Journal of Information Security and Applications 84 (2024) 103826

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

SL3PAKE: Simple Lattice-based Three-party Password Authenticated Key Exchange for post-quantum world

Vivek Dabra [a], Saru Kumari [b,*], Anju Bala [c], Sonam Yadav [d]

[a] Department of Computer Science & Engineering, Panipat Institute of Engineering & Technology, Samalkha, Panipat, India
[b] Department of Mathematics, Chaudhary Charan Singh University, Meerut, India
[c] Computer Science & Engineering Department, Thapar Institute of Engineering & Technology, Patiala, India
[d] Department of Mathematics, SGT University, Gurugram, India

ARTICLE INFO

Keywords:
Post-quantum cryptography
Lattice based cryptography
Ring Learning With Error
Key exchange
Signal leakage attack

ABSTRACT

Three-party Password Authenticated Key Exchange (3PAKE) is a protocol where two parties generate the same session key with the help of a trusted server. With the evolution of quantum computers, there is a growing need to develop the 3PAKE protocols that can resist the quantum attacks. Hence, various 3PAKE protocols have been proposed based on the famous Ring Learning With Error (RLWE) problem. But we find out that all these protocols are vulnerable to signal leakage attacks if their public/private keys are reused. Also, the design of these protocols are pretty complex, thus making these protocols highly inefficient. Hence, to overcome the above issues, we have proposed Simple Lattice-based 3PAKE (SL3PAKE), which is simple in its design and resists signal leakage attack if its public/private keys are reused. The order and flow of messages in the proposed SL3PAKE protocol is quite natural without added complexity, thus makes it simple 3PAKE protocol. Finally, we present the comparative analysis based on communication overhead among the proposed SL3PAKE and other three-party protocols. From the analysis, it has been shown that the proposed SL3PAKE protocol has much less communication overhead/communication rounds than the other three-party protocols.

1. Introduction

Key exchange is the fundamental cryptographic primitive that allows two or more parties to exchange a common secret key securely over an insecure network. It is one of the crucial cryptographic algorithms for the security protocols used over the internet, including the Voice over Internet Protocol (VOIP), Transport Layer Security (TLS), and Session Initiation Protocol (SIP). Unlike other Authenticated Key Exchange (AKE), the Password Authenticated Key Exchange (PAKE) enables the parties involved in key exchange to authenticate each other using human memorable low entropy password. The human memorable low-entropy passwords are easy to remember, and do not require special hardware to store in server's database.

The researchers initially proposed the two-party password-authenticated key exchange (2PAKE) protocols to secure communication between the two parties involved. In this approach, the two parties will share the password beforehand to authenticate each other in each instantiation of the protocol. Over time, researchers have proposed many versions of 2PAKE [1–3]. However, in the 2PAKE protocol, each party has to remember the large number of passwords corresponding to the other parties involved in the key exchange. For instance, if there are n parties involved in 2PAKE, then there will be $\frac{n.(n-1)}{2}$ passwords to be shared among the parties. Also, these passwords must be stored securely by the participating parties. This will leads to the cumbersome task of managing and storing a large number of passwords securely. If any of the participating party's devices get compromised, other parties also have to pay the price of this breach.

To resolve the above issue, another variant of PAKE termed 3PAKE has been proposed. In 3PAKE, an additional party, the server, securely stores the passwords of all participating parties in the key exchange. 3PAKE is essentially the 2PAKE with further use of the server for authentication purposes. The server in 3PAKE acts as a trusted third party which authenticate each of the participating parties. With the deployment of the server, an additional measure of security can be employed to secure the stored passwords of the participating parties. The server will now store the verifiers in its database that is computed from the passwords of legal users. Now, each participating party can use a single password in establishing the authentication of the participating parties. Therefore, the 3PAKE protocol provides a suitable way

* Corresponding author.
E-mail addresses: drvivek.cse@piet.co.in (V. Dabra), saryusiirohi@gmail.com (S. Kumari), anjubala@thapar.edu (A. Bala), sonamyadav20jan@gmail.com (S. Yadav).

https://doi.org/10.1016/j.jisa.2024.103826

Available online 2 July 2024

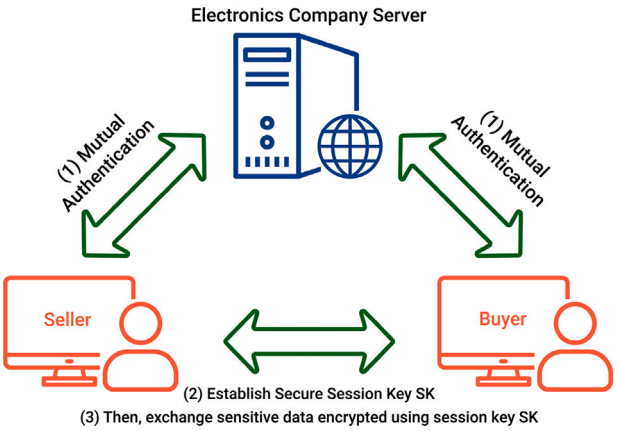

Fig. 1. An example of real-life scenario where 3PAKE can be used for data exchange between buyer and seller.

for large-scale user-to-user communication by eliminating the need to remember different passwords for different users. Thus, as shown in Fig. 1, the proposed 3PAKE can be used to establish the secure session key between buyer and seller with the help of a trusted server of an electronic company. This will further help to exchange the sensitive data between buyer and seller that is encrypted using the established session key.

Many researchers have proposed numerous 3PAKE protocols [4–8] due to the aforementioned advantages. However, these 3PAKE protocols depend upon classical mathematical problems like discrete logarithmic and integer factorization problem for its security. In 1999, Shor showed that quantum computers can solve these mathematical problems in polynomial time [9]. In recent times, quantum computers have evolved significantly [10]. This motivated many researchers to propose the 3PAKE that can resist quantum-attacks. Xu et al. [11] in 2017, proposed the first three-party password-authenticated key exchange (3PAKE) using the RLWE problem [12] of lattices. The protocol's security has been proven in the Random Oracle Model (ROM). Additionally, the protocol has been implemented using the LatticeCrypto library, with claims of its efficiency for practical applications. However, despite the above claims, Choi et al. [13] in 2018, found the Xu et al.'s protocol complicated and computationally inefficient.

Further, Choi et al. [13] proposed a new three-party key exchange protocol named "AtLast". Choi et al. [13] claimed their protocol "AtLast" as a simple three-party PAKE protocol based on the RLWE problem of lattices. The protocol inherits the design of Ding et al.'s [14] RLWE-PPK protocol to three-party settings by using the generic approach of Abdalla et al. [15]. Further, Choi et al. [13] also claimed their protocol "AtLast" to be quantum-secure and resist various known attacks. In addition to the assertions mentioned earlier, we have found that the proposed protocol is susceptible to a signal leakage attack [16] when reusing its public and private keys. In 2019, Liu et al. [17] proposed the provably secure three-party password-authenticated Key Exchange (3PAKE) using the RLWE problem. The security of the protocol is not based on any external cryptographic primitives, but rather on the hardness of the RLWE problem. The protocol establishes its provable security by using a modified version of Bellare et al.'s model [1, 18]. But we find that the protocol's design gets extra complicated to provide the authentication of the participating parties. Furthermore, reusing the protocol's public and private keys makes it vulnerable to modified signal leakage attacks, as discussed in section 6 of the publication by Ding et al. [19]. Recently, Islam [20] proposed PB-3PAKA (Password-based three-party authenticated key agreement protocol) for mobile devices. PB-3PAKA is based on ideal lattices and its security is dependent on hardness assumption of RLWE problem. Through analysis, we have determined that this protocol is susceptible to a signal leakage attack when reusing its public/private keys. Therefore, we have proposed the simple lattice-based three-party password-authenticated key exchange (SL3PAKE). The protocol, as its name suggests, is simple in design and can resist signal leakage attacks when reusing its public/private keys.

1.1. Our motivation & contribution

The motivation behind the design of SL3PAKE is that the existing lattice-based 3PAKE protocols are vulnerable to signal leakage attack if their public/private keys are reused. Further, the protocols are complex in their design to provide authentication to participating parties, and hence, they are not suitable for practical applications. Thus, to overcome all the above issues, SL3PAKE has been proposed. The following are the contributions of our work:

- To overcome signal leakage attacks (SLA) in the previous post-quantum 3PAKE protocols, we have proposed the simple lattice-based three-party password-authenticated key exchange (SL3PAKE) protocol. The security of the protocol solely relies on the hardness assumption of the Ring Learning with Errors (RLWE) problem.
- The protocol is simple as it is designed without extra cryptographic primitives like digital signature, message authentication code. The use of digital signatures and message authentication code for designing key exchange protocols is known as **SIGn-MAc** approach, which is proposed by Krawczyk [21] (also named as SIGMA family of key exchange protocols). The disadvantage of this approach is that it is assumed that digital signatures and message authentication codes are existentially unforgeable and thus, the security of the key exchange protocol is dependent on digital signatures and message authentication codes.
- The concept of mutual authentication will ensure that the participating parties and the server can mutually authenticate each other.
- The informal security of the proposed protocol has been proved against signal leakage attack (SLA).
- Additionally, the formal security of the proposed protocol has been demonstrated using the Real-Or-Random (ROR) model.
- Finally, our concrete parameter of choice, implementations, and comparative analysis based on communication cost shows that the proposed protocol is efficient and practical.

