Two-Round PAKE Protocol over Lattices Without NIZK

Zengpeng Li[1] and Ding Wang[2(✉)]

[1] College of Computer Science and Technology, Qingdao University, Qingdao, China
lizengpeng@hrbeu.edu.cn

[2] School of EECS, Peking University, Beijing 100871, China
wangdingg@pku.edu.cn

Abstract. Reducing the number of communication rounds of Password-based Authenticated Key Exchange (PAKE) protocols is of great practical significance. At PKC'15, Abdalla et al. relaxed the requirements of Gennaro-Lindell's framework for three-round PAKE protocols, and obtained a two-round PAKE protocol under the traditional DDH-based smooth projective hash function (SPHF). At ASIACRYPT'17, Zhang and Yu proposed a lattice-based two-round PAKE protocol via the approximate SPHF. However, the language of Zhang-Yu's SPHF depends on simulation-sound non-interactive zero-knowledge (NIZK) proofs, for which there is *no* concrete construction without random oracle under lattice-based assumptions. To our knowledge, how to design a lattice-based two-round PAKE protocol via an efficient SPHF scheme without NIZK remains a challenge. In this paper, we propose the first two-round PAKE protocol over lattices without NIZK. Our protocol is in accordance with the framework of Abdalla et al. (PKC'15) while attaining post-quantum security. We overcome the limitations of existing schemes by relaxing previous security assumptions (i.e., both the client and the sever need IND-CCA-secure encryption), and build two new lattice-based SPHFs, one for IND-CCA-secure Micciancio-Peikert ciphertext (at the client side) and the other for IND-CPA-secure Regev ciphertext (at the server side). Particularly, our protocol attains provable security.

Keywords: Password-based Authenticated Key Exchange · Smooth projective hash function · Lattice-based · Provable security

1 Introduction

Password-based Authenticated Key Exchange (PAKE) protocols are perhaps the most widely used cryptographic protocols, dating back to Bellovin and Merritt's PAKE protocol (named EKE) in 1992 [1]. They showed how two parties, each of which pre-shares a human-memorized password and communicate over a public network, can verify the authenticity of each other and establish a cryptographically robust session key to protect their ensuing data communications. Their EKE is successful in preventing low-entropy passwords from being offline guessed

F. Guo et al. (Eds.): Inscrypt 2018, LNCS 11449, pp. 138–159, 2019.
https://doi.org/10.1007/978-3-030-14234-6_8

by dictionary attacks, and therefore demonstrates the feasibility of employing password-only protocols to build secure communication channels over public networks, which is a key goal of cryptography. Owing to the practicality of PAKE, Bellovin-Merritt's seminal paper [1] has been followed by hundreds of PAKE proposals with varied security and complexity, such as KOY [2], J-PAKE [3] and OPAQUE [4].

In order to generalize the KOY scheme, Gennaro and Lindell [5] introduced the smooth projective hash function (or SPHF) to instantiate the KOY scheme in the Bellare, Pointcheval, and Rogaway (BPR) security model [6]. It is common to abbreviate the general KOY scheme to Gennaro-Lindell framework. Since then, considerable attention has been devoted to developing secure and efficient PAKE protocols via SPHFs, some notable ones include [7,8].

Most of these existing PAKE protocols under the Gennaro-Lindell framework are three-rounds and depend on an "IND-CCA2-secure" encryption scheme to establish a high-entropy session key. How to reduce the number of rounds and relax the security assumption(s) are two important concerns. At SAC'04, Jiang and Gong [9] relaxed the security of Gennaro-Lindell framework by using the combination of an IND-CPA scheme at the server side and an IND-CCA2 scheme at the client side, and did not require the IND-CCA2 scheme at the server side, but their protocol still needs three rounds. At PKC'15, Abdalla et al. [10] reduced the communication rounds by relaxing the Gennaro-Lindell framework and obtained a two-round PAKE under the traditional DDH-based SPHF. In their protocol, the client requires an indistinguishable against plaintext checkable attacks (or IND-PCA) scheme and the server requires an IND-CPA scheme.[1]

At ASIACRYPT'17, Zhang and Yu [11] proposed a lattice-based two-round PAKE protocol via approximate SPHF. However, the language of their SPHF relies on simulation-sound non-interactive zero-knowledge (NIZK) proofs, for which there is no concrete construction without the random oracle under lattice-based assumptions. In a nutshell, it still remains an open question as to:

> *Whether is it possible to construct a secure and efficient two-round* PAKE *protocol without NIZK via the* LWE*-based* SPHF*s?*

1.1 Our Results and Techniques

In this work, we answer the above question in the affirmative. At PKC'15, Abdalla et al. [10] pointed out that, their IND-PCA-secure PKE scheme is also IND-CCA2-secure for small message space. Inspired by this observation, we first adopt the existing IND-CCA-secure LWE-based Micciancio and Peikert scheme [12] to meet the requirements of IND-PCA-secure PKE scheme, and then follow the SPHF design principles suggested by Katz and Vaikuntanathan [13] and propose one lattice-based MP−SPHF for IND-CCA-secure Micciancio-Peikert ciphertext (at the client side) and the other lattice-based Reg−SPHF for

[1] Note that every IND-CCA2-secure scheme is also an IND-PCA-secure scheme.

IND-CPA-secure Regev ciphertext [14] (at the server side). Finally, armed with Reg−SPHF and MP−SPHF, we construct a two-round PAKE in line with the principles of [10]. In all, we make the following contributions:

- **New two-round PAKE protocol**. Zhang and Yu [11] proposed the first lattice-based two-round PAKE protocol in the random oracle model which is built upon the splittable PKE scheme along with the non-adaptive approximate SPHF.[2] However, their construction depends on the IND-CCA1-secure Katz-Vaikuntanathan [13] scheme and simulation soundness NIZK from lattices in the random oracle model. The main drawbacks are that: the Katz-Vaikuntanathan scheme [13] needs to invoke the Invert$(\cdot)$ algorithm many times until the plaintext is recovered, and there is no concrete construction involving NIZK but without random oracle under lattice-based assumptions. To overcome both limitations, we introduce the Micciancio-Peikert scheme [12] and the Regev scheme [14] to design two lattice-based SPHFs (i.e., MP−SPHF for the client side and Reg−SPHF for the server side) as the building blocks of our lattice-based PAKE.
- **Weaker security assumptions**. Though some one-round PAKE protocols were proposed (e.g., [7,15]), these constructions require stronger (i.e., IND-CCA) assumptions for both client and server sides in the security model. Thus, relaxing the security assumptions is another important issue [8,9]. Abdalla et al. [10] constructed a DDH-based two-round PAKE protocol by introducing the new IND-PCA-secure cryptographic primitive to relax the security requirement of the server side from IND-CCA to IND-PCA. In our PAKE, IND-CCA-secure encryption is required at the client side, while IND-CPA-secure encryption is required at the server side.
- **New security formulation**. When formulating the attacker $\mathcal{A}$'s advantage **Adv**, existing PAKE literature (e.g., [7–9,11,16]) invariably assume that passwords come from a uniformly random distribution, and **Adv** is thus formulated as $Q(\lambda)/|\mathcal{D}| + \text{negl}(\lambda)$ for an attacker making at most $Q(\lambda)$ on-line guesses, where λ is the system security parameter and $\mathcal{D}$ is the password space. However, user-chosen passwords are *not* uniformly distributed, but follow the Zipf's law [17,18]. Thus, we use the formulation $C' \cdot Q(\lambda)^{s'} + \text{negl}(\lambda)$ to more accurately capture $\mathcal{A}$'s advantage **Adv**, where $s' \in [0.15, 0.30]$ and $C' \in [0.001, 0.1]$ [17,18] are constant CDF-Zipf regression parameters of $\mathcal{D}$.

1.2 Related Works

We now give a brief history of PAKE and SPHF.

PAKE. We first remark that we use *flow* to denote the unidirectional communication between the parties, and the *round* can be used to denote the bidirectional communication between the parties. If the messages are sent asynchronously, then the round and flow are the same notation. But if the messages are sent

[2] The non-adaptive approximate SPHF means the adversary can see the projective key *ph* before choosing the word W.

simultaneously, then each round contains two flows. Actually, if the PAKE protocols were divided according to the communication rounds, then there exist three types of PAKE protocols. **(1)** Three-round (or three-flow) PAKE as first introduced by Katz, Ostrovsky and Yung [2] was only achieved based on DDH assumption in the standard model. After that, a series of works [5,8,9,13] were proposed to improve the three-round PAKE protocols. **(2)** Two-round (or two-flow) PAKE as first introduced by Abdalla et al. [10] was achieved by introducing a new cryptographic primitive IND-PCA-secure PKE scheme, followed by Zhang and Yu who proposed the first two-round PAKE over lattices. **(3)** Katz and Vaikuntanathan [15] proposed the general one-round (but two-flow) PAKE framework which requires the client and the server to send messages to each other simultaneously. Alternatively, Groce and Katz [8] extended the Jiang-Gong scheme [9] in the universal composability (UC) framework [19,20] and proved it secure. Afterwards, a series of PAKE in UC-model were discussed [4,21,22].

