Post-Quantum Secure Remote Password Protocol from RLWE Problem

Xinwei Gao[1], Jintai Ding[2](✉), Jiqiang Liu[1](✉), and Lin Li[1]

[1] Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, People's Republic of China
{xinweigao,jqliu,lilin}@bjtu.edu.cn

[2] Department of Mathematical Sciences, University of Cincinnati, Cincinnati 45219, USA
jintai.ding@gmail.com

Abstract. Secure Remote Password (SRP) protocol is an augmented Password-based Authenticated Key Exchange (PAKE) protocol based on discrete logarithm problem (DLP) with various attractive security features. Compared with basic PAKE protocols, SRP does not require server to store user's password and user does not send password to server to authenticate. These features are desirable for secure client-server applications. SRP has gained extensive real-world deployment, including Apple iCloud, 1Password etc. However, with the advent of quantum computer and Shor's algorithm, classic DLP-based public key cryptography algorithms are no longer secure, including SRP. Motivated by importance of SRP and threat from quantum attacks, we propose a RLWE-based SRP protocol (RLWE-SRP) which inherit advantages from SRP and elegant design from RLWE key exchange. We also present parameter choice and efficient portable C++ implementation of RLWE-SRP. Implementation of our 209-bit secure RLWE-SRP is more than 3x faster than 112-bit secure original SRP protocol, 5.5x faster than 80-bit secure J-PAKE and 14x faster than two 184-bit secure RLWE-based PAKE protocols with more desired properties.

Keywords: Post-quantum · RLWE · SRP · PAKE · Protocol Implementation

1 Introduction

1.1 Key Exchange

Key exchange (KE) is an important and fundamental cryptographic primitive. It allows two or multiple parties to agree on same session key, which is later utilized in encryption and other cryptographic primitives. With the ground-breaking Diffie-Hellman key exchange proposed in 1976 [14], public key cryptography came into reality and it has been widely deployed in real world applications. Since public key computations are rather expensive compared with symmetric-based ones, symmetric encryption is adopted to encrypt actual communication data instead of public key encryption. The shared key generated during key exchange is extremely important, especially in constructing real-world security protocols and applications. Important applications of key exchange include Transport Layer Security (TLS), Secure Shell (SSH), Internet Key Exchange (IKE), Internet Protocol Security (IPsec), Virtual Private Network (VPN) etc.

However, Diffie-Hellman and other unauthenticate key exchange protocols are vulnerable to Man-In-The-Middle (MITM) attack, where an adversary in the middle between communicating parties can intercept and tamper messages and pretend himself as legit counterpart. An important line of key exchange protocols that can defeat such attack is authenticated key exchange (AKE). In AKE, authentication mechanisms can ensure one or both sides of key exchange are securely authenticated. HMQV [22] is an example of AKE. There are various approaches to achieve authentication, including public key infrastructure (PKI)-based (using signatures and verified public key with valid certificates), password (and its variants)-based AKE protocol (PAKE) etc. PAKE is an important approach to realize AKE. Examples of PAKE protocols are PAK & PPK [10], J-PAKE [20], EKE [7], SPAKE2 [2] etc. Some additional works include [17,19,24] etc.

In most network security protocols, PKI-based authentication (certificate and signature) is more popular mostly because it is "secure enough". In most cases, server side can be securely authenticated using certificate but client side is not since generally client does not have a valid certificate. This highlights one advantage of PAKE protocols - simpler mutual authentication. In PAKE, mutual authentication can be securely achieved using pre-shared value or tokens for both parties (in most cases, such pre-shared value is password or hash of password). This also saves some rather expensive computations related to public key operations (e.g., compute/verify signature). A shortcoming for basic PAKE protocols is that these constructions directly using password or hash of password as pre-shared value. We can foresee that once server is compromised and actual user password (or its hash value) is leaked, adversary can easily impersonate as the client to authenticate himself. This stresses an crucial problem and challenge for basic PAKE protocols.

A solution to this issue is augmented PAKE, since it only requires the server to store a pre-shared verifier (or a token) which is generated usingsword and other elements, instead of simply storing actual password or hash of password. In execution of augmented PAKE protocol, client needs to enter correct password in order to compute intermediate values correctly and authenticate himself. Server uses the stored verifier to authenticate user. Meanwhile, actual password is not sent to server. The trick is that these intermediate values can be only computed with correct password. Adversary cannot compute such intermediate values thanks to delicate design and hard problems like discrete logarithm problem etc. The advantage of augmented PAKE protocols is that even if attacker owns the verifier, he cannot impersonate as client by sending the verifier he captured since he does not know actual password. Examples of augmented PAKE protocols are SRP [31], PAK-Z [10] etc.

1.2 Post-Quantum World

Diffie-Hellman key exchange protocol and various current public key cryptosystems are constructed based on discrete logarithm problem (DLP), integer factorization problem (IFP), elliptic curve variant of DLP (ECDLP) etc. Currently, there are no public known algorithms on classic computers that can solve these hard problems with large parameters. However, Shor introduced a quantum algorithm in 1994 [29], which suggested that DLP, IFP and ECDLP can be solved efficiently on a sufficient large quantum computer. This is horrible for current public key cryptography since most current widely deployed public key algorithms are constructed based on these hard problems. Once sufficient large quantum computer is built, most current public key algorithms can be broken. We also know that the development of quantum computers within past decades is incredibly fast. In 2015, National Security Agency (NSA) announced their plan of switching to post-quantum cryptography in near future. At PQCrypto 2016, National Institute of Standards and Technology (NIST) announced call for post-quantum cryptography standards. All these facts stress the importance of post-quantum cryptography and severity of deploying post-quantum cryptographic constructions in real world applications.

There are several approaches to build post-quantum cryptography primitives, including lattice-based, multivariate-based, hash-based, code-based, supersingular elliptic curve-based etc. Lattice-based ones are very popular due to strong security and high efficiency. Among all lattice-based constructions, Learning With Errors (LWE) and Ring-LWE based ones are more practical and outstanding due to much better efficiency, robust security and versatility. There are several works have demonstrated real-world efficiency of post-quantum cryptographic primitives, including experimenting "NewHope" RLWE key exchange [5] in Chrome canary build [11], deploying BLISS lattice-based signature in strongSwan VPN [1] etc.

