AN EFFICIENT CRYPTOGRAPHIC TECHNIQUE USING POST-QUANTUM CRYPTOGRAPHY

A Thesis submitted in fulfillment of the requirement for the award of the degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Vivek Dabra

(Registration No. : 951603009)

Under the guidance of:

Dr. Anju Bala
Associate Professor
Thapar Institute of Engineering & Technology
Patiala, Punjab (India)

Dr. Saru Kumari
Assistant Professor
Ch. Charan Singh University
Meerut, Uttar Pradesh (India)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY,
PATIALA – 147004

June 2022

CERTIFICATE

I, Vivek Dabra, hereby declare that the thesis entitled **"An Efficient Cryptographic Technique using Post-Quantum Cryptography"** submitted to the Computer Science and Engineering Department at Thapar Institute of Engineering & Technology, Patiala, Punjab, India is an authenticated record of my own work for the award of the degree of "Doctor of Philosophy" under the supervision of Dr. Anju Bala and Dr. Saru Kumari. This report has not been submitted to any other institution for award of any other degree.

(Vivek Dabra)
Regn. No. 951603009

Place: Patiala, Punjab (India)

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Verified by:

Dr. Anju Bala
(Supervisor)
Associate Professor
Thapar Institute of Engineering & Technology
Patiala, Punjab (India)

08.06.2022

Dr. Saru Kumari
(Co-Supervisor)
Assistant Professor
Ch. Charan Singh University
Meerut, Uttar Pradesh (India)

ABSTRACT

Post-quantum cryptography (PQC) refers to the cryptographic techniques that are secure against cryptanalytic attacks by quantum computers. In PQC, lattice-based cryptography (LBC) is the popular approach for designing public-key cryptographic techniques such as key exchange protocols, digital signature, and encryption techniques. The key exchange protocol is one of the basic cryptographic primitives of the Public Key Infrastructure (PKI). However, the literature about the study of key exchange protocols using lattice-based cryptography is limited, and often the schemes are studied independently. Therefore, a review of lattice-based key exchange protocols has been done in this work. Further, these protocols have been classified under two different categories depending on the reconciliation mechanism used by the protocol. From the comprehensive literature survey, it has been found that the key exchange protocols designed using LWE/RLWE problems of lattices are not secure if their public-private keys are reused. Due to key reuse, these key exchange protocols are vulnerable to signal leakage attack (SLA) and key mismatch attack. Among these signal leakage attacks is the most severe, and all the RLWE-based key exchange protocols are vulnerable to SLA attack. Therefore, an efficient cryptographic key exchange technique that can resist signal leakage attack has been proposed. This key exchange technique has been proposed for different scenarios. These scenarios are described below as.

Firstly, a new Lattice-based Anonymous Password Authenticated Key Exchange (LBA-PAKE) protocol for mobile devices has been proposed. The proposed protocol resists the signal leakage attack and provides key reusability, anonymity, and perfect forward secrecy. Also, the formal security analysis of the proposed LBA-PAKE protocol has been done using the widely adopted Real-Or-Random (ROR) model. Further, the proposed LBA-PAKE protocol and Feng et al.'s protocol have been implemented on the common mobile-server platform for the comparative performance analysis. The experimental results show that the proposed LBA-PAKE protocol is as efficient as Feng et al.'s protocol with an extra shield of security. Next, the modified two-party authenticated key agreement (m-2PAKA) protocol for post-quantum world has been proposed. This protocol is the improvement of Islam [1] provably secure two-party authenticated key agreement (2PAKA) protocol. By the cryptanalysis of Islam's 2PAKA protocols, it has been found that the protocol is vulnerable to improved-signal leakage attack (i-SLA) if its public/private keys are reused. Using i-SLA, the attacker can successfully recover the honest user's long-term private key by instantiating the utmost q number of key exchange

sessions with the honest user using q number of malformed public keys. Hence, the modified two-party authenticated key agreement (m-2PAKA) protocol has been proposed to counter the i-SLA attack. The proposed m-2PAKA protocol inherits the basic design of Islam's protocol, with an additional countermeasure to resist the i-SLA attack.

Finally, a simple lattice-based three-party password-authenticated key exchange (SL3PAKE) protocol has been proposed. The protocol is simple and resists signal leakage attack if its public/private keys are reused. The provable security of the proposed SL3PAKE protocol has been proved using the ROR model. Also, our concrete parameter of choice, implementations, and calculation of communication cost shows that protocol is efficient and practical.

