Efficient Password-Based Authenticated Key Exchange from Lattices

Yi Ding* and Lei Fan

Shanghai Jiao Tong University

Abstract. Protocols for password-based authenticated key exchange (PAKE) allow two users who share only a short, low-entropy password to agree on a cryptographically strong session key. Any unauthorized party who controls the communication channel but does not possess the password cannot participate in the method and is constrained as much as possible from guessing the password. One must ensure that protocols are immune to off-line dictionary attacks in which an adversary exhaustively enumerates all possible passwords in an attempt to determine the correct one. Recently Katz, et al. [6] gave a new framework for realizing PAKE without random oracles, in the common reference string model. In this paper, we instantiate the framework of [6] under the lattices assumptions. Specifically, we modified the lattice-based approximate projective hashing introduced in [9] and plug it into the framework of [6], and we prove our new PAKE is efficient and secure based on the security of GK's PAKE framework [6] in the standard model.

Keywords: PAKE; lattices; smooth projective hashing

1 Introduction

Protocols for authenticated key exchange enable two parties to generate a shared, cryptographically strong key while communicating over an insecure network under the complete control of an adversary. Such protocols are among the most widely used and fundamental cryptographic primitives; indeed, agreement on a shared key is necessary before higher-level tasks such as encryption and message authentication become possible.

Password-based authenticated key exchange (PAKE) protocols enable two users to generate a common, cryptographically-strong key based on an initial, low-entropy, shared secret (i.e., a password). The difficulty in this setting is to prevent off-line dictionary attacks where an adversary exhaustively enumerates potential passwords on its own, attempting to match the correct password to observed protocol executions. Roughly, a PAKE protocol is secure if off-line attacks are of no use and the best attack is an on-line dictionary attack where an adversary must actively try to impersonate an honest party using each possible password. On-line attacks of this sort are inherent in the model of password-based authentication; more importantly, they can be detected by the server as failed login attempts and defended against.

* 1-108 SEIEE Building, No.800 Dongchuan Road, 200240, Shanghai, China.

1.1 Related Work

The first successful password-authenticated key agreement methods were Encrypted Key Exchange methods [14] described by Steven M. Bellovin and Michael Merritt in 1992. Although several of the first methods were flawed, the surviving and enhanced forms of EKE effectively amplify a shared password into a shared key, which can then be used for encryption and/or message authentication. The first provably-secure PAKE protocols were given in work by M. Bellare, D. Pointcheval, and P. Rogaway [2] and in work [15] by V. Boyko, P. MacKenzie, and S. Patel both presented in Eurocrypt 2000. These protocols were proven secure in the random oracle model.

Katz, Ostrovsky, and Yung (KOY) [19] demonstrated the first efficient PAKE protocol with a proof of security in the standard model. Their protocol was later abstracted by Gennaro and Lindell (GL) [18], who gave a general framework that encompasses the original KOY protocol as a special case. These protocols are secure even under concurrent executions by the same party, but require a common reference string (CRS). While this may be less appealing than the plain model, reliance on a CRS does not appear to be a serious drawback in practice for the deployment of PAKE, where common parameters can be hard-coded into an implementation of the protocol. The KOY/GL framework requires a CCA-secure encryption scheme (such as Cramer-Shoup cryptosystem [17]) with an associated smooth projective hash function, and its extensions require four rounds in order to achieve mutual authentication. Almost all subsequent work on efficient PAKE in the standard model [4], [10], and [5] can be viewed as extending and building on the KOY/GL framework.

A different PAKE protocol in the CRS model is given by Jiang and Gong [8], later abstracted and generalized by Groce and Katz [6]. Comparing to KOY/GL framework, the new JG/GK framework only requires a CCA-secure encryption scheme, and a CPA-secure encryption scheme with an associated smooth projective hash function [3]. It also achieves mutual authentication in three rounds. In their work [6], Groce and Katz mentioned their framework will significantly improve efficiency when basing the protocol on lattice assumptions. Katz and Vaikuntanathan [9] first instantiated the KOY/GL PAKE protocol under lattice assumptions. The most technically difficult aspect of their work is the construction of a lattice-based CCA-secure encryption scheme with an associated approximate smooth projective hash system. Instantiating the exact smooth projective hash from lattice assumptions is stated as an open question in [12], and we argue that an approximate SPH is not suffice for the JG/GK PAKE framework. In order to plug into the JG/GK's framework, we use an approximate lattice-based SPH and an error correcting code(ECC) to do the job of an exact lattice-based SPH.

The use of lattice-based assumptions in cryptography has seen a bloom of activity in recent years. In part, this is due to a general desire to expand the set of assumptions on which cryptosystems can be based (i.e., beyond the standard set of assumptions related to the hardness of factoring and solving the discrete logarithm problem). Cryptographic primitives based on lattices are appealing because of known worst-case/average-case connections between lattice problems, as well as because several lattice problems are currently immune to quantum attacks. Also, the best-known algorithms for several lattice problems CVP/SVP require exponential time. Even restricting to classical attacks, the best known algorithms for solving several lattice problems require exponential time (in contrast to the sub-exponential algorithms known, e.g., for factoring). Finally, relying on lattices can potentially yield very efficient constructions because the basic lattice operations manipulate relatively small numbers and are inherently parallelizable.

