Achieving One-Round Password-based Authenticated Key Exchange over Lattices

Zengpeng Li and Ding Wang

Abstract—Password-based authenticated key exchange (PAKE) protocol, a widely used authentication mechanism to realize secure communication, allows protocol participants to establish a high-entropy session key by pre-sharing a low-entropy password. An open challenge in PAKE is how to design a quantum-resistant round-optimal PAKE. To solve this challenge, lattice-based cryptography is a promising candidate for post-quantum cryptography. In addition, Katz and Vaikuntanathan (ASIACRYPT’09) design the first three-round PAKE protocol by leveraging the smooth projective hash function (SPHF) over lattices. Subsequently, Zhang and Yu (ASIACRYPT’17) optimized Katz-Vaikuntanathan’s approximate SPHF via a splittable public key encryption. They then constructed a two-round PAKE by using the simulation-sound non-interactive zero-knowledge (NIZK) proofs, but how to construct a lattice-based simulation-sound NIZK remains an open research question. In other words, how to design a one-round PAKE via an efficient lattice-based SPHF still remains a challenge. In this work, we attempt to fill this gap by proposing a lattice-based SPHF with adaptive smoothness. We then obtain a one-round PAKE protocol over lattices with rigorous security analysis by integrating the proposed SPHF into the one-round framework proposed by Katz and Vaikuntanathan (TCC’11). Furthermore, we explore the possibilities of achieving two-round PAKE and universal composable (UC) security from our SPHF, and show the potential application of our PAKE in Internet of Things (IoTs) where communication cost is the main consideration.

Index Terms—Password-based Authenticated Key Exchange; Smooth Projective Hash Function; Lattice-based Cryptography.

1 INTRODUCTION

Password-based authentication still constitutes the most widespread concepts of authentication [1], [2], especially on the Internet today, e.g., [3], [4]. Password-based authenticated key exchange (PAKE) protocol is an important cryptographic primitive that enables two players (e.g., a client and a server) to generate a high-entropy session key by using a common, low-entropy password. Then, the players could utilize the common session key to protect communications over an insecure network (see Fig. 1). Building on earlier literatures such as the EKE (a.k.a., encrypted key exchange) scheme [5] and the Bellare-Pointcheval-Rogaway (BPR) model [6], the classical Kata-Ostroskyy-Yung (in short KOY) framework [7] is proposed under the decisional Diffie-Hellman (DDH) assumption. Subsequently, Gennaro and Lindell (hereafter GL scheme) [8] generalized the KOY scheme by introducing the smooth projective hash function (SPHF) in the BPR security model. Since then, considerable attention [9]–[12] has been devoted to the development of efficient PAKE protocols via SPHFs.

The concept of SPHF was first denoted by Cramer and Shoup [13]. The authors used SPHF to obtain the first encryption scheme, which is indistinguishable against (adaptive) chosen ciphertext attacks (or IND-CCA2) from an encryption scheme that is indistinguishable against chosen plaintext attacks (or IND-CPA), under the DDH assumption in the standard model. A number of DDH-based or lattice-based SPHFs have been successively proposed (e.g., [9], [14], [15]). With these SPHFs as the building block, constant-round PAKE protocols (e.g., [9], [10], [12], [15]) can be obtained. However, most of these PAKE protocols are built on DDH-based SPHFs in the group (or pair) setting.

According to our investigation, most of the existing PAKE protocols (e.g., [9], [16]–[18]) under the GL framework [8] need at least three rounds, and they leverage IND-CCA2 encryption schemes to establish a high-entropy session key. However, how to reduce communication rounds and relax the security requirement of the underlying primitive remain two research challenges that have yet to be resolved. Notably, Jiang and Gong [16] relaxed the security of GL framework by using the combination of an IND-CPA scheme at the user-side and an IND-CCA2 scheme at the server side. However, their PAKE protocol still needs three rounds of communication. In 2015, Abdalla et al. [19] improved the GL framework and obtained a two-round PAKE protocol under the DDH-based SPHF, where the client requires an IND-CPA-secure scheme and the server...
requires an indistinguishable against plaintext checkable attack (or IND-PCA) resistant scheme. Note that an IND-CCA2-secure scheme implies an IND-PCA-secure scheme. However, most of mentioned PAKE protocols (e.g., [16]–[18]) under the DDH assumptions are insecure in the coming quantum era.

Motivations. In fact, our main motivation is to design a quantum-resistant round-optimal PAKE protocol. With advances in quantum computing, quantum attacks against conventional cryptographic primitives (e.g., DDH-based ones) are becoming a reality. Not surprisingly, post-quantum cryptographic primitives are receiving increased attention. In special, the cryptographic primitive over lattices is one popular line of research. In lattice-based cryptography, the worst-case hardness of lattice assumptions (e.g., the Short Integer Solution (SIS) and the Learning with Errors (LWE)) [25], [26] have been demonstrated to be a great success for attribute-based encryption (ABE), functional encryption (FE) and fully homomorphic encryption (FHE) [27].

Accordingly, many traditional cryptographic primitives have been re-constructed depending on the assumptions over lattices, including lattice-based SPHF schemes and PAKE protocols. In addition, there are only three lattice-based PAKE constructions under the learning with errors (or LWE) assumptions (i.e., [9], [14], [15]). A comparative summary of lattice-based SPHFs is presented in Table 1. Apparently, as shown in Table 1, it still remains an open research question as to:

Is it possible to construct an efficient one-round PAKE protocol via the lattice-based SPHF?

1.1 Our Contributions and Techniques

In this work, we answer the above question in the affirmative. More specifically, we optimize the IND-CCA-secure lattice-based LWE scheme of Micciancio and Peikert [22] leveraging the label and the deterministic rounding function respectively, and then construct a one-round PAKE protocol over our new optimized SPHF scheme by adopting the general one-round PAKE framework of Katz and Vaikuntanathan [23]. In summary, our contributions are four-fold:

1) **Neat LWE-based approximate SPHF.** Inspired by SPHF of Benhamouda et al. [15] for the hash proof system, we construct a word-independent approximate SPHF via the Micciancio-Peikert trapdoor [22], in accordance with the principle of Katz and Vaikuntanathan [9]. As our construction focuses on the practicality and flexibility of one-round lattice-based PAKE protocols, we avoid the underpinning “simple error correcting code” (or ECC) [14]. We also adopt a simple deterministic rounding function to obtain the common session key. Although the rounding function of Benhamouda et al. [15] is more accurate in rounding the hash value, it is more complex than the deterministic rounding function [9] we use.

2) **One-round PAKE over lattices.** With our new lattice-based SPHF, we design an efficient one-round PAKE protocol over lattices in the standard model, and provide a formal security analysis. Benhamouda et al. [28] followed Katz-Vaikuntanathan’s framework [23] and proposed a one-round PAKE protocol via trapdoor-SPHF, whose security relies on the DDH assumption. Zhang and Yu [14] proposed a two-round PAKE over lattices, and its main drawback is that the IND-CCA-secure encryption scheme depends on simulation-sound non-interactive zero-knowledge (NIZK) proofs [21]. So far, how to construct the lattice-based simulation-sound NIZK in standard setting is still an open question. At PKC’18, Benhamouda et al. [15] demonstrated how one can obtain the one-round PAKE. However, how to instantiate the protocol over lattices remains an open problem. In this paper, we explore the potential of SPHF over lattices to construct a concrete one-round PAKE over lattices, thereby realizing the idea of Benhamouda et al. [15].

3) **New security bound.** We employ Zipf’s law in passwords [29], [30] to quantify the advantage Adv of the adversary (in short, Adv is representing the advantage). In other words, we set $\text{Adv} = C \cdot \frac{Q(\lambda)}{|D| + \text{negl}(\lambda)}$ for an attacker making at most $Q(\lambda)$ on-line guesses, where $\lambda$ is the system security parameter and $D$ is the password space. This new security bound is $3 \sim 4$ order of magnitude more accurate than the conventional uniformly-random security bound (i.e., $\text{Adv} = \frac{Q(\lambda)}{|D| + \text{negl}(\lambda)}$) that has been used in the security proofs of most existing protocols (e.g., [14], [15], [31]–[33]). We then demonstrate the effectiveness of our approach by evaluating a large-scale real-world dataset, comprising 524.65 million mail.163.com passwords. Notably, the ‘163 Mail’ is the largest mail provider in China.

4) **Some potential applications.** We show that our SPHF can be used to design lattice-based two-round and three-round PAKE protocols in the standard model. Further, one can probably extend our one-round PAKE to achieve universally composable security (UC) according to [17], [23], but it is beyond the scope of this paper. We also explore the applications of our PAKE in Internet of Things (IoTs) inspired by various applications [34], [35].

2 Related Work

Before describing our constructions, we will briefly review the literatures of PAKE and SPHF.