SPHF. Cramer and Shoup [23] first proposed the concept of SPHF which is a special kind of hash proof system and defined on the NP language L over a domain X. Concretely, there are two basic keyed functions (i.e., $\mathsf{Hash}(\cdot)$ and $\mathsf{ProjHash}(\cdot)$) in SPHFs. The participants can compute $\mathsf{Hash}(\cdot)$ by taking as input the private hashing key hk and a word W. Similarly, the one can compute the function $\mathsf{ProjHash}(\cdot)$ by taking the public projective hashing key ph, a witness w and a word W, where the word W contains the message msg and corresponding labeled IND-CCA ciphertext $\mathbf{c}$. Notably, the output distributions of the two functions are statistically indistinguishable for a word W over the language L.

2 Preliminaries

We denote vector $\mathbf{x}$ via bold lower-case letter and matrix $\mathbf{A}$ via bold upper-case letter, and λ the security parameter. An m-dimension lattice can be written as $\varLambda = \{\mathbf{Bs} \mid \mathbf{s} \in \mathbb{Z}^n\}$, where $\mathbf{B} \in \mathbb{Z}^{m \times n}$ is called basis of $\varLambda$ for $m \geq n\lceil \log q \rceil$. Notably, the determinant of $\varLambda$ is $det(\varLambda) = \sqrt{det(\mathbf{B}^T\mathbf{B})}$. Meanwhile, we adopt the typical deterministic rounding function of [13] to discard the noise elements.

Definition 1 (The Square-Signal Function, [13]). *The typical deterministic rounding function (a.k.a., the so-called square-signal) was defined as* $R(x) = \lfloor 2x/q \rceil \pmod 2$. *The value of* $R(h)$ *can be viewed as a number in* $[-\frac{(q-1)}{2}, \cdots, \frac{(q-1)}{2}]$ *and output* $b \in \{0,1\}$.

Definition 2 (Hamming Metric). *For any two strings of equal length* $x, y \in \{0,1\}^v$*, the* Hamming *distance is one of several string metrics for measuring the edit distance between two strings. We write it* $\mathsf{HD}(x, y)$.

2.1 Lattice Background and Learning with Errors

Definition 3 ([24]). *A distribution ensemble* $\chi = \chi(\lambda)$ *over the integers is called* B*-bounded (denoted* $|\chi| \leq B$*) if there exists:* $\Pr_{x \xleftarrow{\$} \chi}[|x| \geq B] \leq 2^{-\tilde{\Omega}(n)}$.

Definition 4 (Decision-$\mathsf{LWE}_{n,q,\chi,m}$). *Assume given an independent sample* $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times 1}$*, where the sample is distributed according to either: (1)* $\mathcal{A}_{\mathbf{s},\chi}$ *for a uniform random* $\mathbf{s} \in \mathbb{Z}_q^n$ *(i.e.,* $\{(\mathbf{A}, \mathbf{b}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}, \mathbf{s} \leftarrow \mathbb{Z}_q^{n\times 1}, \mathbf{e} \leftarrow \chi^{m\times 1}, \mathbf{b} = \mathbf{A}\cdot\mathbf{s} + \mathbf{e} \pmod q\}$*), or (2) the uniform distribution (i.e.,* $\{(\mathbf{A}, \mathbf{b}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}, \mathbf{b} \leftarrow \mathbb{Z}_q^{m\times 1}\}$*). Then, the above two distributions are computationally indistinguishable.*

Remark 1. Reductions between the LWE assumption and approximating the shortest vector problem in lattices (for appropriate parameters) were shown in [14,25–27], here we omit the corollary of these schemes' results.

Lemma 1 (From [12]). *The* PPT *algorithm* $\mathsf{Invert}(\cdot)$ *can be used to invert the injective trapdoor function* $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^T \cdot \mathbf{A} + \mathbf{e} \pmod q$*, and satisfies the following requirements:*

- *The algorithm takes as input the following parameters:* **(1).** *a parity-check matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *along with* **(2).** *a* $\mathbf{G}$*-trapdoor* $\mathbf{R} \in \mathbb{Z}^{\bar{m}\times n\ell_q}$*, where* $\mathbf{A}\cdot\binom{\mathbf{R}}{\mathbf{I}} = \mathbf{H}\cdot\mathbf{G}$ *for the invertible tag* $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ *of* $\mathbf{R}$*.* **(3).** *an* LWE *instance* $\mathbf{b}$ *satisfying* $\mathbf{b} = \mathbf{s}^T \cdot \mathbf{A} + \mathbf{e} \pmod q$*.*
- *The algorithm outputs the secret vector* $\mathbf{s}$ *(which depends on the value of* $\mathbf{b}^T \cdot \binom{\mathbf{R}}{\mathbf{I}}$*.) and the noise vector* $\mathbf{e} = \mathbf{b} - \mathbf{A}^T\mathbf{s}$*.*

2.2 Smooth Projective Hash Functions

Cramer and Shoup [23] first introduced the projective hash function families at EUROCRYPT'02. SPHF acts as an important type of the projective hash function which requires the existence of a domain X and an underlying NP language $L \subseteq X$ such that it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$. More precisely, an SPHF contains four PPT algorithms over $L \subseteq X$

$$\mathsf{SPHF} = (\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash}),$$

which is defined as follows:

- $\mathsf{HashKG}(L)$ inputs an NP language L and outputs a hash key hk.
- $\mathsf{ProjKG}(hk, L, W)$ inputs an NP language L, a hk, and a word $W \in L$ and outputs a projective hash key ph.
- $\mathsf{Hash}(hk, L, W)$ inputs an NP language L, a hk, and a $W \in L$ and outputs a hash value h over $\{0,1\}^v$ for some positive integer $v = \Omega(\lambda)$.
- $\mathsf{ProjHash}(ph, L, W, w)$ inputs an NP language L, a ph, a $W \in L$, and a witness w and outputs a projective hash value $p \in \{0,1\}^v$.

Meanwhile, the SPHFs satisfy the notions of (approximate) correctness and smoothness:

- **Approximate Correctness:** We say the property of approximate correctness (i.e., ε-correct) holds, if the Hamming metric between $\mathsf{Hash}(hk, L, W)$ and $\mathsf{ProjHash}(hk, L, W)$ is larger than $\varepsilon \cdot v$, then the probability of Hamming distance is negligible, i.e.,

$$\Pr[\mathsf{HD}(\mathsf{Hash}(hk, L, W), \mathsf{ProjHash}(hk, L, W)) > \varepsilon \cdot v] = \mathrm{negl}(\lambda).$$

- **Smoothness:** We say the property of the smoothness holds, if the following two distributions are statistical indistinguishable:

$$(1)\{(ph, h) \mid hk \leftarrow \mathsf{HashKG}(L), ph = \mathsf{ProjKG}(hk, L, W), \underline{h \leftarrow \mathsf{Hash}(hk, L, W)}\}.$$
$$(2)\{(ph, h) \mid hk \leftarrow \mathsf{HashKG}(L), ph = \mathsf{ProjKG}(hk, L, W), \underline{h \leftarrow \{0,1\}^v}\}.$$

Here, we stress that we call the approximate SPHF as SPHF if $\varepsilon = 0$, i.e., ε-correct. However, obtaining the 0-correct in lattice setting is not easy, thus our constructed $\mathsf{Reg{-}SPHF}$ and $\mathsf{MP{-}SPHF}$ are also approximated SPHFs.

2.3 The Bellare-Pointcheval-Rogaway Security Model

In this subsection, we follow the definition of Bellare, Pointcheval, and Rogaway [6] which is the follow-up work of [28–30].

Participants, Passwords, and Initialization. For any execution of the protocol, there is an initialization phase during which public parameters are established. We assume a fixed set U of protocol users. For every distinct U_1, $U_2 \in \mathsf{U}$, we assume that U_1 and U_2 share a password pw_{U_1,U_2}, (i.e., pw). Meanwhile, each pw_{U_1,U_2} is independently sampled from the password space $D(\lambda)$ according to the Zipf's law [17,18].

Execution of the Protocol. In reality, a protocol describes the behaviours of the user after receiving inputs from their environment. In the formal model, the adversary $\mathcal{A}$ will decide the inputs for the user, and each user is allowed to instantiate an unlimited number of instances and can run the protocol multiple times (possibly concurrently) with different partners. We denote instance i of user U as Π_U^i. Each instance may be used only once. The adversary is given oracle access to these different instances; furthermore, each instance maintains (local) state which is updated during the course of the experiment. In particular, each instance Π_U^i maintains local state that includes the following variables:

- sid_U^i, session id; pid_U^i, partner id; skey_U^i, session key id.
- acc_U^i, a boolean variable denoting acceptance at the end of the execution.
- term_U^i, a boolean variable denoting termination at the end of the execution.