ACKNOWLEDGMENT

Before discussing my journey of Ph.D., I would like to thank the almighty God who gave me strength and courage to overcome all the obstacles and complete this endeavour. Without acknowledging the people who supported me throughout this journey, this thesis would be incomplete. I know words are never enough to express the gratitude; I am just delivering the phrase for the acceptance of regards.

Firstly, I would like to express my sincerest thanks to my parents. With their consent, support, and motivation, I thought to accept this biggest challenge in my life. They have been the true source of real inspiration for me. Secondly, I would like to thank my supervisor, Dr. Anju Bala and Dr. Saru Kumari, who has supported me throughout my Ph.D. work with his patience and knowledge; while providing me with the room to work in my own way. Apart from providing me with excellent supervision, active cooperation, and constant encouragement throughout this journey, he also shared their invaluable experiences with me to succeed in life. I will always remain indebted to him.

I am also grateful to the head of the department, Prof. (Dr.) Maninder Singh, Dean of Student Affair Prof. (Dr.) Inderveer Chana, Ph.D. Coordinator, Dr. Sushma Jain, and members of my doctoral committee, Dr. Shreelekha Pandey, and Dr. Mukesh Singh for their constructive suggestions and ensuring the correct pace of my work. I am also obliged to the Director, Prof. (Dr.) Prakash Gopalan, Dean (RSP), Prof. (Dr.) Rafat Siddique, and the management of Thapar Institute of Engineering and Technology, who provided me with all the necessary resources and facilities to complete my work. I would like to thank all the lab technician of computer science & engineering department for their unconditional support. This will include Mr. Gurdeep Rongla, Mr. Rakesh Sharma, Mr. Hariom Sharma, Mr. Gursev Singh Preet and Mr. Divam Rahega.

The chain of my gratitude will definitely be incomplete if I forget to thank my complete family, my father Shri. Ashok Kumar, my mother Mrs. Neelam, my sister Anu and my wife Jagriti Kamboj, for their unconditional love, support, and encouragement in every phase of my life. It was due to my father's, mother, and my wife confidence and vision that motivated me to overcome every obstacle during the research. I would also like to express heartfelt thanks to Mr. Vikas Kumar, who always believed in me and whose blessings have truly played the role of game-changer in my life. I would also like to thank my friends and colleagues with whom I have traveled this journey of research. A special thanks to Dr. Sumedha Arora, Dr. Rajat Chaudhary and Mr. Vijay Soni. These people have made my research journey all the

more memorable and pleasant. As one cannot mention the names of all well-wishers, friends, and beloved ones, I would like to pay my regards to one and all who supported me during this journey of knowledge.

(**Vivek Dabra**)

List of Publications

SCI Indexed International Journals

1. Vivek Dabra, Anju Bala, and Saru Kumari, “LBA-PAKE: Lattice-based anonymous password authenticated key exchange for mobile devices", *IEEE Systems Journal*, pp. 1-11, September 2020. (Early Access, DOI: 10.1109/JSYST.2020.3023808) (IEEE, IF 3.931)

2. Vivek Dabra, Anju Bala, and Saru Kumari, “Flaw and amendment of a two-party authenticated key agreement protocol for post-quantum environments", *Journal of Information Security and Applications*,vol. 61, p. 102889, June 2021. (Elsevier, IF 3.872)

3. Vivek Dabra, Anju Bala, and Saru Kumari, “Reconciliation based key exchange schemes using lattices: a review", *Telecommunication Systems*, vol. 77, pp. 413–434, February 2021. (Springer, IF 2.314)

Communicated Article

1. Vivek Dabra, Saru Kumari, Anju Bala, Sonam Yadav “SL3PAKE: Simple Lattice-based Three-party Password Authenticated Key Exchange for post-quantum world", *IEEE Transactions on Information Forensics and Security*. (IEEE, IF 7.178).