1.2 Organization

Building on ideas of Groce and Katz [6], we show a new construction of an efficient PAKE based on lattices. Our work is to theirs as the work of Katz and Vaikuntanathan [9] is to that of Katz-Ostrovsky-Yung [19], and Gennaro-Lindell [18]; namely, we present a instantiation of lattice-based PAKE under the JG/KV framework.

In section II we first introduce some notation and review the necessary background on lattices. Then we define the secure PAKE protocol, and give a brief description of the JG/GK PAKE framework. Lastly we introduce the lattice-based approximate smooth projective hash system presented in [9] and argue that it cannot directly plug into the JG/GK PAKE framework. We present our PAKE protocol in section III, and later give a brief proof of the security under the LWE assumption.

2 Preliminaries

Notation. Throughout the paper we say that a function ε: $R^* \rightarrow R^*$ is negligible if $\varepsilon(n)$ is smaller than all polynomial fractions for sufficiently large n. We use column notation for vectors and use $(x_1, ..., x_n)$ to denote the column vector with entries x_1, ..., x_n. We use square brackets to enclose matrices and row vectors. Define the statistical distance, denoted $\Delta(X;Y)$, as $\Delta(X;Y) = 1/2\Sigma_{s\in\Omega}|\Pr[X = s] = \Pr[Y = s]|$, let X and Y be two random variables taking values in some finite set Ω. We say that X is uniform over Ω if $\Delta(X;U_\Omega) \leq \sigma$ where U_Ω is a uniform random variable over Ω.

2.1 Lattices and the LWE assumption

A lattice is defined as the set of all integer combinations

$$\mathcal{L}(b_1, ..., b_n) = \{\sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z}, \text{for } 1 \leq i \leq n\}$$

of n linearly independent vectors $b_1, ..., b_n$ in $\mathbb{R}^n$. The set of vectors $b_1, ..., b_n$ is called a *basis* for the lattice. A basis can be represented by the matrix $\mathrm{B} = [b_1, ..., b_n] \in \mathbb{R}^{n\times n}$ having the basis vectors as columns. Using matrix notation, the lattice generated by a matrix $\mathrm{B} \in \mathbb{R}^{n\times n}$ can be defined as $\mathcal{L}(\mathrm{B}) = \{\mathbf{Bx} : x \in \mathbb{Z}^n\}$, where $\mathbf{Bx}$ is the usual matrix-vector multiplication.

Lattice problems: The most well known computational problems on lattices are the following.

- Shortest Vector Problem (SVP): Given a lattice basis B, find the shortest non-zero vector in $\mathcal{L}(\mathrm{B})$.

- Closest Vector Problem (CVP): Given a lattice basis B and a target vector t (not necessarily in the lattice), find the lattice point v $\in \mathcal{L}(\mathrm{B})$ closest to t.
- Independent Vectors Problem (SIVP): Given a lattice basis B $\in \mathbb{Z}^{n\times n}$, find n linearly independent lattice vectors S $= [s_1, ..., s_n]$ (where $s_i \in \mathcal{L}(\mathrm{B})$ for all i) minimizing the quantity $\|S\| = \max_i \|s_i\|$.

The LWE hardness assumption Security of all our constructions reduces to the LWE (learning with errors) problem, a classic hard problem on lattices defined by Regev [13].

Definition 1. Consider a prime q, a positive integer n, and a distribution χ over $\mathbb{Z}_q$, all public. An $(\mathbb{Z}_q, n, \chi)$-LWE problem instance consists of access to an unspecified challenge oracle $\mathcal{O}$, being, either, a noisy pseudo-random sampler $\mathcal{O}_s$ carrying some constant random secret key $s \in \mathbb{Z}_q^n$, or, a truly random sampler $\mathcal{O}_u$, whose behaviors are respectively as follows:

- $\mathcal{O}_s$: outputs samples of the form $(u_i, v_i) = (u_i, u_i^T s + x_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where, $s \in \mathbb{Z}_q^n$ is a uniformly distributed persistent value invariant across invocations, $x_i \in \mathbb{Z}_q$ is a fresh sample from χ and u_i is uniform in $\mathbb{Z}_q^n$.
- $\mathcal{O}_u$: outputs truly uniform random samples from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The $(\mathbb{Z}_q, n, \chi)$-LWE problem allows repeated queries to the challenge oracle $\mathcal{O}$. We say that an algorithm $\mathcal{A}$ decides the $(\mathbb{Z}_q, n, \chi)$-LWE problem if $|\mathrm{Pr}[\mathcal{A}^{\mathcal{O}_s} = 1] - \mathrm{Pr}[\mathcal{A}^{\mathcal{O}_u} = 1]|$ is non-negligible for a random $s \in \mathbb{Z}_q^n$.

Regev [13] shows that for certain noise distributions χ denoted $\overline{\Psi}_\alpha$, the LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction.

2.2 Password-Based Authenticated Key Exchange

We now recall the security model for password-based authenticated key exchange of Bellare et al. [2].

Protocol participants. Each participant in the password-based key exchange is either a client $C \in \mathcal{C}$ or a server $S \in \mathcal{S}$. The set of all users or participants $\mathcal{U}$ is the union $\mathcal{C} \cup \mathcal{S}$.