Adversarial Model. The adversary $\mathcal{A}$ is allowed to fully control the external network, namely that he is able to do whatever one wants, such as he can **(1)** block, inject, modify, and delete messages; **(2)** request any session keys adaptively. Formally, we model how the adversary $\mathcal{A}$ interacts with various instances by the following oracles:

- $\mathsf{Send}(U_C, i, M)$. The oracle sends the message M to instance $\Pi^i_{U_C}$. Upon receiving the message from the oracle Send, the instance $\Pi^i_{U_C}$ then runs according to the protocol specification, updating state as the approach. We remark that, the output of $\Pi^i_{U_C}$ (i.e., the message sent by the instance) is given to $\mathcal{A}$.
- $\mathsf{Execute}(U_C, i, U_S, j)$. The oracle executes the protocol between instances $\Pi^i_{U_C}$ and instances $\Pi^j_{U_S}$. The outputs of the oracle is the protocol transcript, i.e., the ordered messages can be exchanged between the instances.
- $\mathsf{Reveal}(U_C, i)$. The oracle allows the adversary to learn session keys from previous and concurrent executions and outputs the session key skey^i_U. Meanwhile, erasures the improper session keys.
- $\mathsf{Test}(U_C, i)$. The oracle allows the adversary to query it only once and outputs a random bit b. If $b = 1$, then the adversary is obtained a session key skey^i_U. If $b = 0$, then the adversary is obtained a uniform session key. Lastly, the adversary guesses a random bit b'. If $b = b'$ then the adversary is successful.

Partnering. Let U_C, $U_S \in U$. Instances $\Pi^i_{U_C}$ and $\Pi^j_{U_S}$ are partnered if: (1) $\mathsf{sid}^i_U = \mathsf{sid}^i_{U_C} \neq$ NULL; and (2) $\mathsf{pid}^i_{U_C} = U_C$ and $\mathsf{pid}^j_{U_S} = U_S$.

Correctness. If the instance $\Pi^i_{U_C}$ and instance $\Pi^j_{U_S}$ are partnered then there exist $\mathsf{acc}^i_{U_C} = \mathsf{acc}^j_{U_S} =$ TRUE and $\mathsf{skey}^i_{U_C} = \mathsf{skey}^j_{U_S}$ and they both obtained the common session key.

Definition 5. *For all* PPT *adversaries $\mathcal{A}$ making at most $Q(\lambda)$ on-line guessing attacks, if it holds that $\mathbf{Adv}_{\mathcal{A},\Pi}(\lambda) \leq C' \cdot Q^{s'}(\lambda) + \text{negl}(\lambda)$, then the* PAKE *protocol Π is a secure protocol, where $s' \in$ [0.15, 0.30] and $C' \in$ [0.001, 0.1] are constant* CDF-Zipf *regression parameters depending on the password space $\mathcal{D}$ [17,18].*

Remark 2. In most existing PAKE studies (e.g., [7–9,11]) and other kinds of password-based protocols (e.g., two-factor authentication [31] and password authenticated keyword search [32]), passwords are assumed to follow a uniformly random distribution, and the real attacker's advantage $\mathbf{Adv}$ is thus formulated as $Q(\lambda)/|D| + \text{negl}(\lambda)$, where $|D|$ is the size of the password dictionary D, and $Q(\lambda)$ is the number of $\mathcal{A}$'s active on-line password guessing attempts (which is analogous to Q_{send} in [9], q_{send} in [33], n_{se} in [16] and q_s in [7,31,32]). Instead, we prefer the CDF-Zipf model [17,33], and the attacker $\mathcal{A}$'s advantage $\mathbf{Adv}$ can be formulated as $C' \cdot Q^{s'}(\lambda) + \text{negl}(\lambda)$ for the Zipf parameters C' and s'. Figure 1 shows that the traditional uniform-model based formulation $Q(\lambda)/|D| + \text{negl}(\lambda)$ always significantly underestimates the real attacker $\mathcal{A}$'s $\mathbf{Adv}$ ($\forall Q(\lambda) \in [1, |D|]$). Fortunately, the CDF-Zipf based formulation $C' \cdot Q^{s'}(\lambda) + \text{negl}(\lambda)$ well approximates $\mathcal{A}$'s advantage $\mathbf{Adv}$: $\forall Q(\lambda) \in [1, |D|]$, *the largest deviation* between $C' \cdot Q^{s'}(\lambda) + \text{negl}(\lambda)$ and

Adv is as low as 0.617%. This CDF-Zipf based formulation is also drastically more accurate than other occasionally used formulations like the Min-entropy model in [10] and Becerra et al.'s obscure one (see Eq. 1 in [34]) which undesirably defeats the advantage of the *quantitativeness* of provable security.

3 Reg−SPHF from the Regev Scheme

We now describe how to follow the Katz-Vaikuntanathan framework [13] to design a SPHF for the ciphertext of the Regev scheme [14].

It is well known that the Regev scheme is one of the most classical IND-CPA-secure scheme under the decisional LWE assumption. The others are the Gentry-Peikert-Vaikuntanathan (a.k.a., dual-Regev) scheme [35] and the Lindell-Peikert scheme [36]. In line with the principles of the SPHF of Katz-Vaikuntanathan (KV) construction [13], we adopt Regev scheme as the building block to design the SPHF, for simplicity, we abbreviate it to Reg−SPHF. We remark that the other lattice-based PKE schemes also can be used to design the SPHF which follows the framework of Katz-Vaikuntanathan.

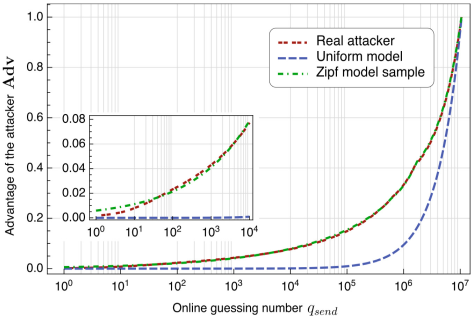

Fig. 1. Online guessing advantages **Adv** of the real attacker, the uniform-modeled attacker and our Zipf-modeled attacker (using 15.25 million 000Webhost passwords [17]).

- $hk \leftarrow$ Reg.HashKG(params): inputs a random vector $\mathbf{h} \leftarrow \mathbb{Z}_q^{n\times 1}$ and outputs the hashing key $hk := \mathbf{h} \in \mathbb{Z}_q^{m\times 1}$.
- $ph \leftarrow$ Reg.ProjKG(params, $hk = \mathbf{h}, pk = \mathbf{A}$): inputs $\mathbf{h}$ and the public key of IND-CPA-secure scheme $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, then outputs the projective hashing key $ph\!: = \mathbf{p}_{reg} = \mathbf{A}\cdot\mathbf{h} \in \mathbb{Z}_q^{n\times 1}$. We stress that, in Reg−SPHF setting, we only obtain the "approximate correctness".
- $h \leftarrow$ Reg.Hash($hk = \mathbf{h}, W := (c, \mathbf{m})$) :
  1. The algorithm inputs $\mathbf{h}$ and the word W, where the word W contains a ciphertext $c = \mathbf{c} \in \mathbb{Z}_q^{m\times 1}$ and the plaintext $\mathbf{m}$.
  2. The hash function works as follows:

$$\begin{aligned} h &= \mathsf{Hash}(hk = \mathbf{h}, W := (c, \mathbf{m})) \\ &= R\Big(\Big[\mathbf{c} - (\lfloor \frac{q}{2} \rfloor \cdot \mathbf{m})\Big]^T \cdot \mathbf{h}\Big) = R\Big(\big[\mathbf{r}^T \cdot \mathbf{A}\big] \cdot \mathbf{h}\Big) \\ &= R\Big((\mathbf{r}^T \cdot \mathbf{A}) \cdot \mathbf{h} \pmod q \in \mathbb{Z}_q\Big) \in \{0, 1\}. \end{aligned}$$

  3. Obtains $b := h \pmod 2 \in \{0, 1\}$, where h is a number in $[-(q-1)/2, \cdots, (q-1)/2]$ and outputs $b = 0$ if $h < 0$, otherwise, outputs $b = 1$.

- $p = \mathsf{Reg.ProjHash}(ph = \mathbf{p}_{reg}, W := (c, \mathbf{m}); w = \mathbf{s})$(Projection)
  1. The algorithm inputs $ph = \mathbf{p}_{reg} \in \mathbb{Z}_q^{n\times 1}$, the word W, and the witness $\mathbf{s} \in \mathbb{Z}_q^{n\times 1}$.
  2. The algorithm computes

$$\begin{aligned} p &= \mathsf{Reg.ProjHash}(ph = \mathbf{p}_{reg}, W := (c, \mathbf{m}); w = \mathbf{r}) \\ &= R\Big(\mathbf{r}^T \cdot \mathbf{p}_{reg}\Big) = R\Big(\mathbf{r}^T \cdot (\mathbf{A}\mathbf{h}) \pmod q\Big) \in \{0, 1\}. \end{aligned}$$

  3. Obtains the result of $b := p \pmod 2 \in \{0, 1\}$, and outputs $b = 0$ if $p < 0$, otherwise, outputs $b = 1$.

Below, we analyze the two important properties of Reg−SPHF.