2.1 PAKE

using SPHF on a labeled IND-CCA-secure PKE scheme. In the following year, Jiang and Gong [16] relaxed the security of Gennaro-Lindell framework using random oracles. In this setting, an IND-CPA-secure scheme can satisfy the requirements of the client, but the server still requires an IND-CCA2-secure scheme to preserve the privacy. However, these schemes were only achieved in the standard model (under the stand-alone setting). Groce and Katz [18] extended the Jiang-Gong scheme [16] in the universal composability (UC) framework [17], [36] and proved it secure. Along similar line, Katz and Vaikuntanathan [9] proposed the first three-round PAKE over lattices in ASIACRYPT’09. However, the proposed PAKE requires three rounds.

**Two-round PAKE.** Reducing the number of communication rounds and relaxing the security assumptions are two ongoing research focus in the PAKE literature. Over the years, there were only two successful attempts to reduce the communication rounds (or flows) from three to two. The first scheme, proposed by Abdalla et al. [19], introduced a new cryptographic primitive IND-CPA-secure PKE scheme with an associated SPHF to satisfy the client requirements of the two-round PAKE. They also adopted the IND-CPA-secure scheme with an associated SPHF to meet the requirements of the server. In this setting, the stronger assumption (e.g., IND-CCA2-secure scheme along with an associated SPHF) is no longer required by the server. The second scheme, introduced by Zhang and Yu [14], instantiated the first two-round PAKE over lattices using the splittable public key encryption (or PKE) scheme with the associated non-adaptive approximate SPHF. Unfortunately, their PAKE depends on simulation-sound non-interactive zero-knowledge (NIZK) proofs and how to construct a lattice-based simulation-sound NIZK remains an open question.

**One-round PAKE.** The first one-round PAKE framework based on conventional DDH assumption was designed by Katz and Vaikuntanathan [23], where the client and the server are required to send the message to each other simultaneously. The authors then proved the security of the protocol in the standard model and in the UC framework separately. Benhamouda et al. [28] followed the framework of [23] and designed an efficient one-round PAKE via trapdoor-SPHF on a Cramer-Shoup ciphertext, but their scheme was still based on conventional DDH assumption. Currently, there is no known concrete one-round PAKE protocol construction over lattices in the literature.

### 2.2 SPHF

As illustrated in Fig. 2, SPHF is defined on the NP language $L$ over a domain $X$. At the time of research, there is no efficient adversary that can distinguish between an element (or a point) $x$ in NP language $L$ (i.e., $x \in L$) and an element (or a point) $x \in X \setminus L$ for domain $X$. Furthermore, SPHFs contain two keyed functions (i.e., Hash$(\cdot)$ and ProjHash$(\cdot)$). Concretely, function Hash$(\cdot)$ can be computed by taking as input the hashing key $hk$, and the function ProjHash$(\cdot)$ can be computed by taking as input the projective hashing key $ph$. The output values of both functions are the same (i.e., statistically indistinguishable) for the word $W$ over the language $L$ (i.e., $W \in L$). In other words, the output value satisfies $\text{Hash}(hk, W) = \text{ProjHash}(ph, W, w)$, where $w$ is the witness and the word $W$ contains the labeled IND-CCA ciphertext $c$ and the message $msg$. Thus, even when given $ph$, the adversary cannot guess $\text{Hash}(hk, W \in L)$.

The formal definition is presented in Sec. 3.

![Fig. 2. Smooth projective hash function.](image-url)

**Word-independent CS-SPHF with non-adaptive soundness.** SPHF was first proposed by Cramer and Shoup [13] (CS-SPHF) to facilitate the design of PAKE protocol. The syntax of CS-SPHF is as follows:

\[
\begin{align*}
  \{ & hk \leftarrow \text{HashKG}(L); ph \leftarrow \text{ProjKG}(hk, pk, \perp); \\
  & h \leftarrow \text{Hash}(hk, W); p \leftarrow \text{ProjHash}(ph, W, w) \}\end{align*}
\]

(2.1)

**Word-dependent GL-SPHF with non-adaptive soundness.** Gennaro and Lindell [8] optimized CS-SPHF, but made the projective key $ph$ dependent on the word $W$. We abbreviate it to GL-SPHF, whose syntax is as follows:

\[
\begin{align*}
  \{ & hk \leftarrow \text{HashKG}(L); ph \leftarrow \text{ProjKG}(hk, pk, W); \\
  & h \leftarrow \text{Hash}(hk, W); p \leftarrow \text{ProjHash}(ph, W, w) \}\end{align*}
\]

(2.2)

Both CS-SPHF and GL-SPHF achieve non-adaptive smoothness, namely that $W$ is independent of the project key $ph$ and the adversary cannot see the projective key $ph$ before choosing the word $W$. Namely that even if we can see $ph$, we cannot change $W$.  

### Table 1. A comparative summary of lattice-based SPHF schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Building blocks</th>
<th>SPHF models</th>
<th>Adaptive smooth</th>
<th>Error-correction code</th>
<th>Round</th>
</tr>
</thead>
</table>

† The scheme in [9] builds on the work of [20], [24], while the scheme in [14] builds on the scheme of [9].

• NIZK: Non-Interactive Zero-Knowledge; SPHF: Smooth Projective Hash Function.

• The symbol * implies that there is no concrete PAKE construction, but it provides an interesting point to construct one-round PAKE over lattices.
Word-independent KV-SPHF with adaptive soundness. Katz and Vaikuntanathan [9] achieved adaptive smoothness and made the projective key $ph$ independent on $W$, where adaptive smoothness implies that we can choose $W$ after having seen the projective key $ph$. For convenience, we abbreviate it to KV-SPHF, whose syntax is as follows:

$$\begin{align*}
\{hk & \leftarrow \text{HashKG}(L); ph \leftarrow \text{ProjKG}(hk, pk, \bot); \\
h & \leftarrow \text{Hash}(hk, f(W)); p \leftarrow \text{ProjHash}(ph, W, w)\}.
\end{align*}$$

We say $f$ is adaptive if $A$ sees $ph$ before choosing $W$.

2.3 IND-CCA-Secure Scheme over Lattices

There are two paradigms of IND-CCA2-secure encryption that are built on IND-CPA-secure encryption:

- Dolev-Dwork-Naro paradigm [37]: utilizing one-time signature and one-time NIZK (OT-NIZK).
- Naro-Yung/Sahai paradigm ( [21], [38]): utilizing one-time simulation-sound NIZK (OT-SS-NIZK).

Furthermore, there are a number of variants of the Cramer-Shoup scheme [13] designed to achieve IND-CCA2-secure encryption, by using the hash proof system, such as the schemes of Kiltz et al. [39] and of Kurosawa-Desmedt [40].

However, only three straight constructions of the IND-CCA-secure PKE over lattices have been proposed in the literature. Concretely, Peikert and Waters [41] proposed the first IND-CCA1-secure PKE scheme under the (worst-case) lattice assumption, along with some optimization [20]. However, the schemes are based on lossy trapdoor functions. In a separate work, Katz and Vaikuntanathan [9] (KV) used Regev scheme [25] as the building block and designed the IND-CCA1-secure PKE scheme. However, the decryption algorithm of Katz-Vaikuntanathan scheme needs to invoke the BBDSolve($\cdot$) procedure multiple times. Subsequently, Micciancio and Peikert [22] proposed a new efficient IND-CCA1-secure PKE scheme, by introducing the G-trapdoor function and tweaking the decryption algorithm. In other words, plaintexts are recovered via querying the trapdoor inversion procedure Invert($\cdot$) multiple times.

3 Preliminaries

Below, we first list parameters in Table 2 that will be used in our construction and security analysis.

Furthermore, vectors and matrices are denoted as bold lower-case letter (e.g., $x$) and bold upper-case letter (e.g., $A$), respectively. An $m$-dimension lattice can be denoted as $\Lambda = \{B s | s \in \mathbb{Z}^n\}$, where $B \in \mathbb{Z}^{m \times n}$ is referred to as the basis of $\Lambda$ for the parameter $m \geq n[\log q]$. The determinant of $\Lambda$ is denoted as $\det(\Lambda) = \sqrt{\det(B^T B)}$, $q$-ary lattices is defined as follows,

$$\begin{align*}
\Lambda(A) &= \{A \cdot s | s \in \mathbb{Z}^n_q\} + q\mathbb{Z}^m, \\
\Lambda^-(A) &= \{z \in \mathbb{Z}^m | z^T \cdot A = 0 \pmod{q}\}, \\
\Lambda^u(A) &= \{z \in \mathbb{Z}^m | z^T \cdot A = u \pmod{q}\}.
\end{align*}$$

### Tab. 2. Parameters and Abbreviations.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>Security parameter</td>
</tr>
<tr>
<td>$Q(\lambda)$</td>
<td>$Q(\lambda)$ on-line guesses</td>
</tr>
<tr>
<td>$D$</td>
<td>The password space</td>
</tr>
<tr>
<td>$\text{negl}(\lambda)$</td>
<td>A negligible function in $\lambda$</td>
</tr>
<tr>
<td>$s'$ and $C'$</td>
<td>Parameters of the Zipf distribution</td>
</tr>
<tr>
<td>$A$</td>
<td>The adversary</td>
</tr>
<tr>
<td>$\text{Adv}$</td>
<td>The advantage of the adversary</td>
</tr>
<tr>
<td>$\Pi$</td>
<td>The proposed one-round PAKÉ protocol</td>
</tr>
<tr>
<td>$\text{PPT}$</td>
<td>Probabilistic polynomial time</td>
</tr>
<tr>
<td>$\text{IND-CPA}$</td>
<td>Indistinguishable against chosen plaintext attacks</td>
</tr>
<tr>
<td>$\text{IND-CCA}$</td>
<td>Indistinguishable against (Non-adaptive) chosen ciphertext attacks</td>
</tr>
<tr>
<td>$\text{IND-CCA2}$</td>
<td>Indistinguishable against (Adaptive) chosen ciphertext attacks</td>
</tr>
<tr>
<td>$\text{IND-PCA}$</td>
<td>Indistinguishable against plaintext checkable attack</td>
</tr>
</tbody>
</table>

Note that, $\Lambda(A)$ and $\Lambda^-(A)$ are dual of each other. $\Lambda^u(A)$ is the coset of $\Lambda^-(A)$ for a syndrome $u \in \mathbb{Z}_q^n$.