Long-lived keys. Each client $C \in \mathcal{C}$ holds a password pw_C. Each server $S \in \mathcal{S}$ holds a vector $\mathrm{pw}_S = \{\mathrm{pw}_S[C]\}_{C\in\mathcal{C}}$ with an entry for each client, where $\mathrm{pw}_S[C]$ is the transformed-password, as defined in [2]. In this work(also in [6]), we only consider the symmetric model, in which $\mathrm{pw}_S[C] = \mathrm{pw}_C$, we denote the sharing password as $\pi_{U,U'}$. In general, pw_C and pw_S are also called the long-lived keys of client C and server S, and they may be different.

Protocol execution. The interaction between an adversary $\mathcal{A}$ and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. During the execution, the adversary may create several instances of a participant. While in a concurrent model, several instances may be active at any given time, only one active user instance is allowed for a given intended partner and password in a non-concurrent model. Let U^i denote the instance i of a participant U and let b be a bit chosen uniformly at random. The query types available to the adversary are as follows:

- $Execute(C^i, S^j)$: This query models passive attacks in which the attacker eavesdrops on honest executions between a client instance C^i and a server instance S^j. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.
- $Send(U^i, m)$: This query models an active attack, in which the adversary may tamper with the message being sent over the public channel. The output of this query is the message that the participant instance U^i would generate upon receipt of message m.
- $Reveal(U^i)$: This query models the misuse of session keys by a user. If a session key is not defined for instance U^i or if a $Test$ query was asked to either U^i or to its partner, then return $\perp$. Otherwise, return the session key held by the instance U^i.
- $Test(U^i)$: This query tries to capture the adversary's ability to tell apart a real session key from a random one. If no session key for instance U^i is defined, then return the undefined symbol $\perp$. Otherwise, return the session key for instance U^i if $b = 1$ or a random key of the same size if $b = 0$.

Partnering. The definition of partnering uses the notion of session identifications (sid). In [6] Groce. et al. defined mutual authentication and denoted the security parameter by n. Prior to any execution of the protocol there is an initialization phase during which public parameters are established. They assume a fixed set of protocol participants $\mathcal{U}$. For every distinct $U, U' \in \mathcal{U}$, we assume U and U' share a password $\pi_{U,U'}$. They make the simplifying assumption that each $\pi_{U,U'}$ is chosen independently and uniformly at random from the set $\{1, ..., D_n\}$ for some integer D_n that may depend on n. Denote instance i of user U as Π_U^i, sid_U^i , pid_U^i , and sk_U^i denote the session id, partner id, and session key, respectively. The session id is simply a way to keep track of different executions; let sid_U^i be the (ordered)concatenation of all messages sent and received by Π_U^i. The partner id denotes the user with whom Π_U^i believes it is interacting; we require $pid_U^i \neq U$. acc_U^i is the flag denoting acceptance. Let $U, U' \in \mathcal{U}$. Instances Π_U^i and $\Pi_{U'}^i$ are *partnered* if: $sid_U^i = sid_{U'}^i \neq$ null, and $pid_U^i = U'$, $pid_{U'}^i = U$.

Freshness. The notion of freshness is defined to avoid cases in which adversary can trivially break the security of the scheme. The goal is to only allow the adversary to ask $Test$ queries to $fresh$ oracle instances. More specifically, we say an instance U^i is $fresh$ if it has accepted and if both U^i and its partner are $unopened$, which means $Reveal(U^i)$ has not been made by the adversary.

Correctness. To be viable, a key-exchange protocol must satisfy the following notion of correctness: if Π_U^i and $\Pi_{U'}^j$ are partnered then acc_U^i = $acc_{U'}^j$ = true and sk_U^i = $sk_{U'}^j$, i.e., they both accept and conclude with the same session key. Define oracle $Test(U^i)$ as follows: A random bit b is chosen; if b = 1 the adversary is given sk_U^i, and if b = 0 the adversary is given a session key chosen uniformly from the appropriate space.

Semantic security. Consider an execution of the key exchange protocol Π by an adversary $\mathcal{A}$, in which the latter is given access to the $Reveal$, $Execute$, $Send$, and $Test$ oracles and asks a single $Test$ query to a fresh instance, a single $Test$ query to a fresh instance, and outputs a guess bit b'.

Informally, the adversary can succeed in two ways: (1) if it guesses the bit b used by the Test oracle (this implies secrecy of session keys), or (2) if it causes an instance to accept without there being a corresponding partner (this implies mutual authentication). We denote the event that the adversary succeeds by Succ. The advantage of $\mathcal{A}$ in attacking protocol $\mathit{\Pi}$ is

$$\mathrm{Adv}_{\mathcal{A},\Pi}(\mathrm{k}) = 2 \cdot \mathrm{Pr}[\mathrm{Succ}] - 1,$$

Definition 2. Protocol $\mathit{\Pi}$ is a secure PAKE protocol with explicit mutual authentication if, for all dictionary sizes $\|\mathrm{D_n}\|$ and for all ppt adversaries $\mathcal{A}$ making at most $\mathrm{Q(n)}$ on-line attacks, there exists a negligible function $\mathrm{negl}(\cdot)$ such that

$$\mathrm{Adv}_{\mathcal{A},\Pi}(\mathrm{n}) \leq \mathrm{Q(n)}/\|\mathrm{D_n}\| + \mathrm{negl(n)}.$$

2.3 Smooth Projective Hashing

Smooth projective hash functions were introduced by Cramer and Shoup [3], and later extended by Gennaro and Lindell [18]. To define this notion they rely on the existence of a set X (actually a distribution on sets), and an underlying $\mathcal{NP}$-language $L \subseteq X$ (with an associated $\mathcal{NP}$-relation R). The basic hardness assumption is that it is infeasible to distinguish between a random element in L and a random element in $X \backslash L$. This is called a hard subset membership problem.