1.2. Organization

The roadmap for the rest of the article is as follows: Section 2 discusses the related work. Section 3 is for preliminaries. Section 4 describes the proposed SL3PAKE protocol along side the correctness condition and informal security of the proposed protocol. Also, we have proven the formal security of the proposed SL3PAKE protocol using the ROR model in Section 5. Section 6 demonstrates the chosen concrete parameter along with the implementation results of the proposed protocol. Additionally, this section includes a comparative analysis of the communication cost among the proposed SL3PAKE protocol and other three-party post-quantum protocols. Finally, Section 7 discusses the conclusion and future scope.

2. Related work

In this section, we will examine the two-party PAKE (2PAKE) and three-party PAKE (3PAKE) protocols that utilize Learning with Errors (LWE) [22] and Ring Learning with Errors (RLWE) [12] problems of lattices.

2.1. Lattice-based two-party key exchange (2KE/2AKE/2PAKE)

Researchers find the Lattice-based problems an excellent alternative to design the quantum-resistant cryptographic primitives. The two

famous lattice-based problems are the Learning With Errors (LWE) and Ring Learning With Errors problem. Both problems are the base of the hardness of various key exchange protocols claimed to be quantum secure. There are two different kinds of key exchange protocols that are based on the LWE/RLWE problem. The first is the reconciliation type, and the other is the encryption type. The reconciliation type key exchange protocols [23–25] offer greater bandwidth compared to encryption type key exchange protocols [26,27]. Ding et al. [28] proposed the first reconciliation-based protocol in 2012. They proved the passive security of the scheme against a PPT (Probabilistic Polynomial-Time) adversary based on the difficulty of the LWE/RLWE problems. The scheme is the counterpart of basic Diffie–Hellman key exchange and, therefore, does not provide any authentication of the parties involved within the key exchange. There are several other schemes [14,29] that utilize the error reconciliation method of Ding et al. [28] in their protocol, integrating with different authentication methods or provide implicit authentication to design authenticated key exchange (AKE) protocol. In 2014, Peikert [30] observed that the error reconciliation method of Ding et al.'s [28] scheme generates a biased key that cannot be directly utilized as a secret key and must be uniformly extracted using a strong extractor. He further asserted that his error reconciliation method directly generates an unbiased key that can serve as the secret key. Similar to Ding et al.'s [28] case, many other key exchange schemes [23,24] utilize Peikert's error reconciliation technique, and they integrate their key exchange scheme with the TLS. The authors of these schemes claim that their schemes are efficient enough for real-world applications.

To verify their claims, Gao et al. [31] in 2017 conducted an experiment. The paper compares the efficiency of the schemes proposed by Bos et al. [24] and Ding et al. [28]. As opposed to the Bos et al. scheme, which employs Peikert's error reconciliation procedure, the Ding et al. scheme has its own error reconciliation method. According to this work, the Ding et al. scheme is 11 times quicker than the Bos et al. scheme, while the Ding et al. scheme's error reconciliation method is 1.57 times faster than Peikert's scheme. Many key exchange schemes utilize the reconciliation mechanisms that Ding et al. [28] and Peikert [30] developed. In 2016, Alkim et al. [25] introduced the key exchange scheme that extends Peikert's [30] error reconciliation mechanism. Popularly known as "New Hope", the approach attracted a lot of attention when Google used it to secure their Chrome browser. The test version of Google Chrome [32] utilized a novel, quantum-secure algorithm known as New Hope to commence the testing process. On the same note, Feng et al. in 2018 developed the first perfect lattice-based protocol for mobile devices employing the error reconciliation mechanism of Ding et al.'s scheme [28]. The ROM model has demonstrated the security of the suggested protocol, and authors claim that it provides quantum security [33]. Despite their claims, Dabra et al. demonstrated that their protocol is vulnerable to attacks such as signal leakage attacks, spoofing attacks, manipulation attacks, and attacks on user anonymity. [34]. Recently, Seyhan et al. [35] and Basu et al. [36] proposed the quantum secure 2PAKE schemes for two-party communication. The security of both schemes is dependent on the hardness assumption of the module learning with rounding (MLWR) problem.

2.2. Lattice based three-party key exchange (3PAKE)

In 2017, Xu et al. [11] proposed the first lattice-based 3PAKE using RLWE problem. The protocol is known as RLWE-3PAKE because it relies on RLWE-PAK, as described in the work by Ding et al. [14]. The protocol is provably secure, and its security is proved in the ROM model. Xu et al. [11] implemented the protocol using the LatticeCrypto library with their computation results showing that their proposed protocol is highly efficient. But Choi et al. [13] in 2018, found that Xu et al.'s protocol is complicated and is not efficient. Hence, Choi et al. [13] proposed another three-party lattice-based 3PAKE which is also termed as "AtLast". AtLast is a simple 3PAKE protocol that inherits the design of Ding et al.'s [14] RLWE-Password Protected Key Exchange (RLWE-PPK) protocol to three-party settings by using the generic approach of Abdalla et al. [15]. The proposed protocol has been shown to resist various known attacks. However, our careful analysis revealed that the protocol is vulnerable to signal leakage attacks when public/private keys are reused. In 2019, Liu et al. [17] proposed another provably secure 3PAKE using RLWE problem of lattices. The protocol's error reconciliation method is based on Peikert's reconciliation mechanism [30]. The concept of mutual authentication is there in the proposed protocol in which users and the server can authenticate each other. The protocol's provable security is demonstrated by applying a modified version of Bellare et al.'s model [1,18]. Also, the protocol resists undetectable online/off-line password guessing attacks and enjoys forward secrecy and quantum attacks resistance. The protocol is effective in practice, as demonstrated by the specific parameter selections and implementation evidence. In contrast to Liu et al.'s assertions [17], we discover that the protocol is intricate and prone to signal leakage attack (SLA). Recently, Islam [20] proposed PB-3PAKA three-party protocol using RLWE problem of lattices and claim to be post-quantum secure. However, we have discovered that reusing both the public and private keys of the protocol makes it vulnerable to a signal leakage attack (SLA).

3. Preliminaries

We will go over the basics of lattice cryptography and the password-authenticated key exchange mechanism in this part.

1. **Lattice-based Cryptography**
   A set of points in n-dimensional euclidean space is called a lattice. Let $\mathbb{R}^s$ be the s dimensional Euclidean space. A lattice in $\mathbb{R}^s$ is the set

   $$L(c_1, c_2, \ldots, c_r) = \sum_{i=1}^{r} x_i.c_i \tag{1}$$

   such that $x_i \in \mathbb{Z}$, of all integral combinations of r linearly independent vectors $c_1, c_2, \ldots, c_r$ in $\mathbb{R}^s$, where integer r is a rank and s is the dimension of lattice such that $s \geq r$. In case, if $r = s$ then the lattice is called full rank lattice. The vector c_i, where $i = 1, 2, \ldots, r$ is called lattice basis vectors, which can be represented in matrix form as

   $$B = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_r \end{bmatrix} \in \mathbb{R}^{s \times r}$$

   where c_i is a column vector. Using matrix notation Eq. (1) can be rewritten as

   $$L(B) = Bx : x \in \mathbb{Z}^r$$

   where Bx is a matrix–vector multiplication.
   Short vector problem, closest vector problem, LWE, and RLWE are the famous lattice problems to design the cryptographic primitives in lattice-based cryptography. The LWE and RLWE [12,22] are well-studied lattice problems that are commonly used to build various cryptographic primitives.
2. **Ring Learning With Error problem**
   Let n be a positive integer satisfying the requirement $n = 2k$, where k is any positive integer and q is a large odd prime satisfying the condition $q = 1 \ mod \ 2n$. Let $f(x)$ be cyclotomic polynomial of the form $x^n + 1$ and R be the ring of polynomials over $\mathbb{Z}$ such that ring $R = \frac{\mathbb{Z}[x]}{\langle f(x) \rangle}$. The element in R are polynomials over $\mathbb{Z}$. Define a ring $R_q = \dfrac{\mathbb{Z}_q[x]}{\langle f(x) \rangle}$, the elements of R_q is a polynomial over $\mathbb{Z}_q$. Let a is any element, $a \in R$ such that $a = a_0 + a_1 x + \cdots + a_{n-1} x_{n-1}$ where $a_i \in \mathbb{Z}$. The $L_2 = \sqrt{\sum_{i=0}^{n-1} a_i^2}$

Table 1
Notation table of the proposed SL3PAKE protocol.

SL3PAKE's notations of parameters
q: odd prime number
R: $\frac{\mathbb{Z}[x]}{<f(x)>}$, Quotient ring of the polynomial ring Where f(x) is Cyclotomic polynomial x^n+1 of degree n and $n=2^k$ where k is any positive integer
R_q: R/qR, Quotient Ring
n: Positive security parameter s.t. $q \ mod \ 2n=1$
χ_β: Discrete Gaussian Distribution over R_q with standard deviation β
$e \leftarrow \chi_\beta$: Sample error term e from χ_β
a: random element in R_q
x_S: public key of server S s.t. $x_S = a.s_S + 2.e_S$ where $s_S, e_S \leftarrow \chi_\beta$
s_S: secret key of server S
x_A: public key of client A s.t. $x_A = a.s_A + 2.e_A$ where $s_A, e_A \leftarrow \chi_\beta$
s_A: secret key of client A
x_B: public key of client B s.t. $x_B = a.s_B + 2.e_B$ where $s_B, e_B \leftarrow \chi_\beta$
s_B: secret key of client B
P, Q: String P concatenated with string Q
$\{n, q, \chi_\beta, a, x_S\}$: System or Public parameters
s_S: Confidential secret key of the server S

and norm $L_\infty = \max\{|a_i|\}_{i=1}^{n-1}$. Let β is the standard derivation and define χ_β the discrete Gaussian distribution on R_q.
The RLWE problem is defined as follows: Let a_i be a uniform sample from R_q and e be an error distribution. Let $A_{S,\chi}$ be the distribution with samples $(a_i, b_i) \in R_q \times R_q$, where $b_i = a_i.s + e$. The search variant of the RLWE problem is to find s from T, where T is the number of samples of the type $(a_i, a_i.s + e_i \ mod \ q)$. Now we will define several lemmas in the following manner.