Lemma 2. *The* Reg−SPHF *is a smooth projective hash function for the Regev scheme.*

Below we first prove the approximate correctness. Our goal is to prove $\mathsf{Reg.Hash}(hk = \mathbf{h}, W := (c, \mathbf{m})) = \mathsf{Reg.ProjHash}(ph = \mathbf{p}_{reg}, W := (c, \mathbf{m}); w = \mathbf{s})$ with probability greater than 1/2. The correctness of Reg−SPHF means that the relationship between the hash key hk and the word W from language L equals the relationship between the projective hash key ph and the witness w for any word in L. The smoothness of Reg−SPHF is that the hash value is independent of the projective hash key ph for any word in $X \setminus L$. Moreover, in order to discard the noise elements, we adopt the typical deterministic rounding function $R(x) = \lfloor 2x/q \rceil \pmod 2$ (a.k.a., the so-called square-signal) which was proposed by Katz and Vaikuntanathan [13].

- **Projective (or Correctness).** If the result of $\langle \mathbf{e}, \mathbf{h} \rangle$ is small for n, $m \geq n\sqrt{\log q}$, then the following equation holds

$$\begin{aligned} &R\Big(\mathsf{Reg.Hash}(hk = \mathbf{h}, W := (c, \mathbf{m}))\Big) \\ &= R\Big(\mathsf{Reg.ProjHash}(ph = \mathbf{p}_{reg}, W := (c, \mathbf{m}); w = \mathbf{s})\Big). \end{aligned}$$

Proof. In this paper, we follow the methodology of [13] and adopt the typical deterministic rounding function $R(x) = \lfloor 2x/q \rceil \pmod 2$ to calculate $\mathsf{Hash}(\cdot)$ and $\mathsf{ProjHash}(\cdot)$ respectively. Regarding the following two equations Eqs. (3.1) and (3.2),

$$\begin{aligned} &\Big(\mathsf{Reg.Hash}(hk = \mathbf{h}, W := (c, \mathbf{m}))\Big) \\ &= \Big(\big[\mathbf{r}^T \cdot \mathbf{A}\big] \cdot \mathbf{k}\Big) = \Big((\mathbf{r}^T \cdot \mathbf{A}) \cdot \mathbf{k} \pmod q\Big). \end{aligned} \tag{3.1}$$

$$\begin{aligned} &\Big(\mathsf{Reg.Hash}(hk = \mathbf{h}, W := (c, \mathbf{m}))\Big) \\ &= \Big(\mathbf{r}^T \cdot \mathbf{p}_{reg}\Big) = \Big(\mathbf{r}^T \cdot (\mathbf{A}\mathbf{k}) \pmod q\Big). \end{aligned} \tag{3.2}$$

we can easily find that the above two equations Eqs. (3.1) and (3.2) are equal, then we can utilize the rounding function $R(\cdot)$ and find that the output of $R\Big(\mathsf{Reg.Hash}(hk = \mathbf{h}, W := (c, \mathbf{m}))\Big)$ and $R\Big(\mathsf{Reg.ProjHash}(ph = \mathbf{p}_{reg}, W := (c, \mathbf{m}); w = \mathbf{s})\Big)$ are equal. □

- **Smoothness.** Below we prove the smoothness property of Reg−SPHF.

Proof. Consider the word $W := \big(c, \mathbf{m}\big) \notin L$, that means c is not an encryption of $\mathbf{m}$, under the public key $pk = \mathbf{A}$. Hence the above implies that the following two distributions have negligible statistical distance in λ,

$$(1).\{(ph, h) \mid \mathsf{HashKG}(L) \to \mathbf{h}, \mathsf{ProjKG}(hk, L, W) \to \mathbf{Ah}, \underline{\mathsf{Hash}(hk, L, W) = (\mathbf{r}^T\mathbf{A})\mathbf{h}}\}.$$

$$(2).\{(ph, h) \mid \mathsf{HashKG}(L) \to \mathbf{h}, \mathsf{ProjKG}(hk, L, W) \to \mathbf{Ah}, \underline{h \leftarrow \{0,1\}}\}.$$

We note that, $\mathsf{Hash}(hk, W) = (\mathbf{r}^T\mathbf{A})\mathbf{h}$ given $\mathsf{ProjKG}(hk, pk) = \mathbf{Ah}$. Due to $\mathbf{r}$ is witness vector, thus $\mathsf{ProjKG}(hk, pk)$ provides no information on $\mathsf{Hash}(hk, W)$ and $\mathsf{Hash}(hk, W)$ is uniformly distributed over $\{0, 1\}$, given $\mathsf{ProjKG}(hk, pk)$.

Hence, we conclude that the projective hash function is smooth. □

4 MP−SPHF from the Micciano-Peikert Scheme

The MP construction is IND-CCA1-secure, but the Katz-Vaikuntanathan framework requires the IND-CCA2-secure scheme along with the corresponding SPHF. Hence, we can use either strongly unforgeable one-time signature [37], or a message authentication code (MAC) and weak form of commitment [38] to obtain the IND-CCA2 security. Below, we first present the labeled IND-CCA1-secure scheme. For the sake of simplicity, we omit the generic transformation to IND-CCA2 at this stage which can be found in [37].

We first fix the label by $u \neq 0$ and obtain the labeled MP scheme, then we use it to develop an SPHF. Following the Katz-Vaikuntanathan (KV) construction, below we present an SPHF based on MP scheme, we call it MP−SPHF.

- $hk \leftarrow \mathsf{MP.HashKG}(\mathsf{params})$: samples $\mathbf{k} \leftarrow \mathbb{Z}_q^{n\times 1}$ and sets it as the hashing key $hk := \mathbf{k} \in \mathbb{Z}_q^{m\times 1}$.
- $ph \leftarrow \mathsf{MP.ProjKG}(\mathsf{params}, hk = \mathbf{k}, pk = \mathbf{A}_u)$: inputs the $\mathbf{k}$ and the public key of IND-CCA scheme $\mathbf{A}_u = [\bar{\mathbf{A}} \mid h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n\times m}$ with the fixed label u, then outputs the projective hashing key $ph := \mathbf{p} = \mathbf{A}_u \cdot \mathbf{k} \in \mathbb{Z}_q^{n\times 1}$.
- $h \leftarrow \mathsf{MP.Hash}(hk = \mathbf{k}, W := (c, \mathbf{m}))$:
  1. The algorithm inputs $\mathbf{k}$ and the word W, where W contains a ciphertext $c = (\mathsf{label}, \mathbf{c} \in \mathbb{Z}_q^{m\times 1})$ and the plaintext $\mathbf{m}$.
  2. The hash function works as follows:

$$\begin{aligned} h &= \mathsf{MP.Hash}(hk = \mathbf{k}, W := (c, \mathbf{m})) \\ &= R\Big([\mathbf{c} - (\mathbf{0} \mid \mathsf{encode}(\mathbf{m}))]^T \cdot \mathbf{k}\Big) = R\Big(\big[\mathbf{s}^T \cdot \mathbf{A}_u + \mathbf{e}^T\big] \cdot \mathbf{k}\Big) \\ &= R\Big((\mathbf{s}^T \cdot \mathbf{A}_u) \cdot \mathbf{k} + \mathbf{e}^T \cdot \mathbf{k} \pmod q \in \mathbb{Z}_q\Big) \in \{0, 1\}. \end{aligned}$$

We stress that $\mathbf{e}^T \cdot \mathbf{k}$ is the noise element $\mathbf{e}^T \cdot \mathbf{k}$ and bounded by $|\mathbf{e}^T\mathbf{k}| \leq \|\mathbf{e}^T\| \cdot \|\mathbf{k}\| \leq (r\sqrt{mn}) \cdot (\alpha q\sqrt{mn}) < \varepsilon/2 \cdot q/4$.

3. Outputs $b := h \pmod 2 \in \{0, 1\}$, where h is a number in $[-(q-1)/2, \cdots, (q-1)/2]$ and the algorithm outputs $b = 0$ if $h < 0$, otherwise, outputs $b = 1$.

- $p = \mathsf{MP.ProjHash}(ph = \mathbf{p}, W := (c, \mathbf{m}); w = \mathbf{s})$
  1. It inputs $ph = \mathbf{p} \in \mathbb{Z}_q^{n\times 1}$, the word W, and the witness $\mathbf{s} \in \mathbb{Z}_q^{n\times 1}$.
  2. The algorithm computes and outputs

$$\begin{aligned} p &= \mathsf{ProjHash}(ph = \mathbf{p}, W := (c, \mathbf{m}); w = \mathbf{s}) \\ &= R\Big(\mathbf{s}^T \cdot \mathbf{p}\Big) = R\Big(\mathbf{s}^T \cdot (\mathbf{A}_u\mathbf{k}) \pmod q\Big) \in \{0, 1\}. \end{aligned}$$

  3. Obtains $b := p \pmod 2 \in \{0, 1\}$, and outputs $b = 0$ if $p < 0$, otherwise, outputs $b = 1$.

Lemma 3. *The* MP–SPHF *is a smooth projective hash function for the* MP *scheme.*

Below we first prove our scheme achieves approximate correctness. Similar to the projective property of Reg–SPHF, our goal is to prove $\mathsf{MP.Hash}(hk = \mathbf{k}, W := (c, \mathbf{m})) = \mathsf{MP.ProjHash}(ph = \mathbf{p}, W := (c, \mathbf{m}); w = \mathbf{s})$ with probability greater than 1/2. Moreover, we still use the rounding function $R(x) = \lfloor 2x/q \rceil \pmod 2$ to discard the noise elements.

