Before giving our main quantum-resistant password authentication protocol, we first describe the classical password authentication flow as follows:

- In the registration (i.e., log-on or sign-up) phase, a client is requested to register with its unique username/password (i.e., uid/pw) on the identity server, then the identity server stores the corresponding uid and $h = \mathsf{H}(\mathsf{pw})$ at her local database.
- In the authentication (i.e., log-in or sign-on) phase, the client uses its uid with an associated h' to authenticate the server to get the corresponding service. Upon receiving the uid and h', the server then validates if $h' = h$ for the corresponding uid.
- If (uid, h) matches (uid, h'), i.e., $h' = h$, then the server computes a token $\mathsf{tk}_{\mathsf{msk}} \leftarrow \mathsf{Auth}_{msk}(x)$ under a master secret key msk to, and responds the generated $\mathsf{tk}_{\mathsf{msk}}$ to the client, where the message x contains client's information/attributes, and the additional information (e.g., expiration time *etc.*) that is included in the $\mathsf{tk}_{\mathsf{msk}}$.

This approach cannot prevent an identity provider breaching, and a *single point of failure* will happen with a high probability. Thus, the attacker enables to (i) recover msk and forge arbitrary tokens that enable obtain services, and (ii) obtain hashed passwords to recover client credentials by launching an offline dictionary attack. Furthermore, this approach still cannot prevent quantum attacks.

Quantun-Safe Password Authentication Over Lattices. The essential PHS-based authentication idea of Benhamouda and Pointcheval [17] is that, instead of storing the password in plaintext, the server stores the password in the form of ciphertext. To avoid deriving the hashed password from a key derivation function (KDF), the server derives the hashed password

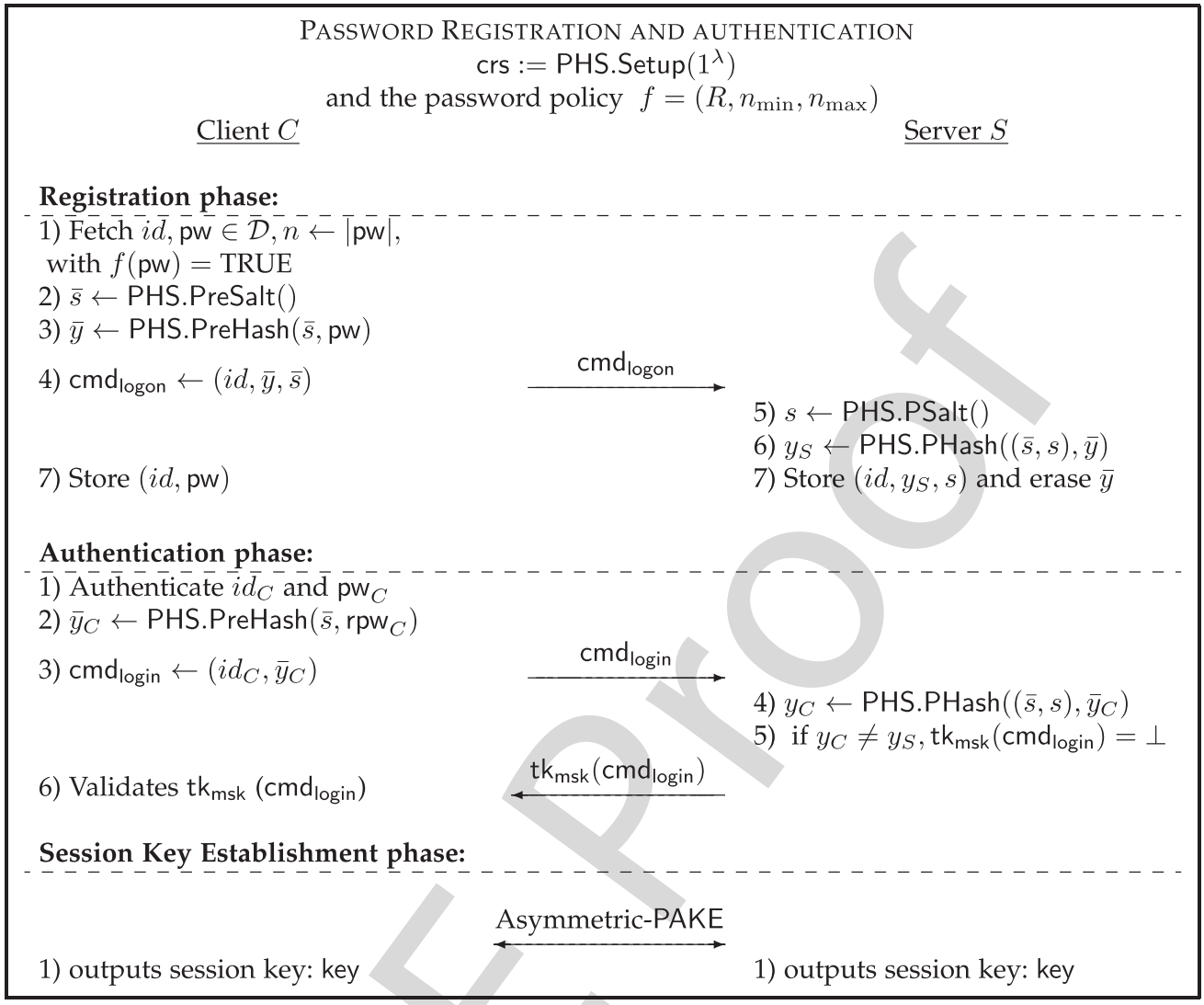

Fig. 2. Password registration and authentication via hashing password. For better illustration, the detailed asymmetric-PAKE construction is omitted here, and we will present the detailed techniques in Section 5.

by utilizing a hard-to-invert password hashing PHash taking as input the password along with a random "salt" (PSalt$(\cdot)$). While a session is executing, the client attempts to send encryptions of the pre-hash value of the password evaluated by PreHash$(\cdot)$ on the password. Correspondingly, the server responses encryptions of the stored hashed password PHash$(\cdot)$. However, the registration and authentication phase are omitted by Benhamouda and Pointcheval [17] while their approach cannot guarantee security in the coming of quantum era. In order to make the asymmetric-PAKE framework more complete, in this paper, we inherit the spirit of Benhamouda and Pointcheval [17] and sketch the phases of registration and authentication while achieving quantum security. To illustrate our construction, the asymmetric-PAKE framework is divided into three phases in the fine-grained form, consisting (1) log-on; (2) log-in; and (3) session key establishment (i.e. asymmetric-PAKE) phases, as depicted in Fig. 2.

Particularly, the first two phases, registration and login can combine them into the *registration-login* protocol. The detailed description of the authentication process is shown in the second part of Fig. 2. Below, we describe the *registration-login* protocol based on PHS.

Registration (or log-on) Phase. Before executing the registration phase, public parameters $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ are shared to the participants as common reference string.

1) *The server (i.e. identity service) assigns the password policy*. The server S first formulates and assigns the password policy f to the client C at the offline stage.
2) *The client types in her username with an associated password*. The client C initiates the *registration* execution by typing in its associated uid and picking its corresponding pw, where uid can be for instance an email address, and this uid will be used later to log-in at the server side. Then the created log-on command $\mathsf{cmd}_{\mathsf{logon}} := (\mathsf{uid}, \mathsf{rpw} \leftarrow \mathsf{PHS.PreHash}(\mathsf{uid}||\mathsf{pw}))$ is sent to the server.
3) *The server generates and returns the token*. Upon receiving the log-on command $\mathsf{cmd}_{\mathsf{logon}}$, the server computes the *password hash* $y_S \leftarrow \mathsf{PHash}(s, \mathsf{rpw})$ by taking as input a salt $s \leftarrow \mathsf{PSalt}()$, a pre-salt $\bar{s} \leftarrow \mathsf{PreSalt}()$, and received rpw. Subsequently, the server stores (uid, y_S, s) and erases rpw while it returns the pre-salt $\bar{s}$ as her login credential.

Finally, if and only if $f(\mathsf{pw}) = \mathrm{TRUE}$ the server accepts a hash value y for client's choice pw.

Authentication (or log-in) Phase. Secure login occurs when the client authenticates him to communicate with the server without the risk of impersonation. Once authenticated, the two participants then establish a session key at the authenticated key exchange phase. In summary, we informally present the authentication phase as follows, and the detailed description depicted in Fig. 2.

1) When a client C expects to authenticate to a server S using a pw_C, it calculates the pre-hash value $\bar{y}_C = \mathsf{PreHash}(\bar{s}_C, \mathsf{pw}_C)$ corresponding to her password. Then, the client creates a lon-on command $\mathsf{cmd}_{\mathsf{login}} := (id, \bar{y}_C))$ and sends $\mathsf{cmd}_{\mathsf{logon}}$ to the server.
2) Upon receiving the command $\mathsf{cmd}_{\mathsf{login}}$, the server splits the latest received message and obtains $\bar{y}_C$, then the server fetches the hash value y_C from her password database and calculates $y'_C := \mathsf{PHash}((\bar{s}_C, s_S), \bar{y}_S) = y$ to decide whether the received password is legitimate or not. If the received password is legitimate we have that the user is eligible to login.
3) The server generates the authentication token $\mathsf{tk}_{\mathsf{msk}}(\mathsf{cmd}_{\mathsf{login}})$ using her local master secret key msk, and the server sends back to the client as its *login credentials*, where Auth is either a MAC or a hash-based signature.
4) The authenticated client and the legal server will establish a session key using the asymmetric-PAKE mechanism, and we detail the construction in the following Section 5.