Lemma 1 ([29]). *"Let R is a polynomial ring over $\mathbb{Z}$. Then,for any $c, d \in R$, $\| c.d \| \leq \sqrt{n}. \| c \| . \| d \|$ and $\| c.d \|_\infty \leq n. \| c \|_\infty . \| d \|_\infty$".*

Lemma 2 ([37,38]). *let $\beta = \omega(\sqrt{logn})$ be a positive real number, the following inequality $Pr_{x \leftarrow \chi_\beta} \| x \| > \beta\sqrt{n} \leq 2^{-n+1}$ holds.*

The characteristic function denoted by Cha() is defined as

$$Cha(m) = \begin{cases} 0 & m \in E \\ 1 & m \notin E \end{cases} \ where \ E = \{-\left\lfloor \frac{q}{4} \right\rfloor, \ldots, \left\lfloor \frac{q}{4} \right\rfloor\}$$

And Mod_2 function is defined as $Mod_2 : \mathbb{Z}_q \times \{0,1\} \rightarrow \{0,1\}$ such that $(m, n) \mapsto \left(m + n.\frac{q-1}{2}\right) \ mod \ q \ \ mod \ 2$, where $m \in \mathbb{Z}_q$ and $n = Cha(m)$.

Lemma 3 ([28]). *Suppose q be an odd prime and R_q is a polynomial over $\mathbb{Z}_q$. Let $l, m \in R_q$ and $l = m + 2e$, where $e \leftarrow \chi_\beta$ such that $\|l - m\| < \frac{q}{4}$ and $n = Cha(m)$. Then $Mod_2(m, n) = Mod_2(l, n)$.*
$Cha(m) = (Cha(m_0), \ldots, Cha(m_{n-1}))$ can be used to extend the $Cha()$ function to elements of R_q, where $m = m_0 + m_1 x + \cdots + m_{n-1}x^{n-1} \in R_q$. Similarly, in the ring R_q, we can expand the $Mod_2()$ function.

4. Proposed SL3PAKE protocol

In this section, we describe the proposed SL3PAKE protocol along with the correctness condition and informal security of the proposed protocol. Table 1 depicts the notations table of the proposed SL3PAKE protocol.

4.1. Setup phase

To finish system initialization, server S performs the subsequent actions:

- Server S chooses the security parameter n as a power of two based on the sensitivity of the application for which the protocol is intended to be used.
- S selects an odd prime integer q such that $q \ mod \ 2n = 1$, a discrete Gaussian distribution χ_β over R_q, and a random element a such that $a \in R_q$.
- Server S calculates the public key x_S as $x_S = a.s_S + 2e_S$, where s_S is the server S secret key and $s_S, e_S \leftarrow \chi_\beta$ at random.
- Server S chooses three one-way hash functions, $h_0()$, $h_1()$, and $h_2()$. The mathematical description of h_0 is as follows:

  $$h_0 : \{0,1\}^* \rightarrow R_q$$

  where, $h_0()$ is the hash function that takes string of variable length as input and output the element e such that $e \in R_q$.
  Also, mathematical description of $h_1()$ is:

  $$h_1 : \{0,1\}^* \rightarrow \{0,1\}^g$$

  where, $h_1()$ is the hash function that takes string of variable length as input and output the message digest of fixed length say g. The $h_1()$ hash function is typically of SHA3 type hash function. The $h_2()$ is described as:

  $$h_2 : \{0,1\}^* \rightarrow \chi_\beta$$

  where, $h_2()$ is the hash function that takes an element e as input such that $e \in R_q$ and output the element x such that x is sampled from χ_β distribution.
- The server saves the hashes of the participants' passwords, $h_0(pw_A)$ and $h_0(pw_B)$ of party A and B, respectively, in advance via secure channels.
- Server S generate the identity of party A, B and server S as $ID_A, ID_B \ and \ ID_S$ respectively and make it publicly available.
- Finally, Server S publishes the system parameters $\{n, q, \chi_\beta, a, x_S, ID_A, ID_B, ID_S, h_0(), h_1(), h_2()\}$ publicly while maintaining the confidentiality of the secret key s_S and the hashes $h_0(pw_A)$, and $h_0(pw_B)$.

4.2. Description of the protocol

The SL3PAKE protocol relies on Ding et al.'s error reconciliation mechanism, which is described in the preliminaries section and includes the signal function (Cha()) and robust extractor ($Mod_2()$). The proposed protocol presented in Table 2 is described as follows:

1. First, client A instantiates the session by inputting its identity ID_A and password pw_A. Client A then computes the public key x_A as $x_A = a.s_A + 2.e_A$ where $s_A, e_A \leftarrow \chi_\beta$. After that the parameters x_A^* and h_{AS} are computed as $x_A^* = x_A + h_0(pw_A)$ and $h_{AS} = h_1(ID_A, ID_S, x_A, x_A^*)$. Finally, client A sends the parameters $\{ID_A, x_A^*, h_{AS}\}$ to client B.
2. After receiving the parameters, client B input its identity ID_B and password pw_B. It then computes the public key x_B as $x_B = a.s_B + 2.e_B$ where $s_B, e_B \leftarrow \chi_\beta$. After that the parameters x_B^* and h_{BS} are computed as $x_B^* = x_B + h_0(pw_B)$ and $h_{BS} = h_1(ID_B, ID_S, x_B, x_B^*)$. Finally, client B sends the parameters $\{ID_A, ID_B, x_A^*, x_B^*, h_{AS}, h_{BS}\}$ to server S.
3. Server S after receiving the parameters from client B, first authenticate both client A and client B. For authentication of client A, server S will compute $x'_A = x_A^* - h_0(pw_A)$ and $h_1(ID_A, ID_S, x'_A, x_A^*)$. Here, x'_A is same as x_A i.e. public key of client A. Then it will verify whether $h_{AS} \stackrel{?}{=} h_1(ID_A, ID_S, x'_A, x_A^*)$ is equal; if it is not, server S will terminate the session; otherwise, it will continue. Similarly, server S will authenticate client B by computing $x'_B = x_B^* - h_0(pw_B)$ and $h_1(ID_B, ID_S, x'_B, x_B^*)$. Here, x'_B is same as x_B i.e. public key of client B. Then it will check whether $h_{BS} \stackrel{?}{=} h_1(ID_B, ID_S, x'_B, x_B^*)$ exists. If the value is not equal, server S will abort the session; otherwise, it will continue.

Table 2
Simple Lattice-based Three-party Password Authenticated Key Exchange (SL3PAKE).

Client A	Client B	Server S
Input $ID_A,\ pw_A$		
$x_A = a.s_A + 2.e_A,\ where\ s_A,\ e_A \leftarrow \chi_\beta$		
$x_A^* = x_A + h_0(pw_A)$		
$h_{AS} = h_1(ID_A, ID_S, x_A, x_A^*)$		
$\xrightarrow{\{ID_A, x_A^*, h_{AS}\}}$		
	Input $ID_B,\ pw_B$	
	$x_B = a.s_B + 2.e_B,\ where\ s_B,\ e_B \leftarrow \chi_\beta$	
	$x_B^* = x_B + h_0(pw_B)$	
	$h_{BS} = h_1(ID_B, ID_S, x_B, x_B^*)$	
	$\xrightarrow{\{ID_A, ID_B, x_A^*, x_B^*, h_{AS}, h_{BS}\}}$	
		$x'_A = x_A^* - h_0(pw_A)$
		Check if $h_{AS} \stackrel{?}{=} h_1(ID_A, ID_S, x'_A, x_A^*)$
		If not equal then abort, otherwise continue
		$x'_B = x_B^* - h_0(pw_B)$
		Check if $h_{BS} \stackrel{?}{=} h_1(ID_B, ID_S, x'_B, x_B^*)$
		If not equal then abort, otherwise continue
		Here, Assert that $x'_A == x_A$ and $x'_B == x_B$
		$x_S = a.s_S + 2e_S,\ where\ s_S, e_S \leftarrow \chi_\beta$
		$c_A = x'_B.s_S + 2.f_{S4},\ where\ f_{S4} \leftarrow \chi_\beta$
		$c_B = x'_A.s_S + 2.f_{S5},\ where\ f_{S5} \leftarrow \chi_\beta$
		$m = h_2(ID_S, ID_A, x_s, x'_A)$
		$n = h_2(ID_S, ID_B, x_s, x'_B)$
		$k_{SA} = (x'_A.s_S + 2.m).m + 2.f_{S1}, f_{S1} \leftarrow \chi_\beta$
		$w_{SA} = Cha(k_{SA})$
		$\sigma_{SA} = Mod_2(k_{SA}, w_{SA})$
		$\alpha_{SA} = h_1(ID_A, ID_B, ID_S, c_A, x'_A, \sigma_{SA})$
		$k_{SB} = (x'_B.s_S + 2.n).n + 2.f_{S3}, f_{S3} \leftarrow \chi_\beta$
		$w_{SB} = Cha(k_{SB})$
		$\sigma_{SB} = Mod_2(k_{SB}, w_{SB})$
		$\alpha_{SB} = h_1(ID_A, ID_B, ID_S, c_B, x'_B, \sigma_{SB})$
		$\xleftarrow{\{c_A, c_B, x_S, w_{SA}, w_{SB}, \alpha_{SA}, \alpha_{SB}\}}$
	$n = h_2(ID_S, ID_B, x_S, x_B)$	
	$k_{BS} = (x_S.s_B + 2.n).n + 2.f_{B1}$	
	$\sigma_{BS} = Mod_2(k_{BS}, w_{SB})$	
	$\alpha_{BS} = h_1(ID_A, ID_B, ID_S, c_B, x_B, \sigma_{BS})$	
	Check if $\alpha_{BS} \stackrel{?}{=} \alpha_{SB}$	
	If not equal then abort, otherwise continue	
	$v_{BA} = c_B.s_B + 2.f_{B2},\ where\ f_{B2} \leftarrow \chi_\beta$	
	$w_{BA} = Cha(v_{BA})$	
	$\sigma_{BA} = Mod_2(v_{BA}, w_{BA})$	
	$sk_{BA} = h_1(ID_A, ID_B, ID_S, x_A^*, x_B^*, \sigma_{BA})$	
	$\xleftarrow{\{ID_B, x_B^*, c_A, x_S, w_{BA}, w_{SA}, \alpha_{SA}\}}$	
$m = h_2(ID_S, ID_A, x_S, x_A)$		
$k_{AS} = (x_S.s_A + 2.m).m + 2.f_{A1}$		
$\sigma_{AS} = Mod_2(k_{AS}, w_{SA})$		
$\alpha_{AS} = h_1(ID_A, ID_B, ID_S, c_A, x_A, \sigma_{AS})$		
Check if $\alpha_{AS} \stackrel{?}{=} \alpha_{SA}$		
If not equal then abort, otherwise continue		
$v_{AB} = c_A.s_A + 2.f_{A2},\ where\ f_{A2} \leftarrow \chi_\beta$		
$\sigma_{AB} = Mod_2(v_{AB}, w_{BA})$		
$sk_{AB} = h_1(ID_A, ID_B, ID_S, x_A^*, x_B^*, \sigma_{AB})$		