A smooth projective hash family is a family of hash functions that operate on the set X. Each function in the family has two keys associated with it: a hash key k, and a projection key $\alpha(k)$. The first requirement (which is the standard requirement of a hash family) is that given a hash key k and an element x in the domain X, one can compute $H_k(x)$. There are two additional requirements: the $projection requirement$ and the $smoothness requirement$. The former is that given a projection key $\alpha(k)$ and an element in $x \in L$, the value of $H_k(x)$ is uniquely determined. Moreover, computing $H_k(x)$ can be done efficiently, given the projection key $\alpha(k)$ and a pair $(x, w) \in R$. The $smoothness requirement$, on the other hand, is that given a random projection key $s = \alpha(k)$ and any element in $x \in X \backslash L$, the value $H_k(x)$ is statistically indistinguishable from random.

We follow (and adapt) the treatment of Katz and Vaikuntanathan [9], who extend their original definition by introducing the approximate smooth projective hashing. Roughly speaking, the differences between [9] definition and that of [18] is that [9] only requires *approximate* correctness. (For readers not familiar with the formal definition of smooth projective hashing, find [18] for more details.)

Formally, an $\varepsilon(n)$-approximate smooth projective hash function is defined by a sampling algorithm that, given *pk*, outputs $(K, G, \mathbb{H} = \{H_k : X \to \{0,1\}^n\}_{k \in K}, S, \alpha : K \times (\{0,1\}^* \times \mathcal{C}) \to S$ such that:

- There are efficient algorithms for (1) sampling a uniform $k \in K$, (2) computing $H_k(x)$ for all $k \in K$ and $x \in X$, and (3) computing $\alpha(k, \mathrm{label}, C)$ for all $k \in K$ and $(\mathrm{label}, C) \in \{0,1\}^* \times \mathcal{C}$.
- **Approximate correctness** Let $\mathrm{Ham(a, b)}$ denote the Hamming distance of two strings $a, b \in \{0,1\}^n$. Then there is an efficient algorithm *H'* that takes as input $s = \alpha(k, \mathrm{label}, C)$ and $\overline{x} = (\mathrm{label}, C, m, r)$ for which $C = \mathrm{Enc}_{pk}(\mathrm{label}, m; r)$ and satisfies: $\mathrm{Pr}[\mathrm{Ham}(H_k(x), H'(s, \overline{x})) \geq \varepsilon \cdot n] = \mathrm{negl(n)}$.

– **Smoothness** For any $x = (\text{lable}, C, m) \in X\ L$, the following two distributions have statistical distance negligible in n: $\{k \leftarrow K; s = \alpha(k, \text{label}, C) : (s, H_k(x))\}$ and $\{k \leftarrow K; s = \alpha(k, \text{label}, C); v \leftarrow \{0,1\}^n : (s, v)\}$.

2.4 The JG/GK PAKE Framework

The JG/GK framework for PAKE in [6] has the following primitives: (1). A CPA-secure public-key encryption scheme $\Sigma' = (\text{Gen}', \text{Enc}', \text{Dec}')$ with an associated smooth projective hash function. and (2). A CCA-secure public-key encryption scheme $\Sigma = (\text{Gen}, \text{Enc}, \text{Dec})$. Jiang-Gong protocol is actually an instant of this framework by letting Σ' be the El Gamal encryption scheme (which is well-known to admit an SPH function), and Σ be the Cramer-Shoup encryption scheme, we recover the JG protocol [8]. Using the CPA-secure encryption system let JG/GK protocols about 25$\%$ faster than the KOY/GL protocols.

We should note that the framework requires an exact smooth projective hash function. Although lattice-based CPA-secure(or CCA-secure) public-key encryption scheme and its variants have been existed since [13] and [7], constructing such an scheme along with an exact smooth projective hash from lattice assumptions is still an open question since first stated in [12]. We argue that constructing exact lattice-based CPA SPH may not have interest on its own(at least for PAKE scheme). First of all, the definition could probably be modified so that approximate CPA SPH along with error-correction satisfies the definition of exact CPA SPH. Second, it is only interesting if exact CPA SPH is more efficient than approximation CPA SPH with error-correction.

3 An Efficient Lattice-based PAKE

In this section we present a new efficient PAKE from lattices based on [6]. Specifically, we use the lattice-based CPA-secure encryption scheme along an approximate smooth projective hash function introduced in [9]. In addition, we use error-correcting code to let it fit in the framework.

3.1 Approximate CPA SPH from Lattice [9]

The encryption scheme is a variant of the scheme presented in [7], and is based on the hardness of the LWE problem [13]. With some modifications to the algorithm, it provides smoothness for the approximate SPH system.

