Achieving Round-Optimal PAKE Protocol via The New Lattice-Based Smooth Projective Hash Functions

March 7, 2019

Abstract

Password-Authenticated Key Exchange (PAKE) protocol is an important cryptographic primitive which can enable two players (i.e., client and server) to generate a common, shared, and high entropy session key by pre-sharing a simple and low-entropy password. We note that most of the existing PAKE protocols secure in the standard model are designed by *smooth projective hash function* (or SPHF), yet only one SPHF over lattices was proposed by Katz and Vaikuntanathan (ASIACRYPT'09) to design a 3-flow (or 3-round) PAKE protocol. On the other hand, Abdalla, Benhamouda, and Pointcheval (PKC'15) proposed a new cryptographic primitive, namely indistinguishable against plaintext checkable attacks (IND-PCA) scheme, the authors pointed out that, in a way, every IND-CCA2 encryption is IND-PCA encryption and inspired us to obtain 2-flow PAKEs by using the IND-PCA-secure scheme. However, their scheme was still based on decisional Diffee-Huffman (DDH) assumption.

Hence, designing 2-flow PAKE via an efficient lattice-based SPHF remains a challenging issue. In this paper, our starting point is the lattice-based SPHF of Katz and Vaikuntanathan (ASIACRYPT'09), which can be followed to design two new types of lattice-based SPHFs, i.e., Reg-SPHF for the IND-CPA-secure ciphertext of Regev scheme (STOC'05) and MP-SPHF for the IND-CCA-secure ciphertext of Micciancio-Peikert scheme (EUROCRYPT'12). Lastly we work along the research line and use our designed lattice-based SPHFs (Reg-SPHF and MP-SPHF) to obtain so far the first 2-flow PAKE protocol via lattice-based SPHFs in the standard model. Moreover, we point out that one can extend our lattice-based SPHFs to improve the existing PAKE protocols in the standard model.

Keywords: Password-Authenticated Key Exchange; Smooth Projective Hash Function; Lattice-based Cryptography.

1 Introduction

The notion of *Password-Authenticated Key Exchange* (hereafter PAKE) was first proposed by Katz, Ostrovsky, and Yung [29] under decisional Diffie-Hellman (DDH) assumption, which was promoted by the previous works, such as the authenticated key exchange of Bellovin and Merritt [10] (BM) and the security model of Bellare, Pointcheval, and Rogaway [7] (BPR). PAKE protocol is an important primitive which can enable two players (i.e., client and server) to generate a common, shared, and high entropy session key by using a simple and low-entropy password. Then we can use the common session key to protect communications over an insecure network. Gennaro and Lindell [24] generalized the KOY scheme (GL) by introducing the *smooth projective hash function* (or SPHF) in the BPR security model. For future convenience, we abbreviate the general KOY scheme to Gennaro-Lindell framework. Since then, PAKE received serious attention. Many cryptographers are devoting to develop the efficient PAKE protocols via the SPHFs [32, 27, 33, 19, 11, 12, 2, 4, 3].

As mentioned above, utilizing the SPHFs as the building block, the constant-round PAKE protocols can be obtained easily. Importantly, the concept of SPHF was first denoted by Cramer and Shoup [22], they used the SPHF to obtain the first indistinguishable against chosen ciphertext attacks (or IND-CCA) security encryption scheme from the indistinguishable against chosen plaintext attacks (or IND-CPA) security under the DDH assumption in the standard model. Roughly speaking, the SPHFs were defined on the NP language L over a domain X and contained two keyed functions (i.e., $\mathsf{Hash}(\cdot)$ and $\mathsf{ProjHash}(\cdot)$). Concretely, there is no efficient adversary 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 a domain X. Furthermore, the function $\mathsf{Hash}(\cdot)$ can be computed by using the hashing key hk and the function $\mathsf{ProjHash}(\cdot)$ can be computed by using the projective hashing key ph. Notably, the outputs of the two functions are same (or, to be precise, statistically indistinguishable) for a word W over the language L (i.e., $W \in L$), namely that $\mathsf{Hash}(hk, W) = \mathsf{ProjHash}(ph, W, w)$, where w is the witness and the word W contains the labeled IND-CCA ciphertext $\mathbf{c}$ and message msg. As a consequence, even knowing ph, the adversary cannot guess $\mathsf{Hash}(hk, W \in L)$. More formal definitions will be found in the following section.

On the other hand, we note that, most of the existing PAKE protocols were designed by using DDH-based SPHFs in the group (or pair) setting. However, with the advancement of the quantum computer, DDH-based cryptography cannot prevent the quantum attacks. Thus post-quantum cryptography is now receiving increasing emphasis. Lattice-based cryptography is one of the typical representatives of post-quantum cryptography. The worst-case hardness of lattice problems (such as the Short Integer Solution (SIS) problem and the Learning with Errors (LWE) problem [48, 26]) have been proved to be a phenomenal success in fully homomorphic encryption (FHE) [25]. Importantly, lattice-based cryptography quickly caught up with pairing-based cryptography. In this case, many cryptography primitives were re-constructed by using lattice-based cryptography. However, there is only one LWE-based SPHF over the lattice [32].

Moreover, most of the existing PAKE protocols under the Gennaro-Lindell framework need "3-flow" and "IND-CCA2" encryption scheme to achieve the shared and high-entropy common session key. How to reduce the number of rounds and relax the security are two important projects. Notably, Jiang and Gong [28] relaxed the security of Gennaro-Lindell framework by using the combination of an IND-CPA scheme and an IND-CCA2 scheme, but they still needed three flows to achieve this goal. Afterwards Abdalla, Eenhamouda, and Pointcheval [4] improved the Gennaro-Lindell framework and obtained a 2-flow PAKE protocol under the DDH-based SPHF. Concretely, in their scheme, the client required an indistinguishable against plaintext checkable attacks (or IND-PCA) scheme and the server required an IND-CPA scheme. [1] Our work is along this research line and we embark on the following question:

How to achieve an efficient two-flow PAKE protocol via the LWE*-based* SPHF*s?*

Our main result answers the above question in the affirmative. In short, Abdalla, Benhamouda, and Pointcheval (ABP) [4] pointed out that their IND-PCA-secure PKE scheme is also the IND-CCA2-secure PKE scheme for small message space. Inspired by the work of Abdalla et al. [4], we adopt the existing the IND-CCA-secure LWE-based Micciancio and Peikert scheme [43] to fulfil requirements of IND-PCA-secure PKE scheme. We then follow the principles of the SPHF of Katz and Vaikuntanathan [32], we give a lattice-based MP-SPHF for IND-CCA-secure Micciancio-Peikert ciphertext and a lattice-based Reg-SPHF for IND-CPA-secure Regev ciphertext [48]. Lastly, armed with Reg-SPHF and MP-SPHF, we develop two-flow PAKE via the lattice-based SPHF in line with the principles of [4]. Below we sketch out main techniques.

[1] We stress that every IND-CCA2 scheme is also IND-PCA scheme.

1.1 Our Results and Techniques

In the following, we sketch the main contributions of our work. The detailed descriptions are present in Section 3, Section 4, and Section 5 respectively.