1.3 Related Works

Secure Remote Password (SRP) Protocol and Important Real-World Applications. Thomas Wu proposed the Secure Remote Password (SRP) protocol in 1998 [31]. SRP protocol is an augmented PAKE protocol designed based on DLP. Compared with basic PAKE protocols, advantages of SRP are: (1) Server only stores a securely pre-shared verifier. Neither user password nor hash of password are stored for both client and server; (2) SRP can stop adversary from impersonating as client even if server is compromised; (3) No one (adversary, malicious server etc.) can recover user's password from verifier; (4) SRP does not require user sending actual password or its variants to servers to authenticate himself. These are major advantages of SRP compared with other PAKE protocols. SRP is also a key exchange protocol which provides mutual authentication and forward secrecy. SRP is standardized in RFC 2945, ISO 11770-4 and IEEE P1363.

SRP has been widely deployed in industry and critical real-world applications. Here we list a few of them:

1. Apple iCloud. SRP is adopted in iCloud to authenticate user when attempting to access iCloud account using iCloud security code (iCSC), where iCSC is set and only known by user (i.e., so-called "password" in SRP). According to Apple security white paper [6], SRP protocol keeps Apple servers away from acquiring information about user's actual iCSC. Moreover, for HomeKit accessories, which are developed for smart home automation are also using 3072-bit SRP to authenticate between iOS device and HomeKit accessory. SRP stops Apple from knowing the setup code printed on HomeKit accessories.
2. 1Password. 1Password is the leading password manager. It adopts SRP to handle user's master password and login attempts. In fact, 1Password security team claims that user's security is not affected in recent "Cloudbleed" bug thanks to the adoption of SRP [18].
3. ProtonMail. Highly-secure email service ProtonMail adopts SRP for secure login since 2017, protecting the security of user's account even if server is compromised.
4. Blizzard. The leading gaming company also uses SRP to authenticate user login. In 2012, their servers were compromised in a security breach, but SRP protects safety of user's password [25].
5. TLS. There are SRP ciphersuites for TLS protocol (e.g., [27]). OpenSSL has supported TLS-SRP ciphersuites.

To the best of our knowledge, there are no practical attacks against SRP while formal security proof is not presented in original SRP paper [31]. They claim that SRP is secure against several attacks with security analysis. Moreover, security of SRP protocol is heavily relied on hardness of discrete logarithm problem, which can be broken by quantum computers in near future. Importance of SRP and threats from quantum computers directly motivate this work.

Post-Quantum Key Exchange Protocols from RLWE. Jintai Ding et al. proposed the first LWE and RLWE based key exchange protocols which are analogues of Diffie-Hellman key exchange in 2012 (denoted as DING12) [16]. DING12 proposed a "robust extractor" (i.e., error reconciliation mechanism), which allows two parties to agree on same key over approximately equal values with overwhelming probability. This work is known as the foundation of error reconciliation-based LWE & RLWE key exchange protocols. There are various variants (e.g., [5,8,9,26]) that share similar protocol structure and error reconciliation techniques as DING12. It is proven secure under passive probabilistic polynomial-time (PPT) attack and enjoys very high efficiency.

Jiang Zhang et al. proposed the first RLWE-based AKE protocol which is a RLWE analogue of HMQV in 2015 [32], where core idea of DING12 and HMQV are well inherited and applied. This protocol is proven secure under Bellare-Rogaway model with enhancements to capture weak perfect forward secrecy. They also present parameter choices and proof-of-concept implementation.

Jintai Ding et al. proposed the first RLWE-based PAKE protocol in 2017 (denoted as PAKE17) [15]. This work also follows idea in DING12 and gives RLWE analogues of classic PAK and PPK protocols. To the best of our knowledge, this is the only work up to now that gives RLWE-based PAKE constructions and proof-of-concept implementation. RLWE-PAK and RLWE-PPK are proven secure under extended security model of PAK and PPK (random oracle model, ROM). Proof-of-concept implementation shows that PAKE17 is efficient. Another lattice-based PAKE construction is given in [21]. It is constructed based on common reference ring (CRS), therefore it is more complex and less efficient.

1.4 Our Contributions

Directly motivated by threats from quantum computers and widely deployed SRP protocol, we first propose a RLWE-based secure remote password protocol which enjoys elegant design and high efficiency (denoted as RLWE-SRP). Our RLWE-SRP protocol can be regarded as RLWE variant of original SRP protocol. Same as SRP, in our RLWE-SRP, user only need to remember his password and server only stores a verifier. The verifier is not actual password and no information about password is revealed. During key exchange, no actual password is transmitted. Even if server is compromised and verifier is captured, adversary cannot impersonate as user to authenticate with server, nor recover actual password. RLWE-SRP enjoys several desired features, including: mutual authentication, high efficiency and resistant to quantum attacks. Leakage of session key of previous sessions will not help adversary to identify user's password and verifier. We also present security analysis following same approach as original SRP paper.

Second, we present practical parameter choice and efficient portable C++ implementation of RLWE-SRP. With current state-of-the-art cryptanalysis tool for RLWE-based cryptosystems, our parameter choice offers 209-bit security. Benchmark shows that our construction and implementation are truly practical. Compared with implementation of original SRP protocol (112-bit security), RLWE-SRP is 3x faster with higher security level. Compared with other PAKE protocols including 80-bit secure J-PAKE [20], 184-bit secure RLWE-PAKE and RLWE-PPK from PAKE17 [15], our implementation is 5.53x, 13.96x and 13.96x faster respectively.

2 Preliminaries

2.1 Ring Learning with Errors

Oded Regev presented LWE problem in 2005 [28] and was extended by Lyubashevsky et al. to Ring-LWE in 2010 [23]. RLWE problem is a direct variant of LWE problem in ring setting. LWE and RLWE have become most popular primitives to build post-quantum cryptography. Hardness of LWE and RLWE problem is directly reduced to NP-hard lattice problems, including Shortest Vector

Problem (SVP) etc. in regular lattice and ideal lattice respectively. Compared with LWE, one significant advantage of RLWE is much reduced key size and communication overhead due to the fact that large matrix is replaced by degree n polynomial in R_q in RLWE. This reduces at least quadratic computation and communication overhead. Currently, there are no algorithms can solve LWE, RLWE and underlying lattice problems with properly chosen parameters on both classical and quantum computers.