5 Asymmetric PAKE in the Standard Model

In this section, we show our main contribution and explain how to transform a protocol for one-round symmetric-PAKE into a one-round asymmetric-PAKE.

5.1 One-Round Symmetric PAKE Over Lattices

An Approximate Bit-PHF. Micciancio-Peikert (in short MP) encryption is a labeled IND-CCA1-secure under the decisional LWE assumption, which is an elegant candidate to design an approximate b-PHF and SPHF based on MP encryption [27], [28], we abbreviate them to MP-b-PHF and MP-SPHF, respectively. An encoding function is used for messages $\mathsf{encode}(\mu \in \{0,1\}) = \mu \cdot (0, \ldots, 0, \lceil q/2 \rceil)^T \in \mathbb{Z}_q^m$ for an odd prime q. In addition, $2 \cdot \mathsf{encode}(\mu) = \mu \cdot (0, \ldots, 0, \mu)^T \in \mathbb{Z}_q^m$. To approximate calculation of the outputs of hashing values, the square-signal function $R(x) = \lfloor 2x/q \rceil \pmod 2$ is introduced by [25]. Very recently, a new cost-consuming rounding function $R(x) = 1/2 + \cos(2\pi x/q)/2$ with a high degree of accuracy is introduced by Benhamouda *et al.* [27]. MP-b-PHF is sketched as follows.

- $hk \leftarrow \mathsf{MP\text{-}b\text{-}PHF.HashKG}(\mathsf{param})$. It outputs a $hk := \mathbf{k}$ that is a random vector from $\mathbb{Z}_q^{m\times 1}$.
- $ph \leftarrow \mathsf{MP\text{-}b\text{-}PHF.ProjKG}(hk = \mathbf{k}, pk = \mathbf{A}_u)$. It outputs a $ph := \mathbf{p} = \mathbf{A}_u \cdot \mathbf{k} \in \mathbb{Z}_q^{n\times 1}$ by taking as input the $hk := \mathbf{k}$ and the public key $\mathbf{A}_u = [\bar{\mathbf{A}} \,|\, h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n\times m}$ of Micciancio-Peikert encryption with the fixed label lbl^u.
- $h \leftarrow \mathsf{MP\text{-}b\text{-}PHF.Hash}(hk = \mathbf{k}, \mathsf{wd} := (\mathsf{ct}, \mathbf{m}))$. It outputs the hashing value h by taking as input $\mathbf{k}$ and wd, where wd contains a plaintext $\mathbf{m}$ and its corresponding ciphertext $\mathsf{ct} = (\mathsf{lbl}^l, \mathbf{c} \in \mathbb{Z}_q^{m\times 1})$. In particular, h is calculated as follows,

$$\begin{aligned} h &= \mathsf{MP\text{-}b\text{-}PHF.Hash}(hk = \mathbf{k}, \mathsf{wd} := (\mathsf{ct}, \mathbf{m})) \\ &= R\Big([\mathbf{c} - (\mathbf{0} \,|\, \mathsf{encode}(\mathbf{m}))]^T \cdot \mathbf{k}\Big) \\ &= R\Big((\mathbf{s}^T\mathbf{A}_u) \cdot \mathbf{k} + \mathbf{e}^T\mathbf{k} \,(\mathrm{mod}\ q) \in \mathbb{Z}_q\Big) \in \{0,1\}, \end{aligned}$$

where the element $\mathbf{e}^T \cdot \mathbf{k}$ is the noise that is bounded by $|\mathbf{e}^T\mathbf{k}| \le \|\mathbf{e}^T\| \cdot \|\mathbf{k}\| \le (r\sqrt{mn}) \cdot (\alpha q\sqrt{mn}) < \varepsilon/2 \cdot q/4$. Next the algorithm outputs $b := h \,(\mathrm{mod}\ 2) \in \{0,1\}$ and sets $b = 0$ if $h < 0$, otherwise, sets $b = 1$.

- $p = \mathsf{MP\text{-}b\text{-}PHF.Proj}(ph = \mathbf{p}, \mathsf{wd} := (\mathsf{ct}, \mathbf{m}); w = \mathbf{s})$. It outputs the projection hashing value p by taking as input a $ph = \mathbf{p} \in \mathbb{Z}_q^{n\times 1}$, a wd, and an $\mathbf{s} \in \mathbb{Z}_q^{n\times 1}$. In particular, p is calculated as follows,

$$\begin{aligned} p &= \mathsf{Proj}(ph = \mathbf{p}, \mathsf{wd} := (\mathsf{ct}, \mathbf{m}); w = \mathbf{s}) \\ &= R\Big(\mathbf{s}^T \cdot \mathbf{p}\Big) \\ &= R\Big(\mathbf{s}^T \cdot (\mathbf{A}_u\mathbf{k}) \,(\mathrm{mod}\ q)\Big) \in \{0,1\}. \end{aligned}$$

Next the algorithm outputs $b := p \,(\mathrm{mod}\ 2) \in \{0,1\}$, sets $b = 0$ if $p < 0$, otherwise, sets $b = 1$.

Theorem 5.1. *The* MP-b-PHF *scheme has the universality and statistical correctness.*

As pointed out by [27], if we obtain an approximate (word-independent) universal MP-b-PHF, then an approximate (word-independent) MP-SPHF can be obtained based on the universal MP-b-PHF in a straightforward way.

Review MP-SPHF From Micciancio-Peikert Scheme. The approximate MP-b-PHF scheme with a concrete rounding function $R(x) = 1/2 + \cos(2\pi x/q)/2$ can be transformed straightforwardly by increasing the size of the output of the hashing function, which means $\mathsf{HashKG}(\cdot)$ samples several independent hashing keys $hk_1, hk_2, \ldots, hk_v$ for $v = \Omega(\lambda)$, and concatenates the outputs of all the corresponding $\mathsf{Hash}(\cdot)$ results. The approximate MP-SPHF is defined as follows.

- $hk \leftarrow \mathsf{MP\text{-}SPHF.HashKG}(\mathsf{param})$. It outputs a hashing key $hk := (\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_v)$ by sampling v times $\mathbf{k}_i \in \mathbb{Z}_q^{m\times 1}$ for $i \in [v]$.
- $ph \leftarrow \mathsf{MP\text{-}SPHF.ProjKG}(hk = \mathbf{k}, pk = \mathbf{A}_u)$. It derives the projection hashing key $ph := \mathbf{p} = (\mathbf{A}_u \cdot \mathbf{k}_1, \mathbf{A}_u \cdot \mathbf{k}_2, \ldots, \mathbf{A}_u \cdot \mathbf{k}_v) \in (\mathbb{Z}_q^{n\times 1})^v$ from $\mathbf{k} := (\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_v)$ and $\mathbf{A}_u = [\bar{\mathbf{A}} \,|\, h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n\times m}$.
- $h \leftarrow \mathsf{MP\text{-}SPHF.Hash}(hk = \mathbf{k}, \mathsf{wd} := (\mathsf{ct}, \mathbf{m}))$. It outputs the hashing value $h = (h_1 \| \ldots \| h_v)$ by taking as input $\mathbf{k}$ and wd. In particular, h_i is calculated as follows,

$$\begin{aligned} h_i &= \mathsf{MP\text{-}SPHF.Hash}(\mathbf{k}_i, \mathsf{wd} := (\mathsf{ct}, \mathbf{m})) \\ &= R\Big([\mathbf{c} - (\mathbf{0} \,|\, \mathsf{encode}(\mathbf{m}))]^T \cdot \mathbf{k}_i\Big) \\ &= R\Big((\mathbf{s}^T \cdot \mathbf{A}_u) \cdot \mathbf{k}_i + \mathbf{e}^T \cdot \mathbf{k}_i \,(\mathrm{mod}\ q)\Big) \in \{0,1\}. \end{aligned}$$

- $p = \mathsf{MP\text{-}SPHF.Proj}(ph = \mathbf{p}, \mathsf{wd} := (\mathsf{ct}, \mathbf{m}); w = \mathbf{s})$. It outputs the projection hashing value p by concatenating

v projected hashing values of MP-b-PHF. Namely, $p = (p_1 \| \ldots \| p_v)$, where p_i is calculated as follows,

$$p_i = \mathsf{Proj}(\mathbf{p}_i, \mathsf{wd} := (\mathsf{ct}, \mathbf{m}); w = \mathbf{s}) = R\Big(\mathbf{s}^T \cdot \mathbf{p}\Big) = R\Big(\mathbf{s}^T \cdot (\mathbf{A}_u \mathbf{k}_i) \,(\mathrm{mod}\ q)\Big) \in \{0,1\}.$$

Lemma 5.2. *The* MP-SPHF *is a smooth projective hash function for the* MP *scheme.*

- *Approximate correctness*. Notably, the *approximate correctness* property follows because

$$\Pr[\mathsf{Hash}(hk_i, \mathsf{wd}) = \mathsf{Proj}(ph_i, \mathsf{wd}, w)] \geq 1 - negl(\lambda),$$

  is obtained using the Hoeffding bound.
- *Correctness*. Katz and Vaikuntanathan [25] provided a generic transformation from an approximate SPHF to an SPHF by utilizing Error Correcting Codes (in short ECCs) that enable to correct an ε-fraction of errors.
- *Smoothness*. It ensures that when $\mathsf{wd} \notin \mathcal{L}$, the output of $\mathsf{Hash}(hk, \mathsf{wd} \notin \mathcal{L})$ is negligibly close to uniform even when knowing the projection hashing key ph. Additionally, as each MP-b-PHF is independent and smooth, the classical hybrid argument approach is used to prove the *smoothness* property by using uniform distributions U_i on first i values $\mathsf{Hash}(hk_i, \mathsf{wd})$, while the others are computed as usual.