4. After client A and client B have successfully authenticated, server S computes its public key x_S as $x_S = a.s_S + 2e_S$ where $s_S, e_S \leftarrow \chi_\beta$. Now, server S computes the parameters c_A *and* c_B which will help client A and client B respectively to establish the common session key between them. The parameter c_A is calculated as $c_A = x'_B.s_S + 2.f_{S4}$ where $f_{S4} \leftarrow \chi_\beta$ and s_S is the secret key of the server S corresponding to public key x_S. Similarly, the parameter c_B is computed as $c_B = x'_A.s_S + 2.f_{S5}$ where $f_{S5} \leftarrow \chi_\beta$.
5. Now, server S will compute the parameters $k_{SA}, w_{SA}, \sigma_{SA}$ *and* α_{SA} to authenticate itself to client A. These parameters are computed as follows: $k_{SA} = (x'_A.s_S + 2.m).m + 2.f_{S1}$ where $m = h_2(ID_S, ID_A, x_s, x'_A)$ *and* $f_{S1} \leftarrow \chi_\beta$. Then, from k_{SA} the w_{SA}, σ_{SA} and α_{SA} are computed as $w_{SA} = Cha(k_{SA})$, $\sigma_{SA} = Mod_2(k_{SA}, w_{SA})$ and $\alpha_{SA} = h_1(ID_A, ID_B, ID_S, c_A, x'_A, \sigma_{SA})$. Similarly, the parameters $k_{SB}, w_{SB}, \sigma_{SB}$ *and* α_{SB} is computed to authenticate server S to client B. Finally, server S sends the parameters $\{c_A, c_B, x_S, w_{SA}, w_{SB}, \alpha_{SA}, \alpha_{SB}\}$ to client B.
6. Client B after receiving the parameters $\{c_A, c_B, x_S, w_{SA}, w_{SB}, \alpha_{SA}, \alpha_{SB}\}$ from server S, first authenticate the server S. The authentication is done by computing the parameters k_{BS}, σ_{BS} *and* α_{BS} as $k_{BS} = (x_S.s_B + 2.n).n + 2.f_{B1}$ where $n = h_2(ID_S, ID_B, x_s, x_B), f_{B1} \leftarrow \chi_\beta$, $\sigma_{BS} = Mod_2(k_{BS}, w_{SB})$ and $\alpha_{BS} = h_1(ID_A, ID_B, ID_S, c_B, x_B, \sigma_{BS})$. Then, the client B will check whether $\alpha_{BS} \stackrel{?}{=} \alpha_{SB}$, if it is not equal then client B will abort the session otherwise it will continue. After successful authentication of server S, client B will now generate the session key sk_{BA} using the parameters $\{v_{BA}, w_{BA}$ and $\sigma_{BA}\}$ as follows: The parameters $\{v_{BA}, w_{BA}$ *and* $\sigma_{BA}\}$ are computed as $v_{BA} = c_B.s_B + 2.f_{B2}$, *where* $f_{B2} \leftarrow \chi_\beta$, $w_{BA} = Cha(v_{BA})$ and $\sigma_{BA} = Mod_2(v_{BA}, w_{BA})$. Finally, session key sk_{BA} is computed as $sk_{BA} = h_1(ID_A,$

$ID_B, ID_S, x_A^*, x_B^*, \sigma_{BA}$) and client B sends the parameters $\{ID_B, x_B^*, c_A, x_S, w_{BA}, w_{SA}, \alpha_{SA}\}$ to client A.

7. Similar to client B, client A after receiving the parameters $\{ID_B, x_B^*, c_A, x_S, w_{BA}, w_{SA}, \alpha_{SA}\}$ from client B, authenticate the server S by computing the parameters $\{k_{AS}, \sigma_{AS}\ and\ \alpha_{AS}\}$. The parameters $\{k_{AS}, \sigma_{AS}\ and\ \alpha_{AS}\}$ is computed as $k_{AS} = (x_S.s_A + 2.m).m + 2.f_{A1}$ where $m = h_2(ID_S, ID_A, x_s, x_A), f_{A1} \leftarrow \chi_\beta$, $\sigma_{AS} = Mod_2(k_{AS}, w_{SA})$, and $\alpha_{AS} = h_1(ID_A, ID_B, ID_S, c_A, x_A, \sigma_{AS})$. The client A will now check whether $\alpha_{AS} \stackrel{?}{=} \alpha_{SA}$, if it is not equal then server S will abort the session otherwise it will continue. After authentication of server S, client B will now generate the session key sk_{AB} using the parameters $\{v_{AB}\ and\ \sigma_{AB}\}$ as follows: The parameters $\{v_{AB}\ and\ \sigma_{AB}\}$ is computed as $v_{AB} = c_A.s_A + 2.f_{A2}\ where\ f_{A2} \leftarrow \chi_\beta$ and $\sigma_{AB} = Mod_2(v_{AB}, w_{BA})$. Finally, session key sk_{AB} is generated as $sk_{AB} = h_1(ID_A, ID_B, ID_S, x_A^*, x_B^*, \sigma_{AB})$. Here, we assert that the session key $sk_{AB} == sk_{BA}$.

4.3. Condition for the correctness of the proposed SL3PAKE protocol

The SL3PAKE protocol (refer to Table 2) demonstrates that for the protocol to be correct, the following conditions must be met: $\sigma_{AB} = \sigma_{BA}, \sigma_{SA} = \sigma_{AS}\ and\ \sigma_{BS} = \sigma_{SB}$

Now, one can easily observe that computing σ_{AB} and σ_{BA} involves the same number of terms as computing σ_{AS}, σ_{SA}, σ_{BS} and σ_{SB}. Thus, the correctness condition for $\sigma_{AB} = \sigma_{BA}$ requires the same error tolerance as for $\sigma_{AS} = \sigma_{SA}$ and $\sigma_{BS} = \sigma_{SB}$. Therefore, we derive the condition of correctness for $\sigma_{AB} = \sigma_{BA}$ and this correctness condition is also true for $\sigma_{AS} = \sigma_{SA}$ and $\sigma_{BS} = \sigma_{SB}$.

Currently, we calculate the values of σ_{AB} and σ_{BA} based on v_{AB} and v_{BA}, respectively. The value of v_{AB} and v_{BA} are computed as:

$$\begin{aligned} v_{AB} &= c_A.s_A + 2.f_{A2},\ where\ f_{A2} \leftarrow \chi_\beta \\ &= a.s_A.s_B.s_S + 2.e_B.s_S.s_A + 2.s_A.f_{S4} + 2.f_{A2} \end{aligned} \tag{2}$$

$$\begin{aligned} v_{BA} &= c_B.s_B + 2.f_{B2},\ where\ f_{B2} \leftarrow \chi_\beta \\ &= a.s_A.s_B.s_S + 2.e_A.s_S.s_B + 2.s_B.f_{S5} + 2.f_{B2} \end{aligned} \tag{3}$$

Subtracting Eqs. (2) and (3), we get

$$\begin{aligned} v_{BA} - v_{AB} = {} & 2.e_A.s_S.s_B + 2.s_B.f_{S5} + 2.f_{B2} \\ & - 2.e_B.s_S.s_A - 2.s_A.f_{S4} - 2.f_{A2} \end{aligned} \tag{4}$$

$$\begin{aligned} \|v_{BA} - v_{AB}\| \le {} & 2.\{\|e_A.s_S.s_B\| + \|s_B.f_{S5}\| + \|f_{B2}\| \\ & + \|e_B.s_S.s_A\| + \|s_A.f_{S4}\| + \|f_{A2}\|\} \end{aligned} \tag{5}$$

Applying Lemma 1 and Lemma 2, we get

$$\begin{aligned} \|v_{BA} - v_{AB}\| &< 2.(2.\beta^3.\sqrt[5]{n} + 2.\beta^2.\sqrt[3]{n} + 2.\beta.\sqrt{n}) \\ &< 4.(\beta^3.\sqrt[5]{n} + \beta^2.\sqrt[3]{n} + \beta.\sqrt{n}) \\ &< 4.\beta.\sqrt{n}(\beta^2.n^2 + \beta.n + 1) \end{aligned} \tag{6}$$

Now, applying Lemma 3 to above equation, we get

$$\begin{aligned} \frac{q}{8} &> 4.\beta.\sqrt{n}(\beta^2.n^2 + \beta.n + 1) \\ q &> 32.\beta.\sqrt{n}(\beta^2.n^2 + \beta.n + 1) \end{aligned} \tag{7}$$

As a result, the above-mentioned criterion for the proposed SL3PAKE protocol's correctness has been obtained.