Let n be the security parameter, and $l = n$ be the message length. The parameters of the system are a prime $q = q(n,l)$, a positive integer $m = m(n,l)$, and a Gaussian error parameter $\beta = \beta(n,l) \in (0,1]$ that defines a distribution $\overline{\Psi}_\beta$. Choose a B $\in Z_q^{m\times n}$ along with l+ 1 vectors u_0,..., $u_l \in Z_q^m$. Let the public key contain the matrix A $= [\text{B}|\text{U}]$, where the columns of $U \in Z_q^{m\times(l+1)}$ are the vectors u_0,..., u_l. The secret key is the trapdoor T for the entire matrix A by running $(\text{A}, \text{T}) \leftarrow \text{TrapSamp}(1^m, 1^{n+l+1}, q)$, where TrapSamp is as described in [1]. Let the public key be A and the secret key is T. To encrypt the message $w \in Z_q^l$ with respect to a public key as above, the sender chooses $s \leftarrow Z_q^n$ uniformly at random, and an error vector $x \leftarrow \overline{\Psi}_{\beta}^{mn}$. The ciphertext is

$$\mathrm{y} = \mathrm{A} \cdot \begin{pmatrix} s \\ 1 \\ w \end{pmatrix} + \mathrm{x}\,(\mathrm{mod\ q})\,.$$

The decryption algorithm is immaterial to understand this paper, we omit their detailed procedure here.

Fix a public key $\mathrm{A} \in Z_q^{m\times(n+l+1)}$ for the system (where we write $\mathrm{A} = [\mathrm{B}|\mathrm{U}]$, as usual), and a dictionary $\mathcal{D} = Z_q^l$. A key for the SPH system is a k-tuple of vectors $(e_1, ..., e_k)$ where each $e_i \leftarrow D_{Z^m,r}$ is drawn independently from the discrete Gaussian distribution. The projection set $S = (Z_q^n)^k$. For a key $(e_1, ..., e_k) \in (Z_q^m)^k$, the projection is $\alpha(e_1, ..., e_k) = (u_1, ..., u_k)$, where $u_i = \mathrm{B}^T e_i$.

We now define the smooth projective hash function $\mathcal{H} = \{H_k\}_{k\in K}$. On input a key $(e_1, ..., e_k) \in K$ and a ciphertext $c = (label, y, m)$, the hash function is computed as follows. First compute

$$z_i = e_i^T \left[y - U \cdot \begin{pmatrix} 1 \\ m \end{pmatrix} \right] \in Z_q.$$

Treat z_i as a number in $[-(q-1)/2...(q-1)/2]$ and output $b_1...b_k \in \{0,1\}^k$. where

$$b_i = \begin{cases} 0 \text{ if } z_i < 0 \\ 1 \text{ if } z_i > 0 \end{cases}$$

On input a projected key $(u_1, ..., u_k) \in S$, a ciphertext $c = (\mathrm{label}, y, m)$ and a witness $s \in Z_q^n$ for the ciphertext, the hash function is computed as $H'_u(c, s) = b_1...b_k$ where

$$b_i = \begin{cases} 0 \text{ if } u_i^{\mathrm{T}} s < 0 \\ 1 \text{ if } u_i^{\mathrm{T}} s > 0 \end{cases}$$

Theorem 1. [9] Let n, l, m, q, β be chosen such that $m \geq 4(n+l)\mathrm{log}q$, $\beta < 1/(2\cdot m^2 n \cdot w(\sqrt{\mathrm{log}n})$, and $\sqrt{q} \cdot w(\sqrt{\mathrm{log}n}) \leq r \leq \varepsilon/(8 \cdot mn^2 \cdot \beta)$. Then, the cryptosystem above is a CPA-secure encryption scheme assuming the hardness of decision LWE problem [13], and $\mathcal{H} = \{H_k\}_{k\in K}$ is an-approximate smooth projective hash system.

Approximate correctness. Let b_i be the i^{th} bit of $H_{(e1,...,ek)}(c)$ and b_i' be the i^{th} bit of $H'_{(e1,...,ek)}(c, s)$. Because $|z_i - u_i^{\mathrm{T}} s| = |e_i^{\mathrm{T}}(Bs + x) - u_i^{\mathrm{T}} s| = |e_i^{\mathrm{T}} x| \leq \|e_i\| \cdot \|x\| < \varepsilon/2 \cdot q/4$. We see that the probability that $b_i \neq b_i'$ is at most $\varepsilon/2$. So the Hamming distance between $H_{(e1,...,ek)}(c)$ and $H'_{(e1,...,ek)}(c, s)$ is at most εk with overwhelming probability.

Smoothness. Smoothness comes from the fact that the pair $(e_i^{\mathrm{T}} B, e_i^{\mathrm{T}} z)$ is statistically close to the uniform distribution over Z_q^{n+1}. We omit the proof here, refer to [9] for further details.