Review One-Round Symmetric PAKE via MP-SPHF. There is limited literature available concerning PAKE over lattices [25], [26], [28] and no literature over lattices deals with asymmetric authentication. The first three-round symmetric-PAKE protocol over lattices is proposed by Katz and Vaikuntanathan [25] under a variant of the BPR model [42]. A two-round symmetric-PAKE protocol via the approximate SPHF is proposed by Zhang and Yu [26]. However, an important tool of their construction depends on simulation-sound NIZK proofs while there is no concrete simulation-sound NIZK construction over lattices. A two-round symmetric-PAKE protocol over lattices is proposed by Li and Wang [28], whose construction bypasses the NIZK utilization, and it requires an IND-CCA-secure encryption with an associated word-independent KV-SPHF for the client side and an associated word-dependent GL-SPHF for the server side. Further, Benhamouda *et al.* [53] gave a draft of one-round symmetric-PAKE over lattices, but its main drawback is the lack of rigorous analysis.[2] Very recently, Li and Wang [48] is inspired by [27], and proposed a one-round symmetric-PAKE over lattices with rigorous analysis, and it only requires an IND-CCA-secure encryption with an associated word-independent KV-SPHF on both sides.

5.2 One-Round Asymmetric PAKE Over Lattices

Our Construction. Along with symmetric-PAKE constructions, some kinds of zero-knowledge proof must be used, as for example SPHF schemes. In a nutshell, in each session, both participants generate fresh SPHF "private" hashing key hk and "public" projection hashing key ph to be employed on incoming messages. The public ph is sent along with the encrypted message, for instance, the client sends the message $(\mathsf{ct}_C \leftarrow \mathsf{Enc}(\mathsf{pw}_C), ph_C)$ to the server while the server sends the message $(\mathsf{ct}_S \leftarrow \mathsf{Enc}(\mathsf{pw}_S), ph_S)$ to the client. If the encrypted message is obtained by encrypting the correct password, meaning that both parties have the same password or outcome of $\mathsf{PHash}((\bar{s}, s), \mathsf{pw})$, then $\mathsf{SPHF.Hash}(hk, \mathsf{wd})$ takes as input the message received and its own hashing key hk, and outputs $\mathsf{SPHF.Proj}(ph, \mathsf{wd}, w)$. Notably, $\mathsf{SPHF.Proj}(ph, \mathsf{wd}, w)$ takes as input the SPHF projection hashing key ph received from its corresponding parter, the local message, and its private witness. Thus, these SPHF hashes (i.e., $\mathsf{SPHF.Proj}(\cdot)$ and $\mathsf{SPHF.Hash}(\cdot)$) are enabled to calculate the session key. Smoothness property guarantees the security of asymmetric-PAKE. Moreover, each participant must retain their private witness (i.e., randomness) used in the (IND-CCA2) encryption, as the outgoing message of $\mathsf{SPHF.Proj}(\cdot)$ requires this witness.

As depicted in Fig. 3, our one-round asymmetric-PAKE protocol builds on by our proposed PHS scheme and the MP-SPHF scheme optimized by Benhamouda *et al.* [27]. Before executing the asymmetric-PAKE, we first execute the modulus switching to transfer the pre-hash value $\bar{y}$ and the hashing value y over $\mathbb{Z}_p$ to over $\mathbb{Z}_q$. Here, we omit how to transfer them. To our knowledge, Benhanmuda and Pointcheval [17] introduced a one-round asymmetric-PAKE using a naive PHS along with CS-SPHF which is based on the labeled Cramer-Shoup encryption. Subsequently, Kiefer and Manulis [36] developed a two-round asymmetric-PAKE using a Pedersen-like PHS along with CS-SPHF and KV-SPHF. In this work, in order to obtain the first post-quantum one-round asymmetric-PAKE in the standard model, armed with the first one-round PAKE over lattices proposed by [28], we integrate the proposed PHS into one-round PAKE protocol.

Notably, traditional PAKE schemes are based on the non-quantum-safe assumptions (e.g., DDH) in BPR model, and the security is reduced to the non-quantum-safe assumption. In the lattice-based crypto setting, the only difference is that the security is reduced to the LWE assumption because building blocks are instantiated using lattice-based instantiations. Additionally, we remark that the protocol is asymmetric because we use a PHS. Also, the utilization of GL-SPHF allows computing the shared key in a secure way. For instance, we note that replay attacks are not possible, since the attacker doesn't have access not only to password, but also to values r and hk_C, or respectively $\bar{r}$ and hk_S. This is implicitly proved as a result of the fact that in the CCA game an adversary enables to query to encryption and decryption oracles, but has no access to underlying random coins, r and $\bar{r}$, or private keys. Together with the hashing properties of the subjacent GL-SPHF scheme, we achieve security in the BPR *standard* model, where we need to use a CRS. Since we are not in the ROM model, we don't need to adapt to quantum ROM model. Hence by following the standard model, we are able to prove the security of our construction based on real-or-random model, whose building blocks are quantum-resistant. Therefore, even a quantum adversary does not have the ability to distinguish protocol messages from random values.

To achieve this goal, the client and the server require an IND-CCA encryption along with different languages for its associated GL-SPHF. The client and the server

2. Note that their scheme was withdrawn from ePrint due to the weakness of the security proof.

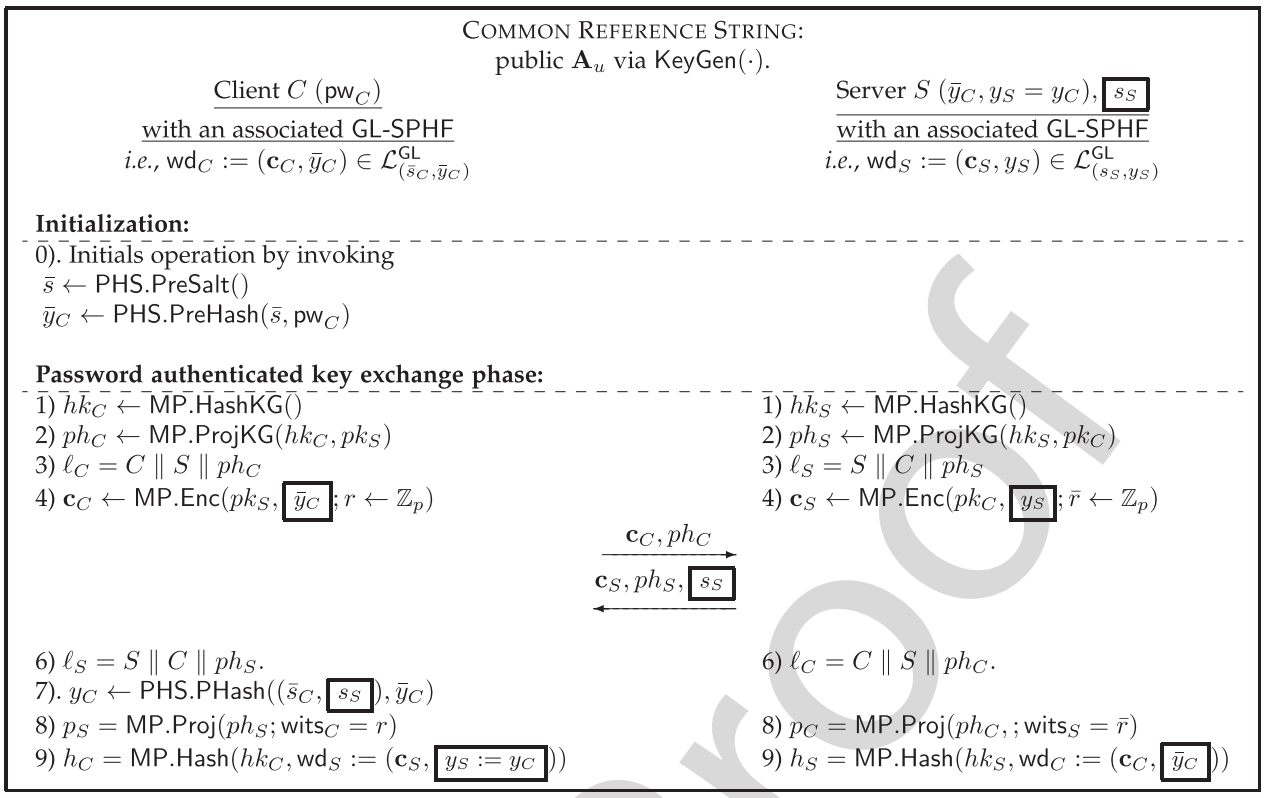

Fig. 3. Asymmetric one-round PAKE via MP-SPHF and PHS [18]. For better illustration, the black box is used to highlight the different inputs compared with the symmetric-PAKE.

have a GL-SPHF for the two following families of languages:

– The word of the client is $\mathsf{wd}_C := (\mathbf{c}_C, \bar{y}_C) \in \mathcal{L}^{\mathsf{GL}}_{(\bar{s}_C, \bar{y}_C)}$,

$$\mathcal{L}^{\mathsf{GL}}_{(\bar{s}_C, \bar{y}_C)} = \{(\ell, \mathsf{ct}^{\mathrm{cca}}) \,|\, \exists\ \mathsf{pw}, \exists\ r, \mathsf{ct}^{\mathrm{cca}} = \mathsf{CCA.Enc}^{\ell}(pk_{\mathrm{cca}}, \underline{\bar{y}_C}; r) \wedge \bar{y}_C \leftarrow \mathsf{PHS.PreHash}(\bar{s}_C, \mathsf{pw}_C)\},$$

where $\bar{s}_C \leftarrow \mathsf{PHS.PreSalt}()$.