4.4. Informal security of the proposed SL3PAKE protocol against signal leakage attack (SLA)

In this section,we will go through how the proposed SL3PAKE protocol is informally secured against Signal Leakage Attacks (SLA).

Let us suppose that either client A or client B is acting as an adversary in the first scenario, and that both clients A and B are adversaries in the second situation, where both are capable of obtaining the server's secret key.

Case 1. In this case, it has been assumed that client A is an adversary (and client B is an honest party) who wants to obtain the server's secret key. To get the secret key of the server, the adversary generates its malformed public key x_A and creates $x_A^* = x_A + h_0(pw_A)$ from it, which it sends to the server. Suppose that the server has authenticated the adversary, and now the server will create its parameters using the public key of the adversary. In the proposed protocol, the value of k_{SA} and m are $(x'_A.s_S + 2.m)m + 2f_{s1}$ and $h_2(ID_A, ID_S, x_A, x_S)$ respectively. Here, x_A and x'_A both represent the same public key of the client A. The value of m calculates the value of k_{SA}, which is then used to calculate the signal function w_{SA}. It is evident that the computation of w_{SA} depends on the adversary public key x'_A. In SLA attack, the idea of an adversary is to vary its public key to observe the changes in signal function and thus recover the secret key (s_S in this case) of the honest party (see [16] for a thorough explanation of the SLA attack). But this idea will not work in the context of the SL3PAKE protocol.

In the case of the proposed SL3PAKE protocol, whenever the adversary changes the value of its public key, then the value of m also changes on each value of the adversary's public key. Due to this, the value of k_{SA} changes, which results in a change in the value of signal function w_{SA} in-deterministically. As a result, the attacker will never be able to retrieve the server's secret key s_S by just viewing the signal function w_{SA}.

Case 2. In this case, we assume that both client A and client B are adversaries who want to get the server's secret key. Here, client A and client B are adversary $\mathcal{A}$ and adversary $\mathcal{B}$ respectively. To get the secret key of the server, both adversary $\mathcal{A}$ and adversary $\mathcal{B}$ generate their malformed public keys x_A and x_B respectively and create x_A^* and x_B^* from them, which they send to the server. Suppose that the server has authenticated both adversary $\mathcal{A}$ and adversary $\mathcal{B}$ and now the server will generate its parameters using public keys of both adversary $\mathcal{A}$ and adversary $\mathcal{B}$. In this protocol the values of k_{SA}, k_{SB}, m and n are $(x'_A.s_S + 2.m)m + 2f_{s1}$, $(x'_B.s_S + 2.n)n + 2f_{s2}$, $h_2(ID_A, ID_S, x_A, x_S)$ and $h_2(ID_B, ID_S, x_B, x_S)$ respectively. Here, x_A and x'_A both represent the same public key of the client A, similarly x_B and x'_B both represent the same public key of the client B. Using the same justification as given in Case 1, we claim that both adversary $\mathcal{A}$ and adversary $\mathcal{B}$ will never be able to recover the server's secret key.

5. Formal security of the proposed SL3PAKE protocol

The proposed SL3PAKE protocol's formal security has been shown in this section using the ROR model [15]. First, we give the brief description of our security model that is designed by following ROR model [15] and finally, we prove the security of the proposed SL3PAKE protocol in Theorem 1.

5.1. Brief description of security model

In the this sub section, we will describe the main components of our security model which is followed from the Abdalla et al.'s ROR model [15]. All the security definitions are directly followed from the ROR model.

(a) **Protocol Participants:** The participants include users and server, The users are represented as Clients C^i and server as S^j. The C^i, S^j denotes the i and j instance of client and server respectively and these instances of client and server are also called oracles.

(b) **Partnering:** The condition for partnering of the two instances C^i and S^j are:

- The instances C^i and S^j must accept. Here acceptance means both oracles holds the session key (sk), a session id (SID) and a partner id (PID).

- The session identification (sid) of both the instances C^i and S^j must be same.
- The partner identification (PID) for C^i is S^j and for S^j is C^i.
- There must be no instance other than C^i and S^j that would accept with PID equal to C^i or S^j.

(c) **Freshness:** The C^i and S^j instance of client and server is fresh when the shared session key between two instances is not known to the adversary by trivial means. In the BPR model [1] this trivial means is asking the Test query after reveal query by the adversary $\mathcal{A}$. But in the ROR model, reveal query is replaced by multiple number of Test query to be asked by adversary $\mathcal{A}$. Hence, instead of mandating the adversary $\mathcal{A}$ to ask Test query on fresh instances, the ROR model embed the notion of freshness in the definition of oracles.

(d) **Adversary:** The adversary $\mathcal{A}$ is assume to have complete control over the communication channel. Thus, adversary $\mathcal{A}$ can read, modify, fabricate and inject new messages onto the communication channel. To simulate the above capabilities of the adversary $\mathcal{A}$, the ROR model consist of different oracle's queries. Using the oracles queries, the interaction between the adversary $\mathcal{A}$ and the participants are simulated and output of oracle query is returned back to the adversary $\mathcal{A}$. In this manner, real-time attack by polynomial-time adversary $\mathcal{A}$ is simulated using ROR model. These oracle queries are as follows:

(i) **Execute**(C^i, S^j): The execute oracle query models the eavesdropping attack between the oracle C^i and S^j. The query outputs the transcript of the honest execution of the protocol between C^i and S^j.

(ii) **Send**(U^i, m): This oracle query models the active attack by an adversary $\mathcal{A}$, against the user instance U^i. The query outputs the message that the user instance U^i would generate upon the receipt of the message m.

(iii) **CorruptS**(S^j): This oracle query models the attack in which the long term master secret key s and secure database of the server has been compromised. The query simulates the attack against the server instance S^j and output the long term secret key and sensitive information stored in the server's database. This query illustrates the perfect forward secrecy of the proposed protocol. Here, CorruptS() oracle query follows the strong-corruption model as not only long term secret key but also secure database has been compromised.

(iv) **Test**(U^i): This query models the semantic security of the session key as described in the ROR model [15]. This oracle query work as follows: First a coin b is flipped at the start of the experiment and its value is kept hidden from the adversary $\mathcal{A}$ and will decide the output of Test() oracle query whenever called. Now, when test oracle query is asked by an adversary $\mathcal{A}$ for user instance U^i, then it checked whether session key for user instance U^i is defined or not. If no session key is defined then, the oracle query will return the output $\perp$, Otherwise, based on the flipped bit b i.e. if $b = 1$ or $b = 0$, the query will return session key or random key of same size respectively.

Semantic Security of the session key in our security model: As discussed previously, in the ROR model the adversary can ask Execute, Send and Test oracle queries but not the Reveal query. Indeed, in ROR model the adversary can ask as many Test queries as he can. In this case, all the Test queries will answered in accordance with the same value of flipped coin bit b which is derived during the start of the experiment. Therefore, during the whole experiment whenever the Test query has been called to any instance of client and server, the output of the Test query will be either all real or all random session key. Also, note that for random case, the same random value should be returned if Test oracle query is asked for two partnered instances. The goal of the adversary is to guess the hidden bit b at the end of the experiment. If adversary $\mathcal{A}$ correctly guesses the bit b of the flipped coin, then he will win the experiment.

An adversary's success, denoted by the symbol SUCC, is when they accurately predicts the outcome of a coin flip.

Definition 1. The advantage of an adversary successfully predicting the bit b of the flipped coin is as follows:

$$Adv_P^{3PAKE}(\mathcal{A}) = |2.Pr[SUCC] - 1|$$

This is the same as violating the proposed 3PAKE protocol P's semantic security. The protocol P is a secure key exchange protocol, if $Adv_P^{3PAKE}(\mathcal{A})$ is small.

Definition 2. As proved by Lyubashevsky et al. [12], an adversary $\mathcal{A}$ cannot solve the RLWE problem in a polynomial-time. Thus, the advantage $(Adv_{R_q}^{RLWE})$ of solving the RLWE problem by the polynomial-time adversary is small. Mathematically,it can be represented as

$$Adv_{R_q}^{RLWE} \leq \epsilon,$$

where ϵ is a low number.