- Firstly, we note that, compared with the decryption algorithm of Katz-Vaikuntanathan (IND-CCA-secure PKE) scheme [32], the Micciancio-Peikert scheme [43] doesn't need to invoke the Invert$(\cdot)$ algorithm many times to recover the plaintext. Hence, in Section 4, we first revisited Micciancio-Peikert scheme, we then use IND-CCA-secure Micciancio-Peikert scheme to replace the IND-CCA-secure Katz-Vaikuntanathan scheme and design lattice-based MP-SPHF by following the approach of Katz and Vaikuntanathan [32]. Similar Reg-SPHF is obtained using analogues of the approach of [32] in Section 3. Moreover, we obtain the final result of the functions via the deterministic rounding function to replace the "simple error correcting code".

- Secondly, we note that, reducing the number of communication rounds is always an important research hotspot. In short, Benhamouda et al. (CRYPTO'13) followed the framework of Katz and Vaikuntananthan (TCC'11) and proposed a new 1-round (but 2-flow) PAKE protocol via trapdoor-SPHF, but their scheme was still based on decisional Diffee-Huffman (DDH) assumption and required the parties send the message synchronously. Afterwards, Abdalla, Benhamouda, and Pointcheval [4] constructed 2-flow PAKE protocol by introducing the new IND-PCA-secure cryptographic primitive, which relaxes the request then to send the message asynchronously. We assume their IND-PCA-secure scheme is secure and no attacks on their scheme were probably not known. Hence, in this paper, armed with we constructed Reg-SPHF and MP-SPHF over lattices, we design 2-flow PAKE protocol in line with the principle of [4].

- Thirdly, inspired by a series of works of Wang et al. [52, 51], we introduce the Zipf distribution [49, 50] to help us analyze the advantage of the adversary. In short, traditionally, we always consider a probabilistic polynomial time (PPT) adversary with advantage to distinguish the real password distribution from the uniform distribution. However, in reality, the real password distribution follows the Zipf distribution. The advantages of the adversary against PAKE protocols can be 2 4 orders of magnitude more accurately captured (formulated) by using the CDF-Zipf model [49, 50] than by using the unrealistic uniform-based assumption of password distribution. Hence, in this paper, we adopt the Zipf distribution to replace the uniform distribution.

1.2 Related Works

Below we give a history brief of PAKE and SPHF as follows.

1.2.1 PAKE

Here, we revisit some important PAKE protocols. We remark that we use flow (or round) to denote the number of communication between the parties. If the messages are sent asynchronously, then the round and flow are same. But if the messages are sent simultaneously, then each one round contains two flows. Actually, Katz and Vaikuntanathan [33] proposed the general 1-round (but 2-flow) PAKE framework which requires the client and the server to send the message to each other simultaneously.

Three-flow PAKE (or Gennaro-Lindell Framework). Katz, Ostrovsky, and Yung [29] proposed the first 3-round PAKE under DDH assumption without SPHF. Followed by Gennaro and Lindell [24], they extended the Katz-Ostrovsky-Yung scheme by using the SPHF on a labeled IND-CCA-secure PKE scheme. The following year, Jiang and Gong [28] relaxed the security of Gennaro-Lindell framework with random oracles and used an IND-CPA scheme for the client side, but still used an IND-CCA2 scheme for the server side. However, these schemes were only achieved in the standard

model (under the stand-alone setting). Groce and Katz [27] extended the Jiang-Gong scheme [28] in the universal composability (UC) framework [18, 21, 20] and proved it secure. Importantly, at AsiaCrypt'09, Katz and Vaikuntanathan [32] worked along this research line and proposed the first lattice-based PAKE protocol, but they needed three rounds.

Two-flow PAKE (or Abdalla-Eenhamouda-Pointcheva Framework). Reducing the number of communication rounds is an advanced subject of PAKE. Over the years many attempts have been made to reduce the communication rounds from three to two. Only one scheme was proposed by Abdalla et al. [4] recently. They proposed a new cryptographic primitive IND-PCA-secure PKE scheme and used it to meet the 2-round requirements.

One-round (but two-flow) PAKE (or Katz-Vaikuntanathan Framework). The minimum round PAKE scheme was designed by Katz and Vaikuntanathan [33]. The authors required the client and the server sends the message to each other simultaneously. They proved the security of the protocol in the standard model and in the UC framework separately. Very recently, Benhamouda et al. [12] designed an efficient one-round PAKE protocol under the DDH-assumption via the trapdoor-SPHF on a Cramer-Shoup ciphertext. However, there is no efficient one-round PAKE protocol over the lattice.

1.2.2 SPHF

The concept of SPHF was first proposed by Cramer and Shoup [22] for designing PAKE protocol. For future convenience, we abbreviate it to CS-SPHF. The syntax is as follows: $\{hk \leftarrow \mathsf{HashKG}(L); ph \leftarrow \mathsf{ProjKG}(hk, pk, \perp); h \leftarrow \mathsf{Hash}(hk, W); p \leftarrow \mathsf{ProjHash}(ph, W, w)\}$. Subsequently, Gennaro and Lindell [24] optimized the CS-SPHF, but made the projective key ph dependent on the word W. We abbreviate it to GL-SPHF. The syntax is as follows, $\{hk \leftarrow \mathsf{HashKG}(L); ph \leftarrow \mathsf{ProjKG}(hk, pk, W); h \leftarrow \mathsf{Hash}(hk, W); p \leftarrow \mathsf{ProjHash}(ph, W, w)\}$. Obliviously, the CS-SPHF and GL-SPHF only achieved non-adaptive smoothness, namely that the word W is independent of the project key ph and the adversary cannot see the projective key ph before choosing the word W. [2] Very recently, Katz and Vaikuntanathan [32] achieved adaptive smoothness [3] and made the projective key ph independent on the word W. We abbreviate it to KV-SPHF. The syntax is as follows, $\{hk \leftarrow \mathsf{HashKG}(L); ph \leftarrow \mathsf{ProjKG}(hk, pk, \perp); h \leftarrow \mathsf{Hash}(hk, f(W)); p \leftarrow \mathsf{ProjHash}(ph, W, w)\}$. We stress that f is an adaptive function which means the adversary can see the projective key ph before choosing the word W.

1.2.3 IND-CPA-Secure PKE Schemes Over Lattices

To the best of knowledge, since the first lattice-based PKE scheme with worst-case/average-case equivalence was first proposed by Ajtai and Dwork [5, 6], various lattice-based PKE schemes were proposed. One of the most classic LWE-based IND-CPA-secure PKE schemes is Regev scheme [48]. Subsequently, Gentry, Peikert, and Vaikuntanathan [26] proposed anther LWE-based IND-CPA-secure PKE scheme which is called dual Regev scheme. In a nutshell, the duality property is that the random vector of Regev scheme's public key is transformed into the random vector of dual Regev scheme's ciphertext. Currently, most of the lattice-based encryption algorithms are designed by using the above two schemes [48, 26] as the building block. Such as fully homomorphic encryption [25, 34, 36, 35, 38, 40], identity-based encryption [26, 37], and other lattice-based private information retrieval protocols [17, 39]. Meanwhile, there are lots of improved or optimized LWE-based IND-CPA-secure PKE schemes [46, 41].

[2] Even if we can see the projective key ph, we cannot change the word W.

[3] We can choose the word W after having seen the projective key ph.