- **Projective (or Approximate Correctness).** If the result of $\langle \mathbf{e}, \mathbf{k}\rangle$ is small for n, $m \geq n\sqrt{\log q}$, then the following equation holds

$$R\Big(\mathsf{Hash}(hk = \mathbf{k}, W := (c, \mathbf{m}))\Big) = R\Big(\mathsf{ProjHash}(ph = \mathbf{p}, W := (c, \mathbf{m}); w = \mathbf{s})\Big).$$

Proof. In this paper, we adopt the typical deterministic rounding function $R(x) = \lfloor 2x/q \rceil \pmod 2$ and follow the methodology of [13] to round the outputs of $\mathsf{Hash}(\cdot)$ and $\mathsf{ProjHash}(\cdot)$ respectively. Regarding the following two equations,

$$\begin{aligned} \Big(\mathsf{Hash}(hk = \mathbf{k}, W := (c, \mathbf{m}))\Big) &= \Big(\big[\mathbf{s}^T \cdot \mathbf{A}_u + \mathbf{e}^T\big] \cdot \mathbf{k}\Big) \\ &= \Big((\mathbf{s}^T \cdot \mathbf{A}_u) \cdot \mathbf{k} + \mathbf{e}^T \cdot \mathbf{k} \pmod q\Big) \quad (4.1) \end{aligned}$$

$$\begin{aligned} \Big(\mathsf{ProjHash}(ph = \mathbf{p}, W := (c, \mathbf{m}); w = \mathbf{s})\Big) &= \Big(\mathbf{s}^T \cdot \mathbf{p}\Big) \\ &= \Big(\mathbf{s}^T \cdot (\mathbf{A}_u\mathbf{k}) \pmod q\Big) \quad (4.2) \end{aligned}$$

We first consider the equation Eq. (4.1) and the Definition 1, the result of $R(h)$ can be viewed as a number in $[-\frac{(q-1)}{2}, \cdots, \frac{(q-1)}{2}]$, and we can obtain the result $b \in \{0, 1\}$. Moreover, the noise element $\mathbf{e}^T \cdot \mathbf{k}$ is bounded by $|\mathbf{e}^T\mathbf{k}| \le \|\mathbf{e}^T\| \cdot \|\mathbf{k}\| \le (r\sqrt{mn}) \cdot (\alpha q\sqrt{mn}) < \varepsilon/2 \cdot q/4$. Hence, the result of $R(\mathbf{e}^T\mathbf{k})$ is identical with 0. Thus, there exists

$$b = \begin{cases} 0, \text{ if } R(h) < 0; \\ 1, \text{ if } R(h) > 0. \end{cases}$$

Consider the equation Eq. (4.2) and the Definition 1, we have that

$$b = \begin{cases} 0, \text{ if } R\big(\mathbf{s}^T \cdot (\mathbf{A}_u\mathbf{k})\big) < 0; \\ 1, \text{ if } R\big(\mathbf{s}^T \cdot (\mathbf{A}_u\mathbf{k})\big) > 0. \end{cases}$$

Obliviously, the above two results are equal since the size of the noise $\|\mathbf{e}^T\mathbf{k}\|$ is bounded by $q\varepsilon/8 < q/4$. □

- **Smoothness.** Below we prove the smoothness property of MP−SPHF.

Proof. Consider the word $W := \big(c, \mathbf{m}\big) \notin L$, that means c is not an encryption of $\mathbf{m}$, under the public key $pk = \mathbf{A}_u$. Hence the above implies that the following two distributions have negligible statistical distance in λ,

$$\begin{aligned}(1).\{(ph, h) \mid \mathsf{HashKG}(L) \to \mathbf{k}, \mathsf{ProjKG}(hk, L, W) \to \mathbf{A}_u\mathbf{k},\\ \underline{\mathsf{Hash}(hk, L, W) = (\mathbf{s}^T\mathbf{A}_u + \mathbf{e}^T)\mathbf{k}}\}.\\ (2).\{(ph, h) \mid \mathsf{HashKG}(L) \to \mathbf{k}, \mathsf{ProjKG}(hk, L, W) \to \mathbf{A}_u\mathbf{k},\\ \underline{h \leftarrow \{0, 1\}}\}.\end{aligned}$$

We note that, $\mathsf{MP.Hash}(hk, W) = (\mathbf{s}^T\mathbf{A}_u + \mathbf{e}^T)\mathbf{k}$ given $\mathsf{MP.ProjKG}(hk, pk) = \mathbf{A}_u\mathbf{k}$. Due to $\mathbf{s}$ is witness vector, thus $\mathsf{MP.ProjKG}(hk, pk)$ provides no information on $\mathsf{MP.Hash}(hk, W)$ and $\mathsf{MP.Hash}(hk, W)$ is uniformly distributed over $\{0, 1\}$, given $\mathsf{MP.ProjKG}(hk, pk)$.

Hence, we conclude that the projective hash function is smooth. □

5 Two-Round PAKE Protocol over Lattices

At ASIACRYPT'17, Zhang and Yu proposed a lattice-based two-round PAKE protocol [11] using simulation-sound NIZK in the random oracle model. At PKC'15, Abdalla et al. [10] proposed the new cryptographic primitive "IND-PCA-secure PKE" to design two-round PAKE protocols without NIZK.[3] However, this PAKE builds on the DDH assumption and cannot prevent quantum attacks. To our knowledge, it remains an open question to construct a two-round PAKE protocol under the LWE setting without NIZK in the random oracle model.

[3] They improved the Gennaro-Lindell framework to reduce the round number to two.

Common Reference String(public key):
$pk_{\text{cca}} \leftarrow \mathsf{MP.KeyGen}(\cdot)$ and $pk_{\text{cpa}} \leftarrow \mathsf{Reg.KeyGen}(\cdot)$.

Client C with a low-entropy password π		Server S with a low-entropy password π
1). Invokes $hk_C^{\text{cca}} \leftarrow \mathsf{MP.HashKG}(\mathsf{params})$ 2). $ph_C^{\text{cca}} \leftarrow \mathsf{MP.ProjKG}(\mathsf{params}, hk_C^{\text{CCA}}, pk)$ 3). Sets label $\ell = C \parallel S \parallel ph_C^{\text{cca}}$ 4). $\mathbf{c}_C^{\text{cca}} \leftarrow \mathsf{MP.Enc}(\mathsf{params}, pk_{\text{cca}}, \pi; r \leftarrow \mathbb{Z}_p)$		
	$\xrightarrow{\mathbf{c}_C^{\text{cca}}, ph_C^{\text{cca}}}$	5). Receives ph_C^{cca} and $\mathbf{c}_C^{\text{cca}}$. 6). Sets label $\ell = C \parallel S \parallel ph_C^{\text{cca}}$ 7). $\mathbf{c}_S \leftarrow \mathsf{Reg.Enc}(\mathsf{params}, pk_{\text{cpa}}, \pi; r \leftarrow \mathbb{Z}_p)$ 8). Invokes $hk_S^{\text{cpa}} \leftarrow \mathsf{Reg.HashKG}(\mathsf{params}, \ell)$ 9). $ph_S^{\text{cpa}} \leftarrow \mathsf{Reg.ProjKG}(\mathsf{params}, \ell, hk_S^{\text{cpa}}, pk)$
10). Receives ph_S^{cpa} and $\mathbf{c}_S^{\text{cpa}}$.	$\xleftarrow{\mathbf{c}_S^{\text{cpa}}, ph_S^{\text{cpa}}}$	
11). $\mathsf{MP.ProjHash}(ph_S^{\text{cpa}}, W_C^{\text{cca}} = (\mathbf{c}_C^{\text{cca}}, \pi); w_C = r)$. Obtains p_S 12). $\mathsf{MP.Hash}(hk_C^{\text{cca}}, W_S^{\text{cpa}} := (\mathbf{c}_S^{\text{cpa}}, \pi))$ Outputs h_C Outputs $\mathsf{key} := (p_S \cdot h_C)$ Erases everything except π and key		11). $\mathsf{Reg.ProjHash}(ph_C^{\text{cca}}, W_S^{\text{cpa}} = (\mathbf{c}_S^{\text{cpa}}, \pi); w_S = \bar{r})$. Obtains p_C 12). $\mathsf{Reg.Hash}(hk_S^{\text{cpa}}, W_C^{\text{cca}} := (\mathbf{c}_C^{\text{cca}}, \pi))$ Outputs h_S Outputs $\mathsf{key} := (p_C \cdot h_S)$ Erases everything except π and key

Fig. 2. A sketch of our two-round lattice-based PAKE protocol.