Let $n \in Z$ be power of 2, $q \in Z$, ring $R_q = Z_q[x]/(x^n + 1)$ where $Z_q = Z/qZ$, $D_{Z^n,\sigma}$ be discrete Gaussian distribution on Z^n with standard deviation σ. For uniform randomly generated public parameter $a \in R_q$, small and fixed term $s \leftarrow D_{Z^n,\sigma} \in R_q$, $e \leftarrow D_{Z^n,\sigma} \in R_q$, let $A_{s,D_{Z^n,\sigma}}$ be distribution over pairs $(a, b = as + e) \in R_q \times R_q$. RLWE assumption implies that for fixed s, distribution $A_{s,D_{Z^n,\sigma}}$ is indistinguishable from uniform distribution on R_q^2 given polynomial many samples. Search version of RLWE problem is to recover s given RLWE samples and decision version is to distinguish RLWE samples from uniform random ones. There are security reductions from hard problems in ideal lattice to RLWE. If one can solve RLWE problem, then he can solve underlying hard lattice problems as well.

RLWE has been applied to construct various cryptography primitives, including public key encryption, digital signatures, key exchange, attribute-based encryption, homomorphic encryption etc.

2.2 Revisit DING12 RLWE Key Exchange Protocol

In 2012, Jintai Ding et al. proposed LWE and RLWE key exchange protocols [16]. This is the first work that gives simple and provable secure LWE and RLWE key exchange. DING12 takes advantage of commutative property and constructs key exchange protocols that are analogues of Diffie-Hellman key exchange. One technical challenge for building key exchange protocol over LWE & RLWE is how to reconcile errors, since public key and key exchange materials k_i, k_j are perturbed by small error terms, therefore values for both parties are approximately equal, not rigorously equal $(g^a)^b \mod q = (g^b)^a \mod q$ in Diffie-Hellman key exchange. To solve this problem, DING12 gives a new error reconciliation mechanism called "robust extractor" (i.e., error reconciliation mechanism). In order to reconcile errors, it is required that the difference between key exchange materials of two parties is approximately equal. One side needs to send a carefully computed "signal" value to the other side. Signal value implies which "region" does coefficient of polynomial belongs to and this will leads to correct error reconciliation and key computation. This work sets the foundation of LWE & RLWE key exchange. Various RLWE-based unauthenticated key exchange constructions including [5,8,9,26] follow this idea. [15,32] and this work also use this notion to construct AKE and PAKE protocols respectively.

Here we recall several core functions and properties

Signal Function. Let $q > 2$ be a prime. Hint functions $\sigma_0(x)$, $\sigma_1(x)$ from Z_q to $\{0, 1\}$ are defined as:

$$\sigma_0(x) = \begin{cases} 0, x \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor] \\ 1, otherwise \end{cases}, \sigma_1(x) = \begin{cases} 0, x \in [-\lfloor \frac{q}{4} \rfloor + 1, \lfloor \frac{q}{4} \rfloor + 1] \\ 1, otherwise \end{cases}$$

Signal function Cha() is defined as: For any $y \in Z_q$, $\text{Cha}(y) = \sigma_b(y)$, where $b \xleftarrow{\$} \{0, 1\}$. If $\text{Cha}(y) = 1$, we denote y is in outer region, otherwise y is in inner region. Signal value from an execution of key exchange is indistinguishable from uniform random bits.

Robust extractor. $\text{Mod}_2()$ is a deterministic function with error tolerance δ. Mod_2 is defined as: $\text{Mod}_2(x, w) = (x + w \cdot \frac{q-1}{2} \mod q) \mod 2$.
Input of $\text{Mod}_2()$ are: (1) $x \in Z_q$ and (2) Signal w. Output of Mod_2 is key bit k. Denote error tolerance as δ. For any $x, y \in Z_q$, if $\|x - y\|_\infty \leq \delta$, $\text{Mod}_2(x, w) = \text{Mod}_2(y, w)$, where $w = \text{Cha}(y)$. Error tolerance $\delta = \frac{q}{4} - 2$ for reconciliation mechanism in DING12, which is the key to ensure correctness of key exchange over LWE and RLWE with overwhelming probability.

Randomness. For any odd $q > 2$, if x is uniformly random in Z_q, then $\text{Mod}_2(x, w)$ is uniformly random conditioned on signal w.

DING12 RLWE key exchange is illustrated in Fig. 1:

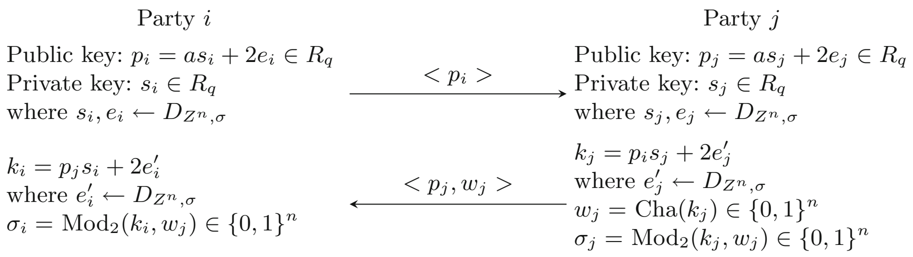

Party i
Party j
Public key: $p_i = as_i + 2e_i \in R_q$
Private key: $s_i \in R_q$
where $s_i, e_i \leftarrow D_{Z^n,\sigma}$
$< p_i >$
Public key: $p_j = as_j + 2e_j \in R_q$
Private key: $s_j \in R_q$
where $s_j, e_j \leftarrow D_{Z^n,\sigma}$
$k_j = p_i s_j + 2e'_j$
where $e'_j \leftarrow D_{Z^n,\sigma}$
$w_j = \text{Cha}(k_j) \in \{0,1\}^n$
$\sigma_j = \text{Mod}_2(k_j, w_j) \in \{0,1\}^n$
$< p_j, w_j >$
$k_i = p_j s_i + 2e'_i$
where $e'_i \leftarrow D_{Z^n,\sigma}$
$\sigma_i = \text{Mod}_2(k_i, w_j) \in \{0,1\}^n$

Fig. 1. DING12 RLWE key exchange protocol

3 Post-Quantum Secure Remote Password Protocol

Our RLWE-SRP protocol is a direct RLWE variant of original SRP [31]. Our design inherits advantages of SRP and strengthen its hardness further by constructing based on RLWE problem.