3.2 The PAKE Protocol

We use the lattice-based CPA-secure encryption system Σ' with an approximate SPH function introduced above and a lattice-based CCA-secure encryption system of [11] denoted as Σ. The protocol relies on a common reference string (CRS) consisting of public keys pk, pk' for Σ and Σ', respectively, and parameters $(K, G, \mathbb{H} = \{H_k : X \to \{0,1\}^n\}_{k \in K}, S, \alpha : K \times (\{0,1\}^* \times \mathcal{C}) \to S$ for approximate SPH function associated with pk'. A high-level depiction of the protocol is given in Figure 1.

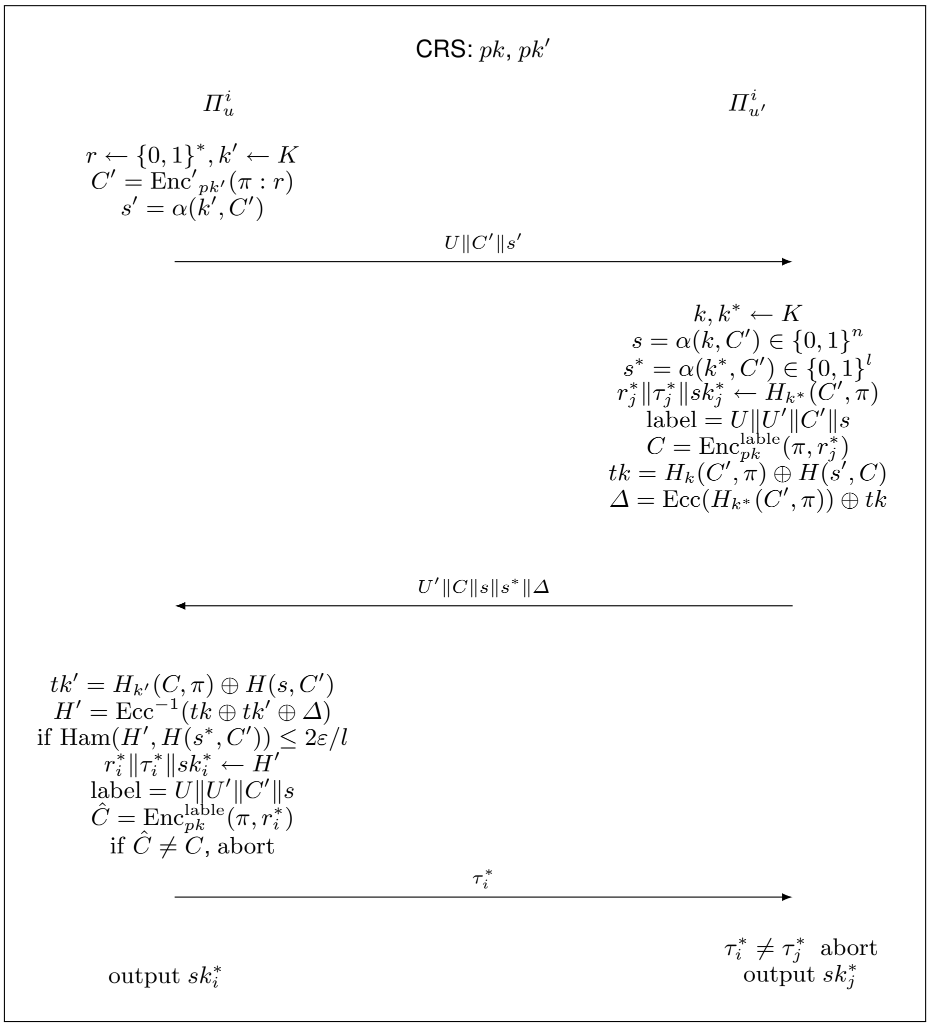
CRS: pk, pk'
Π_u^i
$\Pi_{u'}^i$
$r \leftarrow \{0,1\}^*, k' \leftarrow K$
$C' = \mathrm{Enc'}_{pk'}(\pi : r)$
$s' = \alpha(k', C')$
$U \| C' \| s'$
$k, k^* \leftarrow K$
$s = \alpha(k, C') \in \{0,1\}^n$
$s^* = \alpha(k^*, C') \in \{0,1\}^l$
$r_j^* \| \tau_j^* \| sk_j^* \leftarrow H_{k^*}(C', \pi)$
label $= U \| U' \| C' \| s$
$C = \mathrm{Enc}_{pk}^{\mathrm{lable}}(\pi, r_j^*)$
$tk = H_k(C', \pi) \oplus H(s', C)$
$\Delta = \mathrm{Ecc}(H_{k^*}(C', \pi)) \oplus tk$
$U' \| C \| s \| s^* \| \Delta$
$tk' = H_{k'}(C, \pi) \oplus H(s, C')$
$H' = \mathrm{Ecc}^{-1}(tk \oplus tk' \oplus \Delta)$
if $\mathrm{Ham}(H', H(s^*, C')) \leq 2\varepsilon/l$
$r_i^* \| \tau_i^* \| sk_i^* \leftarrow H'$
label $= U \| U' \| C' \| s$
$\hat{C} = \mathrm{Enc}_{pk}^{\mathrm{lable}}(\pi, r_i^*)$
if $\hat{C} \neq C$, abort
τ_i^*
$\tau_i^* \neq \tau_j^*$ abort
output sk_i^*
output sk_j^*

Fig. 1. An honest execution of the protocol.