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Definitions
ACCE	Authenticated and Confidential Channel Establishment
AKE	Authenticated Key Exchange
App	Application
BR	Bellare-Rogaway
CA	Certificate Authority
CMA	Chosen Message Attack
CVP	Closest Vector Problem
CRT	Chinese Remainder Theorem
IND-CPA	Indistinguishability under Chosen Plaintext Attack
KEM	Key Encapsulation Mechanism
LBAPAKE	Lattice-based Anonymous Password Authenticated Key Exchange
LBC	Lattice-based Cryptography
LSB	Least Significant Bit
LWE	Learning With Errors
MAC	Message Authentication Code
MSB	Most Significant Bit
NIST	National Institute of Standards and Technology
PAKE	Password Authenticated Key Exchange
2PAKA	Two-Party Authenticated Key Agreement
3PAKE	Three-Party Password-Authenticated Key Exchange
PAK	Password Authenticated Key exchange
PFS	Perfect Forward Secrecy
POK	Proof-of-Knowledge
POP	Proof-of-Possession
PPK	Password Protected Key exchange
PPT	Probabilistic Polynomial-Time
PQC	Post-Quantum Cryptography
PRF	Pseudo-Random Function
PSK	Pre-Shared Key
QROM	Quantum Oracle Model
RLWE	Ring Learning With Errors
ROM	Random Oracle Model
ROR	Real-Or-Random Model

SFA	Small Field Attack
SIGn-and-MAc	Signature and MAC
SIP	Session Initiation Protocol
SLA	Signal Leakage Attack
i-SLA	improved-Signal Leakage Attack
SL3PAKE	Simple Lattice-based Three-party Password Authenticated Key Exchange
SMAKE	SIGN-and-MAC Authenticated Key Exchange
SVP	Short Vector Problem
TLS	Transport Layer Security

Chapter 1

Introduction

Cryptographic techniques are the base of all the security practices used in modern era. Almost all the security system relies on cryptographic algorithms for the secure data storage and secure transmission of data. Till now, all these algorithms are dependent on hard mathematical problems for their security such as integer factorization problem, discrete logarithmic problem and elliptic curve discrete logarithmic problem. These problems are assumed to be computationally hard in the past but the invention of quantum computing change the whole perception of researchers. Now, it is believed that these classical mathematical problems will be solved easily by the powerful quantum computers. So, cryptographers are designing the algorithms that are secure in post quantum world which led to development of new field in cryptography known as post-quantum cryptography.

This chapter introduces the post-quantum cryptography (PQC) and its different family. Among the different families of PQC, the lattice family has been chosen; hence, the basic background of lattice-based cryptography (LBC) is presented. The famous LWE/RLWE problems of lattices and their implications in designing key exchange protocols have been discussed. In addition, thesis motivation and contribution have been presented to give an overall picture of the thesis.

1.1 Post-Quantum Cryptography: An Overview

Post-quantum cryptography is an emerging field of cryptography and it refers to cryptographic algorithms that are secure against quantum computers. The cryptographic algorithms which are based on the mathematical problems like factoring the large prime numbers, discrete logarithmic problem etc., are hard to break with present-day resources. Researchers show that these problems can be solved in polynomial time using quantum computers. In 1999, Shor [5] proposed a quantum algorithm for integer factorization that runs in polynomial time. This algorithm can be used to break RSA on an ideal quantum computer as the security of RSA depends on the fact that integer factorization of a large number is hard. Similarly, Grover's [6]

quantum algorithm can be utilized for searching an unstructured database in polynomial time. Researchers claim that although symmetric cryptography offers some resistance, the asymmetric or public key cryptography will be drastically affected by the introduction of quantum computers.

To resolve the above issue, researchers denoted themselves towards developing the cryptographic techniques that can withstand the quantum effect and at the same time provide efficiency equivalent to that of classical algorithms. Therefore, they have proposed many algorithms based on different approaches that can be used in post quantum cryptography. National Institute of Standards & Technology (NIST) classified these approaches into multivariate, hash-based, lattice-based, code-based, supersingular elliptic curve isogeny and symmetric key quantum resistance cryptography. Figure 1.1 shows the classifications of different approaches in post-quantum cryptography.