RLWE-SRP has 3 phases: Verifier setup, Authenticated key exchange and Verification.

- In "Verifier setup" phase, client generates verifier v and sends to server. v is used for authentication and it is the replacement of "pre-shared password" for basic PAKE protocols. Client only need to remember the correct password.

– In "Authenticated key exchange" phase, client and server share session key and authenticate each other. Client does not send password to server to authenticate himself.
– In "Verification" phase, both parties compute hash value of some elements to verify key exchange and authentication are indeed successful. After three phases, a mutual authenticated secure channel is established.

We note that "Verifier setup" is only executed once when user registers to the system, "Authenticated key exchange" and "Verification" are executed for each login attempt.

It is known that SRP is an augmented PAKE protocol, where augmentation refers to server stores a verifier, instead of storing password or hash value of password like basic PAKE protocols. For basic PAKE constructions, if server is compromised, adversary can directly impersonate as user to authenticate with server using captured password or hash of password. SRP solves this problem using "verifier". Our RLWE-SRP inherits this novel property and enhances the security of SRP by constructing based on RLWE problem. Only user with correct password can compute a particular value and use this value in key exchange to authenticate himself.

3.1 Protocol Construction

In this section, we present our post-quantum secure remote password protocol RLWE-SRP. Let $D_{Z^n,\sigma}$ be a discrete Gaussian distribution with standard deviation σ. H is standard secure hash function (e.g., SHA3-256), XOF is Extendable-Output Functions (e.g., SHAKE-128) and "$\|$" denotes concatenation. Definition of Cha() and Mod$_2$() are exactly the same as we recalled in Sect. 2.2. We assume that client and server execute the protocol honestly and have no deliberate malicious behaviours (e.g., client deliberately reveals password, server reveals verifier, leakage of variables stored in memory etc.).

Phase 0: Verifier Setup

Verifier	$v = a \cdot s_v + 2e_v$
Password	pwd
Salt	$salt$
Username	I

$a \leftarrow R_q$ is public parameter in RLWE which is shared by both parties. To compute v, client computes $s_v \leftarrow D_{Z^n,\sigma}$ and $e_v \leftarrow D_{Z^n,\sigma}$. Note that this is different from other random sampling operations, since it is required that s_v and e_v are sampled using specific $seed1 = \text{SHA3-256}(salt\|\text{SHA3-256}(I\|pwd))$ and $seed2 = \text{SHA3-256}(seed1)$ as seed respectively for pseudorandom generator (PRNG). Purpose for this design is to stop attackers from recovering password from leaked verifier. We will elaborate this later.

Client need to send username I, salt $salt$ and verifier v to server to complete verifier setup. Setting up verifier has to go through secure channels (e.g., strong

TLS or other security measures). *salt* and v are stored and indexed by I in server's database. After v is generated, client should delete all variables from memory to prevent potential leakage. If v is stolen, client needs to setup a new verifier and proceed this phase from scratch.

We note that similar setup phase also exists in other PAKE protocols. For other PAKE protocols, pre-shared password or hash of password is stored on server in this phase, where our RLWE-SRP stores verifier. This phase is only executed once for each user during registration process. For each key exchange and authentication process, only phase 1 and 2 are executed.

Phase 1: Authenticated Key Exchange

Client initiation. Randomly samples s_1 and e_1 from $D_{Z^n,\sigma}$ and computes ephemeral public key $p_i = a \cdot s_1 + 2e_1$. Send username I and ephemeral public key p_i to server.

Server Response. Search for client's salt *salt* and verifier v according to username I in database. Randomly samples s_1', e_1' and e_1''' from $D_{Z^n,\sigma}$ and computes ephemeral public key $p_j = a \cdot s_1' + 2e_1' + v$ and $u =$ SHAKE-128(SHA3-256$(p_i \| p_j)$). Compute key exchange material $k_j = v \cdot s_1' + 2e_1''' + p_i \cdot s_1' + u \cdot v$, signal $w_j = \text{Cha}(k_j)$. Send *salt*, p_j and w_j to client.

Client finish. Compute $v = a \cdot s_v + 2e_v$ where s_v and e_v are sampled from $D_{Z^n,\sigma}$ using seed $seed1 =$ SHA3-256$(salt \|$SHA3-256$(I \| pwd))$ and $seed2 =$ SHA3-256$(seed1)$ for PRNG respectively. Randomly samples e_1'' from distribution $D_{Z^n,\sigma}$ and compute $u =$ SHAKE-128(SHA3-256$(p_i \| p_j)$). Compute key exchange material $k_i = (p_j - v) \cdot s_v + 2e_1'' + (p_j - v) \cdot s_1 + u \cdot v$. Compute final shared session key $sk_i =$ SHA3-256$(\text{Mod}_2(k_i, w_j))$. Delete all variables except p_i, p_j and sk_i from memory.

Server finish. Compute final shared session key $sk_j =$ SHA3-256(Mod$_2$ (k_j, w_j)). Delete all variables except p_i, p_j and sk_j from memory.