When a client instance Π_U^i wants to authenticate to the server instance $\Pi_{U'}^i$, the client first chooses a random tape r and then computes an encryption $C' = \text{Enc'}_{pk'}(\pi : r)$ of the shared password π. Then it chooses a random hash key $k' \leftarrow K$ and computes the projection key $s' = \alpha(k', C')$. The client then sends $U\|C'\|s'$ to the server.

Upon receiving the massage $U\|C'\|s'$, the server also chooses two random hash keys $k, k^* \leftarrow K$ and computes the projection key $s = \alpha(k, C')$,and $s^* = \alpha(k^*, C')$. It then computes two hash $H_{k^*}(C', \pi) \in \{0,1\}^l, l < n$ and $H_k(C', \pi) \in \{0,1\}^n$ using the ciphertext C' it received in the first message and the password π that it shares with U. For $H_{k^*}(C', \pi)$ the result is parsed as a sequence of three bit-strings r_j^*, τ_j^*, sk_j^*, where r_j^* will be used as the random tape for an encryption using Enc. The server set $\text{label} = U\|U'\|C'\|s$, and generate an encryption $C = \text{Enc}_{pk}^{\text{lable}}(\pi, r_j^*)$ of shared password π. It then computes hash $H(s', C)$ using the projection key s' it received from the client, and generates a temporary session key $tk = H_k(C', \pi) \oplus H(s', C)$. It also computes $\Delta = \text{Ecc}(H_{k^*}(C', \pi)) \oplus tk$ where $\text{Ecc} : \{0,1\}^l \rightarrow \{0,1\}^n, l < n$ is an error-correcting code correcting a 2ε fraction of errors. Finally, server sends the message $U'\|C\|s\|s^*\|\Delta$ back to the client.

Upon receiving $U'\|C\|s\|s^*\|\Delta$, the client first computes hash $H_{k'}(C, \pi)$ and $H(s, C')$ using the projected key s,C that it received from server and the k' it generated in the first round. It also computes $H(s^*, C') \in \{0,1\}^l$ using the s^* it received from the server. Then it computes $tk' = H_{k'}(C, \pi) \oplus H(s, C')$ and $H' = \text{Ecc}^{-1}(tk \oplus tk' \oplus \Delta) \in \{0,1\}^l$. Next it first check whether $\text{Ham}(H', H(s^*, C')) \leq 2\varepsilon/l$, if it is not the case, the client aborts. Otherwise, it continue to parse r_i^*, τ_i^*, sk_i^* from H', and computes $\hat{C} = \text{Enc}_{pk}^{\text{lable}}(\pi, r_i^*)$. It then checks whether $\hat{C} = C$ it received from the server in the second round. If it is the case, the server has successfully authenticated to the client, and the client then accepts, sends τ_i^* to the server, and output the session key sk_i^*, otherwise, the client aborts.

When the server receives the client's final message τ_i^*, it checks that $\tau_i^* \neq \tau_j^*$ and aborts if that is not the case. Otherwise the client has successfully authenticated to the server, and the server accepts and outputs the session key sk_j^*.

3.3 Proof of Security

Correctness is easily verified. If both parties are honest and there is no adversarial interference, then $H' = \text{Ecc}^{-1}(tk \oplus tk' \oplus \Delta) \in \{0,1\}^l$, note that $tk' = H_{k'}(C, \pi) \oplus H(s, C')$, $tk = H_k(C', \pi) \oplus H(s', C)$, and $\Delta = \text{Ecc}(H_{k^*}(C', \pi)) \oplus tk$, according to **Approximate correctness** of **Theorem 1. [9]**, H' can be corrected back to $H_{k^*}(C', \pi)$ using Ecc^{-1} and so it holds that $r_i^* = r_j^*$, $\tau_i^* = \tau_j^*$. It follows that both parties will accept and output the same session key $sk_i^* = sk_j^*$.

Theorem 2. If Σ' is a lattice-based CPA-secure public encryption scheme with associated approximate smooth projective hash function, Σ is a CCA-secure public-key encryption scheme, and $\text{Ecc} : \{0,1\}^l \rightarrow \{0,1\}^n, l < n$ is an error-correcting code correcting a 2ε fraction of errors. Then the protocol in Figure 1. is a secure PAKE protocol with explicit mutual authentication.

The proof of security of the protocol follows [6] closely; we sketch the main ideas. First, as in [19], [18], we note that for a passive adversary (i.e., one that simply observes interactions between the server and the client), the shared session-key is pseudorandom. This is simply because the transcript of each interaction consists of semantically-secure encryptions of the password π and the projected keys of the approximate SPH system.

It remains to deal with active adversaries that modify the messages sent from the client to the server and back. We can reduce the security to the GK framework's. Briefly, if the adversary sends the client a ciphertext that does not decrypt to the client's password, then the session-key computed by the client is statistically close to uniform conditioned on the adversary's view.

We defer a complete proof to the proof of security in the GK framework [6].

3.4 Comparing with JG's instantiation

We present a instantiation of lattice-based PAKE under the KV framework. Comparing with Jiang and Gong's instantiation in [8] which security is under the decisional discrete logarithm problems, our lattice-based PAKE has many advantages. Using hard lattice problems, such as the shortest vector problem, as the basis of security has advantages over using the factoring or discrete logarithm problems. For instance, lattice operations are more efficient than modular exponentiation and lattice problems remain hard for quantum and subexponentialtime adversaries.