**Definition 3.1** (Hamming Metric). For any two strings of equal length $x, y \in \{0, 1\}^n$, if we write $\text{HD}(x, y)$, then the Hamming distance is one of several string metrics for measuring the edit distance between two strings.

### 3.1 Lattice Background

**Definition 3.2** (Decision-LWE$_{n, q, m, \chi}$, [25]). We first assume that there exist two different distributions

- (1) $A_{x, \chi} := \{(A, b) : A \leftarrow \mathbb{Z}_q^{m \times n}, s \leftarrow \mathbb{Z}_q^{n \times 1}, e \leftarrow \chi^{m \times 1}, b = A \cdot s + e \pmod{\chi}\}$
- (2) the uniform distribution (i.e., $\{(A, b) : A \leftarrow \mathbb{Z}_q^{m \times n}, b \leftarrow \mathbb{Z}_q^{m \times 1}\}$)

If no one can distinguish an independent sample $(A, b) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times 1}$ with overwhelming probability, where the sample is distributed according to either the distribution (1) or the distribution (2), then we can say that the above two distributions are computationally indistinguishable.

**Remark 3.3.** Reductions between approximating the shortest vector problem (SVP) in lattices (for appropriate parameters) and the LWE problem have been discussed by Regev and others [20], [25], [41], [42]. The reduction is out of scope of this work, thus the related corollaries of the reductions are omitted at this stage. More details are available in [20], [25], [41], [42].

Below, we state the Lemma 3.4 that is a key result used to show the correctness of our construction. Before describing the Lemma 3.4, we will first introduce the computable gadget matrix $G \in \mathbb{Z}_q^{m \times N}$ for the parameters $N \geq m[\log q]$ [22]. $G$ is a fixed structure matrix which is composed of a set of the gadget vector $g = (2^0, 2^1, 2^2, \ldots, 2^{\ell - 1}) \in \mathbb{Z}_q^{\ell}$. Along with the matrix $G$, there is an efficiently computable “short preimage” function $G^{-1}(\cdot)$ which is also a deterministic inverse function.

**Lemma 3.4** ([22]). Regarding the parameters $N$, $q$, $m$ and $m'$, there exists an equation $GG^{-1}(M) = M$, if the inverse
function $G^{-1}(\cdot)$ inputs on a matrix $M \in \mathbb{Z}^{n \times m}$ and the output result satisfies $G^{-1}(M) \in \{0, 1\}^{n \times m}$.

**Corollary 3.5.** In order to invert the injective trapdoor function $g_\lambda(s, e) = s^T \cdot A + e \pmod q$, a PPT algorithm $Invert(\cdot)$ should fulfill the following requirements:

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

### 3.2 Smooth Projective Hash Functions

The projective hash function families were first proposed by Cramer and Shoup [13] first at EUROCRYPT’02. SPHF as an important type of projective hash function. In a nutshell, in the projective hash function families, the adversary is computationally hard to distinguish a random element in language $L$ from a random element in the domain $X \setminus L$ for the existence of a domain $X$ along with $L \subseteq X$, where $L$ is an underlying NP language.

As shown in Fig. 2 that is inspired by [43], SPHF acts as a special proof system and contains the required elements: a witness $w$ and a word $W$. $W$ contains the labeled IND-CCA ciphertext $c$ and message $m$. Further, SPHF contains both approximate correctness and smoothness properties. SPHF over $L \subseteq X$ is defined by four PPT algorithms.

- $hk \leftarrow \text{HashKG}(L)$. It takes an NP language $L$ as input and outputs a “private” hash key $hk$.
- $ph \leftarrow \text{ProjKG}(hk, L, W)$. The algorithm takes an NP language $L$, a hash key $hk$, and a word $W \in L$ as input and outputs a “public” projective hash key $ph$.
- $h \leftarrow \text{Hash}(hk, L, W)$. The algorithm takes an NP language $L$, a hash key $hk$, and a word $W \in L$ as input and outputs a hash value $h$ over $\{0, 1\}^v$ for some positive integer $v = \Omega(\lambda)$.
- $\text{ProjHash}(ph, L, W, w)$. The algorithm takes an NP language $L$, a projective hash key $ph$, a word $W \in L$, and a witness $w$ as input and outputs a projective hash value $p$ over $\{0, 1\}^v$.

Meanwhile, SPHF satisfies the notions of (approximate) correctness and smoothness:

- **Approximate Correctness:** If a word $W$ is in language $L$, i.e., $W \in L$ there exists
  \[
  \Pr[H(D(\text{Hash}(hk, L, W)), \text{ProjHash}(hk, L, W))] > \varepsilon \cdot |v| = \text{negl}(\lambda),
  \]
  then the approximate correctness (i.e., $\varepsilon$-correct) property holds.
- **Smoothness:** If any word $W \in X \setminus L$, the following two distributions have a negligible statistical distance in $\lambda$:

  1. $\{(ph, h) \mid hk \leftarrow \text{HashKG}(L), ph = \text{ProjKG}(hk, L, W), h \leftarrow \text{Hash}(hk, L, W)\}$.
  2. $\{(ph, h) \mid hk \leftarrow \text{HashKG}(L), ph = \text{ProjKG}(hk, L, W), h \leftarrow \{0, 1\}^v\}$. (3.6)

then the smoothness property holds.

If the approximate SPHF is 0-correct, then we call it SPHF. However, in the lattice setting, we cannot obtain the 0-correct feature.

### 3.3 PAKE Security Model

Below, we review the PAKE security model by following the definitions of Bellare, Pointcheval, and Rogaway [6].

**Participants, passwords, and initialization.** A fixed set of protocol participants (also known as users) is denoted as $U$. For every distinct $U_1, U_2 \in U$, we assume $U_1$ and $U_2$ share a password $pw_{U_1, U_2}$ (i.e., $pw$). It is assumed that each $pw_{U_1, U_2}$ is independently sampled from the password space $D(\lambda)$ according to Zipf’s law [29].

**Execution of the protocol.** In reality, a protocol describes how users behave after receiving commands (or inputs) from the environment. On the contrary, the adversary provides these inputs in the formal model. In this model, the protocol can be executed by each party with different partners multiple times (possibly concurrently).

Meanwhile, this model allows each party to instantiate an unlimited number of instances, and the adversary is allocated oracle access to these different instances. For convenience, the $i$-th instance of the user $U$ is denoted as $\Pi^i_U$. In particular, each instance can be used in this model only once. Furthermore, the (local) state that is updated during the course of the experiment is maintained by corresponding each instance. Below, we describe the concrete variables of local state maintained by each user instance $\Pi^i_U$:

- $sid^i_U$, session id.
- $pid^i_U$, partner id.
- $skey^i_U$, session key id.
- $acc^i_U$, a boolean variable denoting acceptance at the end of the execution.
- $term^i_U$, a boolean variable denoting termination at the end of the execution.

**Adversarial Model.** The adversary (or malicious party) is allowed to fully control all communication in the external network, namely the adversary is able to do whatever (s)he wants, in the sense of the capability to 1). block, inject, modify, and delete messages; and 2). request any session keys adaptively. Formally, the adversary launches attacks using oracle queries model in the real world. Thus, to define security, we introduce a set of oracles and use these oracles to explain how the adversary can interact with various instances.

The following oracle queries are the case of one-round challenge-response protocol, the adversary first sends the query to the assigned oracle, then the assigned oracle answers the query to the adversary. More formally description are as follows:

- $\text{Send}(U, i, M)$. The oracle is first activated by the adversary, then it sends message $M$ to the instance
Upon receiving $M$ from Send, $\Pi_U^i$ then runs in accordance with the protocol specification and updates the state. The output of $\Pi_U^i$ is sent to the adversary.

- Execute($U_C, i, U_S, j$). The oracle represents the protocol execution between $\Pi_U^i$ and $\Pi_U^j$ without any outside interference from the adversary. To answer this query, the oracle outputs the protocol transcript to the adversary, where the transcript is the complete ordered messages that can be exchanged between the instances.

- Reveal($U_C, i$). The oracle models known key attacks and allows the adversary to learn session keys $r$ the specified instance from previous and concurrent executions, and outputs the session key $sk_U$. Meanwhile, improper session keys are erased.