In this section, armed with the above MP−SPHF and Reg−SPHF, we follow the framework of Abdalla-Benhamouda-Pointcheval and design a new lattice-based two-round PAKE protocol. Here we provide a high view of our protocol:

- **First round.** The client runs the Miccianio-Peikert scheme with an associated MP−SPHF, then sends the first flow message (i.e., the ciphertext of Miccianio-Peikert scheme along with the corresponding signature and the projective hash key of MP−SPHF) to the server.
- **Second round.** Upon receiving the information from the client, the server first checks the legitimacy of the signature, then runs the Regev scheme with an associated Reg−SPHF. Subsequently, the server returns the second flow message (i.e., the ciphertext of the Regev scheme and the projective hash key of Reg−SPHF) to the client.
- **Local computation.** After receiving the messages from the other party, both parties perform the calculation locally on the received message and the local message. Concretely, the client generates the common session key $= (p_S \cdot h_C)$ and the server generates the common session key $= (p_C \cdot h_S)$.

Figure 2 illustrates the detailed description of the two-round PAKE protocol over lattices. Since every IND-CCA2-secure encryption is also IND-PCA-secure, we follow the road-map of [10] and achieve the expected two-round PAKE. Importantly, we do not depend on a simulation-sound NIZK [11] and the detailed explanation can be found in [39]. In this case, we can omit the issue of the gap between correctness and smoothness because the proof of the resulting two-round PAKE works exactly as in [10]. The details are provided in Appendix of [39].

Moreover, as far as we know, if the label of the label-IND-CCA2 encryption scheme is fixed in advance to some public constant, then the resulting scheme is IND-CPA. Hence, we can follow the generic transformation of [39] to convert a label-IND-CCA2 encryption scheme with message space $\{0,1\}$ and label space $\{0,1\}^{\lambda}$ into a IND-CCA2 encryption scheme with message space $\{0,1\}^{v}$ (for some v polynomial in λ) and label space $\{0,1\}^{*}$ according to [37]. In a concrete way, a strongly unforgeable one-time signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ was introduced to achieve the above goal. The client invokes the algorithm Sign and takes as input the ciphertext $\mathbf{c}_C^{\text{cca}}$. Subsequently, the server will verify the signature of the ciphertext using the algorithm Ver. For the sake of explanation, we omit this transformation step in the above protocol.

5.1 Correctness Analysis

In this subsection, we analyze the correctness of our PAKE protocol.

Lemma 4. *If the two communication parties obtained the same common session key, then the correctness holds.*

Proof. For the client side, the client C obtained the session key as follows

$$\begin{aligned}\mathsf{skey}_C &= p'_S \cdot (p_S \cdot h_C) \cdot h'_C\\ &= R\Big(\mathbf{s}^T(\frac{\mathbf{A}_u}{\mathbf{A}})\mathbf{A}\mathbf{h}\Big) \cdot R\Big(\mathbf{s}^T\mathbf{A}\mathbf{h}\Big) \cdot R\Big(\mathbf{r}^T\mathbf{A}\mathbf{k}\Big) \cdot R\Big(\mathbf{r}^T\mathbf{A}(\frac{\mathbf{A}_u}{\mathbf{A}}\mathbf{k})\Big)\\ &= R\Big(\mathbf{s}^T\mathbf{A}_u\mathbf{h}\Big) \cdot R\Big(\mathbf{s}^T\mathbf{A}\mathbf{h}\Big) \cdot R\Big(\mathbf{r}^T\mathbf{A}\mathbf{k}\Big) \cdot R\Big(\mathbf{r}^T\mathbf{A}_u\mathbf{k}\Big)\end{aligned}$$

Meanwhile, for the server side, the server S obtained the session key as follows

$$\begin{aligned}\mathsf{skey}_S &= p'_C \cdot (p_C \cdot h_S) \cdot h'_S\\ &= R\Big(\mathbf{r}^T(\frac{\mathbf{A}}{\mathbf{A}_u})\mathbf{A}_u\mathbf{k}\Big) \cdot R\Big(\mathbf{r}^T\mathbf{A}_u\mathbf{k}\Big)\\ &\qquad \cdot R\Big((\mathbf{s}^T\mathbf{A}_u + \mathbf{e}^T)\mathbf{h}\Big) \cdot R\Big((\mathbf{s}^T\mathbf{A}_u + \mathbf{e}^T)(\frac{\mathbf{A}_u}{\mathbf{A}})\mathbf{h}\Big)\\ &= R\Big(\mathbf{r}^T\mathbf{A}\mathbf{k}\Big) \cdot R\Big(\mathbf{r}^T\mathbf{A}_u\mathbf{k}\Big) \cdot R\Big((\mathbf{s}^T\mathbf{A}_u + \mathbf{e}^T)\mathbf{h}\Big) \cdot R\Big((\mathbf{s}^T\mathbf{A} + \mathbf{e}^T(\frac{\mathbf{A}_u}{\mathbf{A}}))\mathbf{h}\Big)\end{aligned}$$

In order to meet the requirements of the Lemma 3 and the typical deterministic rounding function $R(x) = \lfloor 2x/q \rceil \pmod 2$ from [13], we consider $max\{\|\mathbf{e}^T\mathbf{h}\|, \|\mathbf{e}^T(\frac{\mathbf{A}_u}{\mathbf{A}})\mathbf{h}\|\} \leq q/4$ for the bound of $\|\mathbf{e}^T\mathbf{h}\| \leq mB$ and $\|\mathbf{e}^T(\frac{\mathbf{A}_u}{\mathbf{A}})\mathbf{h}\| \leq O(mB)$.[4] In this setting, the output of the typical deterministic rounding function $R\Big((\mathbf{s}^T\mathbf{A}_u + \mathbf{e}^T)\mathbf{h}\Big) = R\Big((\mathbf{s}^T\mathbf{A}_u)\mathbf{h}\Big)$ and the output of $R\Big((\mathbf{s}^T\mathbf{A} + \mathbf{e}^T(\frac{\mathbf{A}_u}{\mathbf{A}}))\mathbf{h}\Big) = R\Big((\mathbf{s}^T\mathbf{A}))\mathbf{h}\Big)$. Hence, we have that $\mathsf{skey}_C = \mathsf{skey}_S$. □

[4] We use big-O notation to asymptotically bound the growth of a running time to within constant factors.

5.2 Security Analysis

Theorem 1. *The two-round lattice-based* PAKE *protocol from Fig.* 2 *is secure in the BPR model, under the* LWE *assumption.*

Proof. Below we provide the sketched proof because of the space limitation. Roughly speaking, this proof follows the schemes given in Benhamuda et al. [5,7,40], we only check that our primitives ($\mathsf{Reg}-\mathsf{SPHF}$ and $\mathsf{MP}-\mathsf{SPHF}$) fulfill the same properties in order to be able to modularly apply the proof given in [40].

Experiment $\mathsf{Expt}.0$. This is the real attack game, the advantage was denoted by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.0}(\lambda) = \varepsilon$. Then, we incrementally modify the simulate procession to make the trivial attacks possible. In this experiment, all of the private input values of the honest players can be used by the simulator. Following [7,15], there exist three types of Send queries:

- $\mathsf{Send}_0(C, i, S)$-query. In this setting, the adversary asks the instance Π_C^i to initiate an execution with an instance of S. Then S answers the query by a flow and returns it to C.
- $\mathsf{Send}_1(S, j, \mathsf{msg})$-query. The adversary sends the first flow message msg to the instance Π_S^j. The oracle defines his own session key and returns second flow which answered back by the instance Π_S^j.
- $\mathsf{Send}_2(C, i, \mathsf{msg})$-query. The adversary sends the second flow message msg to the instance Π_C^i. The oracle gives no answer back, but defines his own session key, for possible later Reveal or Test queries.

We remark that, if there exists $\pi_C = \pi_{C,S}$, then the client C and the server S are compatible. Actually, the definition of "compatibility" was defined by Katz et al. [30,41] which means even if the password was changed during the execution of the protocol, the changed password does not have an effect on the execution.

Experiment $\mathsf{Expt}.1$. We first modify the way how to deal with the $\mathsf{Execute}$-queries. Concretely, in response to a query $\mathsf{Execute}(U_C, i, U_S, j)$, we use the encryption of dummy passwords π_C^0 and $\pi_{C,S}^0$ from Zipf distribution to replace the ciphertext $\mathbf{c}_C$ and $\mathbf{c}_S$. Apparently, the fake passwords π_C^0 and π_S^0 are not in language L and the random elements are not used in the generation of the fake ciphertext. This is indistinguishable from $\mathsf{Expt}.0$ under the IND-CPA property of the encryption scheme. Moreover, due to the hash key and projective key are known by the players, hence they can compute the common session key

$$\begin{aligned}\mathsf{key} &= \mathsf{Hash}(hk_C, W_S := (\mathbf{c}_S, \pi)) \cdot \mathsf{ProjHash}(ph_S, W_C = (\mathbf{c}_C, \pi); w_C = r)\\ &= \mathsf{Hash}(hk_S, W_C := (\mathbf{c}_C, \pi)) \cdot \mathsf{ProjHash}(ph_C, W_S = (\mathbf{c}_S, \pi); w_S = \bar{r})\\ &= \mathsf{key}\end{aligned}$$

Since we could have first modified the way to compute key, which has no impact at all from the soundness of the SPHF, the unique difference comes from the different ciphertexts. Actually, this is indistinguishable property of the probabilistic encryption scheme, for each $\mathsf{Execute}$-query.