– The word of the client is $\mathsf{wd}_S := (\mathbf{c}_S, y_S) \in \mathcal{L}^{\mathsf{GL}}_{(s_S, y_S)}$,

$$\mathcal{L}^{\mathsf{GL}}_{(s_S, y_S)} = \{(\ell, \mathsf{ct}^{\mathrm{cca}}) \,|\, \exists\ \mathsf{pw}, \exists\ r, \mathsf{ct}^{\mathrm{cca}} = \mathsf{CCA.Enc}^{\ell}(pk_{\mathrm{cca}}, \underline{y_S}; r) \wedge y_S \leftarrow \mathsf{PHS.PHash}((\bar{s}_C, s_S), \bar{y}_C)\}.$$

Notably, $\mathcal{L}^{\mathsf{GL}} \supset \mathcal{L}^{\mathsf{GL}}_{(s,y)}$ holds, and we have $\mathcal{L}^{\mathsf{GL}} = \{(\ell, \mathsf{ct}^{\mathrm{cca}}) \,|\, \exists\ \bar{y}, \exists\ r, \mathsf{ct}^{\mathrm{cca}} = \mathsf{CCA.Enc}^{\ell}(pk_{\mathrm{cca}}, \underline{\bar{y}}; r)\}$ for any (s, y).

Correctness for One-Round Asymmetric PAKE.

Lemma 5.3 (Correctness). *If the magnitude of inner product $\langle \mathbf{e}, \mathbf{k} \rangle$ for n, $m \geq n\sqrt{\log q}$ is small, then $p_S = h_S$ and $p_C = h_S$.*

Proof. To facilitate understanding and evaluation, we round the outputs of $\mathsf{Hash}(\cdot)$ and $\mathsf{Proj}(\cdot)$ respectively by using the typical deterministic rounding $R(x) = \lfloor 2x/q \rceil \pmod 2$ as discussed in [25].

At the client side, we have the following two results,

$$\begin{aligned} p_S :&= \mathsf{Proj}(ph_S, \mathsf{wd}_C = (\mathbf{c}_C, \bar{y}_C); w_C = r := \mathbf{s}_C) \\ &= \underline{R\Big(\mathbf{s}_C^T \cdot \mathbf{p}_S\Big) = R\Big(\mathbf{s}_C^T (\mathbf{A}_u \mathbf{k}_S)\Big)} \end{aligned}$$

$$\begin{aligned} h_C :&= \mathsf{Hash}(hk_C, \mathsf{wd}_S := (\mathbf{c}_S, y_S)) \\ &= R\Big([\mathbf{c}_S - (\mathbf{0} \,|\, \mathsf{encode}(y_S))]^T \cdot \mathbf{k}_C\Big) \\ &= \underline{R\Big((\mathbf{s}_S^T \mathbf{A}_u)\mathbf{k}_C + \mathbf{e}_S^T \mathbf{k}_C\Big)}. \end{aligned}$$

At the server side, we have the following two results,

$$\begin{aligned} p_C &= \mathsf{Proj}(ph_C, \mathsf{wd}_S = (\mathbf{c}_S, y_S); w_S = \bar{r} := \mathbf{s}_S) \\ &= \underline{R\Big(\mathbf{s}_S^T \cdot \mathbf{p}_C\Big) = R\Big(\mathbf{s}_S^T (\mathbf{A}_u \mathbf{k}_C)\Big)} \end{aligned}$$

$$\begin{aligned} h_S :&= \mathsf{Hash}(hk_S, \mathsf{wd}_C := (\mathbf{c}_C, \bar{y}_C)) \\ &= R\Big([\mathbf{c}_C - (\mathbf{0} \,|\, \mathsf{encode}(\bar{y}_C))]^T \cdot \mathbf{k}_S\Big) \\ &= \underline{R\Big((\mathbf{s}_C^T \mathbf{A}_u)\mathbf{k}_S + \mathbf{e}_C^T \mathbf{k}_S\Big)}. \end{aligned}$$

Here the rounding function $R(h)$ is regarded as a number in $[-\frac{(q-1)}{2}, \ldots, \frac{(q-1)}{2}]$ and outputs $b \in \{0, 1\}$. Moreover, the noise element $\mathbf{e}^T \cdot \mathbf{k}$ is bounded by $|\mathbf{e}^T \mathbf{k}| \leq \|\mathbf{e}^T\| \cdot \|\mathbf{k}\| \leq (r\sqrt{mn}) \cdot (\alpha q \sqrt{mn}) < \varepsilon/2 \cdot q/4$. Hence, the result of

TABLE 1
Performance Comparison of Various PAKE Protocols†

Scheme	Assumption	Model	(A)symm	Round(& Flow)	Auth	Building Blocks
Katz-Vaikuntanathan [25]	Lattice	Std	×	3 (& 3)	×	KV-SPHF + Peikert enc
Zhang-Yu [26]	LWE	Std	×	2 (& 2)	×	GL-SPHF + Peikert enc+NIZK
Benhamouda *et al.* [27]	LWE	Std	×	1 (& 2)	×	KV-SPHF + MP enc
Li-Wang [28]	LWE	Std	×	2 (& 2)	×	KV-SPHF + Regev + MP enc
Li-Wang [48]	LWE	Std	×	1 (& 2)	×	KV-SPHF + MP enc
Benhamouda-Pointceval [17]	DDH+SXDH+GXDH	Std	✓	2 (& 2)	✓	GL-SPHF + CS enc+ PHS
Jutla-Roy [20]	MDDH	ROM+UC	✓	4 (& 4)	✓	QA-NIZK +EG enc+PHS
Jarecki-Krawczyk-Xu [22]	DDH	ROM+UC	✓	3 (& 3)	✓̸	Oblivious-PRF
Haase-Labrique [24]	DDH	ROM+UC	✓	4 (& 4)	✓	PBKDF+Hash Functions
Bradley-Jarecki-Xu [47]	DDH+SDH+STOWF	ROM+UC	✓	2 (& 2)	✓	Trapdoor-CKEM(SPHF+Enc)
Ours	LWE	Std	✓	1 (& 2)	✓	(SPHF +MP enc+PHS)

Std: Standard model;
ROM: Random oracle model;
UC: Universal composability;
CS enc: Cramer-Shoup encryption;
EG enc: ElGamal encryption;
MP enc: Micciancio-Peikert encryption;
(A)symm: Asymmetric or Symmetric;
Auth: With authentication or not;
QA-NIZK: Quasi-Adaptive NIZK, which is achieved using GL-SPHF*;*
✓: the scheme does support this property;
✓̸: the scheme does support this property partially;
×: the scheme does not support this property;
Round: unidirectional (asynchronously) communication;
Flow: bidirectional (simultaneously) communication.

$R(\mathbf{e}^T\mathbf{k})$ is identical to 0, and we obtain $p_S = h_S$ and $p_C = h_S$. □

Security for One-Round Asymmetric-PAKE.

Theorem 5.4. *The one-round asymmetric* PAKE *protocol as depicted in Fig. 3 is secure under the* LWE *assumption in the BPR model, if the* PHS *and the* MP-SPHF *are secure over lattices.*

Sketched proof. PAKE has two security goals: one is that no one is enabled to recover the password from a series of exchanges using active attacks (e.g., "Man-in-the-Middle" (MITM) attacks). (2) Another is that no one could be authenticated without knowing the password (or having a correct guess to it). Our modularity analysis strategy follows the methodologies of Katz and Vaikuntanathan [29] and Benhamouda *et al.* [54, Theorem 4], we only need to check that the SPHF associated with MP encryption, and PHS associated with CDGLW commitment, fulfil the same properties. Below, we sketch the RoR analysis strategy. We use the Execute oracle to model the passive security (i.e., eavesdropping) for the two participants, respectively. Then we use the Send oracle to model the active security (i.e., MITM attacks, insertion, deletion, or arbitrarily modification, *etc*) for the two participants. After that, we use the Corrupt oracle to model the setting of the server corruption. Finally, we use the Test oracle to answer the adversary whether her guess is correct or not. Additionally, the "replay attack" falls in an attempt to violate the second security property of PAKE, and it is also covered by our analysis, because the replay attack is one of the lower tier version of a MITM attack, which implies that a valid data transformation is maliciously or fraudulently repeated or delayed. In our construction, replay attacks are not possible, since attacker doesn't have access not only to password, but also to a randomness values r and the hashing key of the client hk_C, or respectively a randomness $\bar{r}$ and the hashing key of the server hk_S. This is implicitly proved as a result of the fact that in the CCA game an adversary has access to encryption and decryption oracles, but has no access to underlying random coins, r and $\bar{r}$, or private keys. Hence, the straight-forward replay attacks will fail. Please see the appendix for our detailed security analysis, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2020.3040776. □

6 Experiments

6.1 Theoretical Analysis

We give a detailed comparison of our asymmetric-PAKE with the existing schemes as shown in Table 1, where "*flow*" means the unidirectional (asynchronously) communication and "*round*" means the bidirectional (simultaneously) communication. Notably, Jutla and Roy [20] and Bradley *et al.* [47] achieved UC secure asymmetric-PAKE protocols via SPHF variants, and Jarecki *et al.* [22] achieves an efficient asymmetric-PAKE against pre-computation attacks using oblivious-PRF and authenticated key exchange to resist key-compromise impersonation (KCI) attacks. However, these schemes cannot remain secure in the quantum coming era. Further, Haase and Labrique [24] gave an asymmetric-PAKE for Industry IoT based on specific number-theoretic assumptions, but the drawbacks of constructions based on number-theoretic assumptions are that the parties need to prove to her partner that he knows the credential in every login, as discussed in [54]. Apparently, compared to the foremost asymmetric-PAKE protocols, the main advantage of our scheme is that our scheme can achieve round-optimal asymmetric-PAKE protocol with the assistance of password-hashing schemes while still secure against quantum attacks. Meanwhile, our asymmetric-PAKE is based on SPHF which allows low-interactivity of the symmetric-PAKE and guarantees that each party does not need to prove to the other that he knows the right password.