- Test($U_C, i$). The oracle allows the adversary to query it, once and only once, and outputs a random bit $b$. If $b = 1$, then the adversary obtains the session key $sk_U$; otherwise, the adversary is given 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 $\Pi_U^i$ and $\Pi_U^j$ are partnered if: (1) $\text{sid}_U^i = \text{sid}_U^j \neq \text{NULL}$; and (2) $\text{pid}_U^i = U_C$ and $\text{pid}_U^j = U_S$.

**Correctness.** If $\Pi_U^i$ and $\Pi_U^j$ are partnered, then there exist $\text{acc}_U^i = \text{acc}_U^j = \text{TRUE}$ and $sk_U^i = sk_U^j$ and both instances can obtain the common session key.

**Definition 3.6 (Advantage of the adversary).** We assume that all PPT adversaries $A$ make at most $Q(\lambda)$ on-line guessing attacks, and the password dictionary $D$ follows the Zipf-like distribution with parameters $C' = 0.062239$ and $s' = 0.155478$. If the advantage of the adversary holds

$$\text{Adv}_{A, U}(\lambda) \leq C' \cdot Q^{s'}(\lambda) + \text{negl}(\lambda),$$

where $Q(\lambda)$ denotes the number of on-line attacks made by $A$, then the PAKE protocol II is said to be secure.

**Remark 3.7.** In most existing PAKE studies (e.g., [9], [14], [15], [31], [35]), passwords are assumed to follow a uniformly random distribution, and the attacker’s advantage is formulated as $Q(\lambda)/D + \text{negl}(\lambda)$ for the password dictionary with size $D$. However, recent research [29], [44], [45] has revealed that passwords chosen by users from various languages follow the Zipf’s law (but not a uniformly random distribution). Thus, we prefer the CDF-Zipf model [29], and the attacker’s advantage can be formulated as $C' \cdot Q^{s'}(\lambda) + \text{negl}(\lambda)$ with $C'$ and $s'$ being the Zipf parameters. Fig. 3 shows that the traditional uniform-model based formulation $Q(\lambda)/D + \text{negl}(\lambda)$ always $(\forall Q(\lambda) \in [1, D])$ significantly underestimates the real attacker’s Adv. Fortunately, the Zipf based formulation $C' \cdot Q^{s'}(\lambda) + \text{negl}(\lambda)$ well approximates the real attacker’s Adv: $(\forall Q(\lambda) \in [1, D])$, the largest deviation between $C' \cdot Q^{s'}(\lambda) + \text{negl}(\lambda)$ and Adv is 0.491%.

**4 Our SPHF via Micciancio-Peikert Scheme**

Below, we detail the labeled IND-CCA1-secure Micciancio-Peikert scheme [22]. At present, there are only a small number of IND-CCA1-secure PKE schemes [9], [20], [22], [26], [46], [47] over lattices. Compared with the Katz-Vaikuntanathan scheme [9], the main advantage of the Micciancio-Peikert scheme is that it does not require the invoking of the Invert($\cdot$) algorithm multiple times to recover the plaintext during decryption. Furthermore, the IND-CCA1-secure Micciancio-Peikert scheme can be converted to the level of IND-CCA2 security via relatively generic transformations using either a message authentication code (MAC) along with a weak form of commitment [48] or strongly unforgeable one-time signature [37]. For simplicity, we only present the label-IND-CCA1-secure Micciancio-Peikert scheme in this section, and omit the phase of transformation to obtain the IND-CCA2-secure scheme. The detail is as follows:

**4.1 Labeled CCA1 Micciancio-Peikert (MP) Scheme**

- params $\leftarrow$ MP.Setup($\lambda, m, \bar{m}, n, q, \ell_q$) : Takes the security parameter $\lambda$, the integers $m, \bar{m}, n$, and the model $q$ as input, where $m = m + n \ell_q$ and $\ell_q = \lceil \log q \rceil = O(\log q)$; then outputs the parameters $\text{params} := (\lambda, m, \bar{m}, n, q, \ell_q)$.

- $(sk, pk) \leftarrow$ MP.KeyGen($\text{params}$) :

1) Takes the params as input and samples a public matrix $A \leftarrow \mathbb{Z}_q^{n \times m}$ along with the trapdoor matrix $R \leftarrow \mathbb{Z}_{\omega(\sqrt{n \log n})}^{m \times n \ell_q}$ by invoking the trapdoor generation algorithm (i.e., $(P, T) \leftarrow \text{TrapGen}(\text{params}, A, R)$). The dot production between the public key $P$ and the trapdoor $T$ is 0 (i.e., $P \cdot T = 0 \pmod q$).

2) Let the matrix $A_i := -A \cdot R \pmod q \in \mathbb{Z}_q^{n \times n \ell_q}$ and the public key $P := \left[ \begin{array}{c} A + A_i \\ -A \end{array} \right] \in \mathbb{Z}_q^{n \times m}$ is generated. Then, we denote $(n \ell_q \times n \ell_q)$-dimension identity matrix by $I_{n \ell_q \times n \ell_q}$ and generate the trapdoor (i.e., secret key) $T := [R \mid I_{(n \ell_q \times n \ell_q)}]$. 

**Fig. 3.** Guessing advantages of the real attacker, the uniform attacker (e.g., [14], [15], [23], [31]) and our Zipf attacker (using 524.65 million mail.163.com passwords).
3) Outputs both secret key $sk := T \in \mathbb{Z}_{q}^{m \times n e_{q}}$ and public key $pk := P \in \mathbb{Z}_{q}^{n \times m}$.

- $c \leftarrow \text{MP.Enc}(pk = P, m \in \{0, 1\}^{n e_{q}}, \text{label} = u)$:
  1) In order to encrypt the message $m \in \{0, 1\}^{n e_{q}}$, the algorithm first maps $m \in \{0, 1\}^{n e_{q}}$ to encode($m$) = $S \cdot m \in \mathbb{Z}_{q}^{n e_{q}}$ for any $S \in \mathbb{Z}^{n e_{q} \times n e_{q}}$ of lattice $A$, and takes the public key $P$ as input.
  2) Then, sample a nonzero label $u \leftarrow \mathcal{U}$ and let
     $$A_{u} = [\bar{A} | A_{1} + h(u)G]$$
     $$= [\bar{A} | h(u)G - \bar{A}R] \in \mathbb{Z}_{q}^{n \times m}.$$  (4.1)
     We note that $A_{u} \cdot T = h(u)G$ for the gadget matrix $G \in \mathbb{Z}_{q}^{n \times n e_{q}}$.

- $\textbf{Remark 4.1.}$ In the ring $\mathcal{U} = \mathbb{Z}_{q}[x] \setminus \langle f(x) \rangle$ for $q = p^{r}$, we denote the label set $\mathcal{U} = \{u_{1}, \ldots, u_{r}\} \subset \mathcal{U}$ with the “unit difference property. More concretely, if the difference $u_{i} - u_{j} \in \mathcal{U}$ for any $i \neq j$, then the label matrix $h(u_{i} - u_{j}) = h(u_{i}) - h(u_{i}) \in \mathbb{Z}_{q}^{n \times n}$ is invertible.

3) Sample a random vector $s \leftarrow \mathbb{Z}_{q}^{n \times 1}$, a noise vector $\tilde{e} \leftarrow D_{\bar{Z}, \alpha q}$, and another noise vector $\hat{e} \leftarrow D_{\bar{Z}, \alpha q}$, where $s^{2} = (||\hat{e}||^{2} + \bar{m}(\alpha q)^{2}) \cdot \omega(\sqrt{\log n})$. Then, the algorithm samples $\tilde{e}$ and $\hat{e}$ together, and obtains the noise vector $e = (\hat{e}, \tilde{e}) \in \mathbb{Z}_{q}^{n \times 1}$.

4) Computes and outputs the ciphertext over $\mathbb{Z}_{q}^{n \times 1}$,
   $$c = A_{u}^{T} \cdot s + e + (0 | \text{encode}(m)) \pmod{q}.$$  (4.2)

5) Lastly, outputs the ciphertext as follows $c = (u, c) \in \mathcal{U} \times \mathbb{Z}_{q}^{n \times 1}$.