1.2.4 IND-CCA-Secure PKE Schemes Over Lattices

To our knowledge, there are only three efficient IND-CCA-secure public key encryption (or PKE) schemes over lattices. Concretely, Peikert and Waters [47] proposed the first IND-CCA-secure PKE scheme under the (worst-case) lattice assumption, along with some optimizations [45], but their schemes were based on lossy trapdoor functions. Importantly, Katz and Vaikuntanathan [32] (KV) adopted Regev scheme [48] as the building block and designed the IND-CCA-secure PKE scheme. However, the decryption algorithm of Katz-Vaikuntanathan scheme needs to invoke the $\mathsf{BBDSolve}(\cdot)$ procedure many times. Subsequently, Micciancio and Peikert [43] proposed a new efficient IND-CCA-secure PKE scheme by introducing the **G**-trapdoor function and tweaking the decryption algorithm, i.e., they recovered the plaintext by attempting to query trapdoor inversion procedure $\mathsf{Invert}(\cdot)$ many times.

1.3 Paper Organization

In Section 2 we review related notions. In Section 3, we describe our Reg-SPHF scheme. In Section 4, we describe the improved SPHF over lattices. In Section 5, we describe the proposed 2-flow PAKE protocol via our lattice-based SPHFs. In Section 6, we present three ways to extend our scheme. In Section 7, we give a conclusion.

2 Preliminaries

We denote the security parameter by λ. We denote vector $\mathbf{x}$ via bold lower-case letter and matrix $\mathbf{A}$ via bold upper-case letter. An m-dimensional lattice can be written $\Lambda = \{\mathbf{Bs} \mid \mathbf{s} \in \mathbb{Z}^n\}$, where $\mathbf{B} \in \mathbb{Z}^{m \times n}$ is called basis of Λ for $m \geq n\lceil \log q \rceil$. Notably, the determinant of Λ is $det(\Lambda) = \sqrt{det(\mathbf{B}^T\mathbf{B})}$. We define the following q-ary lattices as

$$\begin{aligned} \Lambda(\mathbf{A}) &= \{\mathbf{A} \cdot \mathbf{s} \mid \mathbf{s} \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m \\ \Lambda^{\perp}(\mathbf{A}) &= \{\mathbf{z} \in \mathbb{Z}^m \mid \mathbf{z}^T \cdot \mathbf{A} = \mathbf{0} \pmod q\} \\ \Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}) &= \{\mathbf{z} \in \mathbb{Z}^m \mid \mathbf{z}^T \cdot \mathbf{A} = \mathbf{u} \pmod q\} \end{aligned}$$

We stress that $\Lambda(\mathbf{A})$ and $\Lambda^{\perp}(\mathbf{A})$ are dual of each other. $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A})$ is the coset of $\Lambda^{\perp}(\mathbf{A})$ for a syndrome $\mathbf{u} \in \mathbb{Z}_q^n$.

In order to discard the noise elements, we adopt the typical deterministic rounding function which is similar to the scheme of Katz and Vaikuntanathan [32].

Definition 2.1 (The Square-Signal Function, [32])**.** *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 2.2 (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 2.3 ([15] Def2.1)**.** *A distribution ensemble* $\chi = \chi(\lambda)$ *over the integers is called* B*-bounded (denoted* $|\chi| \leq B$*) if there exists:*

$$\Pr_{x \xleftarrow{\$} \chi}[|x| \geq B] \leq 2^{-\tilde{\Omega}(n)}$$

Lemma 2.4 (Matrix-vector Leftover Hash Lemma (LHL), [16] Lemma 2.1)**.** *Let $\lambda \in \mathbb{Z}$, $n,\ q \in \mathbb{N}$, $m \geq n\log q + 2\lambda$, $\mathbf{r} \xleftarrow{R} \{0,1\}^m$ and $\mathbf{y} \xleftarrow{R} \mathbb{Z}_q^n$. Sample a uniform random matrix $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{m\times n}$, then the statistical distance between the distributions $(\mathbf{A}, \mathbf{A}^T\mathbf{r})$ and $(\mathbf{A}, \mathbf{y})$ is as follows:*

$$\Delta\big((\mathbf{A}, \mathbf{A}^T \cdot \mathbf{r}), (\mathbf{A}, \mathbf{y})\big) \leq 2^{-\lambda} \tag{2.1}$$

Lemma 2.5 ([42] Lemma 4.4)**.** *1. For $\forall\ k > 0$, $\Pr[|e| > k\cdot\sigma, e \leftarrow \mathcal{D}_\sigma^1] \leq 2\cdot\exp(-\frac{k^2}{2})$;*

2. For $\forall\ k > 0$, $\Pr[\|\mathbf{e}\| > k\cdot\sigma\cdot\sqrt{m}, \mathbf{e} \leftarrow \mathcal{D}_\sigma^m] \leq k^m\cdot\exp(\frac{m}{2}\cdot(1-k^2))$.

Remark 2.6. *Throughout the paper, we suppose $\sigma \geq 2\sqrt{n}$. Therefore, if $\mathbf{e} \leftarrow \mathcal{D}_\sigma^m$ then we have, on average, that $\|\mathbf{e}\| \approx \sqrt{m}\cdot\sigma$. Lemma 2.5 (2) implies that $\|\mathbf{e}\| \leq 2\sigma\sqrt{m}$ with overwhelming probability.*

Definition 2.7 (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 2.8. *Regev and others [48, 45, 47, 44] show that reductions between the LWE assumption and approximating the shortest vector problem in lattices (for appropriate parameters). We omit the corollary of these schemes' results. More details will be found in [48, 45, 47, 44].*

Lemma 2.9 ([43])**.** *For any $N \geq m\lceil\log q\rceil$ there exists a computable gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{m\times N}$ and an efficiently computable deterministic inverse (a.k.a., "short preimage") function $\mathbf{G}^{-1}(\cdot)$. The inverse function $\mathbf{G}^{-1}(\mathbf{M})$ takes as input a matrix $\mathbf{M} \in \mathbb{Z}_q^{m\times m'}$ for any m' and outputs a matrix $\mathbf{G}^{-1}(\mathbf{M}) \in \{0,1\}^{N\times m'}$ such that $\mathbf{G}\mathbf{G}^{-1}(\mathbf{M}) = \mathbf{M}$.*

Lemma 2.10 (From [43])**.** *In order to invert the injective trapdoor function $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^T\cdot\mathbf{A} + \mathbf{e} \pmod q$, there exists a PPT algorithm $\mathsf{Invert}(\cdot)$ with 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 $\mathbf{A}\cdot\begin{pmatrix}\mathbf{R}\\ \mathbf{I}\end{pmatrix} = \mathbf{H}\cdot\mathbf{G}$ 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 $\mathbf{b}^T\cdot\begin{pmatrix}\mathbf{R}\\ \mathbf{I}\end{pmatrix}$.) and the noise vector $\mathbf{e} = \mathbf{b} - \mathbf{A}^T\mathbf{s}$.*

2.2 Smooth Projective Hash Functions

Cramer and Shoup [22] 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 $X \setminus L$. The main feature of SPHF is that, the outputs of the function $\mathsf{Hash}(hk, W)$ and $\mathsf{ProjHash}(ph, W, w)$ are same (or, to be precise, statistically indistinguishable) for a word W over the language L (i.e., $W \in L$), where w is the witness and the word W contains the labeled IND-CCA ciphertext $\mathbf{c}$ and message msg. More precisely, an SPHF over $L \subseteq X$ is defined by four probabilistic polynomial time (PPT) algorithms.

- $hk \leftarrow \mathsf{HashKG}(L)$. The algorithm takes an NP language L as input and outputs a "private" hash key hk.
- $ph \leftarrow \mathsf{ProjKG}(hk, L, W)$. 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.