RLWE-SRP protocol is illustrated in Fig. 2:

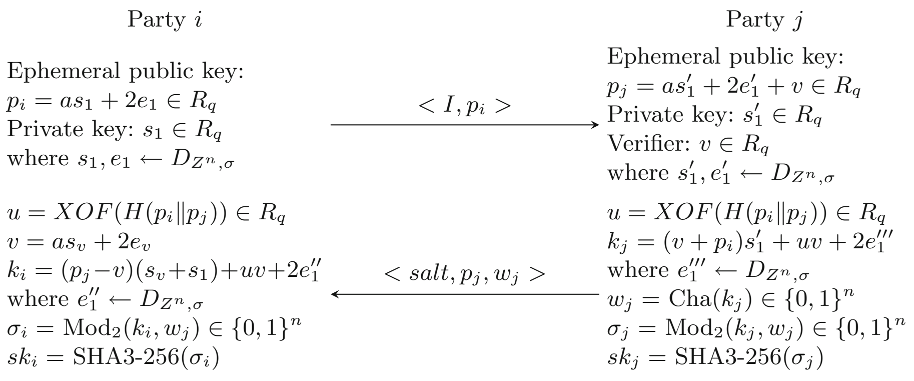
Phase 2: Verification

Verification steps of RLWE-SRP are identical to original SRP protocol and given as follows:

$C \to S$: Client computes $M_1 =$ SHA3-256$(p_i \| p_j \| sk_i)$ and sends to server. Server computes $M_1' =$ SHA3-256$(p_i \| p_j \| sk_j)$. If $M_1 = M_1'$, key exchange is successful and client is authenticated.

$S \to C$: Server computes $M_2' =$ SHA3-256$(p_i \| M_1' \| sk_j)$ and sends to client. Client computes $M_2 =$ SHA3-256$(p_i \| M_1 \| sk_i)$. If $M_2 = M_2'$, key exchange is successful and mutual authentication is achieved.

Note that reader may consider that verification step gains additional roundtrip at first glance. However in practice, this might not be the case. For verification step from $C \to S$, client can send encrypted data alongside M_1. Server first verifies identity with above approach, then decrypt the data using shared



Party i

Ephemeral public key:
$p_i = as_1 + 2e_1 \in R_q$
Private key: $s_1 \in R_q$
where $s_1, e_1 \leftarrow D_{Z^n,\sigma}$

$\xrightarrow{< I, p_i >}$

Party j

Ephemeral public key:
$p_j = as_1' + 2e_1' + v \in R_q$
Private key: $s_1' \in R_q$
Verifier: $v \in R_q$
where $s_1', e_1' \leftarrow D_{Z^n,\sigma}$

$u = XOF(H(p_i \| p_j)) \in R_q$
$k_j = (v + p_i)s_1' + uv + 2e_1'''$
where $e_1''' \leftarrow D_{Z^n,\sigma}$
$w_j = \text{Cha}(k_j) \in \{0,1\}^n$
$\sigma_j = \text{Mod}_2(k_j, w_j) \in \{0,1\}^n$
$sk_j = \text{SHA3-256}(\sigma_j)$

$\xleftarrow{< salt, p_j, w_j >}$

$u = XOF(H(p_i \| p_j)) \in R_q$
$v = as_v + 2e_v$
$k_i = (p_j - v)(s_v + s_1) + uv + 2e_1''$
where $e_1'' \leftarrow D_{Z^n,\sigma}$
$\sigma_i = \text{Mod}_2(k_i, w_j) \in \{0,1\}^n$
$sk_i = \text{SHA3-256}(\sigma_i)$

Fig. 2. Post-quantum secure remote password protocol

key generated from authenticated key exchange phase. If server fails to verify the identity of client, then he cannot decrypt the data. Same approach applies to server, i.e., for verification step from $S \rightarrow C$, server can send encrypted data alongside M_2. Client first verifies identity of server, then decrypt the data.

3.2 Correctness

Correctness of RLWE-SRP is guaranteed with properly chosen parameter. We know that $sk_x = \text{Mod}_2(k_x, w_j)$ ($x = i$ or j) and core notion of error reconciliation in DING12 is the difference between key exchange materials k_i and k_j is very small. If k_i and k_j are sufficiently close (i.e., $\|k_i - k_j\|_\infty \leq$ error tolerance δ), correctness of RLWE-SRP can be guaranteed with overwhelming probability. As we recalled in Sect. 2.2 and [16], if $\|k_i - k_j\|_\infty \leq \frac{q}{4} - 2$, then both sides can derive same output from $\text{Mod}_2()$, i.e., same session key.

For k_i and k_j:

$$\begin{aligned} k_i &= (p_j - v)s_v + (p_j - v)s_1 + uv + 2e_1'' \\ &= \boxed{as_v s_1'} + 2e_1' s_v + \boxed{as_1 s_1'} + 2s_1 e_1' + \boxed{uv} + 2e_1''. \end{aligned} \quad (1)$$

$$\begin{aligned} k_j &= vs_1' + p_i s_1' + uv + 2e_1''' \\ &= \boxed{as_v s_1'} + 2e_v s_1' + \boxed{as_1 s_1'} + 2e_1 s_1' + \boxed{uv} + 2e_1'''. \end{aligned} \quad (2)$$

Note that in (1) and (2), terms in boxes have significantly larger l_∞-norm than other small error terms. If $\|k_i - k_j\|_\infty = \|(2e_1' s_v + 2s_1 e_1' + 2e_1'') - (2e_v s_1' + 2e_1 s_1' + 2e_1''')\|_\infty \leq \|8se + 4e\|_\infty \leq \frac{q}{4} - 2$, where s and e are sampled from same distribution $D_{Z^n,\sigma}$. Correctness of key exchange is guaranteed with overwhelming probability if above inequality holds.

Lemma 1 **([30], lemma 2.5).** *For* $\sigma > 0$, $r \geq 1/\sqrt{2\pi}$, $\Pr[\|x\|_2 > r\sigma\sqrt{n}; x \leftarrow D_{Z^n,\sigma}] < (\sqrt{2\pi e r^2} \cdot e^{-\pi r^2})^n$.

Our technique for choosing parameters is exact same as [15,16,32]. Moreover, $\|k_i - k_j\|_\infty$ and error tolerance are very similar with above three constructions, therefore we omit details here. Parameter choice which guarantees overwhelming success probability (much higher than $1 - 2^{-1024}$) is presented in Sect. 4.1. We note that much more compact parameter q can be chosen easily using Lemma 1.