- $m \leftarrow \text{MP.Dec}(sk, c = (u, c))$ : In order to decrypt the ciphertext $c = (u, c)$ for $u$, the algorithm proceeds with the following steps:
  1) The algorithm first parses $c$ into $u$ and $c$. If $u = 0$, then outputs empty value; otherwise, the algorithm invokes the algorithm Invert($A_{u}, c, R, h(u) \in \mathbb{Z}_{q}^{n \times n}$) (Lemma 3.5) and obtains $c = A_{u}^{T}z_{1} + e_{1}$, (i.e., obtains secret vector $z_{1} \in \mathbb{Z}_{q}^{n \times 1}$ and noise vector $e_{1} = (\hat{e}_{1}, \tilde{e}_{1}) = c - A_{u}^{T}z_{1} \in \mathbb{Z}_{q}^{n \times 1}$). Then, the algorithm checks whether $||\hat{e}_{1}|| \geq \alpha q\sqrt{m}$ or $||\tilde{e}_{1}|| \geq \alpha q\sqrt{2mne_{q}} \cdot \omega(\sqrt{\log n})$, and the output is assigned empty value.
  2) Otherwise, the decryption algorithm invokes Invert($A_{u}, c - (0 | \text{encode}(m)), R, h(u)$) and obtains $c - (0 | \text{encode}(m)) = A_{u}^{T} \cdot z_{2} + e_{2}$ for $||\hat{e}_{2}|| < \alpha q\sqrt{m}$ and $||\tilde{e}_{2}|| < \alpha q\sqrt{2mne_{q}} \cdot \omega(\sqrt{\log n})$. Let $v = c - e_{2} = A_{u}^{T} \cdot z_{2} + (0 | \text{encode}(m)) \pmod{q}$, and the algorithm requires $v$ to parse $v = (\tilde{v}, \hat{v}) \in \mathbb{Z}_{q}^{n \times 1} \times \mathbb{Z}_{q}^{n \times 1}$ for $\tilde{v} \in \Lambda(A^{T})$.
  3) Finally, computes and outputs the plaintext decode($v^{T} \cdot [\tilde{T} | T] \pmod{q}$) $\in \{0, 1\}^{n e_{q}}$.

The proofs of security and correctness are similar to that of the IND-CCA1 MP scheme. To be self-contained, we now provide the two important properties via the following Lemma 4.2 and Theorem 4.3 respectively.

**Lemma 4.2 (Correctness, [22]).** If the correctness property of the above Micciancio-Peikert scheme is hold, then the probability of decryption error is only $2^{-\Omega(n)}$.

**Theorem 4.3 (Security, [22]).** The above Micciancio-Peikert scheme is labeled IND-CCA1-secure under the hardness of decisional LWE$_{n,m,q,x}$ assumption.

The correctness and security analysis can be obtained in a straightforward method following the work of [22]. To save space, the detailed analysis is omitted, and more details are referred to [22].

### 4.2 Our Contribution: MP-SPHF

It is known that the Micciancio-Peikert scheme is labeled IND-CCA1-secure under the decisional LWE assumption. Further, if we introduce a one-time signature scheme, then we can use it to sign the ciphertext under the secret key of the signature, resulting in an IND-CCA2-secure scheme.

Below, we use it to develop an associated SPHF. Following the Katz-Vaikuntanathan (KV-SPHF) construction and introducing the deterministic rounding function of [9], we present a neat lattice-based SPHF scheme with adaptive soundness based on Micciancio-Peikert scheme. The scheme is also hereafter referred to as MP-SPHF.

- $hk \leftarrow \text{HashKG}($params$)$. The algorithm samples a random vector $k$ from $\mathbb{Z}_{q}^{n \times 1}$. Then, it outputs the hashing key $hk := k \in \mathbb{Z}_{q}^{n \times 1}$.
- $ph \leftarrow \text{ProjKG}($params, $hk = k, pk = A_{u}$). The algorithm takes the hashing key $k$ and the public key $A_{u} = [A | h(u)G - AR] \in \mathbb{Z}_{q}^{n \times m}$ with the fixed label $u$ as input, then outputs the projective hashing key $ph := p = A_{u} \cdot k \in \mathbb{Z}_{q}^{n \times 1}$. In this setting, to obtain the “approximate correctness”, we modify the public key. However, in MP-SPHF scheme, the public key of MP scheme is $P := [A | A_{1}] = [A | -A \cdot R] \in \mathbb{Z}_{q}^{n \times m}$.
- $h \leftarrow \text{Hash}(hk = k, W := (c, m))$. The smooth hash function executes as follows:
  1) The algorithm inputs on the hashing key $k$ and the word $W$, where $W$ contains a ciphertext $c = (\text{label}, c \in \mathbb{Z}_{q}^{n \times 1})$ and the plaintext $m$.
  2) The hash function works as follows:
     $$h = \text{Hash}(hk = k, W := (c, m))$$
     $$= R\left(\left[c - (0 | \text{encode}(m))\right]^{T} \cdot k\right)$$
     $$= R\left(s^{T} \cdot A_{u} \cdot k + e^{T} \cdot k \pmod{q}\right),$$
     and outputs the hash values over the set $\{0, 1\}$, where the noise element $e^{T} \cdot k$ is bounded by $|e^{T}k| \leq ||e^{T}|| \cdot ||k|| \leq (\sqrt{mne_{q}}) \cdot (\alpha q\sqrt{m}) < e/2 \cdot q/4$.
  3) Next, the algorithm obtains the result of $b := h \pmod{2} \in \{0, 1\}$, namely: the value $b$ is a number in $[-(q - 1)/2, \ldots, (q - 1)/2]$ and the algorithm outputs $b = 0$ if $h < b$; otherwise, outputs $b = 1$.
- $p \leftarrow \text{ProjHash}(ph = p, W := (c, m); w = s)$. The projection hash function works as follows,
1) The algorithm takes as input a projected key $ph = p \in \mathbb{Z}_p^{n \times 1}$, the word $W$, and the witness $s \in \mathbb{Z}_p^{n \times 1}$.
2) The algorithm computes and outputs
   \[ p = \text{ProjHash}(ph = p, W := (c, m); w = s) \]
   \[ = R \left( s^T \cdot (A_u k) \pmod q \right) \in \{0, 1\}. \quad (4.4) \]
3) Next, the algorithm obtains the result of \( b := p \pmod 2 \in \{0, 1\} \), outputs \( b = 0 \) if \( p < 0 \); otherwise, outputs \( b = 1 \).

**Theorem 4.4.** The MP-SHPF is a smooth projective hash function for the MP scheme.

**Proof:** We will now prove Theorem 4.4 via the following two steps, which are used to respectively demonstrate approximate correctness and smoothness.

**Regarding projective (or approximate correctness).**
Our goal is to prove \( \text{Hash}(hk = k, W := (c, m)) = \text{ProjHash}(ph = p, W := (c, m); w = s) \) with probability greater than 1/2. In lattice-based setting, the correctness of SHPF indicates that the relationship between the hash key \( hk \) and the word \( W \) from language \( L \) equates the relationship between the projective hash key \( ph \) and the witness \( w \) for any word in \( L \).

**Lemma 4.5 (Approximate Correctness).** If the magnitude of inner product \( (e, k) \) is small for the parameters \( n, m \geq n \sqrt{\log q} \), then the results of the rounding function \( y_{\text{hash}} = R \left( \text{Hash}(hk = k, W := (c, m)) \right) \) and \( y_{\text{proj}} = R \left( \text{ProjHash}(ph = p, W := (c, m); w = s) \right) \) are satisfying the following requirement:
\[
\text{Pr}[\text{HD}(y_{\text{hash}}, y_{\text{proj}}) > \varepsilon \cdot v] = \text{negl}(\lambda). \quad (4.5)
\]
**Proof:** In this paper, we adopt the typical deterministic rounding function \( R(x) = [2x/q] \pmod 2 \) and follow the methodology of [9] to round the respective outputs of Hash(·) and ProjHash(·). In order to prove the following equations hold, we have that
\[
R \left( \text{Hash}(hk = k, W := (c, m)) \right) = R \left( \text{ProjHash}(ph = p, W := (c, m); w = s) \right) = R \left( s^T \cdot (A_u k) \pmod q \right). \quad (4.6)
\]
Consider the left side of the above equation, we can view \( R(h) \) as a number in \([-\frac{(q-1)}{2}, \cdots, \frac{(q-1)}{2}]\) and output \( b \in \{0, 1\} \). Moreover, the noise element \( e^T \cdot k \) is bounded by \( |e^T k| \leq \|e^T\| \cdot \|k\| \leq (r \sqrt{mn}) \cdot (\alpha q/\sqrt{m}) < \varepsilon/2 \cdot q/4 \). Hence, the result of \( R(e^T k) \) is identical with 0. Thus,
\[
b = \begin{cases} 
0, & \text{if } R(h) < 0; \\
1, & \text{if } R(h) > 0. 
\end{cases} \quad (4.7)
\]
Consider the right side of the above equation, we have the following results
\[
b = \begin{cases} 
0, & \text{if } R(s^T \cdot (A_u k)) < 0; \\
1, & \text{if } R(s^T \cdot (A_u k)) > 0. 
\end{cases} \quad (4.8)
\]
This completes the proof. \( \Box \)

**Smoothness.** Now, we prove the smoothness property of SHPF. The smoothness of SHPF is that the hash value is independent of the projective hash key \( ph \) for any word in \( X \setminus L \). Moreover, the typical deterministic rounding function \( R(x) = [2x/q] \pmod 2 \) (a.k.a., so-called square-signal) has harmonic coefficients \( r_j \) decreasing as \( \Theta(1/j) \) in absolute value. Using such a rounding function, one would attempt to invoke the trapdoor inversion for \( q/2 \) many multiples of \( c \), more details about the case in [9]. Furthermore, the word \( W := (c, m) \notin L \) implies that the ciphertext \( c \) is not an encryption of message \( m \) under the public key \( pk = A_u \). Hence, the following two distributions have a negligible statistical distance in \( \lambda \),
\[
1). \{ (ph, h) | \text{HashKG}(L) \rightarrow k, \text{ProjKG}(hk, L, W) \rightarrow A_u k, \\
\quad \text{Hash}(hk, L, W) = (s^T A_u + e^T) k \}. \quad (4.9)
2). \{ (ph, h) | \text{HashKG}(L) \rightarrow k, \\
\quad \text{ProjKG}(hk, L, W) \rightarrow A_u k, \\
\quad h \leftarrow \{0, 1\} \}.
\]
We note that, \( \text{Hash}(hk, W) = (s^T A_u + e^T) k \) given \( \text{ProjKG}(hk, pk) = A_u k \). Due to \( s \) being the witness (or random vector), no information on \( \text{Hash}(hk, W) \) can be provided by \( \text{ProjKG}(hk, pk) \), and \( \text{Hash}(hk, W) \) is uniformly distributed over \( \{0, 1\} \), given \( \text{ProjKG}(hk, pk) \).