- $h \leftarrow \mathsf{Hash}(hk, L, W)$. 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 $v = \Omega(\lambda)$.
- $\mathsf{ProjHash}(ph, L, W, w)$. 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, the SPHFs satisfy the notions of (approximate) correctness and smoothness:

- **Approximate Correctness:** If there exists

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

  then the approximate correctness (i.e., ε-correct) property holds.

- **Smoothness:** If the following two distributions have a negligible statistical distance in λ:

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

  then the smoothness property holds.

Here, we must stress that, if the approximate SPHF is 0-correct, then we call it SPHF. However, in lattice setting, we cannot obtain the 0-correct.

2.3 The Bellare-Pointcheval-Rogaway Security Model

In this subsection, we follow the definition of Bellare, Pointcheval, and Rogaway [7] which is the follow-up work of [8, 9, 31].

Participants, passwords, and initialization. We note that, 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 participants (also called users). For every distinct U_1, $U_2 \in \mathsf{U}$, 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 drawn from the password space $D(\lambda)$ according to the Zipf's law [49].

Execution of the protocol. In the real world, a protocol describes how users' behave after receiving input from their environment. In the formal model, these inputs are provided by the adversary. In this model, each party can execute the protocol multiple times (possibly concurrently) with different partners. Meanwhile, this model allows each party to have an unlimited number of instances. 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 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 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 the adversary.

- $\mathsf{Execute}(U_C, i, U_S, j)$. 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 complete 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). $\mathsf{sid}^i_U = \mathsf{sid}^i_{U_C} \neq \text{NULL}$; 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 $\mathsf{acc}^i_{U_C} = \mathsf{acc}^j_{U_S} = \text{TRUE}$ and $\mathsf{skey}^i_{U_C} = \mathsf{skey}^j_{U_S}$ and they both obtained the common session key.

Advantage of the adversary.

Definition 2.11 ([33]). *For all* PPT *adversaries* $\mathcal{A}$ *making at most* $Q(\lambda)$ *on-line 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.*

Remark 2.12. *Here we remark that, in the existing uniform-model, the advantage of* $\mathcal{A}$ *is based on* $Q(\lambda)/D + \text{negl}(\lambda)$ *for all dictionary sizes* D. *But In this paper, we follow the methodology of [50, 51]. We consider the Zipf-model[50], the advantage of* $\mathcal{A}$ *is based on* $C' \cdot Q^{s'}(\lambda) + \text{negl}(\lambda)$ *for the Zipf parameters* C' *and* s'.

3 SPHF From Regev Scheme

In this section, we first review the Regev scheme [48], we then follow the methodology of Li et al. [36] and present the multi-bit Regev scheme which contains a sequence of LWE instances, for future convenience, we abbreviate it to MReg. Below, we present our contribution MReg scheme.

3.1 Multi-Bit Regev Scheme (MReg Scheme)

- $\mathsf{params} \leftarrow \mathsf{MReg.Setup}(1^\lambda)$:

  1. The $\mathsf{Setup}(\cdot)$ takes the parameters λ, $\chi = \chi(\lambda)$, $n = n(\lambda)$ and $m = m(\lambda) = O(n \log(q))$ as denoted above so that the (m, n, q, χ)-LWE assumption achieves at least 2^λ security against known attacks.
  2. Chooses a parameter $t = O(\log(n))$ (the number of secret keys) and outputs $\mathsf{params} = (n, q, \chi, m, t)$.

- $sk \leftarrow \mathsf{MReg.KeyGen}(\mathsf{params})$:

  1. Samples $\mathbf{t} \leftarrow \mathbb{Z}_q^{(n-t)\times 1}$, $i \in [t]$ and outputs $sk_i = \mathbf{s}_i \leftarrow (\mathbf{I}_i, -\mathbf{t}_i^T)^T = (0, \cdots, 1, \cdots, 0 \mid -t_{i,1}, \cdots, -t_{i,n}) \in \mathbb{Z}_q^{1\times n}$, where the i-th position is 1, then chooses $\mathbf{B} \leftarrow \mathbb{Z}_q^{m\times(n-t)}$ uniformly and t vectors $\mathbf{e}_i \leftarrow \chi^m$;
  2. Outputs $pk \leftarrow \mathbf{A} = [\mathbf{b}_1|\cdots|\mathbf{b}_t|\mathbf{B}] \in \mathbb{Z}_q^{m\times n}$ where $\mathbf{b}_i = \mathbf{B}\mathbf{t}_i + \mathbf{e}_i \in \mathbb{Z}_q^m$ and $sk \leftarrow \mathbf{S} := \{\mathbf{s}_1, \cdots, \mathbf{s}_t\} \in \mathbb{Z}_q^{n\times t}$.

- $\mathbf{c} \leftarrow \mathsf{MReg.Enc}(\mathsf{params}, pk, \mathbf{m})$:

  1. Sets $\mathbf{m} := \left(\mathbf{u}^{1\times t} \mid \mathbf{0}^{1\times(n-t)}\right) = \left(u_1, \cdots, u_t \mid 0, \cdots, 0\right) \in \mathbb{Z}_q^{1\times n}$, $u_i \in \{0, 1\}$;
  2. Chooses $\mathbf{r} \leftarrow \mathbb{Z}_q^{m\times 1}$ and computes $\mathbf{c} = \mathbf{A}^T \cdot \mathbf{r} + \lfloor \frac{q}{2} \rfloor \cdot \mathbf{m} \in \mathbb{Z}_q^{n\times 1}$, where the size of ciphertext is $O(n \log^2 q)$.

  Below, we present the decryption algorithm. We show a flexibility decryption algorithm, that's means we can decrypt the ciphertext in a bit-by-bit manner.

- $u_i \leftarrow \mathsf{MReg.bitDec}(\mathsf{params}, sk, \mathbf{c})$:

  1. If we want to get the i-th position bit of message, then we sample i-th column vector $\mathbf{s}_i$ from $\mathbf{S}$;
  2. Computes and outputs $\langle \mathbf{c}, \mathbf{s} \rangle = \langle \mathbf{A}^T \cdot \mathbf{r} + \lfloor \frac{q}{2} \rfloor \cdot \mathbf{m}, \rangle = \lfloor \frac{q}{2} \rfloor \cdot u_i + \mathbf{r}^T \cdot \mathbf{e}_i \pmod q$.

  If $\|\langle \mathbf{c}, \mathbf{S} \rangle\| \leq mB < q/4$, then sets $u_i = 1$ and otherwise sets $u_i = 0$. Outputs u_i.