3.3 Security of RLWE-SRP

We are able to prove the security of our RLWE-SRP protocol under Universally Composable security model (UC-secure) [12]. UC-secure is one of the strongest simulation-based security model. It overcomes shortcomings in game-based models and takes more general simulation-based techniques. It allows secure arbitrary compositions with other primitives once they are proven secure. However, due to page limitation, we do not present formal security proofs in conference version of this paper. Instead, we take similar approach as original SRP paper to show that our RLWE-SRP remain secure with same attacks as original SRP paper suggested. In original SRP paper [31], they do not present formal security analysis for SRP protocol. They show that SRP remain secure under various practical attacks. We also consider such attacks for our RLWE-SRP and categorize them into four major aspects. Moreover, we consider threat from quantum adversary, which attempts to break RLWE-SRP by attacking RLWE problem.

Reduction to RLWE. Fortunately, our RLWE-SRP is built based on RLWE problem, whose hardness is directly reduced to hard lattice problems. Public key, secret key and verifier are constructed as RLWE problem requires. Since no public known algorithms can solve lattice/LWE/RLWE problems on classic and quantum computers with properly chosen parameters, our construction remain secure against such adversaries. In comparison with original SRP, which relies on hardness Diffie-Hellman problem, can be solved easily by quantum adversaries. For RLWE-based constructions (including RLWE-SRP), adversary cannot recover secret key or error term from ephemeral public key. Properly chosen parameters guarantee hardness of RLWE instance. If adversary can break our protocol by recovering secret key, then he can also break RLWE problem and underlying lattice problems.

Hardness of password recovery. Practically, it is possible that server is compromised, therefore adversary may get hands on the verifier. Adversary may attempt to recover user's password from verifier. However, this will not work for both SRP and RLWE-SRP. This is also an advantage for SRP and RLWE-SRP. From construction of RLWE-SRP, we can see that: (1) Only the client who knows correct password can be only computed with correct s_v and e_v, i.e., correct $seed1$ and $seed2$. $salt$ is 256-bit long and uniform randomly generated by client. Client does not need to store s_v and e_v since they can be computed with $salt$, I and correct pwd, where pwd is only known to himself. I and $salt$ are publicly transmitted; (2) Adversary who obtain *seed*1 or *seed*2 cannot compute *pwd* reversely thanks to strong SHA3-256 function, randomness from 256-bit *salt* and strong PRNG; If one can compute *pwd* with public-known elements, then he can at least break hash function and PRNG. (3) v is indistinguishable from uniform random since $v = a \cdot s_v + 2e_v$ is a standard RLWE sample.

If adversary wishes to recover password from v, he first need to solve RLWE problem to recover s_v, then break PRNG in order to find correct seeds, and find preimage of hash value. If one can distinguish v from uniform random, then he can solve decision-RLWE problem, which implies that adversary can solve hard lattice problems. To summarize, If one can recover *pwd* from verifier v, then he can at least solve RLWE problem, break SHA3-256 hash function and PRNG simultaneously.

No password leakage from key exchange. In original SRP paper, they consider password leakage from final shared key and key exchange executions. For leakage of password, we discuss this issue in above subsection, showing that if the adversary can recover password, they he can break various cryptographic primitives. For leakage from shared session key, since we use error reconciliation mechanism in [16] and they proved that signal value and public key leak no information about final shared key, therefore our construction is strong. Output of $\text{Mod}_2()$ and signal w are indistinguishable from uniform random, which is also proven in [16]. Ephemeral public key is also indistinguishable from uniform random, whose security is guaranteed by hardness of RLWE problem and properly chosen parameters.

We note that public keys in RLWE-SRP are ephemeral, namely all parties need to generate fresh keys for key exchange and authentication, therefore our RLWE-SRP is also forward secure like ephemeral Diffie-Hellman key exchange. Adversary cannot decrypt past sessions if current session is compromised.

RLWE-SRP is also resistant to Denning-Sacco attack [13], where attacker tries to impersonate user with captured session key or recover password with brute-force search. Analysis in original SRP implies that such attack can be detected and defeated in verification phase. Since RLWE-SRP uses exact same verification phase as original SRP without any modification, our RLWE-SRP also enjoys this nice property.

No impersonation even if verifier is stolen. Since the pre-shared verifier v can only be computed with correct password, adversary cannot impersonate as user to authenticate himself to the server if server is compromised and verifier is leaked. In k_i computation, adversary has to generate correct s_v (not verifier $v = a \cdot s_v + e_v$) in order to impersonate as legit user. However, s_v and e_v can only be computed with correct password, therefore adversary cannot impersonate as user even if he owns v. However, the adversary can impersonate as server for user to login. This works for almost all security protocols and applications once such information is leaked. In RLWE-SRP, since user does not send actual password to server and password cannot be recovered from verifier or during key exchange, adversary cannot acquire user's actual password even if he impersonates as legit server. This nice property also holds for original SRP. Adversary can recover s_v

and e_v only by solving RLWE problem given lost verifier $v = a \cdot s_v + 2e_v$, which is known be very hard.

Rest of security analysis in original SRP paper are mathematical constraints on choosing parameters regarding to discrete logarithm computations since SRP is built based on DLP. Our RLWE-SRP is built based on RLWE problem, therefore we can safely ignore these analysis. Concrete parameter choice and security level estimation are presented in Sect. 4.

Compared with the RLWE-based PAKE protocol in [15], advantages of our RLWE-SRP come from the usage of verifier. Using actual password or hash of password directly is not strong enough as we discussed above. In [15], if password or hash of password is stolen, adversary can immediately impersonate as legit user and authenticate to user, while this attack does not work for RLWE-SRP. Another advantage is that if the hash function is broken, i.e., one can find preimage of hash value, password can be recovered successfully in [15]. For RLWE-SRP, we first use $seed1 = \text{SHA3-256}(salt \| \text{SHA3-256}(I \| pwd))$ to generate a seed using hash function, then we generate $s_v \in R_q \leftarrow D_{Z^n,\sigma}$ with $seed1$ for PRNG. If adversary can recover password from RLWE-SRP, then he can break hash function, PRNG and RLWE problem.

4 Instantiation, Implementation and Performance

In this section, we introduce parameter choice, security level estimation, implementation details and benchmark of RLWE-SRP. In order to demonstrate efficiency of our protocol and implementation, we also compare performance with original SRP protocol [31] and J-PAKE [20] protocols which are vulnerable to quantum attacks, and two RLWE-based PAKE protocols in [15].