Hence, the smoothness property of the projective hash function can be concluded. \( \Box \)

## 5 Our One-Round PAKE via MP-SHPF

Since the first SHPFs were initially introduced by Cramer and Shoup [13], various extensions have been proposed. A significant breakthrough was given in [49] (PKC’13) which presented an explicit instantiation of KV-SHPF [23] by using the Cramer-Shoup ciphertext. The existing lattice-based constructions of PAKE are only described in [9], [50]. At ASIACRYPT’09, Katz and Vaikuntanathan [9] followed the framework of KOY-GL (i.e., Katz-Ostrovsky-Yung (KOV) [7] and Gennaro-Lindell [8]), and proposed the first three-round PAKE protocol based on lattices in a variant of the Bellare-Pointcheval-Rogaway (BPR) model [6]. Benhamouda et al. [15], [50] gave a draft of one-round PAKE over lattices, but we cannot find the detailed security analysis. Importantly, their scheme was withdrawn from eprint due to the weakness of the security proof. Before describing our one-round PAKE based on MP-SHPF, we first give a high-level of the KOY-GL framework.

1) Each party in the system first sends an encryption of the password synchronously, then the party uses an SHPF taking as input the encrypted password of the partner to generate an output, lastly, the party
uses the output to determine whether there exists the same password.

2) The common session key is generated by multiplying two hash values, where the two hash values are coming from the Hash(·) and ProjHash(·) with the encrypted password respectively.

Here we remark that, 1) if the encrypted passwords are the same, the hash function Hash and ProjHash compute the hash values in their own way but obtain the same results. 2) Conversely, if the passwords are different, the two hash functions compute the values independently but output the different results, which meets the smoothness feature. In a word, the smoothness property implies that the values computed by the hash function of each player are independent. Therefore, an SPHF on a labeled IND-CCA-secure encryption scheme is necessary to achieve this aim. But to this aim, the scheme only depends on the encryption and doesn’t need the decryption algorithm, thus, no one need to know or store the secret key $sk$.

5.1 Our One-Round PAKE over Lattices

Our protocol follows the framework of Katz and Vaikuntanathan [23] which requires an IND-CCA2-secure encryption scheme with an associated word-independent KV-SPHF for the both sides. As shown in Fig. 4, we initialize the one-round PAKE protocol over lattices by using IND-CCA-secure encryption scheme. Note that we assume the server keeps the password $pw$ in plain-text just for ease of presentation. In practice, due to the Zipf’s law in user-chosen passwords [29], password-storing functions shall be slow enough to resist password guessing attacks. As suggested by the latest NIST SP800-63B “Authentication and Lifecycle Management” guidelines [51], the server shall use a memory-hard hash function (e.g., Scrypt and Lyra2 [52]) to hash the password with a random salt. Furthermore, the correctness of the established session key is implied by the Lemma 4.5.

To our knowledge, once the label is fixed at advanced to some constant, then the resulting scheme degrades to an IND-CPA-secure scheme. To solve this problem, Zhang and Yu [14] provided an idea to achieve an IND-CCA2-secure scheme by combining simulation-sound NIZK over lattices in random oracle setting. But the main trouble is that we don’t know how to obtain simulation-sound NIZK over lattices in standard setting. Benhamouda et al. [15] provided a generic transformation using the idea of [37], which can be used to convert an labeled-IND-CCA1-secure scheme to an IND-CCA2-secure scheme. To do so, in the BPR model [6], we first upgrade the labeled-IND-CCA1-secure MP scheme [22] to the IND-CCA2-secure scheme by using a strong one-time signature scheme $\text{Sgn} = (\text{Sgn.Gen}, \text{Sgn.Sign}, \text{Sgn.Ver})$. The client generates the signature for each ciphertext, while the server verifies the correctness of the signature.

5.2 Security Proof

Theorem 5.1. The one-round PAKE protocol (in short II) in Fig. 4 is secure in the BPR model under the LWE assumptions, if the Micciancio-Peikert scheme is IND-CCA-secure with an associated MP-SPHF.

Proof: Our proof follows the approaches of Katz and Vaikuntanathan [23] and Benhamouda et al. [28]. To apply the modularity approach of the proof given in [28, Theorem 4], the main work of this proof is to check whether our primitives (i.e., MP encryption and MP-SPHF) fulfill the same properties. Essentially, we have discussed both correctness and smoothness properties of MP-SPHF in the preceding section, below we detail the rigorous security analysis of the protocol based on MP-SPHF.

We first fix a polynomial-time adversary $A$ attacking our protocol II. We then construct a sequence of experiments $\text{Expt.0, Expt.1, \ldots, Expt.6}$ with the original experiment corresponding to Expt.0. Let $\text{Adv}^\text{sphf}_{\lambda}^\text{Expt.0}(\lambda)$ denote the advantage $\text{Adv}$ of $A$ in experiment Expt. $i$. To prove the desired bound on $\text{Adv}^\text{sphf}_{\lambda}^\text{Expt.6}(\lambda)$, we bound the effect of each change in the experiment on the advantage of $A$, and then show that

$$\text{Adv}^\text{sphf}_{\lambda}^\text{Expt.6}(\lambda) \leq C' \cdot Q^s(\lambda).$$

**Experiment Expt.0.** This is the real attack game, whose advantage is quantified as $\text{Adv}^\text{Expt.0}_{\lambda}(\lambda) = \varepsilon$. Then, in order to make the trivial attacks possible, we incrementally change the process of simulation. In this experiment, all honest players have their private input values that can be used by the simulator. Following [23], [28], there exist two types of Send queries:

- $\text{Send}_0(C, i, S)$-query. The adversary first requires the oracle $\text{Send}_0(\cdot)$ to initiate an execution between an instance $\Pi_C$ of $C$ and an instance of $S$. Then, $C$ queries $S$ to initiate the execution and $S$ answers the query by a flow and sends to communicate with $C$.
- $\text{Send}_1(C, i, \text{msg})$-query. The adversary sends msg to $\Pi_C$ via the oracle. The oracle provides no answer/response, but defines (or computes) his/her own session key, for possible later Reveal or Test queries from the $\Pi_C$.

**Experiment Expt.1.** Expt.1 is identical to Expt.0 except that we modify the way how to deal with Execute-queries. Concretely, in response to $\text{Execute}(U_C, i, U_S, j)$, we use the encryption of fake password $pw_0$ from the Zipf distribution to replace the ciphertext $ct_C$ and $ct_S$. Clearly, the fake password $pw_0$ is not in language $L$. Moreover, due to the hash key and projective key are known by the players, they can compute the common session key:

$$\text{key}_C = \text{Hash}(hk_C, W_S := (ct_S, pw)) \cdot \text{ProjHash}(ph_S, W_C := (ct_C, pw); w_C = r)$$
$$= \text{Hash}(hk_S, W_C := (ct_C, pw)) \cdot \text{ProjHash}(ph_C, W_S := (ct_S, pw); w_S = \bar{r})$$
$$= \text{key}_S.$$
Due to the soundness property of the SPHF, there is no impact to compute key via the different way using either the initial way or the modified way. In fact, this is indistinguishable property of the probabilistic encryption scheme, for each Execute-query. Thus, we can obtain

$$|\text{Adv}_A^{\text{Expt.1}}(\lambda) - \text{Adv}_A^{\text{Expt.0}}(\lambda)| \leq \text{negl}(\lambda) \quad (5.3)$$

by using a series of hybrid hops.

**Experiment Expt.2.** In this experiment, again, we modify the manner how one responds to Execute-queries. We sample a random value from uniform distribution, and use it to replace the common session key. In this setting, the “password” is not satisfied, and the indistinguishability property is guaranteed by the smoothness, i.e.,

$$|\text{Adv}_A^{\text{Expt.2}}(\lambda) - \text{Adv}_A^{\text{Expt.1}}(\lambda)| \leq \text{negl}(\lambda). \quad (5.4)$$

**Experiment Expt.3.** Expt.3 is identical to Expt.2, except that we change the way of how one deals with Send1-queries. Concretely, in this experiment, to answer the query of Send1(C, i, msg) along with msg = (phS, cS), the simulator in the name of the U_T introduces a decryption oracle (or knowing the decryption key more precisely) to decrypt the “unused” received message msg = (phS, cS). Thus, there are three cases as follows:

1) If msg has been altered (or generated) by the adversary, then one can invoke the decryption oracle to recover the password pw contained in word W by decrypting the ciphertext. Hence, there exists the following two cases:
   a) One can assert that the adversary A wins the game and the game then is terminated, if they are both correct W ∈ L and consistent with the values of the receiver (pwC, S = pw).

b) One needs to choose key at random, if they are both incorrect and/or inconsistent with the values of receiver.