Theorem 1. *For a large size password and identity dictionary (say N bits such that value of N is large enough), as well as a wide range of hash algorithms $(h_0(), h_1(), h_2())$, large size of discrete Gaussian distribution space χ_β and small advantage of solving RLWE problem i.e. $Adv_{R_q}^{RLWE} \leq \epsilon$, the proposed SL3PAKE protocol is a secure key exchange protocol.*

Now, Let $Adv_P^{3PAKE}(\mathcal{A})$ represents the advantage of breaking the semantic security of the proposed SL3PAKE protocol P by an adversary $\mathcal{A}$, then

$$\begin{aligned} Adv_P^{3PAKE}(\mathcal{A}) \leq{}& \frac{q_{h0}^2}{q^n} + \frac{q_{h1}^2}{|RS_{h1}|} + \frac{(q_s + q_{exe})^2}{|RS_{\chi_\beta}|} \\ &+ \frac{2.q_s}{|D_{IDA}| * |D_{pwA}| * |D_{IDB}| * |D_{pwB}|} \\ &+ 2.Adv_{R_q}^{RLWE} \end{aligned} \tag{8}$$

where $q^n, |RS_{h1}|, |RS_{\chi_\beta}|, |D_{pwA}|/|D_{IDA}|, |D_{pwB}|/|D_{IDB}|$ are the $h_0()$ and $h_1()$ range spaces, Gaussian distribution range space χ_β, size of password dictionary/size of identity dictionary of client A and client B respectively. Also, $Adv_{R_q}^{RLWE}$ represents the advantage of solving RLWE problem by the polynomial-time adversary. If the above equation holds, then the advantage of an adversary $Adv_P^{3PAKE}(\mathcal{A})$ in breaking the semantic security of SL3PAKE is negligible, thus the proposed SL3PAKE is semantically secure.

Proof. A sequence of games G_i is used to prove the above theorem, where $i = 0, 1, 2, 3, 4$. Let $SUCC_i$ denotes the event of correctly guessing the bit b in the game G_i by an adversary $\mathcal{A}$.

Game G_0: The game G_0 is the actual attack against the protocol P by an adversary $\mathcal{A}$. In-game G_0, adversary $\mathcal{A}$ tries to guess a hidden bit b. By Definition 1:

$$Adv_P^{3PAKE}(\mathcal{A}) = |2.Pr[SUCC_0] - 1| \tag{9}$$

Game G_1: The game G_1 simulate the eavesdropping attack using the execute oracle query (Execute(U_i, S_i)). Finally, adversary $\mathcal{A}$ uses the test query to determine the concealed bit b. The test query returns either a genuine or a random session key as a result. In the proposed protocol, the session key is calculated as $sk_{AB} = h_1(ID_A, ID_B, ID_S, x_A^*, x_B^*, \sigma_{AB}) = h_1(ID_A, ID_B, ID_S, x_A^*, x_B^*, \sigma_{BA}) = sk_{BA}$. For the computation of session key, adversary need to find parameters σ_{AB}, or σ_{BA} as other parameters are obtained through execute oracle query. Now, σ_{AB}, and σ_{BA} are calculated from v_{AB}, v_{BA} which in turn computed from s_A, s_B,

and s_S. Thus, the adversary $\mathcal{A}$ requires the parameters s_A, s_B, and s_S to calculate σ_{AB}, or σ_{BA}. As a result, the eavesdropping attack has no effect on the chance of an adversary $\mathcal{A}$ winning the game. So,

$$Pr[SUCC_0] = Pr[SUCC_1] \tag{10}$$

Game G_2: Using the send, execute, and hash oracle queries, the game G_2 simulates an active attack. By replicating real-time hash functions, this game aims to spot hash value collisions. In the proposed SL3PAKE protocol, the three hash functions $h_0(), h_1()$, and $h_2()$ have been utilized. Let q_{h0} and q_{h1} represent the number of hash queries for the hash functions $h_0()$ *and* $h_1()$, respectively. q^n and $|RS_{h1}|$ are the corresponding range spaces of $h_0()$ *and* $h_1()$, respectively. Now, as a result of the Birthday paradox, the risk of hash values colliding is $\leq \frac{q_{h0}^2}{2.q^n} + \frac{q_{h1}^2}{2.RS_{h1}}$. Using send and execute oracle queries, the probability of collision of the parameters x_A^*, x_B^*, x_S is $\leq \frac{(q_s+q_{exe})^2}{2.|RS_{\chi_\beta}|}$ (using Birthday paradox). Here, q_{exe} and q_s are the number of execute and send queries respectively, while $|RS_{\chi_\beta}|$ is the discrete Gaussian distribution's χ_β range space. Thus, overall probability can be written as

$$\begin{aligned}|Pr[SUCC_1] - Pr[SUCC_2]| \leq \frac{q_{h0}^2}{2.q^n} + \frac{q_{h1}^2}{2.|RS_{h1}|} \\ + \frac{(q_s+q_{exe})^2}{2.|RS_{\chi_\beta}|}\end{aligned} \tag{11}$$

Game G_3: The game G_3 simulates online dictionary attack. In this game, the adversary will try the online guesses of the identity and password combination at server's end using the send oracle query. Each send query eliminates one possible id/password combination. Mathematically, it can be represented as

$$\begin{aligned}|Pr[SUCC_2] - Pr[SUCC_3]| \leq \frac{q_s}{|D_{IDA}| * |D_{pwA}|} \\ * \frac{1}{|D_{IDB}| * |D_{pwB}|}\end{aligned} \tag{12}$$

q_s is the number of send queries, $|D_{IDA}|, |D_{IDB}|$ is the size of client A's and B's identity dictionary, and $|D_{pwA}|, |D_{pwB}|$ is the size of client A's and B's password dictionary, respectively. The number of incorrect identification and password attempts should be recorded in real time by server S.

Game G_4: The corrupt server oracle query is simulated in the game G_4. In this game, the adversary obtains the server S's long-term secret key as well as the parameters that are recorded in the server's database for clients A and B. Currently, an adversary's goal is to use the values of the intercepted parameters to determine the prior session key. As a result, the attacker will have access to the parameters $\{ID_A, ID_B, x_A^*, x_B^*, h_{AS}, h_{BS}\}$ and $\{c_A, c_B, x_S, w_{SA}, w_{SB}, \alpha_{SA}, \alpha_{SB}\}$. The adversary $\mathcal{A}$ uses x_A^* and x_B^* to compute the value of x_A and x_B respectively, as $x_A = x_A^* - h_0(pw_A)$ and $x_B = x_B^* - h_0(pw_B)$. The session key sk_{AB} and sk_{BA} have the parameters σ_{AB} and σ_{BA} respectively. Now, σ_{AB} and σ_{BA} are computed from the parameters v_{AB} and v_{BA} respectively, which in turn are computed from s_A and s_B respectively. Now, fetching s_A and s_B from x_A and x_B respectively are same as solving RLWE problem. Therefore, we have the following result

$$|Pr[SUCC_3] - Pr[SUCC_4]| \leq Adv_{R_q}^{RLWE} \tag{13}$$

All the games are played now. If the adversary $\mathcal{A}$ does not succeed in breaking the semantic security of the proposed protocol, then the last way out is to call Test oracle query $Test(U^i)$. The adversary $\mathcal{A}$ will now try to determine what the output bit b of the Test oracle query will be. Thus, the probability of winning the game G_4 is given by:

$$Pr[SUCC_4] = \frac{1}{2} \tag{14}$$

From (9), we have

$$\begin{aligned}Adv_P^{3PAKE}(\mathcal{A}) &= 2.|Pr[SUCC_0] - \frac{1}{2}| \\ &= 2.|Pr[SUCC_0] - Pr[SUCC_4]|\end{aligned} \tag{15}$$

By using triangular inequality and using Eq. (10), we get

$$\begin{aligned}Adv_P^{3PAKE}(\mathcal{A}) \leq 2.(&|Pr[SUCC_1] - Pr[SUCC_2]| \\ &+ |Pr[SUCC_2] - Pr[SUCC_3]| \\ &+ |Pr[SUCC_3] - Pr[SUCC_4]|)\end{aligned} \tag{16}$$

putting the values from (11)–(14), we get:

$$\begin{aligned}Adv_P^{3PAKE}(\mathcal{A}) \leq 2.(&\frac{q_{h0}^2}{2.q^n} + \frac{q_{h1}^2}{2.|RS_{h1}|} + \frac{(q_s+q_{exe})^2}{|2.RS_{\chi_\beta}|} \\ &+ \frac{q_s}{|D_{IDA}| * |D_{pwA}| * |D_{IDB}| * |D_{pwB}|} \\ &+ Adv_{R_q}^{RLWE})\end{aligned} \tag{17}$$

$$\begin{aligned}Adv_P^{3PAKE}(\mathcal{A}) \leq &\frac{q_{h0}^2}{q^n} + \frac{q_{h1}^2}{|RS_{h1}|} + \frac{(q_s+q_{exe})^2}{|RS_{\chi_\beta}|} \\ &+ \frac{2.q_s}{|D_{IDA}| * |D_{pwA}| * |D_{IDB}| * |D_{pwB}|} \\ &+ 2.Adv_{R_q}^{RLWE}\end{aligned} \tag{18}$$

hence proved.

6. Concrete parameters & implementation of SL3PAKE

In this section, we present our choices of parameters and the performance analysis of the SL3PAKE protocol. The performance examination utilized security parameter values of n at 128, 256, and 512. Also, the comparative analysis based on communication cost among the proposed SL3PAKE protocol and the other three party protocols has been done.

6.1. Parameters of choice, implementations and communication cost of the proposed SL3PAKE protocol

As previously stated, we selected 128, 256, and 512 for n such that $n = 2k$, where k is any positive integer. The discrete Gaussian distribution χ_β has a standard deviation of $\beta = \frac{8}{2\sqrt{2\pi}} \approx 1.596$. As mentioned in section 4.3, the criterion for the proposed protocol's correctness is $q > 32.\beta.\sqrt{n}(\beta^2.n^2 + \beta.n + 1)$, as stated in section 4.3. Thus, for $\beta = 1.596$ and $n = 512$, the value of q is chosen as $q = 1931502101$. This value of q results in less than 1% of error rate (error rate is defined as number of session between party A and party B when session key do not match divided by total number sessions between party A and party B). Here, total number of sessions used to calculate error rate is 1000 i.e. the protocol is instantiated 1000 times to calculate the error rate.