6.2 Benchmark and Experimental Analysis

Below we show benchmark results for our protocol. Table 2 gives the execution time of each algorithm construction.

TABLE 2
Parameter Selection

Case	(n, q, σ)	Prime (m, b, bits)	Dual (m, b, bits)
Classical	(592, 2^{12}, 1.0)	(549, 132,442)	(544, 130, 438)
Quantum	(752, 2^{15}, 1.323)	(716, 132,489)	(737, 130, 485)

TABLE 3
Communication Comparison of Round-Optimal Asymmetric-PAKE Protocols for Classical and Quantum Computers

Communication	Asymmetric		Symmetric
	Client	Server	Client&Server
Classical	26362 bits	32950 bits	26362 bits
Quantum	29602 bits	40320 bits	29602 bits

TABLE 4
Performance Comparison of Round-Optimal Asymmetric-PAKE Protocols for Classical and Quantum Computers

Performance	Symmetric		Asymmetric		
	Client	Server	Client	Server	Logon
Classical	361 ms	357 ms	116 ms	361 ms	13 ms
Quantum	473 ms	471 ms	116 ms	473 ms	18 ms

Our C++ reference implementation is complied with gcc-10.2.0 and the measurement is obtained on a workstation with an Intel(R) Core(TM) i7-8750H running at 2.20 GHz.

Parameter Selection for LWE. Following the recommended parameters of Frodo [55], we use a classical parameter set that could guarantee 128-bit of security against classical attacks as discussed by NIST PQC round 2.[3] In particular, we characterise LWE instances by parameters m, n, α, q and the special case $m = n$, $q = n^c, \alpha \cdot q = \sqrt{n}$, i.e., $\alpha \approx n^{1/2-c}$ for constant c in the following way. For example, as discussed in [56], the authors set $n = 256$, $q = 2^{16} - 1 = 65537$ and $\sigma = \alpha \cdot q/\sqrt{2\pi} \approx 25.53$ following [57] and picking $m = 838$ (or $m = 784$). In this work, the implementation of our protocol are following Frodo (i.e., LWE-based key exchange) [55] and the parameter finding scripts are available at https://github.com/lwe-frodo/.

In Tables 3 and 4 we show timings for the classical and quantum set of parameters, respectively. Our proof of concept was successful in the sense that the schoolbook implementation is achievable. We can optimize algorithms to improve performance even more for practical applications. The MP encryption scheme allows to achieve CCA-security, thus being useful for the construction of one-round asymmetric PAKE schemes. On the other hand, Regev construction is more efficient, but its security is restricted to the CPA-attacks. Each encryption scheme has a corresponding SPHF construction, which was used as a *proof of knowledge of plaintext*. Hence it is an important building block in the construction of key agreement protocols. In Table 5, we show timing information for both SPHF schemes, based on Regev and MP constructions.

3. https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf

TABLE 5
Performance Measurement

Operation (n, m, q, σ)	Classical		Quantum	
	Time	Size	Time	Size
HashKG	$140\mu s$	12600 bits	210μ	16010 bits
ProjKG	$48ms$	333340 bits	$73ms$	423356 bits
Enc	$416ms$	23810 bits	$512ms$	30242 bits
Hash	$1.9ms$	450 bits	$2.5ms$	549 bits
Proj	$7.2ms$	450 bits	$12.8ms$	549 bits
PreHash	$18\mu s$	6588 bits	$28\mu s$	10740 bits
PHash	$27\mu s$	6588 bits	$41\mu s$	10740 bits

7 Conclusion

The major goal of this paper is to adapt the lattice-based PHS in order to integrate it into the symmetric-PAKE construction, obtaining the round-optimal asymmetric-PAKE based on SPHF. To achieve this goal, we show how to instantiate Pedersen-like PHS using the commitment of Cabarcas *et al*. [38]. In addition, once obtained the lattice-based PHS, then we show how to find a solution to integrate the PHS into the symmetric-PAKE protocol while maintaining security against quantum computer attacks. Importantly, the existing lattice-based SPHF-based symmetric-PAKE could be transformed into in the asymmetric-PAKE using our PHS, such as [19], [25], [27], [28], [48]. As a next step in this line of research we leave the question of how to check the password policy when the password was hashed by password-hashing approaches in the asymmetric-PAKE setting, and we will continue to explore how to design more lattice-based PAKE for emerging applications [58], [59], [60].

Acknowledgments

The authors thank the anonymous reviewers for their invaluable comments. This research was supported in part by the National Natural Science Foundation of China under Grant 61802006 and Grant 61802214, in part by the National Natural Science Foundation of Shandong province, China under Grant ZR2019BF009, in part by the applied basic research project of Qingdao under Grant 19-6-2-6-cg, and in part by the Foundation of Guizhou Provincial Key Laboratory of Public Big Data under Grant No. 2019BDKFJJ007.

References

[1] G. Hatzivasilis, "Password-hashing status," *Cryptography*, vol. 1, no. 2, 2017, Art. no. 10.
[2] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. IEEE Symp. Security Privacy*, 2012, pp. 553–567.
[3] I. Haitner, E. Omri, and H. Zarosim, "Limits on the usefulness of random oracles," *J. Cryptol.*, vol. 29, no. 2, pp. 283–335, 2016.
[4] M. I. González Vasco, A. Perez Del Pozo, and C. Soríente, "A key for John Doe: Modeling and designing anonymous password-authenticated key exchange protocols," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2919013.
[5] M. Shirvanian, N. Saxena, S. Jarecki, and H. Krawczyk, "Building and studying a password store that perfectly hides passwords from itself," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 5, pp. 770–782, Sep./Oct. 2019.
[6] W. Li, X. Li, J. Gao, and H. Y. Wang, "Design of secure authenticated key management protocol for cloud computing environments," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2909890.

[7] P. Liu, S. Li, and Q. Ding, "An energy-efficient accelerator based on hybrid CPU-FPGA devices for password recovery," *IEEE Trans. Comput.*, vol. 68, no. 2, pp. 170–181, Feb. 2019.

[8] Q. Yang, K. Xue, J. Xu, J. Wang, F. Li, and N. Yu, "AnFRA: Anonymous and fast roaming authentication for space information network," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 486–497, Feb. 2019.

[9] Z. Ba, Z. Qin, X. Fu, and K. Ren, "CIM: Camera in motion for smartphone authentication," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 11, pp. 2987–3002, Nov. 2019.

[10] L. Wu, J. Wang, K. R. Choo, and D. He, "Secure key agreement and key protection for mobile device user authentication," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 319–330, Feb. 2019.

[11] Q. Xie, D. S. Wong, G. Wang, X. Tan, K. Chen, and L. Fang, "Provably secure dynamic ID-based anonymous two-factor authenticated key exchange protocol with extended security model," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 6, pp. 1382–1392, Jun. 2017.

[12] J. Katz, R. Ostrovsky, and M. Yung, "Efficient password-authenticated key exchange using human-memorable passwords," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2001, pp. 475–494.

[13] R. Gennaro and Y. Lindell, "A framework for password-based authenticated key exchange," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2003, pp. 524–543.

[14] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proc. IEEE Comput. Soc. Symp. Res. Security Privacy*, 1992, pp. 72–84.

[15] S. M. Bellovin and M. Merritt, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise," in *Proc. 1st ACM Conf. Comput. Commun. Secur.*, 1993, pp. 244–250.

[16] C. Gentry, P. D. MacKenzie, and Z. Ramzan, "A method for making password-based key exchange resilient to server compromise," in *Proc. Annu. Int. Cryptol. Conf.*, 2006, pp. 142–159.

[17] F. Benhamouda and D. Pointcheval, "Verifier-based password-authenticated key exchange: New models and constructions," Cryptology ePrint Archive, Report 2013/833, [Online]. Available: https://eprint.iacr.org/2013/833.

[18] F. Kiefer and M. Manulis, "Zero-knowledge password policy checks and verifier-based PAKE," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2014, pp. 295–312.

[19] Z. Zhang, K. Yang, X. Hu, and Y. Wang, "Practical anonymous password authentication and TLS with anonymous client authentication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1179–1191.

[20] C. S. Jutla and A. Roy, "Smooth NIZK arguments," in *Proc. Theory Cryptogr. Conf.*, 2018, pp. 235–262.

[21] D. Pointcheval and G. Wang, "VTBPEKE: Verifier-based two-basis password exponential key exchange," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 301–312.

[22] S. Jarecki, H. Krawczyk, and J. Xu, "OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2018, pp. 456–486.

[23] P. Dupont, J. Hesse, D. Pointcheval, L. Reyzin, and S. Yakoubov, "Fuzzy password-authenticated key exchange," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2018, pp. 393–424.

[24] B. Haase and B. Labrique, "AuCPace: Efficient verifier-based PAKE protocol tailored for the IIoT," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2019, no. 2, pp. 1–48, 2019.

[25] J. Katz and V. Vaikuntanathan, "Smooth projective hashing and password-based authenticated key exchange from lattices," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2009, pp. 636–652.

[26] J. Zhang and Y. Yu, "Two-round PAKE from approximate SPH and instantiations from lattices," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2017, pp. 37–67.