2) If the msg is used previously (or a response to a previous flow sent by the simulator), then the simulator knows the hash key and obtains the projective key. Therefore, the simulator can compute the common session key by using the hash key and the projective key. Specifically, key = Hash(hkC, W_S := (cS, pw)) · ProjHash(phS, W_C := (cS, pw); w_C = r), where cS is not generated by using the randomness. This is similar to Expt.2.

For convenience, we define the first case (1a) as Event $E_v$, whose probability is computed in Expt.6. We note that the change in case (1a) can only increase the advantage of $A$. The second change in case (1b) only increases the advantage of the adversary by a negligible term due to it being indistinguishable under the adaptive-smoothness. Meanwhile, the third change in case (2) has no impact on the way how to compute the key, so finally

$$|\text{Adv}_A^{\text{Expt.3}}(\lambda) - \text{Adv}_A^{\text{Expt.2}}(\lambda)| \leq \text{negl}(\lambda). \quad (5.5)$$

**Experiment Expt.4.** We change the way of how Send1-queries response is formulated. In this experiment, two cases will appear after a “used” message msg = (phS, cS) is sent. In more detail:

- If there exists an instance $\Pi'_S$ of $U_S$ partnered with an instance $\Pi'_C$ of $U_C$, then set $key = \text{sky}'_C = \text{sky}'_S$.
- Otherwise, one chooses key at random.

Note that, in the first case, the “used” message is a replay of a previous flow. Thus, the common session key remains identical. In the second case, as in [23], [28], due to adaptive-smoothness, even if when hashing keys and ciphertexts are re-used, all the hash values are random.

---

**Fig. 4.** One-round Lattice-based PAKE Protocol Based on Miccianio-Peikert Scheme
looking. Hence, the indistinguishable holds and there exists

$$|\text{Adv}_{A}^{\text{Expt.}4}(\lambda) - \text{Adv}_{A}^{\text{Expt.}3}(\lambda)| \leq \text{negl}(\lambda). \quad (5.6)$$

**Experiment Expt.5.** We now modify the way how we deal with Send_0-queries. In this experiment, instead of encrypting the correct and real passwords, one encrypts the fake $pw_0$ (similar to Expt.1 for Execute-queries in answering Send_0$(C, i, S)$). This is necessary to simulate the decryption of Send_1-queries, and therefore indistinguishability holds for IND-CCA-secure PKE scheme. Therefore, we have

$$|\text{Adv}_{A}^{\text{Expt.}5}(\lambda) - \text{Adv}_{A}^{\text{Expt.}4}(\lambda)| \leq \text{negl}(\lambda). \quad (5.7)$$

**Experiment Expt.6.** This experiment is identical to Expt.5, with the exception of using the dummy private inputs for the hash key $hk$ and the projective key $ph$. Concretely, $hk$ and $ph$ do not depend on $W$, and the distributions of these keys are independent of the auxiliary private inputs. Hence,

$$|\text{Adv}_{A}^{\text{Expt.}6}(\lambda) - \text{Adv}_{A}^{\text{Expt.}5}(\lambda)| \leq \text{negl}(\lambda). \quad (5.8)$$

Putting them (i.e., Eq.(5.8)+Eq.(5.7)+⋯+Eq.(5.3)) together, we obtain the following result

$$\text{Adv}_{A}^{\text{Expt.}6}(\lambda) \geq \text{Adv}_{A}^{\text{Expt.}0}(\lambda) - \text{negl}(\lambda) = \varepsilon - \text{negl}(\lambda). \quad (5.9)$$

Actually, Expt.6 is only used to determine whether the adversary wins $Ev$. So the advantage is exactly: $\text{Adv}_{A}^{\text{Expt.}6}(\lambda) = \Pr[Ev]$. Therefore, we have $\varepsilon \leq \Pr[Ev] + \text{negl}(\lambda)$. As discussed earlier, event $Ev$ implies that the adversary $A$ has encrypted the password (i.e., $pw$), where the password is in the correct ($W \in L$) and consistent with the values of the receiver ($pw_{C,S} = pw$). Actually, the witnesses (or the randomness) for the honest parties are never used during the simulation, we can assume that they are chosen at the very end and only used to verify whether event $Ev$ happened:

$$\Pr[Ev] = \Pr[\exists k : pw_{C,S}(k) = pw_k, W \in L]. \quad (5.10)$$

In the above equation, $k$ is the index of the reception of $k$-th Send_1-query. In other words, it has to first guess the private values, and then once it has correctly guessed these values, it has to find a word in the language. Hence,

$$\Pr[Ev] \leq C' \cdot Q_{\varepsilon}^{\varepsilon}(\lambda), \quad (5.11)$$

where $C' \cdot Q_{\varepsilon}^{\varepsilon}(\lambda)$ is the best chance of success that the adversary can have in finding a word $W$ in $L$. Lastly, combining all the above inequalities, we have

$$\varepsilon \leq C' \cdot Q_{\varepsilon}^{\varepsilon}(\lambda) + \text{negl}(\lambda). \quad (5.12)$$

This completes the proof.

6 **Performance evaluation**

We compare the proposed PAKE protocol with several others in this section, and the comparative summary is presented in Table 3. We observe that many follow-up works and optimizations [9]–[12] were proposed following the first PAKE scheme of [7]. Hence, we focus only on SPHF-based PAKE protocols. Prior to discussing the comparison results, we first remark that we use flow to denote the unidirectional communication between the participants. However, *round* can be used to denote the bidirectional communication between participants. If the messages are sent asynchronously, then the round and flow use the same notation, such as in the one-round (and two-flow) protocol [15, 23, 28]. On the contrary, if the messages are sent simultaneously, then each round contains two flows, such as in the two-round (and two-flow) protocol [14, 19, 54]. More specifically, our PAKE protocol is the first one-round PAKE over lattices with rigorous security analysis.

As shown in Table 3, we note that Benhamouda et al. [15, 50] constructed one-round PAKE over lattices, but their scheme only has a sketchy construction without a detailed security proof. Zhang and Yu [14] proposed a two-round PAKE over the lattice, but their IND-CCA-secure scheme depends on the simulation-sound NIZK proofs. As discussed earlier, constructing an efficient lattice-based simulation-sound NIZK remains an open research problem. Benhamouda et al. [15] also sketched a one-round PAKE framework based on their SPHF, sadly, no detailed construction is presented. Differing from these existing works [14, 15, 50], we present an improved MP-SPHF via the deterministic rounding function $R(\cdot)$, then used the improved MP-SPHF to design the one-round PAKE.

On the topic of protocol security, in the same Table 3, the security of KOY and GL scheme relies on the IND-CCA-secure encryption scheme. In other words, the encryption of the client and the server uses the IND-CCA-secure encryption scheme. However, the client of Jiang-Gong scheme [16] and Groc-Katz scheme [18] discards the IND-CCA-secure encryption scheme in the first round to encrypt the password $pw$, instead of using the IND-CPA-secure encryption scheme to encrypt the randomness in the first round, under the CRS model. Subsequently, to reduce the number of communication rounds, Abdalla, Benhamouda, and Pointcheval [19] designed a two-round PAKE protocol by using the proposed IND-PCA-secure encryption scheme. Note that, in this case, only the server uses the IND-PCA-secure encryption but the client uses the IND-CPA-secure encryption. However, in our one-round case and previous works, such as [15, 23], both parties (client and server) are required to adopt IND-CCA-secure encryption scheme.

Table 4 provides a comparison of our scheme with state-of-the-art [9, 14, 15]. All these schemes have the same magnitude of $hk$, $ph$, and key, without considering the size of the $pw$. Furthermore, except that our PAKE
As every IND-CCA2 scheme proposed by Katz-Vaikuntanathan to design a PAKE protocol is round-optimal, one can observe that a significant optimization is potentially the magnitude of the ciphertext. In other words, under equivalent conditions, our protocol has a better communication complexity (i.e., $\Omega((m+n)\log q)$) than previous works [14, 23] (i.e., $\Omega((m+mn)\log q)$) and supports a longer password.

7 Extensions and Applications

In this section, we discuss potential extensions to the proposed PAKE protocol.

7.1 Two-Round PAKE via Regev and MP Scheme

Abdalla et al. [19] proposed a cryptographic primitive IND-PCA-secure PKE scheme, which can be used to meet the two-round PAKE requirements. However, they presented only a generic IND-PCA-secure PKE scheme. As every IND-CCA2-secure PKE scheme is also an IND-PCA-secure PKE scheme, in this work we assume their scheme to be secure (also there is no known attack on their scheme, at the time of this research). Notably, the client needs an IND-PCA-secure PKE scheme, but the server needs an IND-PCA-secure PKE scheme. In this setting, we can follow the methodology in [19] and adopt our improved Miccianio-Peikert (i.e., labeled IND-CCA-secure) scheme to instantiate the IND-PCA-secure PKE scheme. Similarly, we use Regev’s scheme [25] as the building block, following the SPHF approach of Katz-Vaikuntanathan to design SPHF on the Regev ciphertext.