For future convenience, we define this experiment as Event Ev_0 whose probability is computed in $\mathsf{Expt}.8$. Thus we can obtain $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.1}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.0}(\lambda)| \leq \text{negl}(\lambda)$ by using a series of hybrid hops.

Experiment $\mathsf{Expt}.2$. In this experiment, again, we modify the way of the $\mathsf{Execute}$-queries response. We sample a random value from uniform distribution, then use it to replace the common session key. In this setting, the "password" is not satisfied, the indistinguishability property is guaranteed by the smoothness, i.e., $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.2}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.1}(\lambda)| \leq \text{negl}(\lambda)$.

Experiment $\mathsf{Expt}.3$. This experiment is identical to the $\mathsf{Expt}.2$ except that we change the way how to deal with the Send_1-queries. Concretely, in this experiment, we use a Miccianio-Peikert decryption oracle (or alternatively knowing the decryption key of Miccianio-Peikert scheme) to decrypt the "unused" received message $\mathsf{msg} = (ph_C, \mathbf{c}_C)$, three cases can appear:

1. If the msg has been altered (even generated) by the simulator in the name of the client C, then one can obtain the word W by checking whether the ciphertext $\mathbf{c}_C$ contains the expected password $\pi_{S,C}$ or not, along with the label $\ell = C||S||ph_C$. Then, there exist two cases:
   (a) If they are correct $W \in L$ (or the expected password is encrypted) and consistent with the receiver's values, then one can assert that the adversary $\mathcal{A}$ succeeds (i.e., $b' = b$) and terminates the game.
   (b) If they are not both correct and consistent with the receiver's values, then one chooses key at random.
2. If the msg is used previously (or, it is a replay of a previous flow sent by the simulator which in the name of the client C), then, in particular, the simulator knows the hash key and obtains the projective key, then the simulator can compute the common session key by using the hash key and the projective key. Namely $\mathsf{key} = \mathsf{Hash}(hk_C, W_S := (\mathbf{c}_S, \pi)) \cdot \mathsf{ProjHash}(ph_S, W_C = (\mathbf{c}_C, \pi); w_C = r)$, where we stress that $\mathbf{c}_S$ is not generated by using the randomness, which is similar to $\mathsf{Expt}.2$.

For future convenience, we define the first case (1a) as Event Ev_1 whose probability is computed in $\mathsf{Expt}.6$. We note that the change of the case (1a) can only increase the advantage of $\mathcal{A}$. Actually, the second change in the case (1b) only increases the advantage of the adversary by a negligible term due to it is indistinguishable under the adaptive-smoothness. Meanwhile, the third change in the case (2) does not affect the way the key is computed, so finally $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.3}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.2}(\lambda)| \leq \text{negl}(\lambda)$.

Experiment $\mathsf{Expt}.4$. This experiment is identical to the $\mathsf{Expt}.3$ except that we change the way how to deal with the Send_2-queries. Concretely, in this experiment, the simulator can query a Regev decryption oracle (or alternatively knowing the decryption key of the Regev scheme), namely that the simulator in the name of the server instance $\Pi_{U_S}^{j}$ sends the second flow $\mathsf{msg} = (ph_S, \mathbf{c}_S)$ to the client instance $\Pi_{U_C}^{i}$. Three cases can appear:

1. If the msg has been altered (even generated) by the simulator in the name of the server S, in order to response to first flow message $\mathsf{msg} = (ph_C, \mathbf{c}_C)$ that sent by the client instance $\Pi^i_{U_C}$, then one can obtain the word W by checking whether the ciphertext $\mathbf{c}_C$ contains the expected password $\pi_{S,C}$ or not, along with the label $\ell = C||S||ph_C$. Then, there exist two cases:
   (a) If they are correct $W \in L$ (or the expected password is encrypted) and consistent with the receiver's values, then one can assert that the adversary $\mathcal{A}$ succeeds (i.e., $b' = b$) and terminates the simulation.
   (b) If they are not both correct and consistent with the receiver's values, then one chooses key at random.
2. After receiving the first flow $\mathsf{msg} = (ph_C, \mathbf{c}_C)$, if the msg is used previously (or, it is a replay of a previous flow sent by the simulator which in the name of the client ${\Pi^j_{U_S}}'$), then, $\Pi^i_{U_C}$ and ${\Pi^j_{U_S}}'$ are partners. In particular,
   (a) If S and C are compatible, then the simulator knows the hash key and obtains the projective key, then the simulator can compute the common session key by using the hash key and the projective key. Namely $\mathsf{key} = \mathsf{Hash}(hk_C, W_S := (\mathbf{c}_S, \pi)) \cdot \mathsf{ProjHash}(ph_S, W_C = (\mathbf{c}_C, \pi); w_C = r)$, where we stress that $\mathbf{c}_S$ is not generated by using the randomness, which is similar to $\mathsf{Expt}.2$.
   (b) Otherwise, we choose a random common session key.

For future convenience, we define the first case (1a) as Event Ev_2 whose probability is computed in $\mathsf{Expt}.8$. We note that the change of the case (1a) can only increase the advantage of $\mathcal{A}$. Actually, the second change in the case (1b) only increases the advantage of the adversary by a negligible term due to it is indistinguishable under the adaptive-smoothness property. Meanwhile, the third change in the case (2a) does not affect the way the key is computed, so finally $|\mathsf{Adv}^{\mathsf{Expt}.4}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Expt}.3}_{\mathcal{A}}(\lambda)| \leq \mathrm{negl}(\lambda)$.

Experiment $\mathsf{Expt}.5$. We change the way of the Send_1-queries response. Now two cases will appear after a "used" message $\mathsf{msg} = (ph_C, \mathbf{c}_C)$ is sent.

- If there exists an instance $\Pi^i_{U_C}$ of U_C partnered with an instance $\Pi^j_{U_S}$ of U_S, then set $\mathsf{key} = \mathsf{skey}^i_C = \mathsf{skey}^j_S$.
- Otherwise, one chooses key at random.

Note that, in the first case, due to the "used" message is a reply of a previous flow, thus the common session key remains identical. In the second case, Due to the adaptive-smoothness [7,15], even if when hashing keys and ciphertexts are re-used, all the hash values are random looking. Hence, the indistinguishable holds and there exists $|\mathsf{Adv}^{\mathsf{Expt}.5}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Expt}.4}_{\mathcal{A}}(\lambda)| \leq \mathrm{negl}(\lambda)$.

Experiment $\mathsf{Expt}.6$. We change the way the Send_1-queries respond. Now two cases will appear after a "used" message $\mathsf{msg} = (ph_S, \mathbf{c}_S)$ is send.

- If there exists an instance Π^j_S of U_S partnered with an instance Π^i_C of U_C, then set $\mathsf{key} = \mathsf{skey}^i_C = \mathsf{skey}^j_S$.
- Otherwise, one chooses key at random.

Similar to the $\mathsf{Expt}.5$, the indistinguishability holds and there exists $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.6}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.5}(\lambda)| \leq \mathrm{negl}(\lambda)$.

Experiment $\mathsf{Expt}.7$. We now modify the way how to deal with the Send_0-queries. We remark that, in previous experiments, we don't need to know the random $\mathbf{r}_C$ (a.k.a., witness $w_C = \mathbf{r}_C$) which can be used to obtain the ciphertext $\mathbf{c}_C$. In this experiment, instead of encrypting the correct and real passwords, one encrypts the fake π_0 which does as in $\mathsf{Expt}.1$ for $\mathsf{Execute}$-queries to answer the query $\mathsf{Send}_0(C, i, S)$. Due to it is necessary to simulate the decryption of the Send_1-queries, then the indistinguishability holds for IND-CCA-secure Miccianio-Peikert PKE scheme. Therefore, we have $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.7}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.6}(\lambda)| \leq \mathrm{negl}(\lambda)$.