Now, using the above-selected parameters, the LWE estimator has been used to compute the standard security level of the proposed SL3PAKE protocol. The LWE estimator provides information on the degree of security offered by certain sets of parameters in the LWE/RLWE based cryptosystem. This is accomplished by calculating the attack complexities of the decoding, Blum–Kalai–Wasserman (BKW), lattice reduction, meet-in-the-middle, reducing Bounded Distance Decoding (BDD) to unique-SVP, and exhaustive search attacks. Both the time and space complexity of the aforementioned attacks are estimated using the LWE estimator [39]. Fig. 4 shows the screenshot of sage commands used to determine the classical security level of the SL3PAKE protocol using the LWE estimator. From the snapshot, it can be seen that the standard deviation β of discrete Gaussian χ_β is represented by $alpha$ variable of the Sage code. For more information refer to LWE estimator article [39].

The output of sage commands presented in Fig. 4 shows that the proposed SL3PAKE provides 55-bit of classical security with the above parameter of choice.

The representation of various cryptographic operations are:

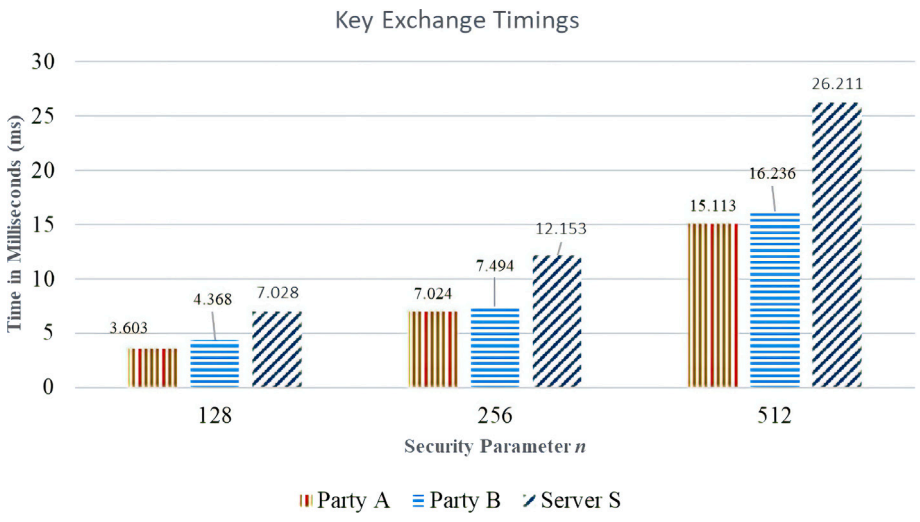

Fig. 2. Key exchange timings of the proposed SL3PAKE protocol.

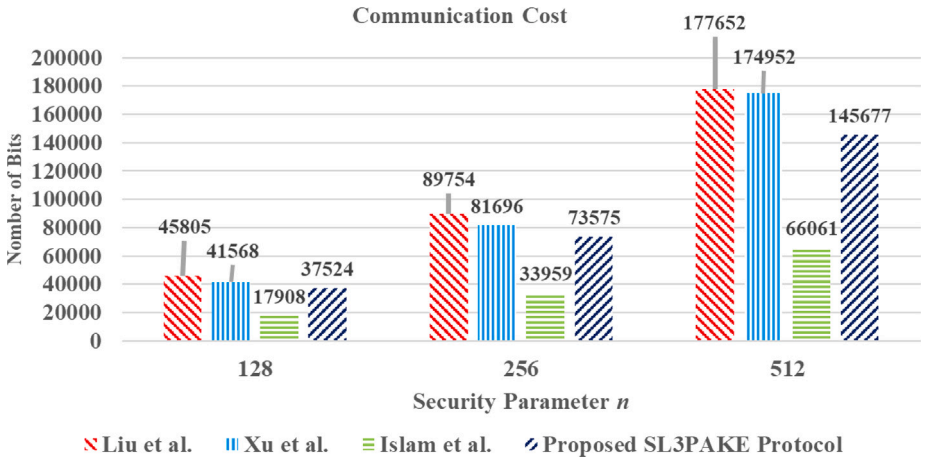

Fig. 3. Comparative analysis based on communication cost.

```
In [19]: load("estimator.py")
In [20]: n, alpha, q=512, alphaf(4, 1931502101), 1931502101
In [21]: set_verbose(1)
In [22]: _=estimate_lwe(n, alpha, q,skip=["arora-gb"])
```

Fig. 4. Sage commands to calculate classical security level using LWE estimator.

Table 3
Cryptographic operations running time (in milliseconds) for $\beta = 1.5965$ and $q = 1931502101$.

Operation	$n = 512$	$n = 256$	$n = 128$
T_{samp}	.076	.048	.025
T_{rmul}	3.867	1.546	.613
T_{h0}	.137	.059	.058
T_{h1}	1.368	.881	.339
T_{h2}	2.210	.963	.404
T_{Cha}	1.296	.823	.345
T_{Mod}	.810	.508	.211

- T_{samp} shows the average time taken to sample an element x from the discrete Gaussian distribution χ_β.
- T_{rmul} is the average time spent multiplying two polynomial elements c, d, where $c \in R_q$ and d is a sample from χ_β.
- T_{h0} depicts the average time taken to compute the $h_0()$ function.
- T_{h1}: stands the average time spent to compute the $h_1()$ function.
- T_{h2}: represents for the average time taken to compute $h_2()$ function.
- T_{Cha} depicts the average time spent to compute the characteristic function $Cha()$.
- T_{Mod} shows the average time taken to compute the $Mod_2()$ function.

The following are the setups for the proposed SL3PAKE's performance analysis:

Intel(R)Core(TM)i5-7200U processors with 8 GB RAM power both the client and server. The hash function $h_1()$ output has been produced using the SHA3-224() hash function. The computational time of the cryptographic operations are shown in Table 3.

The Fig. 2 represents the average key exchange timings for Party A, B, and Server S over 1000 instantiations of the proposed SL3PAKE protocol. The analysis shows that the proposed SL3PAKE protocol requires more computational resources from the server than from the users. This behavior is quite favorable as, in most of the cases, the server has high-end resources while the user has low-end resources.

Thus, the server can efficiently perform computationally intensive tasks than the user.

Now, finally we will compute the communication cost of SL3PAKE protocol. The Table 2 shows that the SL3PAKE protocol involves four rounds of communication among party A, party B, and server S. Thus, to calculate the communication cost, we will calculate the bit size of each parameter exchanged during the four round of communication. The parameters exchanged during four rounds of communications are $\{ID_A, x_A^*, h_{AS}\}$, $\{ID_A, ID_B, x_A^*, x_B^*, h_{AS}, h_{BS}\}$, $\{c_A, c_B, x_S, w_{SA}, w_{SB}, \alpha_{SA}, \alpha_{SB}\}$ and $\{ID_B, x_B^*, c_A, x_S, w_{BA}, w_{SA}, \alpha_{SA}\}$. The bit size of (ID_A, ID_B) is 32 bits each, $(x_A^*, x_B^*, c_A, c_B, x_S)$ is $n * \log_2(q)$ bits each, (w_{SA}, w_{SB}, w_{BA}) is n bits each and $(h_{AS}, h_{BS}, \alpha_{SA}, \alpha_{SB})$ is 224 bits each. Thus, combining together the overall communication complexity of the proposed protocol is $9n\log_2(q)+4n+6\times224+4\times32 = 9n\log_2(q)+4n+1472$ bits.

Similarly, the communication complexity of Xu et al. [11], Liu et al. [17] and Islam et al. [20] are $10n\log_2(q)+5n+1440$, $11n\log_2(q)+4n+1856$ and $4n\log_2(q)+2n+1856$ receptively. Now, with the above communication complexities of the proposed protocol and other two protocols, the comparative analysis based on communication cost has been presented in Fig. 3 for $n = \{128, 256, 512\}$ and $q = 1931502101$. Fig. 3 shows that the proposed SL3PAKE protocol has a lower communication cost than Xu et al.'s [11] and Liu et al.'s [17] protocols but is higher than Islam et al.'s [20] protocol. The proposed SL3PAKE protocol has a provision to prevent signal leakage attacks (SLA), which accounts for the higher communication cost than Islam et al.'s protocol. As previously mentioned, the protocol developed by Islam et al. is susceptible to signal leakage attacks (SLA), therefore, additional measures must be implemented to defend against SLA attacks. This extra measures has been discussed in Section 4.4 (Informal security of the proposed SL3PAKE protocol).

Thus, the proposed SL3PAKE protocol is efficient three-party protocol and at the same time also resists signal leakage attack (SLA). Besides, the proposed SL3PAKE protocol only needs four communication rounds while Xu et al.'s [11] protocol needs six communication rounds, Liu et al.'s [17] needs seven communication rounds and Islam et al. [20] needs eight communication rounds.

7. Conclusion & future scope

In this work, we have designed a 3PAKE protocol termed as SL3PAKE using the RLWE problem of lattices. The ROR model proves the provable security of the protocol. Additionally, the proposed protocol, with its simple design, can resist signal leakage attacks when reusing its public/private keys. The concrete choice of parameters and implementation result show that the protocol is efficient and favorable for client/server scenarios. Finally, we calculate the communication cost of the protocol. Based on the computation and communication calculation, we can observe that the proposed SL3PAKE protocol is efficient for use in real-world applications. In future work, we can verify the provable security of the protocol using the Quantum Random Oracle Model (QROM) model.