[27] F. Benhamouda, O. Blazy, L. Ducas, and W. Quach, "Hash proof systems over lattices revisited," in *Proc. IACR Int. Workshop Public Key Cryptogr.*, 2018, pp. 644–674.

[28] Z. Li and D. Wang, "Two-round PAKE protocol over lattices without NIZK," in *Proc. 14th Int. Conf. Inf. Secur. Cryptol.*, 2018, pp. 138–159.

[29] J. Katz and V. Vaikuntanathan, "Round-optimal password-based authenticated key exchange," in *Proc. Theory Cryptogr. Conf.*, 2011, pp. 293–310.

[30] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Annu. Int. Cryptol. Conf.*, 1991, pp. 129–140.

[31] A. Everspaugh, R. Chatterjee, S. Scott, A. Juels, and T. Ristenpart, "The pythia PRF service," in *Proc. 24th USENIX Conf. Secur. Symp.*, 2015, pp. 547–562.

[32] D. Dolev, C. Dwork, and M. Naor, "Non-malleable cryptography (extended abstract)," in *Proc. 23rd Annu. ACM Symp. Theory Comput.*, 1991, pp. 542–552.

[33] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *Proc. Annu. Int. Cryptol. Conf.*, 1998, 1998, pp. 13–25.

[34] R. Gennaro and Y. Lindell, "A framework for password-based authenticated key exchange[1]," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 2, pp. 181–234, 2006.

[35] R. Cramer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn., Advances.*, 2002, pp. 45–64.

[36] F. Kiefer and M. Manulis, "Blind password registration for verifier-based PAKE," in *Proc. 3rd ACM Int. Workshop ASIA Public-Key Cryptogr.*, 2016, pp. 39–48.

[37] K. Nguyen, B. H. M. Tan, and H. Wang, "Zero-knowledge password policy check from lattices," in *Proc. Int. Conf. Inf. Secur.*, 2017, pp. 92–113.

[38] D. Cabarcas, D. Demirel, F. Göpfert, J. Lancrenon, and T. Wunderer, "An unconditionally hiding and long-term binding post-quantum commitment scheme," IACR Cryptol. ePrint Archive, Report 2013/833. [Online]. Available: http://eprint.iacr.org/2015/628

[39] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Proc. Annu. Cryptol. Conf.*, 2012, pp. 868–886.

[40] Z. Li, C. Ma, and D. Wang, "Achieving multi-hop PRE via branching program," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 45–58, Firstquarter 2020.

[41] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs, "Learning with rounding, revisited - new reduction, properties and applications," in *Proc. 33rd Annu. Cryptol. Conf.*, 2013, pp. 57–74.

[42] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2000, pp. 139–155.

[43] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proc. Annu. Int. Cryptol. Conf.*, 1993, pp. 232–249.

[44] M. Bellare and P. Rogaway, "Provably secure session key distribution: The three party case," in *Proc. 27th Annu. ACM Symp. Theory Comput.*, 1995, pp. 57–66.

[45] J. Katz, R. Ostrovsky, and M. Yung, "Efficient and secure authenticated key exchange using weak passwords," *J. ACM*, vol. 57, no. 1, pp. 3:1–3:39, 2009.

[46] J. Y. Hwang, S. Jarecki, T. Kwon, J. Lee, J. S. Shin, and J. Xu, "Round-reduced modular construction of asymmetric password-authenticated key exchange," in *Proc. 11th Int. Conf. Secur. Cryptogr. Netw.*, 2018, pp. 485–504.

[47] T. Bradley, S. Jarecki, and J. Xu, "Strong asymmetric PAKE based on trapdoor CKEM," in *Proc. Annu. Int. Cryptol. Conf.*, 2019, pp. 798–825.

[48] Z. Li and D. Wang, "Achieving one-round password-based authenticated key exchange over lattices," *IEEE Trans. Service Comput.*, 2019. [Online]. Available: http://dx.doi.org/10.1109/TSC.2019.2939836

[49] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 708–722, Jul./Aug. 2018.

[50] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2776–2791, Nov. 2017.

[51] S. Agrawal, P. Miao, P. Mohassel, and P. Mukherjee, "PASTA: Password-based threshold authentication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 2042–2059.

[52] Z. Li, Z. Yang, P. Szalachowski, and J. Zhou, "Building low-interactivity multi-factor authenticated key exchange for industrial Internet-of-Things," *IEEE Internet Things J.*, 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2020.3008773

[53] O. Blazy, C. Chevalier, L. Ducas, and J. Pan, "Exact smooth projective hash function based on LWE," Cryptol. ePrint Archive, Report 2013/821, 2013. [Online]. Available: https://eprint.iacr.org/2013/821

[54] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud, "New techniques for SPHFs and efficient one-round PAKE protocols," in *Proc. Annu. Cryptol. Conf.*, 2013, vol. 8042, pp. 449–475.

[55] J. W. Bos *et al.*, "Frodo: Take off the ring! practical, quantum-secure key exchange from LWE," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1006–1018.
[56] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *J. Math. Cryptol.*, vol. 9, no. 3, pp. 169–203, 2015.
[57] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proc. 37th Annu. ACM Symp. Theory Comput.*, 2005, pp. 84–93.
[58] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 69–78, Jan. 2015.
[59] J. Wei, X. Chen, X. Huang, X. Hu, and W. Susilo, "RS-HABE: Revocable-storage and hierarchical attribute-based access scheme for secure sharing of e-health records in public cloud," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2947920.
[60] X. Zhang, X. Chen, J. K. Liu, and Y. Xiang, "DeepPAR and deepDPA: Privacy preserving and asynchronous deep learning for industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2081–2090, Mar. 2020.

Zengpeng Li received the PhD degree from Harbin Engineering University (HEU), China, in 2018. During his doctoral program, he was a research assistant with the University of Auckland, New Zealand and Virginia Commonwealth University, USA, respectively. Currently, he is joining Nankai University (NKU), China. Prior to joining NKU, he has worked as a faculty member of Qingdao University (QDU), China, a postdoctoral research fellow at the Singapore University of Technology and Design (SUTD), Singapore, and a postdoctoral research associate at Lancaster University, United Kingdom, respectively. His primary research interests are in cryptographic protocol and secure distributed computing. In particular, his research efforts mainly focus on secure computing on encrypted data, verifiable computation, and password-based cryptography.

Ding Wang received the PhD degree in information security from Peiking University, China, in 2017. Currently, he is a full professor at Nankai University, China, and also serves as the Deputy-Director of Tianjin Key Laboratory of Network and Data Security Technology. Before joining Nankai University, China, he was a lecture and supported by the "Boya Postdoctoral Fellowship" at Peking University, China. He has been involved in the community as the PC Chair, a TPC member and a reviewer for more than 50 international conferences and journals. His works appear in prestigious venues like ACM CCS,Usenix Security, NDSS, ESORICS, DSN, the *IEEE Transactions on Dependable and Secure Computing* and the *IEEE Transactions on Information Forensics and Security*, and seven of them are selected as "ESI highly cited papers". He received the "2018 ACM China Doctoral Dissertation Award", and the "2017 China Computer Federation (CCF) Outstanding Doctoral Dissertation Award". His research interests include password, authentication, and provable security.

Eduardo Morais received the PhD degree from the University of Campinas, Brazil, in 2016. Currently he is an independent researcher and a cryptographic engineer working with zero knowledge proofs at the Input Output Hong Kong (IOHK) Ltd., Hong Kong. Before joining IOHK Ltd., he was an independent researcher and a cryptographic engineer at the International Netherlands Group (ING) Ltd., in Netherland. His interests are focused on privacy-preserving technologies, zero-knowledge proof, and post-quantum cryptography.

APPENDIX A
PROOF OF THEOREM 5.4

Below, we provide the detailed analysis for the Theorem 5.4.

Experiment $\mathsf{Expt}(0)$. This is the real attack experiment, the advantage of a polynomial time adversary $\mathcal{A}$ is quantified as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(0)}(\lambda) = \varepsilon$. Importantly, all honest participants have their private inputs that are enabled to use by the simulator. Notably, the actual passwords are never used by the simulator before any corruption, or the end of the game.

Below, we consider a $\mathcal{A}$ who wins (success) when its guessing-bit b' is equal to the challenge bit b. To make the trivial attacks possible, we incrementally change the process of simulation by a series of experiments $\mathsf{Expt}(i)$, and the advantage of $\mathcal{A}$ is denoted as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(i)}(\lambda) = 2 \cdot \Pr[b = b'] - 1$ in the experiment $\mathsf{Expt}(i)$.

Following [29], [54], we assume that $\mathcal{A}$ can only statistically corrupt participants, in particular, $\mathcal{A}$ can only corrupt a password or a hashed password when no concrete instance of the associated participants is involved in a process of an execution. Send queries are separated two types as follows.

- $\mathsf{Send}_0(C_i, S_j, \mathsf{Start})$-query. The adversary $\mathcal{A}$ initiates an execution between an instance Π_C^i of C and an instance Π_S^j of S by querying the oracle $\mathsf{Send}_0(\cdot)$. In particular, the adversary $\mathcal{A}$ enables to ask C to query S, and S responses the query by interacting with C in a flow.
- $\mathsf{Send}_1(C, i, \mathbf{u})$-query. The adversary $\mathcal{A}$ is able to query the oracle $\mathsf{Send}_1(\cdot)$ by sending the flow (*i.e.*, $\mathbf{u}$) from Π_C^i to Π_S^i. Then the oracle $\mathsf{Send}_1(\cdot)$ enables to calculate his/her own session key but returns noting.