3.1.1 Correctness

We analyze the noise magnitude along the execution of encryption and decryption algorithms. THe correctness of the MReg scheme is as follows:

Lemma 3.1 (Correctness)**.** *If the the noise magnitude was bounded by $\lfloor q/2 \rfloor/2$ for the ciphertext $\mathbf{c} = \mathbf{A}^T \cdot \mathbf{r} + \lfloor \frac{q}{2} \rfloor \cdot \mathbf{m} \pmod q \in \mathbb{Z}_q^{n\times 1}$, then the ciphertext can be decrypted correctly.*

Proof. Consider the secret key $sk_i = \mathbf{s}_i \in \mathbb{Z}_q^{n\times 1}$ for the ciphertext, we have that

$$\mathbf{c}^T \cdot \mathbf{s}_i = \lfloor \frac{q}{2} \rfloor \cdot u_i + \mathbf{r}^T \cdot \mathbf{e}_i \pmod q,$$

with $\|\mathbf{r}^T \cdot \mathbf{e}_i\| < E$. We remark that, for $\mathbf{e} \leftarrow \chi^m$ and $\forall e_i \leftarrow \chi$, $|x_i| \leq B$ (where $B \ll q$ is a bound on the values of χ). By definition, we get $\langle \mathbf{c}, \mathbf{s}_i \rangle = \lfloor q/2 \rfloor \cdot u_i + \mathbf{r}^T \cdot \mathbf{A} \cdot \mathbf{s}_i = \lfloor q/2 \rfloor u_i + \mathbf{r}^T \cdot \mathbf{e}_i \pmod q$ with $||\mathbf{r}^T \cdot \mathbf{e}_i|| \leq \sqrt{m} \cdot \sqrt{m}B := E$. Moreover, due to there exist t bits plaintexts, so the whole of the noises were bounded by $t\|\mathbf{r}^T \cdot \mathbf{e}_i\| \leq tE < \lfloor q/2 \rfloor 2$. □

3.1.2 Security

Below we use the main result to prove the security of multi-bit encryption scheme.

Theorem 3.2 ([34])**.** *If the (n, q, χ, m)-LWE assumption is hard for the parameters $m > n \in \mathbb{N}$, $q \in \mathbb{N}$, $t = O(\log(n))$, and $\chi \leftarrow \mathbb{Z}$, then the following two distributions $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable.*

- $\mathcal{X}$ *is the distribution on* $m \times n$ *matrices* $[\mathbf{b}_1 = \mathbf{B}\mathbf{t}_1 + \mathbf{e}_1 \pmod q | \cdots | \mathbf{b}_t = \mathbf{B}\mathbf{t}_t + \mathbf{e}_t \pmod q | \mathbf{B}]$ *where* $\mathbf{B} \in \mathbb{Z}_q^{m\times(n-t)}$ *is chosen uniformly at random and where, for all* $1 \le i \le t$, $\mathbf{t}_i$ *is sampled uniformly from* $\mathbb{Z}_q^{(n-t)}$ *and* $\mathbf{e}_i$ *is sampled from a discrete Gaussian distribution* χ.
- $\mathcal{Y}$ *is the uniform distribution on* $\mathbb{Z}_q^{m\times n}$.

The following theorem formalizes the key result used to show the security of MReg scheme. We show the scheme is IND-CPA-secure based on the LWE assumption by using Theorem 3.2 to show that the scheme is indistinguishable from the orginal Reg scheme [48].

Theorem 3.3. *Let* $\mathsf{params} = (n, q, \chi, m, t)$ *be such that the* $\mathsf{LWE}_{n,q,\chi,m}$ *assumption holds and* $m = O(n \log q)$. *Then the* MReg *scheme is IND-CPA-secure.*

Proof. Below we present a sketch of the proof:

- Firstly, we apply Theorem 3.2 to show that, under LWE assumption, the matrix $\mathbf{A} = [\mathbf{b}_1, \cdots, \mathbf{b}_t \mid \mathbf{B}] \in \mathbb{Z}_q^{m\times n)}$ is computationally indistinguishable from a randomly chosen matrix;
- Secondly, we use the leftover hash lemma to show that, there is no efficient adversary can tell the difference between $\mathbf{c}$ and uniformly distribution.

This concludes the proof of the theorem. □

Below, we will describe how to follow the Katz-Vaikuntanathan framework [32] to design the SPHF for the ciphertext of Regev scheme [48].

3.2 Reg-SPHF From Regev Scheme

It is well known that Regev scheme is one of the most classical IND-CPA-secure scheme under the decisional LWE assumption. The others are Gentry-Peikert-Vaikuntanathan (a.k.a., dual-Regev) scheme [26] and Lindell-Peikert scheme [41]. In this paper, In line with the principles of the SPHF of Katz-Vaikuntanathan (KV) construction [32], we adopt Regev schem as the building block to design the SPHF, for simplicity, we abbreviate it to Reg-SPHF. We remark that the others lattice-based PKE scheme also can be used to design the SPHF which follows the form of Katz-Vaikuntanathan.

- $hk \leftarrow \mathsf{Reg.HashKG}(\mathsf{params})$: The algorithm samples a random vector $\mathbf{h}$ from $\mathbb{Z}_q^{n\times 1}$. Then outputs the hashing key $hk := \mathbf{h} \in \mathbb{Z}_q^{m\times 1}$.
- $ph \leftarrow \mathsf{Reg.ProjKG}(\mathsf{params}, hk = \mathbf{h}, pk = \mathbf{A})$: The algorithm takes the hashing key $\mathbf{h}$ and the public key $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ as input, 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".
- $h \leftarrow \mathsf{Reg.Hash}(hk = \mathbf{h}, W := (c, \mathbf{m}))$: (Smooth hash function)
  1. The algorithm takes the hashing key $\mathbf{h}$ and the word W as input, 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. Next the algorithm obtains the result of $b := h \pmod 2 \in \{0,1\}$, namely that the value h is a number in $[-(q-1)/2, \cdots, (q-1)/2]$ and the algorithm outputs $b = 0$ if $h < 0$, otherwise, outputs $b = 1$.

- $p = \mathsf{Reg.ProjHash}(ph = \mathbf{p}_{reg}, W := (c, \mathbf{m}); w = \mathbf{s})$(Projection)

1. The algorithm takes a projected key $ph = \mathbf{p}_{reg} \in \mathbb{Z}_q^{n\times 1}$, the word W, and the witness $\mathbf{s} \in \mathbb{Z}_q^{n\times 1}$ as input.
2. The algorithm computes and outputs

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

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

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

Lemma 3.4. *The* Reg-SPHF *is a smooth projective hash function for the Regev scheme.*

Proof. Below we prove the approximate correctness and smoothness respectively.

- **Projective (or Correctness).** Below we prove our scheme achieves 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 $1/2$. In lattice-based setting, we sketch that the correctness of Reg-SPHF is 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 [32].

  Lemma 3.5 (Correctness)**.** *If the inner product of $\langle \mathbf{e}, \mathbf{h} \rangle$ is small for the parameters n, $m \geq n\sqrt{\log q}$, then the outputs of the rounding function $R\Big(\mathsf{Reg.Hash}(hk = \mathbf{h}, W := (c, \mathbf{m}))\Big)$ and $R\Big(\mathsf{Reg.ProjHash}(ph = \mathbf{p}_{reg}, W := (c, \mathbf{m}); w = \mathbf{s})\Big)$ are identical.*

  Proof. In this paper, we adopt the typical deterministic rounding function $R(x) = \lfloor 2x/q \rceil \pmod 2$ and follow the methodology of [32] to round the outputs of $\mathsf{Hash}(\cdot)$ and $\mathsf{ProjHash}(\cdot)$ respectively. Consider the following equations,

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

$$\Big(\mathsf{Reg.ProjHash}(ph = \mathbf{p}_{reg}, W := (c, \mathbf{m}); w = \mathbf{s})\Big) = \Big(\mathbf{r}^T \cdot \mathbf{p}_{reg}\Big) = \Big(\mathbf{r}^T \cdot (\mathbf{Ak}) \pmod q\Big) \quad (3.2)$$

  Obliviously, the above two equations Eq.(3.1) and Eq.(3.2) are equal, then utilizing the rounding function $R(\cdot)$ which was denoted in Definition 2.1, 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. Consider the word $W := (c, \mathbf{m}) \notin L$, 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 SPHF From Micciano-Peikert Scheme

In this section, we present the construction of Micciano-Peikert scheme [43] in a simple way. To the best of our knowledge, the MP construction is IND-CCA1-secure, we can obtain the IND-CCA2 security via relatively generic transformations using either strongly unforgeable one-time signature [23], or a message authentication code (MAC) and weak form of commitment [14].