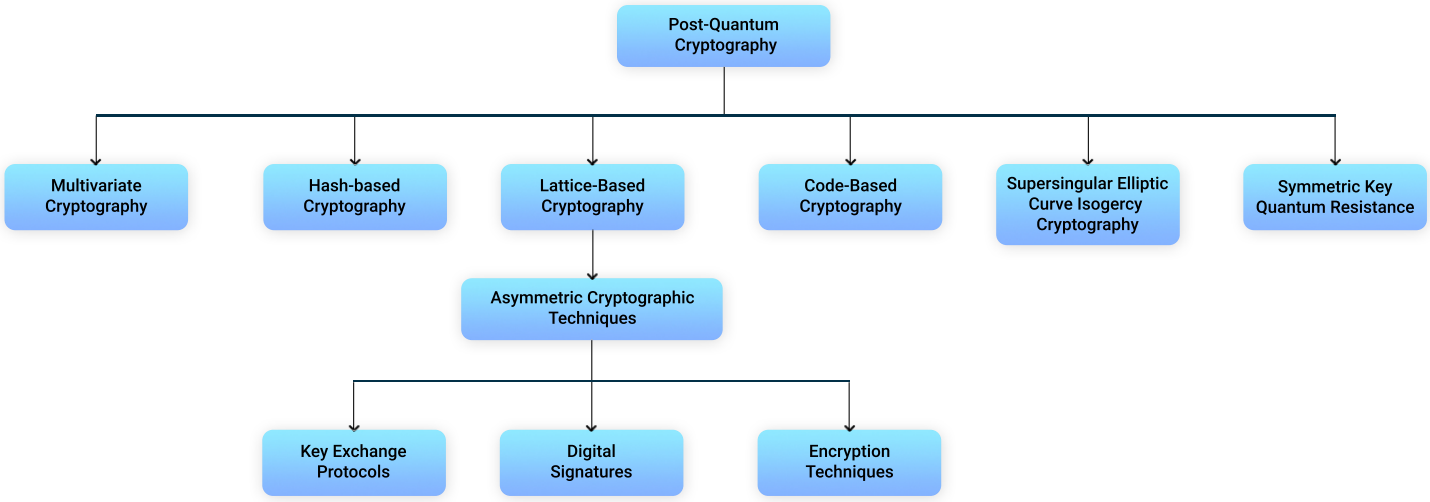


Figure 1.1: Classification of different approaches in Post-Quantum Cryptography

The lattice-based family is the most popular among these approaches. Almost all the public-key cryptographic techniques like key exchange protocols, digital signatures, and encryption techniques have been designed using lattice-based cryptography. Key exchange is one of the important cryptographic technique for the security of protocols on the internet like Transport Layer Security (TLS), Session Initiation Protocol (SIP), Voice over Internet Protocol (VoIP) etc. There are different lattice counterparts proposed taking inspiration from Diffie Hellman key exchange protocols. Most of these protocols are dependent on learning with errors (LWE) [7] or ring learning with errors (RLWE) [8] problems. Both these problems have been studied thoroughly by the researchers and they served as the basis of many key exchange schemes in lattice-based cryptography. Thus, the basic background of the lattice-based cryptography and Key Exchange protocols have been discussed in the below subsections.

1.1.1 Lattice-Based Cryptography

Lattices are set of points in n dimensional plane. Mathematically, description is as follows: Let $\mathbb{R}^k$ be the k dimensional Euclidean space. A lattice in $\mathbb{R}^k$ is the set

$$L(a_1, a_2, ..., a_m) = \sum_{i=1}^{m} x_i a_i : x_i \in \mathbb{Z} \tag{1.1}$$

of all integral combinations of m linearly independent vectors $a_1, a_2, ..., a_m$ *in* $\mathscr{R}^k$ where $k \geq m$. The integers m and k are called rank and dimension of lattice respectively. If m=k, then the lattice is called full rank lattice. The sequence of vectors $a_1, a_2, ..., a_m$ is called lattice basis which can be represented in matrix form as $B = [a_1, a_2, ..., a_m] \in \mathbb{R}^{k \times m}$ where basis vectors are occurring on columns. Using this matrix notation, equation (1) can be expressed as

$$L(B) = Bx : x \in \mathbb{Z}^m \tag{1.2}$$

where Bx is the usual matrix-vector multiplication.
Some of the common notations used throughout the thesis have been described in Table 1.1. If there is any symbol which is not explicitly mentioned, then the meaning of symbols can be referenced from the notation Table 1.1.