Experiment $\mathsf{Expt}(1)$. $\mathsf{Expt}(1)$ proceeds as in $\mathsf{Expt}(0)$ except that we modify how to generate the salts. Particular, the $\mathbf{s} = \mathbf{U} \cdot \mathbf{s}_p := s \leftarrow \mathsf{PSalt}(\mathsf{param})$ is replaced by $\mathbf{s}_p := \bar{s} \leftarrow \mathsf{PreSalt}(\mathsf{param})$, so that a trapdoor τ is known for any salt s. Thus, we have $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(0)}(\lambda)| = 0$ under the salt indistinguishability.

Experiment $\mathsf{Expt}(2)$. $\mathsf{Expt}(2)$ proceeds as in $\mathsf{Expt}(1)$ except that the approach how to modify the query of Execute-oracle between two *compatible* participants. In response to $\mathsf{Execute}(U_C, i, U_S, j)$, the encryption of the fake password $\mathsf{pw}_0 \notin L$ that is sampled from the uniform distribution is used to replace the actual ciphertext $\mathbf{c}_C$ and $\mathbf{c}_S$ who are generated by computing $\mathbf{c}_C \leftarrow \mathsf{MP.Enc}(pk, \bar{y}_C; r)$ and $\mathbf{c}_S \leftarrow \mathsf{MP.Enc}(pk, y_S; \bar{r})$, respectively. This is indistinguishable guaranteed by the IND-CCA-secure encryption for each $\mathsf{Execute}(\cdot)$-query. Additionally, in this case, the hashing keys are known by the participants, hence they enable to calculate locally the common session key as follows,

$$\begin{aligned} \mathsf{key}_C &= \mathsf{MP.Hash}(hk_C, \mathsf{wd}_S \in \mathcal{L}_{(s_S, y_S)}^{\mathsf{GL}}) \cdot \\ &\quad \mathsf{MP.Hash}(hk_S, \mathsf{wd}_C \in \mathcal{L}_{(\bar{s}_C, \bar{y}_C)}^{\mathsf{GL}}) = \mathsf{key}_S. \end{aligned} \tag{A.1}$$

Thus, the correctness of SPHF guarantees that the common session key does not change anything as Equation A.1. Notably, the setting of two *compatible* users imply that the following two equations $\mathsf{Hash}(hk_C, \mathsf{wd}_S \in \mathcal{L}_{(s_S, y_S)}^{\mathsf{GL}}) = \mathsf{Proj}(ph_C, \mathsf{wd}_S \in \mathcal{L}_{(s_S, y_S)}^{\mathsf{GL}}; w_S = \bar{r})$ and $\mathsf{Proj}(ph_S, \mathsf{wd}_C \in \mathcal{L}_{(\bar{s}_C, \bar{y}_C)}^{\mathsf{GL}}; w_C = r) = \mathsf{Hash}(hk_S, \mathsf{wd}_C \in \mathcal{L}_{(\bar{s}_C, \bar{y}_C)}^{\mathsf{GL}})$ are hold. Thus, we have $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(2)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(1)}(\lambda)| \leq \mathrm{negl}(\lambda)$ using a classical hybrid hops.

Experiment $\mathsf{Expt}(3)$. In experiment $\mathsf{Expt}(3)$, the approach on how to respond to Execute-queries is modified between two *compatible* participants. To be specific, a truly random value is sampled from a uniform, and it is utilized to substitute for the common session key. At this point, the "password" is not satisfied, and the indistinguishability property is guaranteed by the smoothness of SPHF, *i.e.*, $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(2)}(\lambda)| \leq \mathrm{negl}(\lambda)$.

Experiment $\mathsf{Expt}(4)$. $\mathsf{Expt}(4)$ proceeds as in $\mathsf{Expt}(3)$ except how to response the Execute-queries between two *incompatible* participants. Explicitly, in response to $\mathsf{Execute}(U_C, i, U_S, j)$-queries, the actual ciphertext $\mathbf{c}_C$ and $\mathbf{c}_S$ are replaced by the encryption of the fake password $\mathsf{pw}_0 \notin L$ that is sampled from the uniform. This is indistinguishable guaranteed by the IND-CCA-secure encryption for each $\mathsf{Execute}(\cdot)$-query. Further, the hashing key and projective key are known by the participants, thus, they are enabled to obtain the session key:

$$\begin{aligned} \mathsf{key}_C &= \mathsf{MP.Hash}(hk_C, \mathsf{wd}_S := (\mathbf{c}_S, \mathsf{pw})) \cdot \\ &\quad \mathsf{MP.Proj}(ph_S, w_C = r) \\ &= \mathsf{MP.Hash}(hk_S, \mathsf{wd}_C := (\mathbf{c}_C, \mathsf{pw})) \cdot \\ &\quad \mathsf{MP.Proj}(ph_C, w_S = \bar{r}) = \mathsf{key}_S. \end{aligned}$$

Soundness property of the SPHF guarantees, using either the initial approach or the modified approach, there is no impact on the generation of session key key (*i.e.*, $\mathsf{key}_C = \mathsf{key}_S$). Thus, for each Execute-query, we can obtain $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(4)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(3)}(\lambda)| \leq \mathrm{negl}(\lambda)$ by using a series of hybrid hops, which is guaranteed by the indistinguishable property of the probabilistic encryption scheme.

Experiment $\mathsf{Expt}(5)$. In experiment $\mathsf{Expt}(5)$, the responsion of Execute-queries is modified again. To be specific, two independent random values are sampled from a uniform distribution and they are used to replace session keys key_C and key_S. At this point, the "password" is not satisfied, and the indistinguishability property is underlying based on the smoothness of SPHF, which implies that $\mathsf{Hash}(hk_C, \mathsf{wd}_S \in \mathcal{L}_{(s_S, y_S)}^{\mathsf{GL}})$ and $\mathsf{Hash}(hk_S, \mathsf{wd}_C \in \mathcal{L}_{(\bar{s}_C, \bar{y}_C)}^{\mathsf{GL}})$ are computational indistinguishable to independent random values. Thus, we obtain $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(5)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}(4)}(\lambda)| \leq \mathrm{negl}(\lambda)$.

Experiment $\mathsf{Expt}(6)$. $\mathsf{Expt}(6)$ proceeds as in $\mathsf{Expt}(5)$, except that how one responses the Send_1-queries. Concretely, to answer the $\mathsf{Send}_1(C, i, \mathbf{u}_C)$-query for the flow $\mathbf{u}_C = (ph_C, \mathbf{c}_C)$ sent from the instance U_C^i to the instance U_S^j, where $\mathbf{c}_C \leftarrow \mathsf{MP.Enc}(pk, \bar{y}_C; r)$, we first assume the password hashing function $\mathsf{PHS.PHash}(\cdot)$ of the server is corrupted, then a decryption oracle who knows the decryption key more precisely and decrypts another flow $\mathbf{u} = (ph_S, \mathbf{c}_S)$ is inducted by the simulator in the name of U_S^j, then the simulator answers and computes the session key using the salt s_S (and $\mathsf{PHS.PHash}(\cdot)$). Thus, three cases will happen as follows:

1) If the adversary has altered (or generated) the flow $\mathbf{u}_C$, then the pre-hash value $\bar{y}_C$ can be recovered to the adversary using the decryption oracle. Hence, the case (1) is divided into the following two sub-cases:

a) If the validation $\mathsf{Checkable}(\mathsf{param}, \tau_{\bar{s}}, s, \bar{y}, y)$ outputs 1, the session key $= \perp$ is sampled by the adversary $\mathcal{A}$. Later, if the adversary $\mathcal{A}$ tests the session key by querying Test-query, which implies that the adversary $\mathcal{A}$ wins the game, and the simulation is terminated.
b) Otherwise, the session key key is sampled at random as $\bar{y}$ and y are not both consistent with the values of receiver.

2) If the simulator is replied by the message $\mathbf{u}$ for a previous flow, then the session key can be calculated by key $= \mathsf{Hash}(hk_C, \mathsf{wd}_S := (\mathbf{c}_S, y_S)) \cdot \mathsf{Proj}(ph_S, w_C = r)$, where $\mathbf{c}_S$ is generated without using the randomness rather using the same approach with $\mathsf{Expt}.(3)$ and $\mathsf{Expt}.(5)$. Further, $\mathbf{c}_C$ is used to reply the Send_1-query.

In a word, no session key is computed in $\mathsf{Expt}(6)$. Moreover, case (1a) captures the increasement of the adversary $\mathcal{A}$'s advantage, whose probability will compute in the following $\mathsf{Expt}(8)$. Case (1b) reflects the advantage of $\mathcal{A}$ increases in a negligible term becasue it captures the changes in this case are indistinguiable under the adaptive-smoothness of the GL-SPHF. In addition, case (2) captures the computed session key key can not be effectted by the changes in this case since $\mathsf{wd}_C := (\mathbf{c}_C, \bar{y}_C) \in \mathcal{L}^{\mathsf{GL}}_{(\bar{s}_C, \bar{y}_C)}$. Therefore, $|\mathsf{Adv}^{\mathsf{Expt}(6)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Expt}(5)}_{\mathcal{A}}(\lambda)| \leq \mathrm{negl}(\lambda)$.