4.1 Micciano-Peikert (MP) Scheme (Labeled CCA1 Scheme)

- $\mathsf{params} \leftarrow \mathsf{MP.Setup}(\lambda, m, \bar{m}, n, q, \ell_q)$: Takes the security parameter λ, the integers m, $\bar{m}$, n, and the model q as input, where $m = \bar{m} + t$ and $\ell_q = \lceil \log q \rceil = O(\log q)$, then outputs the parameters $\mathsf{params} := (\lambda, m, \bar{m}, n, q, \ell_q)$. We remark that, for future convenience, we denote $t = n\ell_q$.

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

  1. Takes the params as input and samples a public matrix $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ along with the trapdoor matrix $\mathbf{R} \leftarrow \mathcal{D}_{\mathbb{Z}, \omega(\sqrt{\log n})}^{\bar{m} \times t}$ by invoking the trapdoor generation algorithm, i.e., $(\mathbf{P}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(\mathsf{params}, \bar{\mathbf{A}}, \mathbf{R})$. Obliviously, the dot production between the public key $\mathbf{P}$ and the trapdoor $\mathbf{T}$ is 0, i.e., $\mathbf{P} \cdot \mathbf{T} = 0 \pmod q$.
  2. Lets the matrix $\mathbf{A}_1 := -\bar{\mathbf{A}} \cdot \mathbf{R} \pmod q \in \mathbb{Z}_q^{n \times t}$ and generates the public key $\mathbf{P} := [\bar{\mathbf{A}} \mid \mathbf{A}_1] = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}} \cdot \mathbf{R}] \in \mathbb{Z}_q^{n \times m}$. Then denotes $(t \times t)$-dimension identity matrix by $\mathbf{I}_{t \times t}$ and generates the trapdoor (i.e., secret key) $\mathbf{T} := \left(\frac{\mathbf{R}}{\mathbf{I}_{(t \times t)}}\right) \in \mathbb{Z}_q^{m \times t}$.
  3. Outputs the the secret key $sk := \mathbf{T}$ and the public key $pk := \mathbf{P}$.

- $c \leftarrow \mathsf{MP.Enc}(pk = \mathbf{P}, \mathbf{m} \in \{0,1\}^t, \mathsf{label} = u)$:

  1. In order to encrypt the message $\mathbf{m} \in \{0,1\}^t$, the algorithm first maps $\mathbf{m} \in \{0,1\}^t$ to $\mathsf{encode}(\mathbf{m}) = \mathbf{S} \cdot \mathbf{m} \in \mathbb{Z}^t$ for any bisis $\mathbf{S} \in \mathbb{Z}^{t \times t}$ of lattice Λ, and takes the public key $\mathbf{P}$ as input.
  2. Moreover, samples a nonzero label $u \leftarrow \mathcal{U}$ and lets $\mathbf{A}_u = [\bar{\mathbf{A}} \mid \mathbf{A}_1 + h(u)\mathbf{G}] = [\bar{\mathbf{A}} \mid h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n \times m}$. We note that, $\mathbf{A}_u \cdot \mathbf{T} = h(u)\mathbf{G}$ for the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times t}$.

     Remark 4.1. *For the ring $\mathcal{U} = \mathbb{Z}_q[x] \setminus (f(x))$ for $q = p^e$, we denote the label set $\mathcal{U} = \{u_1, \cdots, u_\ell\} \subset \mathcal{U}$ with the "unit differences" 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. Samples a random vector $\mathbf{s} \leftarrow \mathbb{Z}_q^{n\times 1}$, a noise vector $\bar{\mathbf{e}} \leftarrow \mathcal{D}_{\mathbb{Z},\alpha q}^{\bar{m}}$, and another noise vector $\hat{\mathbf{e}} \leftarrow \mathcal{D}_{\mathbb{Z},s}^{t}$, where $s^2 = (\|\bar{\mathbf{e}}\|^2 + \bar{m}(\alpha q)^2) \cdot \omega(\sqrt{\log n})^2$. Then the algorithm cascades $\bar{\mathbf{e}}$ and $\hat{\mathbf{e}}$ together and obtains the noise vector $\mathbf{e} = (\bar{\mathbf{e}}, \hat{\mathbf{e}}) \in \mathbb{Z}^{m\times 1}$.
4. Computes and outputs

$$\mathbf{c} = \mathbf{A}_u^T \cdot \mathbf{s} + \mathbf{e} + (\mathbf{0} \mid \mathsf{encode}(\mathbf{m})) \pmod q \in \mathbb{Z}_q^{m\times 1}.$$

5. Lastly, outputs the ciphertext $c = (u, \mathbf{c}) \in \mathcal{U} \times \mathbb{Z}_q^{m\times 1}$.

- $\mathbf{m} \leftarrow \mathsf{MP.Dec}(sk, c = (u, \mathbf{c}))$: In order to decrypt the ciphertext $c = (u, \mathbf{c})$ for the label u, the algorithm proceeds the following steps:

1. The algorithm first parses c into label u and vector $\mathbf{c}$. If $u = 0$, then outputs $\perp$. Otherwise the algorithm invokes the algorithm $\mathsf{Invert}(\mathbf{A}_u, \mathbf{c}, \mathbf{R}, h(u) \in \mathbb{Z}_q^{n\times n})$ (Lemma 2.10) and gets $\mathbf{c} = \mathbf{A}_u^T\mathbf{z}_1 + \mathbf{e}_1$, namely that obtains the secret vector $\mathbf{z}_1 \in \mathbb{Z}_q^{n\times 1}$ and noise vector $\mathbf{e}_1 = (\bar{\mathbf{e}}_1, \hat{\mathbf{e}}_1) = \mathbf{c} - \mathbf{A}_u^T\mathbf{z}_1 \in \mathbb{Z}^{m\times 1}$(*i.e.*, $\mathbb{Z}^{\bar{m}\times 1} \times \mathbb{Z}^{t\times 1}$). Then the algorithm checks whether $\|\bar{\mathbf{e}}_1\| \geq \alpha q\sqrt{\bar{m}}$ or $\|\hat{\mathbf{e}}_1\| \geq \alpha q\sqrt{2\bar{m}t} \cdot \omega(\sqrt{\log n})$, and outputs $\perp$.
2. Otherwise, the decryption algorithm invokes $\mathsf{Invert}(\mathbf{A}_u, \mathbf{c} - (\mathbf{0} \mid \mathsf{encode}(\mathbf{m}), \mathbf{R}, h(u))$ and obtains $\mathbf{c} - (\mathbf{0} \mid \mathsf{encode}(\mathbf{m})) = \mathbf{A}_u^T \cdot \mathbf{z}_2 + \mathbf{e}_2$ for $\|\bar{\mathbf{e}}_2\| < \alpha q\sqrt{\bar{m}}$ and $\|\hat{\mathbf{e}}_2\| < \alpha q\sqrt{2\bar{m}t} \cdot \omega(\sqrt{\log n})$. Lets $\mathbf{v} = \mathbf{c} - \mathbf{e}_2 = \mathbf{A}_u^T \cdot \mathbf{z}_2 + (\mathbf{0} \mid \mathsf{encode}(\mathbf{m}))$ (mod q) and requires $\mathbf{v}$ to parse $\mathbf{v} = (\bar{\mathbf{v}}, \hat{\mathbf{v}}) \in \mathbb{Z}_q^{\bar{m}\times 1} \times \mathbb{Z}_q^{t\times 1}$ for $\bar{\mathbf{v}} \in \Lambda(\bar{\mathbf{A}}^T)$.
3. Finally, computes and outputs the plaintext $\mathsf{decode}(\mathbf{v}^T \cdot \left[\frac{\mathbf{R}}{\mathbf{I}}\right] \pmod q) \in \{0,1\}^t$.