7.2 UC-Secure One-Round PAKE Over Lattices

We can also achieve UC-secure one-round PAKE protocol without increasing the number of rounds. For example, similar to approaches in [17], [18], [23], [55] and in particular [23], we can obtain the UC-secure one-round PAKE protocol over lattices. Key to this is for each participant to generate, apart from its encrypted message ($pw$) and its hash key $hk$, the NIZK proof $\pi$ that participants used to encrypt its hash key $hk$ where $\text{ProjKG}(hk) = ph$. We then can formulate the ideal functionality $\mathcal{F}_{\text{PAKE}}$ for PAKE protocol by utilizing the approach in [17].

7.3 Potential Application: IoT Device Authentication

Lastly, we can use it and implement the two-round PAKE protocol over lattices. Our scheme is also an approaching PKE scheme, following the methodology in [19] and adopt our improved Miccianio-Peikert (i.e., labeled IND-CCA-secure) scheme to instantiate the IND-PCA-secure PKE scheme. Similarly, we use Regev’s scheme [25] as the building block, following the SPHF approach of Katz-Vaikuntanathan to design SPHF on the Regev ciphertext.

Tab. 3. Properties comparison of related PAKE protocols.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Assumption</th>
<th>Security</th>
<th>SPHF</th>
<th>UC</th>
<th>Rounds(Flow)</th>
<th>Client security</th>
<th>Server security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Katz et al. [7]</td>
<td>DDH</td>
<td>CCA</td>
<td>+</td>
<td>x</td>
<td>3(3)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Gennaro-Lindell [8]</td>
<td>DDH</td>
<td>CCA</td>
<td>GL</td>
<td>+</td>
<td>3(3)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Jiang-Gong [16]</td>
<td>DDH</td>
<td>CCA&amp;CPA</td>
<td>GL</td>
<td>x</td>
<td>3(3)</td>
<td>CPA&amp;CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Katz-Vaikuntanathan [9]</td>
<td>LWE</td>
<td>CCA</td>
<td>GL</td>
<td>x</td>
<td>3(3)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Groce-Katz [18]</td>
<td>general</td>
<td>CCA&amp;CPA</td>
<td>CS</td>
<td>✓</td>
<td>3(3)</td>
<td>CPA&amp;CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Katz-Vaikuntanathan [23]</td>
<td>DLINE</td>
<td>CCA</td>
<td>CS</td>
<td>✓</td>
<td>1(2)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Canetti et al. [10]</td>
<td>DDH</td>
<td>CCA</td>
<td>+</td>
<td>x</td>
<td>3(3)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Benhamouda et al. [28]</td>
<td>SXDH</td>
<td>CCA</td>
<td>KV</td>
<td>✓</td>
<td>1(2)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Abdalla et al. [19]</td>
<td>general</td>
<td>PCA&amp;CPA</td>
<td>GL&amp;KV</td>
<td>x</td>
<td>2(2)&amp;12)</td>
<td>CPA</td>
<td>PCA</td>
</tr>
<tr>
<td>Abdalla et al. [53]</td>
<td>DDH</td>
<td>CCA</td>
<td>+</td>
<td>x</td>
<td>3(3)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Zhang-Yu [14]</td>
<td>LWE</td>
<td>CPA+NIZK</td>
<td>GL</td>
<td>x</td>
<td>2(2)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Benhamouda et al. [15]</td>
<td>LWE</td>
<td>CCA</td>
<td>KV</td>
<td>✓</td>
<td>1(2)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
<tr>
<td>Our scheme</td>
<td>LWE</td>
<td>CCA</td>
<td>KV</td>
<td>x</td>
<td>1(2)</td>
<td>CCA</td>
<td>CCA</td>
</tr>
</tbody>
</table>

† implies two PKE schemes, one is CPA-secure and the other is CCA-secure; +: achieving PAKE without SPHF.

Tab. 4. Performance comparison of PAKE protocols over lattices.

| Scheme | $|pw|$ | $|hk|$ | $|ph|$ | $|key|$ | $|ct|$ |
|--------|------|------|------|------|------|
| Katz-Vaikuntanathan [9] | $\ell$ | $O(m \log q)$ | $O(n \log q)$ | $O(c)$ | $\Omega(mn \log q)$ |
| Zhang-Yu [14] | $e \in \mathbb{Z}_q^n$ | $u = \mathbf{B}_{\text{LWE}} e^T [y - U(1, pw)^T]$ | $\mathbf{y} = A_{\text{VK}} (s, 1, pw)^T + x$ |
| Benhamouda et al. [15] | $k \in \mathbb{Z}_q^m$ | $u = \mathbf{A}_u k$ | $p \cdot h$ | $c = \mathbf{A}_u^T s + e + (0 | \text{encode}(pw))$ |

Our scheme | $n \ell_q$ | $O(m \log q)$ | $O(n \log q)$ | $O(c)$ | $\Omega(mn \log q)$ |

$A_{\text{VK}} = [B_{\text{VK}} | U]; A = [B | U]; A_u = [A | h(u)G - AR] \in \mathbb{Z}_q^{n \times m}; \ell$ is random number and $\ell_q = [\log q] = O(\log q)$.

Fig. 5. Potential IoT device authentication.
devices (like RFID tags, smart cards, and wireless sensors) as shown in Fig. 5. For example, a number of authentication schemes (e.g., [56], [57]) augment passwords with what a user has (e.g., phones and cards) and/or what a user is (e.g., fingerprints and iris) to achieve multi-factor security for security-critical applications. However, with the advent of quantum computing [58], these conventional schemes that base their security on assumptions such as the intractability of the integer factorization problem and the discrete logarithm problem) will be insecure. Very recently, Banerjee et al. [59] have developed a novel cryptography circuit that can be used to protect low-power IoT devices via lattice-based cryptography in the coming age of quantum computing. Thus, this necessitates the design of quantum-resistant authentication schemes such as our proposed protocol.

Fig. 6. System Model: One-Round PAKE for IoTs.

We present a high-level description for our system model in Fig. 6. In this model, the resource-constrained IoT devices may connect to the resource-rich server (e.g., the cloud data center) via some public channel on the Internet. The goal of attacker, as shown in Fig. 6, is to gain access to the server. To prevent such an attacker, our one-round PAKE can be used. Specifically, the client and the server first share a low-entropy password pw, and then the client (and the server) gets and sends the projective hashing key phC (resp. phS), as well as the encryption of the pw under the public key pkC (resp. pkS) to each other, where ph is obtained by the projective key generation algorithm ProjKG and the encryption of the pw is obtained via the IND-CCA-secure encryption algorithm. Finally, the both parties (e.g., client and server) compute the session key key via the corresponding inner scalar product between ProjHash and Hash.

8 CONCLUSION

With constant advances in quantum computing, efficient (e.g., round-optimal) quantum resistant authentication protocols are increasingly becoming a real-world necessity. The cryptographic primitive over lattices is one popular post-quantum cryptography. In this paper, we first revisited the methodology of KV-SPHF over lattices. Then, we designed a word-independent lattice-based SPHF (i.e., MP-SPHF) with adaptive smoothness from the Micciancio-Feikert scheme [22], and an efficient lattice-based one-round PAKE protocol whose security is also demonstrated. Furthermore, we pointed out the potentials of constructing two-round and UC-secure one-round lattice-based PAKE protocols by combining our MP-SPHF with additional assumptions. Our PAKE protocol is particularly suitable for resource-constrained devices in an IoT environment, where communication cost is a key consideration.

ACKNOWLEDGMENT

This research was supported in part by the National Natural Science Foundation of China under Grant No.61802006 and No.61802214, in part by the China Postdoctoral Science Foundation under Grants No.2018M640026 and No.2019T120019, in part by the National Natural Science Foundation of Shandong province, China under Grant No.ZR2019BF009, and in part by the applied basic research project of Qingdao city.

REFERENCES


Zengpeng Li received his Ph.D. degree in cryptography from Harbin Engineering University, China, in 2017. Currently, he is a Lecturer at Collage of Computer Science and Technology, Qingdao University, China, and a postdoctoral researcher at Lancaster University, UK. His research focus is lattice-based cryptography, distributed secure computing, and password-based cryptography. He is a member of IEEE, CCF, IEICE and CACR.

Ding Wang (M’17) received his Ph.D. degree in Information Security from Peking University, China, in 2017. Currently, he is a Lecturer and supported by the “Boya Postdoctoral Fellowship” at Peking University. He has been involved in the community as a TPC member and a reviewer for over 50 international conferences and journals. His works appear in prestigious venues like ACM CCS, Usenix Security, NDSS, IEEE TDSC and IEEE TIFS, and seven of them are selected as “ESI highly cited papers”. He received the 2018 ACM China Doctoral Dissertation Award. His research interests focus on password, authentication and provable security.