Experiment $\mathsf{Expt}(7)$. $\mathsf{Expt}(7)$ proceeds as in $\mathsf{Expt}(6)$ except that the approach how to response the $\mathsf{Send}_1(\cdot)$-queries. In $\mathsf{Expt}(6)$, when the hashed password of the server S is not corrupted (*i.e.*, $\mathsf{PHS.PHash}(\cdot)$), then the ciphertexts $\mathbf{c}_S$ is generated at the server side without knowing the randomness. Thus, in $\mathsf{Expt}(7)$, to response $\mathsf{Send}_1(\cdot)$-queries, the ciphertexts $\mathbf{c}_S$ is generated at the server side using the dummy password 0 rather the correct hashed password $y_S \leftarrow \mathsf{PHS.PHash}(\mathsf{param}_{\mathsf{ph}}, (\bar{s}, s), \bar{y})$. Hence, there exists $|\mathsf{Adv}^{\mathsf{Expt}(7)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Expt}(6)}_{\mathcal{A}}(\lambda)| \leq \mathrm{negl}(\lambda)$ that is guaranteed by the indistinguishability of IND-CCA-secure encryption.

Experiment $\mathsf{Expt}(8)$. $\mathsf{Expt}(8)$ proceeds as in $\mathsf{Expt}(7)$ except that the approach how to response the $\mathsf{Send}_1(\cdot)$-queries. Concretely, the second flow $\mathbf{u}_S$ is defined the message sent from the server instance U^j_S to the client instance U^i_C, which consists of a slat s, the server's projection key ph_S, and $\mathbf{c}_S \leftarrow \mathsf{MP.Enc}(pk, y_S; r)$, *i.e.*, $\mathbf{u}_S = (s, ph_S, \mathbf{c}_S)$. To answer the $\mathsf{Send}_1(S, i, \mathbf{u}_S)$-query after sending the flow $\mathbf{u}_S$, we first assume the password pre-hashing function $\mathsf{PHS.PreHash}(\cdot)$ of the client is corrupted [4], then a decryption oracle with the decryption key enables to decrypt the previous flow $\mathbf{u} = (ph_C, \mathbf{c}_C)$ introduced by the simulator in the name of U^i_C. Then the simulator answers and computes the session key using the salt s_S (and $\mathsf{PHS.PHash}(\cdot)$). Thus, four cases will happen as follows:

1) If the flow $\mathbf{u}_S = (s, ph_S, \mathbf{c}_S)$ is generated (or altered) by the adversary rather the server S (via a Send_1-query) after receiving the flow $\mathbf{u}_C$ sent by C (via a Send_0-query), then the hash value y_S can be recovered to the adversary by decrypting the ciphertexts $\mathbf{c}_S$. Thus, case (1) can be divided into two sub-cases as follows.
   a) If the equation $y_S \leftarrow \mathsf{PHS.PHash}(\mathsf{param}_{\mathsf{ph}}, (\bar{s}, s), \bar{y})$ is hold, then it implies that the password pw at the client side is compatible with $(\bar{y}_C, y_S, s)$ at the server side, where $y_S = y_C$, then the session key is sampled as key $= \perp$ by the adversary. Later, if the adversary $\mathcal{A}$ tests the session key by querying Test-query, which implies that the adversary $\mathcal{A}$ wins the game, and the simulation is terminated.
   b) Otherwise, the session key key is sampled at random.
2) If the flow $\mathbf{u}_S = (s, ph_S, \mathbf{c}_S)$ is from some instances of S^j, then C^i is partnered with S^j.
   a) If the instance U^i_C and the instance U^j_S are compatible, then the session key of U^i_C is set to equal to the one of U^j_S that is calculated earily.
   b) Otherwise, the session key key is sampled at random.

Note that, case (1a) captures the increasement of the adversary $\mathcal{A}$'s advantage. Meanwhile, case (1b) and case (2b) reflect the advantage of $\mathcal{A}$ increases in a negligible term because it captures the changes in this case are indistinguishable under the smoothness of the KV-SPHF. Additionally, case (2a) captures the computed session key key can not be effected by the changes in this case since $\mathsf{wd}_S := (\mathbf{c}_S, y_S) \in \mathcal{L}^{\mathsf{GL}}_{(s_S, y_S)}$. Therefore, $|\mathsf{Adv}^{\mathsf{Expt}(8)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Expt}(7)}_{\mathcal{A}}(\lambda)| \leq \mathrm{negl}(\lambda)$.

Experiment $\mathsf{Expt}(9)$. $\mathsf{Expt}(9)$ proceeds as in $\mathsf{Expt}(8)$ except that the approach how to response the $\mathsf{Send}_0(\cdot)$-queries. Similar to $\mathsf{Expt}(6)$, in $\mathsf{Expt}(8)$, when the pre-hashed password of the client C is not corrupted (*i.e.*, $\mathsf{PHS.PreHash}(\cdot)$), then the ciphertexts $\mathbf{c}_C$ is generated at the client side without knowing the randomness. Thus, in $\mathsf{Expt}(9)$, to response $\mathsf{Send}_0(\cdot)$-queries, the ciphertexts $\mathbf{c}_C$ is generated at the client side using the dummy password 0 rather the correct the pre-hashed password $\bar{y}_C \leftarrow \mathsf{PHS.PreHash}(\mathsf{param}_{\mathsf{ph}}, \bar{s}, \mathsf{pw}_C)$. Hence, there exists $|\mathsf{Adv}^{\mathsf{Expt}(9)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Expt}(8)}_{\mathcal{A}}(\lambda)| \leq \mathrm{negl}(\lambda)$ that is guaranteed by the indistinguishability of IND-CCA-secure encryption scheme.

Experiment $\mathsf{Expt}(10)$. $\mathsf{Expt}(10)$ proceeds as in $\mathsf{Expt}(9)$ except how to answer the Send_1-query for the replayed message $(ph_C, \mathbf{c}_C)$ sent by the adversary. In addition to the corruption of the pre-hashed password of the client C, if the flow is honestly generated, then the session key is sampled at random. The indistinguishability of the IND-CCA-secure encryption guarantees that there is no adversary can distinguish the encryption of the dummy pre-hashed password 0 from the correct pre-hashed value. Further, given ph_S, the smoothness of KV-SPHF guarantees that the hash value of $\mathbf{c}_C$ under the hashing key hk_S (*i.e.*, $h_C \leftarrow \mathsf{MP.Hash}(hk_S, \mathsf{wd}_C := (\mathbf{c}_C, \mathsf{pw}))$) is indistinguishable from the uniform. Hence, there exists $|\mathsf{Adv}^{\mathsf{Expt}(10)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Expt}(9)}_{\mathcal{A}}(\lambda)| \leq \mathrm{negl}(\lambda)$. Notably, in $\mathsf{Expt}(10)$, all session keys returned by the Test-queries are completely independent and random (except for the partnered compatible users, for which they are equal).

Moreover, the simulation does not use the password directly during the simulation, and no information will be leaked to with the exception of corruption. Hence, the adversary wins the game if the following two cases happen.

1) If a flow $\mathbf{u}_C$ sent by the simulator to the server S with non-corrupted the hashed password, then a valid pre-

4. In symmetric case and Benhamouda and Pointcheval scheme (ePrint2013/833), in general, we assume the password of the client is corrupted.

hash value $\bar{y}_C$ can be extracted by the simulator, such that the validation $\mathsf{Checkable}(\mathsf{param}, \tau_{\bar{s}}, s, \bar{y}, y)$ outputs 1. If the session key is obtained by the adversary using Test-query, then refer to the $\mathsf{Expt}(6)$.

2) Or if a flow $\mathbf{u}_S$ sent by the simulator to the client C, then a valid hash value y_S can be extracted by the simulator, such that the equation $y_S \leftarrow \mathsf{PHS.PHash}(\mathsf{param}_{\mathsf{ph}}, (\bar{s}, s), \bar{y})$ is hold, where the salt s is drawn by the adversary (as described in $\mathsf{Expt}(8)$). If the session key is obtained by the adversary using Test-query, then refer to the $\mathsf{Expt}(6)$.

Here, passwords are chosen independently at random from a distribution $\mathcal{D}$ of min-entropy β (*i.e.*, not in the related-password model) rather a uniform distribution. Hence, we conclude the following two events if no corruption occurs.

- Event Ev_1, it happens with probability at most $1/2^{\beta} + \mathrm{negl}(\lambda)$ for each flow since "the second pre-image property" and "the pre-hash entropy preservation". Below we conclude the two main reasons.
  1) "Second pre-image property" ensures that $\bar{y}_C \leftarrow \mathsf{PHS.PreHash}(\mathsf{param}_{\mathsf{ph}}, \bar{s}_C, \mathsf{pw})$.
  2) "Pre-hash entropy preservation" ensures that the probability of finding $\bar{y}_C$ without knowing pw nor y_S cannot be more than $1/2^{\beta} + \mathrm{negl}(\lambda)$.
- Event Ev_2, it happens with probability at most $1/2^{\beta} + \mathrm{negl}(\lambda)$ for each flow since "the entropy preservation".

Finally, the probability of the adversary win the last game is

$$\Pr[Ev] \leq Q(\lambda) \cdot 1/2^{\beta} + \mathrm{negl}(\lambda),$$

(*i.e.*, at most $1/2^{\beta} + \mathrm{negl}(\lambda)$) without corruptions, if passwords are sampled independently. $Q(\lambda)$ is the number of active query sessions.

Further, if corruptions occur on either the client or the server, but if the adversary never executes a Test-query for a client with a corrupted pre-hash password, then the case is same with above. Otherwise, if the flow sent by the adversary to the server S can be extracted using a valid pre-hashed $\bar{y}_C$ for $\bar{y}_S$ and a corrupted and hashed value $\bar{y}_S$, then the adversary will win the game. To prevent the above attack happening, an adversary $\mathcal{B}$ is constructed against the tight one-wayness of the password hashing. Thus, the probability of the adversary win the last game is

$$\Pr[Ev] \leq Q(\lambda) \cdot 1/2^{\beta} + \mathrm{Adv}_{\mathcal{B},\mathsf{owf}}(\lambda) + \mathrm{negl}(\lambda).$$