4.1 Parameter Choice and Security Level Estimation

We first choose practical parameter for our protocol. We choose $\sigma = 3.192$, $n = 1024$ and $q = 1073479681$ (approximately 30 bits). Our choice of q can guarantee the correctness of our protocol with extremely low failure probability (much lower than 2^{-1024}). Our 30-bit q is 2-bit smaller than modulus in [9,15], where [9] claims roughly $2^{-131072}$ failure probability with very similar $k_i - k_j$ norm and error tolerance for reconciliation mechanism. Moreover, our choice of q can instantiate NTT for quick polynomial multiplication efficiently since it holds $q \equiv 1 \mod 2n$. This technique is also adopted in various implementations of RLWE-based protocols (e.g., [5]) as well. $\sigma = 3.192$ and $n = 1024$ follow parameter choices of various RLWE-based key exchange protocols (e.g., [9,15,32] etc.). We set statistical distance between sampled distribution and discrete Gaussian distribution to be 2^{-128} to preserve high statistical quality and security.

Security level of our parameter choice is estimated using LWE estimator in [4]. The state-of-the-art LWE estimator gives a thorough security estimation for both LWE and RLWE-based cryptosystems. It evaluates security level of cryptosystems by computing attack complexity of exhaustive search, BKW, lattice reduction, decoding, reducing BDD to unique-SVP and MITM attacks. Due to the fact that currently there are no known attacks that take advantage of the ring structure in RLWE, therefore LWE estimator can be used to estimate security level of RLWE-based constructions. LWE estimator is regarded as most accurate, robust and easy-to-use tool of security level estimation for LWE and RLWE-based constructions. Given any parameters, LWE estimator outputs computation and space complexity of various attacks. [9] and various works also use this tool to estimate security of their parameter choice.

Instruction for estimation is given as follows:

- load("https://bitbucket.org/malb/lwe-estimator/raw/HEAD/estimator.py")
- n, alpha, q = 1024, alphaf(8, 1073479681), 1073479681
- set_verbose(1)
- _ = estimate_lwe(n, alpha, q, skip=["arora-gb"])

With above estimation approach, our parameter choice offers 209-bit security. We note that since LWE estimator is constantly updated, estimation result from latest version of LWE estimator may different from what we present here.

We note that above security claim is actually pessimistic, since public and private keys in our RLWE-SRP are ephemeral-only, while the attacks listed in LWE estimator requires large amount of RLWE samples to work. Given only one sample from our protocol execution, it is much more difficult to break the protocol by attempting to solve RLWE problem and recover s.

4.2 Implementation, Performance and Comparison

We present portable C++ implementation of RLWE-SRP. Our implementation is done using our modified version of NFLlib library [3]. NFLlib is an efficient NTT-based library for RLWE-based cryptography. We modify latest version of NFLlib to adapt to our design for s_v and e_v, i.e., passing certain seed to random number generator when generating random numbers in s_v and e_v. Our implementation only runs on single core and does not utilize parallel computing techniques. Efficiency of our non-constant time implementation benefits from efficient protocol design and SSE-optimized discrete Gaussian sampling and NTT computations. It is portable and can run on more outdated devices.

For hash functions, we choose Keccak sponge function family (standardized and known as SHA3-*). Hashing computations are done using SHA3-256. We also take advantage of an extendable output function (XOF) in Keccak family - SHAKE-128. It can be used as hash function with desired output length. It is used in generating u in our implementation, i.e., extending SHA3-256 output of $(p_i \| p_j)$ to 4096 bytes with SHAKE-128 ($u =$ SHAKE-128(SHA3-256$(p_i \| p_j)$, 4096)). Each coefficient is stored in "uint32_t" type variable, therefore we use SHAKE-128 to extend SHA3-256$(p_i \| p_j)$ to 4096 bytes to instantiate polynomial coefficients of u.

We test implementation of this work, two PAKE protocols that vulnerable to quantum attacks (original SRP [31] and J-PAKE [20]), the only two quantum-resistant RLWE-based PAKE protocols till now in [15] on same server equipped with 3.4 GHz Intel Xeon E5-2687W v2 processor and 64 GB memory, running 64-bit Ubuntu 14.04. All implementations are compiled by gcc or g++ compiler version 4.8.4 with same optimization flags "-O3 -march=native -m64". We report average runtime over 10,000 executions of all protocols in Table 1:

Table 1. Performance comparison of multiple PAKE protocols

	Security level	Security assumption	Client (ms)	Server (ms)
RLWE-SRP	209-bit	RLWE	0.286	0.257
Original SRP [31]	112-bit	DLP	0.805 (2.81x)	0.804 (3.13x)
J-PAKE [20]	80-bit	DLP	1.499 (5.24x)	1.495 (5.82x)
RLWE-PAK [15]	184-bit	RLWE	3.472 (12.14x)	4.053 (15.77x)
RLWE-PPK [15]	184-bit	RLWE	3.488 (12.20x)	4.041 (15.72x)

Number in parentheses is number of times of corresponding runtime for each protocol compared with this work as baseline. Compared with original SRP protocol, our implementation achieves 3x speedup. Compared with J-PAKE, RLWE-PAK and RLWE-PPK, our implementation is 5.53x, 13.96x and 13.96x faster respectively. Benchmark proves the efficiency of our protocol and implementation. We remark that for our RLWE-SRP protocol, client side computation costs slightly more time than server side since client needs to compute verifier v (i.e., sampling s_v and e_v then NTT) while server does not.

Communication cost of all PAKE protocols compared in Table 1 is given in Table 2. Here we assume that for all PAKE protocols, size of username is 64 bytes, salt is 256 bits, output length of hash function is 256 bits. For original SRP and J-PAKE, 80-bit security implies choosing 1024-bit prime modulus, 112-bit security implies choosing 2048-bit prime modulus.