4.1.1 Correctness

Lemma 4.2. *The above scheme has only $2^{-\Omega(\lambda)}$ probability of decryption error.*

The error probability can be made zero by changing $\mathsf{KeyGen}(\cdot)$ and $\mathsf{Enc}(\cdot)$ so that they resample $\mathbf{R}$, $\bar{\mathbf{e}}$, and/or $\hat{\mathbf{e}}$ in the rare event that they violate the corresponding bounds given in the proof below.

Proof. $(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{params})$ i.e., $(\mathbf{P}, \mathbf{T}) \leftarrow \mathsf{MP.KeyGen}(\mathsf{params})$ where $\mathbf{T} = \left[\frac{\mathbf{R}}{\mathbf{I}}\right]$. Then we have $s_1(\mathbf{R}) \leq O(\sqrt{t}) \cdot \omega(\sqrt{\log n})$ except with probability $2^{-\Omega(\lambda)}$. Now consider the random choices made by $\mathsf{Enc}(\mathbf{P}, \mathbf{m})$ for arbitrary $\mathbf{m} \in \{0,1\}^t$. Then we have both $\|\bar{\mathbf{e}}\| \leq \alpha q\sqrt{\bar{m}}$ and $\|\hat{\mathbf{e}}\| \leq \alpha q \cdot O(2\bar{m}t \cdot \omega(\sqrt{\log n})$, except with probability $2^{-\Omega(\lambda)}$. Letting $\mathbf{e} = (\bar{\mathbf{e}}, \hat{\mathbf{e}})$, we have $\|\mathbf{e}^T \cdot \mathbf{T}\| = \|\mathbf{e}^T \cdot \left[\frac{\mathbf{R}}{\mathbf{I}}\right]\| \leq \|\bar{\mathbf{e}}^T \cdot \mathbf{R}\| + \|\hat{\mathbf{e}}^T \cdot \mathbf{I}\| \leq \alpha q \cdot O(t) \cdot \omega(\log n)$. In particular, for large enough $\alpha = 1/\big(O(t) \cdot \omega(\log n)\big)$ we have $\mathbf{e}^T \cdot \mathbf{T} \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-T})$. Therefore, the call to Invert made by $\mathsf{Dec}(sk = \mathbf{T}, c = (u, \mathbf{c}))$ returns $\mathbf{e}$. It follows that for $\mathbf{v} = (\bar{\mathbf{v}}, \hat{\mathbf{v}}) = \mathbf{c} - \mathbf{e}$ (mod q), we have $\bar{\mathbf{v}} \in \Lambda(\bar{\mathbf{A}}^T)$ as needed. Finally, $\mathbf{v}^T \cdot \mathbf{T} = \mathbf{v}^T \cdot \left[\frac{\mathbf{R}}{\mathbf{I}}\right] = \mathbf{s}^T\mathbf{A}_u \cdot \left[\frac{\mathbf{R}}{\mathbf{I}}\right] + (\mathbf{0} \mid \mathsf{encode}(\mathbf{m}))^T \cdot \left[\frac{\mathbf{R}}{\mathbf{I}}\right] = \mathbf{s}^T \cdot h(u)\mathbf{G} + \mathsf{encode}(\mathbf{m})^T \pmod q$, which is in the coset $\mathsf{encode}(\mathbf{m}) \in \Lambda(\mathbf{G})/2\Lambda(\mathbf{G})$, and so Dec outputs $\mathbf{m}$ as desired. □

4.1.2 Security

The proof of security and correctness are similar to the IND-CCA1 MP scheme.

Theorem 4.3 (Security,from [43])**.** *The above scheme is IND-CCA1-secure under the hardness of decisional $\mathsf{LWE}_{n,m,q,\chi}$ assumption.*

Proof. The security proof can be obtained directly by following [43]. Hence, we omit the further details.

4.2 MP-SPHF From MP Scheme

It is well known that the Micciancio-Peikert scheme is labeled IND-CCA1-secure under the decisional LWE assumption. To our knowledge, if the label u is fixed or a constant at advance, the labeled IND-CCA1 MP scheme will degrade into an IND-CPA MP scheme. Moreover, if we introduce an one-time signature scheme, and we use it to sign the ciphertext under the secret key of the signature, then we can obtain an IND-CCA2-secure scheme.

In this setting, we first fix the label by $u \neq 0$ and obtain an IND-CPA 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.

- $hk \leftarrow \mathsf{MP.HashKG}(\mathsf{params})$: The algorithm samples a random vector $\mathbf{k}$ from $\mathbb{Z}_q^{n\times 1}$. Then outputs the hashing key $hk := \mathbf{k} \in \mathbb{Z}_q^{m\times 1}$.

- $ph \leftarrow \mathsf{MP.ProjKG}(\mathsf{params}, hk = \mathbf{k}, pk = \mathbf{A}_u)$: The algorithm takes the hashing key $\mathbf{k}$ and the public key $\mathbf{A}_u = [\bar{\mathbf{A}} \mid h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n\times m}$ with the fixed label u as input, then outputs the projective hashing key $ph := \mathbf{p} = \mathbf{A}_u \cdot \mathbf{k} \in \mathbb{Z}_q^{n\times 1}$. In this setting, we must stress that, the public key of MP scheme is $\mathbf{P} := [\bar{\mathbf{A}} \mid \mathbf{A}_1] = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}} \cdot \mathbf{R}] \in \mathbb{Z}_q^{n\times m}$, but in MP-SPHF scheme, in order to obtain the "approximate correctness", we modify the public key.

- $h \leftarrow \mathsf{MP.Hash}(hk = \mathbf{k}, W := (c, \mathbf{m}))$: (Smooth hash function)

  1. The algorithm takes the hashing key $\mathbf{k}$ and the word W as input, where the word 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:

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

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

  3. Next the algorithm obtains the result of $b := h \pmod 2 \in \{0,1\}$, namely that the value h is a number in $[-(q-1)/2, \cdots, (q-1)/2]$ and the algorithm outputs $b = 0$ if $h < 0$, otherwise, outputs $b = 1$.

- $p = \mathsf{MP.ProjHash}(ph = \mathbf{p}, W := (c, \mathbf{m}); w = \mathbf{s})$(Projection)

  1. The algorithm takes a projected key $ph = \mathbf{p} \in \mathbb{Z}_q^{n\times 1}$, the word W, and the witness $\mathbf{s} \in \mathbb{Z}_q^{n\times 1}$ as input.
  2. The algorithm computes and outputs

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

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

Lemma 4.4. *The MP-SPHF is a smooth projective hash function for the MP scheme.*

Proof. Below we prove the approximate correctness and smoothness respectively.

- **Projective (or Approximate Correctness).** Below we 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.