4 Conclusion

We show a new construction of an efficient PAKE based on lattices. Our work is built on ideas of Groce and Katz [6], just like the work of Katz and Vaikuntanathan [9] instantiated the Katz-Ostrovsky-Yung [19] framework, and Gennaro-Lindell [18] framework. Namely, we present a instantiation of lattice-based PAKE under the JG/KV framework [6].

We first reviewed the definition of a secure PAKE protocol, and then give a brief description of the JG/GK PAKE framework. Then we introduced the lattice-based approximate smooth projective hash system presented in [9] and argue that it cannot directly plug into the JG/GK PAKE framework. We constructed a new PAKE protocol using the modified GK PAKE framework, and gave a brief proof of the security under the LWE assumption.

Acknowledgments. The authors would like to thank anonymous referees for valuable comments. Yi Ding would like to thank Prof. Jonathan Katz for kind response upon query on smooth projective hashing and he especially feels grateful to Dr. Hong-Sheng Zhou for an instructive discussion on the definition of lattice-based PAKE.

References

1. J. Alwen and C. Peikert: Generating Shorter Bases For Hard Random Lattices. In: STACS: Symposium on Theoretical Aspects of Computer Science, pp. 75–86, (2009)

2. M. Bellare, D. Pointcheval, and P. Rogaway: Authenticated Key Exchange Secure Against Dictionary Attacks. In: Advances in Cryptology, Eurocrypt 2000, volume 1807 of LNCS, pp. 139–155. Springer, (2000)
3. R. Cramer and V. Shoup: Universal Hash Proofs And A Paradigm For Adaptive Chosen Ciphertext Secure Public-key Encryption. In: Advances in Cryptology, Eurocrypt 2002, volume 2332 of LNCS, pp. 45–64. Springer, (2002)
4. R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie: Universally Composable Password-based Key Exchange. In: Advances in Cryptology, Eurocrypt 2005, volume 3494 of LNCS, pp. 404–421. Springer, (2005)
5. R. Gennaro: Faster And Shorter Password-authenticated Key Exchange. In: 5th Theory of Cryptography Conference, TCC 2008, volume 4948 of LNCS, pp. 589–606. Springer, (2008)
6. A. Groce, J. Katz: A New Framework For Efficient Password-based Authenticated Key Exchange. In: 17th ACM Conf. on Computer and Communications Security, pp. 516–525. ACM Press, New York, (2010)
7. C. Gentry, C. Peikert, and V. Vaikuntanathan: Trapdoors For Hard Lattices And New Cryptographic Constructions. In: 40th Annual ACM Symposium on Theory of Computing (STOC), pp. 197–206. ACM Press, (2008)
8. S. Jiang and G. Gong: Password Based Key Exchange With Mutual Authentication. In: 11th Annual International Workshop on Selected Areas in Cryptography (SAC), volume 3357 of LNCS, pp. 267–279. Springer, (2004)
9. J. Katz and V. Vaikuntanathan: Password-based Authenticated Key Exchange Based on Lattices. In: Advances in Cryptology, Asiacrypt 2009, volume 5912 of LNCS, pp. 636–652. Springer, (2009)
10. J. Katz, P. D. MacKenzie, G. Taban, and V. D. Gligor: Two-server Password-only Authenticated Key Exchange. In: 3rd Intl. Conference on Applied Cryptography and Network Security (ACNS), volume 3531 of LNCS, pp. 1–16. Springer, (2005)
11. C, Peikert: Public-Key Cryptosystems From The Worst-case Shortest Vector Problem. In Proceedings of STOC'2009, pp.333–342, (2009)
12. C. Peikert, V. Vaikuntanathan, B. Waters: A Framework For Efficient And Composable Oblivious Transfer. In: Advances in Cryptology, Crypto 2008, volume 5157 of LNCS, pp. 554–571. Springer (2008)
13. O. Regev: On Lattices, Learning With Errors, Random Linear Codes, And Cryptography. In: 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 84–93. ACM Press, (2005)
14. S. Bellovin and M. Merritt: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: Proc. IEEE Computer Society Symposium on Research in Security and Privacy, pp. 72–84, Oakland, CA, (1992)
15. V. Boyko, P. MacKenzie, and S. Patel: Provably Secure Password Authenticated Key Exchange Using Diffie-Hellman. In: Advances in Cryptology, Eurocrypt 2000, pp. 156–171, (2000)
16. O. Goldreich and Y. Lindell: Session-Key Generation using Human Passwords Only, In: Crypto '01, LNCS 2139, pp. 408–432. Springer-Verlag, (2001)
17. R. Cramer and V. Shoup: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: proceedings of Crypto 1998, LNCS 1462, pp. 13–25, (1998)
18. R. Gennaro and Y. Lindell: A Framework For Password-based Authenticated Key Exchange. ACM Trans. Information and System Security, 9(2):181–234, (2006)
19. J. Katz, R. Ostrovsky, and M. Yung: Efficient And Secure Authenticated Key Exchange Using Weak Passwords. Journal of the ACM, 57(1):78–116, (2009)