CRediT authorship contribution statement

Vivek Dabra: Writing – original draft, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Saru Kumari:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision. **Anju Bala:** Validation, Supervision. **Sonam Yadav:** Writing – original draft, Methodology, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Bellare M, Pointcheval D, Rogaway P. Authenticated key exchange secure against dictionary attacks. In: International conference on the theory and applications of cryptographic techniques. Springer; 2000, p. 139–55.

[2] Boyko V, MacKenzie P, Patel S. Provably secure password-authenticated key exchange using Diffie-Hellman. In: International conference on the theory and applications of cryptographic techniques. Springer; 2000, p. 156–71.

[3] MacKenzie P, Patel S, Swaminathan R. Password-authenticated key exchange based on RSA. In: International conference on the theory and application of cryptology and information security. Springer; 2000, p. 599–613.

[4] Steiner M, Tsudik G, Waidner M. Refinement and extension of encrypted key exchange. Oper Syst Rev 1995;29(3):22–30.

[5] Sun H-M, Chen B-C, Hwang T. Secure key agreement protocols for three-party against guessing attacks. J Syst Softw 2005;75(1–2):63–8.

[6] Lu R, Cao Z. Simple three-party key exchange protocol. Comput Secur 2007;26(1):94–7.

[7] Chung H-R, Ku W-C. Three weaknesses in a simple three-party key exchange protocol. Inform Sci 2008;178(1):220–9.

[8] Guo H, Li Z, Mu Y, Zhang X. Cryptanalysis of simple three-party key exchange protocol. Comput Secur 2008;27(1–2):16–21.

[9] Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev 1999;41(2):303–32.

[10] 20 Years of quantum computing growth. 2021, https://www.statista.com/chart/17896/quantum-computing-developments/. [Last accessed 30 June 2021].

[11] Xu D, He D, Choo K-KR, Chen J. Provably secure three-party password authenticated key exchange protocol based on ring learning with error. IACR Cryptol ePrint Arch 2017;2017:360.

[12] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings. In: Annual international conference on the theory and applications of cryptographic techniques. Springer; 2010, p. 1–23.

[13] Choi R, An H, Kim K. Atlast: Another three-party lattice-based pake scheme. In: 2018 symposium on cryptography and information security (SCIS 2018), IEICE technical committee on information security. 2018.

[14] Ding J, Alsayigh S, Lancrenon J, Saraswathy R, Snook M. Provably secure password authenticated key exchange based on RLWE for the post-quantum world. In: Cryptographers' track at the RSA conference. Springer; 2017, p. 183–204.

[15] Abdalla M, Fouque P-A, Pointcheval D. Password-based authenticated key exchange in the three-party setting. In: International workshop on public key cryptography. Springer; 2005, p. 65–84.

[16] Ding J, Alsayigh S, Saraswathy R, Fluhrer S, Lin X. Leakage of signal function with reused keys in RLWE key exchange. In: Communications (ICC), 2017 IEEE international conference on. IEEE; 2017, p. 1–6.

[17] Liu C, Zheng Z, Jia K, You Q. Provably secure three-party password-based authenticated key exchange from RLWE. In: International conference on information security practice and experience. Springer; 2019, p. 56–72.

[18] Bellare M, Rogaway P. Provably secure session key distribution: the three party case. In: Proceedings of the twenty-seventh annual ACM symposium on theory of computing. 1995, p. 57–66.

[19] Ding J, Fluhrer S, Rv S. Complete attack on RLWE key exchange with reused keys, without signal leakage. In: Australasian conference on information security and privacy. Springer; 2018, p. 467–86.

[20] Islam SH, Basu S. PB-3PAKA: Password-based three-party authenticated key agreement protocol for mobile devices in post-quantum environments. J Inf Secur Appl 2021;63:103026.

[21] Krawczyk H. SIGMA: The 'SIGn-and-MAc'approach to authenticated diffie-hellman and its use in the IKE protocols. In: Annual international cryptology conference. Springer; 2003, p. 400–25.

[22] Regev O. On lattices, learning with errors, random linear codes, and cryptography. J ACM 2009;56(6):34.

[23] Bos J, Costello C, Ducas L, Mironov I, Naehrig M, Nikolaenko V, et al. 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. ACM; 2016, p. 1006–18.

[24] Bos JW, Costello C, Naehrig M, Stebila D. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: Security and privacy (SP), 2015 IEEE symposium on. IEEE; 2015, p. 553–70.

[25] Alkim E, Ducas L, Pöppelmann T, Schwabe P. Post-quantum key exchange-A new hope. In: USeNIX security symposium, vol. 2016, 2016.

[26] Alkim E, Ducas L, Pöppelmann T, Schwabe P. NewHope without reconciliation. IACR Cryptol ePrint Archive 2016;2016:1157.
[27] Alkim E, Avanzi R, Bos J, Ducas L, de la Piedra A, Pöppelmann PST, et al. NewHope. Technical report, National Institute of Standards and Technology, 2017 ....
[28] Ding J, Xie X, Lin X. A simple provably secure key exchange scheme based on the learning with errors problem. IACR Cryptol ePrint Arch 2012;2012:688.
[29] Zhang J, Zhang Z, Ding J, Snook M, Dagdelen Ö. Authenticated key exchange from ideal lattices. In: Annual international conference on the theory and applications of cryptographic techniques. Springer; 2015, p. 719–51.
[30] Peikert C. Lattice cryptography for the internet. In: International workshop on post-quantum cryptography. Springer; 2014, p. 197–219.
[31] Gao X, Ding J, Saraswathy R, Li L, Liu J. Comparison analysis and efficient implementation of reconciliation-based RLWE key exchange protocol. Technical report, Cryptology ePrint Archive, Report 2017/1178, 2017; 2017, http://eprint.iacr.org. ....
[32] Experimenting with post-quantum cryptography. 2021, https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html. [Last accessed 30 June 2021].
[33] Feng Q, He D, Zeadally S, Kumar N, Liang K. Ideal lattice-based anonymous authentication protocol for mobile devices. IEEE Syst J 2018.
[34] Dabra V, Bala A, Kumari S. LBA-PAKE: Lattice-based anonymous password authenticated key exchange for mobile devices. IEEE Syst J 2020.
[35] Seyhan K, Akleylek S. A new password-authenticated module learning with rounding-based key exchange protocol: Saber. PAKE. J Supercomput 2023;79(16):17859–96.
[36] Basu S, Seyhan K, Islam SH, Akleylek S. MLWR-2PAKA: A hybrid module learning with rounding-based authenticated key agreement protocol for two-party communication. IEEE Syst J 2023.
[37] Micciancio D, Regev O. Worst-case to average-case reductions based on Gaussian measures. SIAM J Comput 2007;37(1):267–302.
[38] Gentry C, Peikert C, Vaikuntanathan V. Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on theory of computing. 2008, p. 197–206.
[39] Albrecht MR, Player R, Scott S. On the concrete hardness of learning with errors. J Math Cryptol 2015;9(3):169–203.

Vivek Dabra received the Master degree in information security from Thapar Institute of Engineering & Technology in 2016. He is currently pursuing Ph.D. in Computer Science & Engineering Department (CSED) of Thapar Institute of Engineering & Technology. His research interests are cryptography, post quantum cryptography (specifically lattice-based cryptography) and cloud security.

Saru Kumari is currently an Associate Professor with the Department of Mathematics, Chaudhary Charan Singh University, Meerut, Uttar Pradesh, India. She received her Ph.D. degree in Mathematics in 2012 from the same University. She has published more than 290 research papers in reputed international journals and conferences, including more than 260 research papers in various SCIE Indexed Journals. She received the Best Paper award from the Journal of Network and Computer Applications, Elsevier in 2020, IEEE Consumer Electronics Magazine in 2022, and Vehicular Communication in 2022. She is on the editorial board of more than a dozen International Journals of high repute, under IEEE, Elsevier, Springer, Wiley, and others such as IEEE Transactions on Intelligent Transport Systems, IEEE Systems Journal, Computer Standards & Interfaces, Elsevier; AEÜ - International Journal of Electronics and Communications, Elsevier; International Journal of Communication Systems, Wiley; Concurrency and Computation: Practice and Experience, Wiley; Telecommunication Systems, Springer; Human-centric Computing and Information sciences, Springer; Transactions on Emerging Telecommunications Technologies, Wiley, etc. She served as lead/Guest Editor of many Special Issues in SCIE Journals of Elsevier, Springer, and Wiley. She has been on the Advisory Committee and Technical Program Committee for many International conferences. Her current research interests include Applied Cryptography, Information Security, Internet of Things, Information Fusion, Blockchain Technology, Security, and Artificial Intelligence.

Anju Bala is working as an assistant professor in the Department of Computer Science and Engineering, Thapar University, Patiala, India. She received her B.E. in Computer Science And Engineering and Mtech from Punjabi University and Ph.D. in the research area of Cloud Computing from Thapar Institute of Engineering & Technology, Patiala. She has more than 50 research publications in reputed journals and conferences and guided more than 35 ME thesis in the same area.

Sonam Yadav received her M.Sc in Mathematics from University of Delhi and Ph.D. in Lattice-based Cryptography from Shree Guru Gobind Singh Tricentenary University. Her current interests include Lattice-based Cryptography and Post-quantum Cryptography.