  Lemma 4.5 (Approximate Correctness)**.** *If the inner product of $\langle \mathbf{e}, \mathbf{k} \rangle$ is small for the parameters n, $m \geq n\sqrt{\log q}$, then the outputs of the rounding function $R\Big(\mathsf{Hash}(hk = \mathbf{k}, W := (c, \mathbf{m}))\Big)$ and $R\Big(\mathsf{ProjHash}(ph = \mathbf{p}, W := (c, \mathbf{m}); w = \mathbf{s})\Big)$ are identical.*

  Proof. In this paper, we adopt the typical deterministic rounding function $R(x) = \lfloor 2x/q \rceil \pmod 2$ and follow the methodology of [32] to round the outputs of $\mathsf{Hash}(\cdot)$ and $\mathsf{ProjHash}(\cdot)$ respectively. Consider the following two equations,

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

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

  Consider the equation Eq.(4.1) and the Definition 2.1, we can view the $R(h)$ as a number in $[-\frac{(q-1)}{2}, \cdots, \frac{(q-1)}{2}]$ and output $b \in \{0, 1\}$. Moreover, the noise element $\mathbf{e}^T \cdot \mathbf{k}$ is bounded by $|\mathbf{e}^T \mathbf{k}| \leq \|\mathbf{e}^T\| \cdot \|\mathbf{k}\| \leq (r\sqrt{mn}) \cdot (\alpha q \sqrt{mn}) < \varepsilon/2 \cdot q/4$. Hence, the result of $R(\mathbf{e}^T \mathbf{k})$ 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 2.1, 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. 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,

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

  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. □

References

[1] M. Abdalla, F. Ben Hamouda, and D. Pointcheval. Tighter reductions for forward-secure signature schemes. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 292–311. Springer, Heidelberg, Feb. / Mar. 2013.

[2] M. Abdalla, F. Benhamouda, O. Blazy, C. Chevalier, and D. Pointcheval. SPHF-friendly non-interactive commitments. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, Heidelberg, Dec. 2013.

[3] M. Abdalla, F. Benhamouda, and P. MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*, pages 571–587. IEEE Computer Society Press, May 2015.

[4] M. Abdalla, F. Benhamouda, and D. Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 332–352. Springer, Heidelberg, Mar. / Apr. 2015.

[5] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[6] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(65), 1996.

[7] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.

[8] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, Aug. 1994.

[9] M. Bellare and P. Rogaway. Provably secure session key distribution: The three party case. In *27th ACM STOC*, pages 57–66. ACM Press, May / June 1995.

[10] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

[11] F. Ben Hamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, Heidelberg, Feb. / Mar. 2013.

[12] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, Aug. 2013.

[13] O. Blazy, C. Chevalier, L. Ducas, and J. Pan. Exact smooth projective hash function based on LWE. Cryptology ePrint Archive, Report 2013/821, 2013. http://eprint.iacr.org/2013/821.

[14] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[15] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, Aug. 2012.

[16] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In R. Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, Oct. 2011.

[17] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, Aug. 2011.

[18] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.

[19] R. Canetti, D. Dachman-Soled, V. Vaikuntanathan, and H. Wee. Efficient password authenticated key exchange via oblivious transfer. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 449–466. Springer, Heidelberg, May 2012.

[20] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Heidelberg, May 2005.

[21] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, Heidelberg, Apr. / May 2002.

[22] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, Apr. / May 2002.

[23] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[24] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Heidelberg, May 2003. http://eprint.iacr.org/2003/032.ps.gz.

[25] C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[26] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[27] A. Groce and J. Katz. A new framework for efficient password-based authenticated key exchange. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS 10*, pages 516–525. ACM Press, Oct. 2010.

[28] S. Jiang and G. Gong. Password based key exchange with mutual authentication. In H. Handschuh and A. Hasan, editors, *SAC 2004*, volume 3357 of *LNCS*, pages 267–279. Springer, Heidelberg, Aug. 2004.

[29] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer, Heidelberg, May 2001.

[30] J. Katz, R. Ostrovsky, and M. Yung. Forward secrecy in password-only key exchange protocols. In S. Cimato, C. Galdi, and G. Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 29–44. Springer, Heidelberg, Sept. 2003.

[31] J. Katz, R. Ostrovsky, and M. Yung. Efficient and secure authenticated key exchange using weak passwords. *J. ACM*, 57(1):3:1–3:39, 2009.

[32] J. Katz and V. Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, Heidelberg, Dec. 2009.

[33] J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, Heidelberg, Mar. 2011.

[34] Z. Li, S. D. Galbraith, and C. Ma. Preventing adaptive key recovery attacks on the GSW levelled homomorphic encryption scheme. In *Provable Security - 10th International Conference, ProvSec 2016, Nanjing, China, November 10-11, 2016, Proceedings*, pages 373–383, 2016.

[35] Z. Li, C. Ma, G. Du, and W. Ouyang. Dual lwe-based fully homomorphic encryption with errorless key switching. In *22nd IEEE International Conference on Parallel and Distributed Systems, ICPADS 2016, Wuhan, China, December 13-16, 2016*, pages 1169–1174, 2016.

[36] Z. Li, C. Ma, E. Morais, and G. Du. Multi-bit leveled homomorphic encryption via dual.lwe-based. In *Information Security and Cryptology - 12th International Conference, Inscrypt 2016, Beijing, China, November 4-6, 2016, Revised Selected Papers*, pages 221–242, 2016.

[37] Z. Li, C. Ma, and D. Wang. Towards multi-hop homomorphic identity-based proxy re-encryption via branching program. *IEEE Access*, 2017, doi=10.1109/ACCESS.2017.2740720.

[38] Z. Li, C. Ma, and D. Wang. Leakage resilient leveled fhe on multiple bit message. *IEEE Transactions on Big Data*, 2017, doi=10.1109/TBDATA.2017.2726554.

[39] Z. Li, C. Ma, D. Wang, and G. Du. Toward single-server private information retrieval protocol via learning with errors. *Journal of Information Security and Applications*, 34:280–284, 2017.

[40] Z. Li, C. Ma, and H.-S. Zhou. Multi-key fhe on multi-bit messages. *SCIENCE CHINA Information Sciences*, 2017, doi = https://doi.org/10.1007/s11432-017-9206-y.

[41] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In A. Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, Feb. 2011.

[42] V. Lyubashevsky. Lattice signatures without trapdoors. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, Apr. 2012.

[43] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, Apr. 2012.

[44] P. Mukherjee and D. Wichs. Two round multiparty computation via multi-key FHE. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016.

[45] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009.

[46] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, Aug. 2008.

[47] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.

[48] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[49] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian. Zipf's law in passwords. *IEEE Trans. Inf. Forensics Security*, 2017, doi=10.1109/TIFS.2017.2721359.

[50] D. Wang and P. Wang. On the implications of Zipf's law in passwords. In I. G. Askoxylakis, S. Ioannidis, S. K. Katsikas, and C. A. Meadows, editors, *ESORICS 2016, Part I*, volume 9878 of *LNCS*, pages 111–131. Springer, Heidelberg, Sept. 2016.

[51] D. Wang and P. Wang. Two birds with one stone: Two-factor authentication with security beyond conventional bound. *IEEE Trans. Dependable Sec. Comput.*, 2017, doi=10.1109/TDSC.2016.2605087.

[52] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang. Targeted online password guessing: An underestimated threat. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16*, pages 1242–1254. ACM Press, Oct. 2016.