Experiment $\mathsf{Expt}.8$. This experiment is identical to the $\mathsf{Expt}.5$ except that we adopt the dummy private inputs for the hash key hk and the projective key ph. Concretely, hk and ph do not depend upon the word W, the distributions of these keys are independent of the auxiliary private inputs, hence there exists $|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.8}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.7}(\lambda)| \leq \mathrm{negl}(\lambda)$. Putting them together, we can obtain

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.8}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.0}(\lambda) - \mathrm{negl}(\lambda) = \varepsilon - \mathrm{negl}(\lambda).$$

Actually, the $\mathsf{Expt}.8$ is only used for declaring whether $\mathcal{A}$ won the event Ev or not. So the advantage is exactly: $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Expt}.6}(\lambda) = \Pr[Ev]$. Therefore, we have

$$\varepsilon \leq \Pr[Ev_0] + \Pr[Ev_1] + \Pr[Ev_2] + \mathrm{negl}(\lambda).$$

As mentioned earlier, **(1).** the event Ev_0 means that $\mathcal{A}$ wins the $\mathsf{Expt}.1$ during the $\mathsf{Execute}(\cdot)$ queries. $\Pr[Ev_0] = \Pr[\exists k_0 \in Q_e^{s'}(\lambda) : \pi_{C,S}(k_0) = \pi_C(k), W \in L]$, where $k_0 \in Q_e^{s'}(\lambda)$ is the index of the recepient of k_0-th $\mathsf{Execute}$-query and $Q_e^{s'}(\lambda)$ is the number of the $\mathsf{Execute}$-queries. **(2).** The event Ev_1 means that the adversary has encrypted (π) that are correct $(W \in L)$ and consistent with the receiver's values $(\pi_{C,S} = \pi)$. Since the random values (or witness) for the honest players are never used during the simulation, we can assume we choose them at the very end only to check whether event Ev_1 happened:

$$\Pr[Ev_1] = \Pr[\exists k_1 \in Q_{s1}^{s'}(\lambda) : \pi_{C,S}(k_1) = \pi_C(k), W \in L],$$

where $k_1 \in Q_{s1}^{s'}(\lambda)$ is the index of the recepient of k_1-th Send_1-query and $Q_{s1}^{s'}(\lambda)$ is the number of the Send_1-queries. Similarly, **(3).** the event Ev_3 means that the adversary has encrypted (π) that are correct $(W \in L)$ and consistent with the receiver's values $(\pi_{C,S} = \pi)$. Since the random values (or witness) for the honest players are never used during the simulation, we can assume we choose them at the very end only to check whether event Ev_2 happened:

$$\Pr[Ev_2] = \Pr[\exists k_2 \in Q_{s3}^{s'}(\lambda) : \pi_{C,S}(k_2) = \pi_C(k), W \in L],$$

where $k_2 \in Q_{s2}^{s'}(\lambda)$ is the index of the recepient of k_2-th Send_1-query and $Q_{s2}^{s'}(\lambda)$ is the number of the Send_2-queries.

In other words, it first has to guess the private values, and then once it has guessed them, it has to find a word in the language, hence, there exists

$$\Pr[Ev_1] + \Pr[Ev_2] + \Pr[Ev_3] \leq C' \cdot \left(Q_e^{s'}(\lambda) + Q_{s1}^{s'}(\lambda) + Q_{s2}^{s'}(\lambda) \right) \times Succ^L(\lambda),$$

where $Succ^L(\lambda)$ is the best success an adversary can get in finding a word in a language L. Then, by combining all the inequalities, one can get

$$\varepsilon \leq C' \cdot \left(Q_e^{s'}(\lambda) + Q_{s1}^{s'}(\lambda) + Q_{s2}^{s'}(\lambda) \right) \times Succ^L(t) + \mathrm{negl}(\lambda).$$

This completes the proof. □

Table 1. A comparison of related PAKE protocols under LWE assumption.

Scheme	SPHF	Rounds	Client&Server	Framework	Building blocks
Katz-Vaikuntanathan[13]	KV[13]	3	CCA & CCA	KV[13]	Peikert Enc[25]
Zhang-Yu[11]	GL[5]	2	CCA & CCA	GL[5]	Peikert Enc[25]
Bonhamouda et al.[39]	KV[13]	1(2-flow)	CCA & CCA	KV[15]	Mic-Pei Enc[12]
Our scheme	KV[13]	2	CCA & CPA	ABP[10]	Mic-Pei Enc[12] & Regev enc[14]

KV−SPHF implies that adaptive smoothness and the *ph* dependents on W.
GL−SPHF implies that non-adaptive smoothness and the *ph* independent on W.

6 Conclusion

In this paper, we first design two types of new lattice-based SPHFs (i.e., the IND-CCA-secure MP−SPHF at client side and the IND-PCA-secure Reg−SPHF at server side) by following the KV−SPHF methodology. Then, we construct the first lattice-based two-round PAKE protocol via Reg−SPHF and MP−SPHF, avoiding using the simulation-sound NIZK in random oracle model as compared to the foremost two-round PAKE protocol by Zhang and Yu at ASIACRYPT'17 [11]. Besides, as shown in Table 1, our protocol builds on weaker security assumptions than those state-of-the-art PAKE protocols [11,13,39] from the LWE assumption.

Acknowledgments. The authors would like to thank the anonymous reviewers for their helpful advice and comments. This work was supported by the National Natural Science Foundation of China (No. 61802006 and No. 61802214).

References

1. Bellovin, S.M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: Proceedings of the IEEE S&P 1992, pp. 72–84 (1992)
2. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_29
3. Hao, F., Ryan, P.: J-PAKE: authenticated key exchange without PKI. In: Gavrilova, M.L., Tan, C.J.K., Moreno, E.D. (eds.) Part II. LNCS, vol. 6480, pp. 192–206. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17697-5_10
4. Jarecki, S., Krawczyk, H., Xu, J.: OPAQUE: an asymmetric PAKE protocol secure against pre-computation attacks. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 456–486. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_15
5. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_33
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_11
7. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 449–475. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_25
8. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: Proceedings of the ACM CCS 2010, pp. 516–525 (2010)
9. Jiang, S., Gong, G.: Password based key exchange with mutual authentication. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 267–279. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30564-4_19
10. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 332–352. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_15
11. Zhang, J., Yu, Y.: Two-round PAKE from approximate SPH and instantiations from lattices. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 37–67. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70700-6_2
12. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
13. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 636–652. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_37
14. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of ACM STOC 2005, pp. 84–93 (2005)

15. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_18
16. Abdalla, M., Benhamouda, F., MacKenzie, P.: Security of the J-PAKE password-authenticated key exchange protocol. In: Proceedings of IEEE S&P 2015, pp. 571–587 (2015)
17. Wang, D., Wang, P.: On the implications of Zipf's law in passwords. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016, Part I. LNCS, vol. 9878, pp. 111–131. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_6
18. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf's law in passwords. IEEE Trans. Inform. Foren. Secur. **12**(11), 2776–2791 (2017)
19. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_22
20. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_24
21. Gentry, C., MacKenzie, P., Ramzan, Z.: A method for making password-based key exchange resilient to server compromise. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 142–159. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_9
22. Dupont, P.-A., Hesse, J., Pointcheval, D., Reyzin, L., Yakoubov, S.: Fuzzy password-authenticated key exchange. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 393–424. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_13
23. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
24. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_50
25. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Proceedings of ACM STOC 2009, pp. 333–342 (2009)
26. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Proceedings of ACM STOC 2008, pp. 187–196 (2008)
27. Li, Z., Ma, C., Wang, D.: Leakage resilient leveled FHE on multiple bit message. IEEE Trans. Big Data. https://doi.org/10.1109/TBDATA.2017.2726554
28. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_21
29. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: Proceedings of ACM STOC 1995, pp. 57–66 (1995)
30. Katz, J., Ostrovsky, R., Yung, M.: Efficient and secure authenticated key exchange using weak passwords. J. ACM **57**(1), 3:1–3:39 (2009)

31. Jarecki, S., Krawczyk, H., Shirvanian, M., Saxena, N.: Two-factor authentication with end-to-end password security. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 431–461. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_15
32. Huang, K., Manulis, M., Chen, L.: Password authenticated keyword search. In: Proceedings of PAC 2017, pp. 129–140 (2017)
33. Wang, D., Wang, P.: Two birds with one stone: two-factor authentication with security beyond conventional bound. IEEE Trans. Depend. Secure Comput. **15**(4), 708–722 (2018)
34. Becerra, J., Iovino, V., Ostrev, D., Šala, P., Škrobot, M.: Tightly-secure PAK(E). In: Capkun, S., Chow, S.S.M. (eds.) CANS 2017. LNCS, vol. 11261, pp. 27–48. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02641-7_2
35. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of ACM STOC 2008, pp. 197–206 (2008)
36. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_21
37. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. **30**(2), 391–437 (2000)
38. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. **36**(5), 1301–1328 (2007)
39. Benhamouda, F., Blazy, O., Ducas, L., Quach, W.: Hash proof systems over lattices revisited. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 644–674. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_22
40. Abdalla, M., Ben Hamouda, F., Pointcheval, D.: Tighter reductions for forward-secure signature schemes. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 292–311. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_19
41. Katz, J., Ostrovsky, R., Yung, M.: Forward secrecy in password-only key exchange protocols. In: Cimato, S., Persiano, G., Galdi, C. (eds.) SCN 2002. LNCS, vol. 2576, pp. 29–44. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36413-7_3