Table 2. Communication cost comparison of multiple PAKE protocols

	Security level	Client→Server (Byte)	Server→Client (Byte)
RLWE-SRP	209-bit	3936	4032
Original SRP [31]	112-bit	352	320
J-PAKE [20]	80-bit	640	640
RLWE-PAK [15]	184-bit	4192	4256
RLWE-PPK [15]	184-bit	4192	4224

We can see that due to mathematical structure of RLWE and parameter choice, RLWE-based protocols have much larger communication cost than discrete logarithm-based ones. In fact, all RLWE-based constructions have much larger communication cost compared with current public key algorithms (e.g., Diffie-Hellman, ECDH, RSA etc.) due to public key in RLWE-based constructions is degree n polynomial in R_q modulus q. Size of public key is estimated to be $n \cdot \lceil \log_2 q \rceil / 8$ bytes. We can also choose smaller q by increasing key exchange failure probability to more practical value (e.g., 2^{-60}) and get roughly 18-bit modulus q. Moreover, choosing slightly smaller σ can reduce size of modulus q by 1 bit. With much smaller q and slightly smaller σ, security level of parameter choice can be increased. Compared with two RLWE-based PAKE protocols in [15], RLWE-SRP has smaller communication cost due to 2-bit smaller modulus q.

5 Conclusions

In this paper, we present the first practical and efficient RLWE-based post-quantum secure remote protocol. Our RLWE-SRP enjoys various nice security properties, including: (1) Resistant to quantum attacks; (2) Mutual authenticated key exchange; (3) No actual password is sent to server; (4) Attacker cannot impersonate client to authenticate even if verifier is stolen; (5) Attacker cannot recover user password with stolen verifier etc. Our 209-bit secure parameter choice and efficient implementation further highlight the practicality of this work. Compared with PAKE protocols that are vulnerable to quantum attacks and two RLWE-PAKE protocols, our RLWE-SRP implementation is much faster. We believe our RLWE-SRP is one step forward for secure remote password protocol towards post-quantum world.

Acknowledgement. We would like to thank anonymous reviewers for valuable feedbacks. This work is supported by China Scholarship Council, National Natural Science Foundation of China (Grant No. 61672092) and Fundamental Research Funds for the Central Universities (Grant No. 2017YJS038). Jintai Ding is partially supported by NSF grant DMS-1565748 and US Air Force grant FA2386-17-1-4067.

References

1. Bliss - strongSwan. https://wiki.strongswan.org/projects/strongswan/wiki/BLISS
2. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_14
3. Aguilar-Melchor, C., Barrier, J., Guelton, S., Guinet, A., Killijian, M.-O., Lepoint, T.: NFLlib: NTT-based fast lattice library. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 341–356. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_20
4. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptology **9**(3), 169–203 (2015)

5. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange-a new hope. IACR Cryptology ePrint Archive 2015, 1092 (2015)
6. Apple: iOS Security. https://www.apple.com/business/docs/iOS_Security_Guide.pdf
7. Bellovin, S.M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: Proceedings of 1992 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 72–84. IEEE (1992)
8. Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: take off the ring! practical, quantum-secure key exchange from lwe. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1006–1018. ACM (2016)
9. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 553–570. IEEE (2015)
10. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_12
11. Braithwaite, M.: Experimenting with Post-Quantum Cryptography. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html
12. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings of 42nd IEEE Symposium on Foundations of Computer Science 2001, pp. 136–145. IEEE (2001)
13. Denning, D.E., Sacco, G.M.: Timestamps in key distribution protocols. Commun. ACM **24**(8), 533–536 (1981)
14. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
15. Ding, J., Alsayigh, S., Lancrenon, J., Saraswa, R.V., Snook, M.: Provably secure password authenticated key exchange based on RLWE for the post-quantum world. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 183–204. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_11
16. Ding, J., Xie, X., Lin, X.: A simple provably secure key exchange scheme based on the learning with errors problem. IACR Cryptology EPrint Archive 2012, 688 (2012)
17. Dousti, M.S., Jalili, R.: Forsakes: a forward-secure authenticated key exchange protocol based on symmetric key-evolving schemes. Adv. Math. Commun. **9**(4), 471–514 (2015). http://aimsciences.org/journals/displayArticlesnew.jsp?paperID=11939
18. Goldberg, J.: Three layers of encryption keeps you safe when ssl/tls fails. https://blog.agilebits.com/2017/02/23/three-layers-of-encryption-keeps-you-safe-when-ssltls-fails/
19. Gonzláez, S., Huguet, L., Martínez, C., Villafañe, H.: Discrete logarithm like problems and linear recurring sequences. Adv. Math. Commun. **7**(2), 187–195 (2013). http://aimsciences.org/journals/displayArticlesnew.jsp?paperID=8550
20. Hao, F., Ryan, P.Y.A.: Password authenticated key exchange by juggling. In: Christianson, B., Malcolm, J.A., Matyas, V., Roe, M. (eds.) Security Protocols 2008. LNCS, vol. 6615, pp. 159–171. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22137-8_23

21. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 636–652. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_37
22. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_33
23. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_1
24. Micheli, G.: Cryptanalysis of a noncommutative key exchange protocol. Adv. Math. Commun. **9**(2), 247–253 (2015). http://aimsciences.org/journals/displayArticlesnew.jsp?paperID=11174
25. Morhaime, M.: Important security update. http://us.blizzard.com/en-us/securityupdate.html
26. Peikert, C.: Lattice cryptography for the internet. In: Mosca, M. (ed.) PQCrypto 2014. LNCS, vol. 8772, pp. 197–219. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11659-4_12
27. Perrin, T., Wu, T., Mavrogiannopoulos, N., Taylor, D.: Using the secure remote password (SRP) protocol for TLS authentication. https://tools.ietf.org/html/rfc5054
28. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM (JACM) **56**(6), 34 (2009)
29. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
30. Stephens-Davidowitz, N.: Discrete gaussian sampling reduces to CVP and SVP. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1748–1764. Society for Industrial and Applied Mathematics (2016)
31. Wu, T.D., et al.: The secure remote password protocol. In: NDSS, vol. 98, pp. 97–111 (1998)
32. Zhang, J., Zhang, Z., Ding, J., Snook, M., Dagdelen, Ö.: Authenticated key exchange from ideal lattices. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 719–751. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_24