Table 1.1: Notation Table

Commonly Used Notations
q: large positive prime number
$\mathbb{Z}$: Set of integer numbers
$\mathbb{Z}_q$: $\mathbb{Z}$ modulo q
$s \in \mathbb{Z}_q^n$: Here s is a vector of n integers modulo q
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 postive integer
R_q: R/qR, Quotient Ring
$\lfloor . \rfloor$: Floor Function
$\lceil . \rceil$: Ceil Function
$\lceil . \rfloor$: Rounding Function
$A \mid\mid B$: String A concatenated with string B

There are various problems on lattices like SVP(Short Vector Problem), CVP (Closest Vector Problem), LWE (Learning with errors Problem), RLWE (Ring learning with errors Problem) that are useful for designing cryptographic primitives. LWE and RLWE are the problems that are widely used for designing the cryptographic primitives as these problems are well-studied [7, 8]. These problems are defined as follows:

Learning with errors (LWE) Problem: "Fix a size $n \geq 1$, a modulus $q \geq 2$ and an error probability distribution ψ on $\mathbb{Z}_q$. Let $A_{s,\psi}$ on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ be the probability distribution obtained by randomly choosing a vector $a \in \mathbb{Z}_q^n$, secret key be $s \in \mathbb{Z}_q^n$ and choosing error $e \in \mathbb{Z}_q$ according to ψ. Then, output is $(a, < a, s > +e)$, where additions are performed in $\mathbb{Z}_q$ i.e. modulo q. An algorithm will solve LWE problem with modulus q and error distribution ψ if for any $s \in \mathbb{Z}_q^n$ given an arbitrary number of independent samples from $A_{s,\psi}$, it outputs the secret key s with high probability in polynomial time" [7].

Ring learning with errors (RLWE) Distribution: "For security parameter λ, let $f(x) = x^n + 1$ where $n = n(\lambda)$ is a power of 2. Let $q \geq 2$ be a prime integer, $R = \mathbb{Z}[x]/ < f(x) >$, $R_q = R/qR$ and ψ be a discrete Gaussian distribution over R_q with small standard deviation say β. A distribution $A_{s,\psi}$ is created by taking the samples $(a_i, b_i) \in R_q \times R_q$, where a_i is uniformly sampled from R_q and $b_i = a_i s + e$, where s be drawn from R_q uniformly and error e is sampled from ψ as $e \leftarrow Sample(\psi)$" [8].
RLWE Assumption: "The RLWE assumption states that for a fixed s sampled from distribution ψ as $s \leftarrow Sample(\psi)$, the distribution $A_{s,\psi}$ is computationally indistinguishable from the uniform distribution on $R_q \times R_q$, given polynomial number of samples" [8].

1.1.2 Key exchange protocols overview

Key exchange protocols are designed to provide secure communication among two or multiple parties over an insecure network. These protocols are widely used for providing the security in various applications domains like smart grid [9], smart city [10], smart healthcare [11] etc. The common session key generated among the parties acts as a means to provide secure channel (secure channel is an establishment of the channel among the communicating parties by which they can exchange messages securely using symmetric encryption under the derived session key). The design of key exchange protocols to provide a secure channel needs some of the basic requirements to be fulfilled. These basic requirements are described below:

- **Authentication:** The key exchange protocol designed for session key generation must ensure that each party involved in the key exchange should be able to verify the identity of the peer involved.

- **Secrecy:** The key exchange protocol also needs to protect the session key from the third party i.e. no third party should be able to deduce anything about the session key created by two trustworthy peers.

- **Consistency:** If two honest parties create a shared session key, they must both have a consistent picture of who the session's peers are. The key exchange protocol must ensure

that the binding between the session key generated and the identities of the honest parties must be strong enough. The inability of this property leads to an attack called an identity misbinding attack which is more commonly known as an unknown key-share attack.

The famous Diffie-Hellman [12] key exchange protocol directly generates the shared secret key (also called session key) using public/private key pairs of the parties involved. Unlike the famous Diffie-Hellman key exchange and its other variants, the LWE/RLWE key exchange protocols do not directly yield the shared secret key from public/private keys of the parties involved. With LWE/RLWE key exchange protocols, the parties involved in the key exchange only compute approximately equal values of shared secret key. To cope with this issue, the notion of reconciliation has been introduced. The idea is that one of the parties sends a hint to the other party, so that they agree on the same shared secret. This is known as noisy Diffie-Hellman. The idea of reconciliation mechanism of the LWE/RLWE-based key exchange protocols has been discussed in below subsection.

1.1.3 Basic approach of reconciliation mechanism

The idea of reconciliation mechanism is to construct Diffie-Hellman styled key exchange protocol where both parties generate the same shared secret naturally without the need of encryption and decryption. These type of key exchange protocols offers better bandwidth requirement as compared to encryption based key exchange protocol.

To understand the reconciliation mechanism, consider the following RLWE based key exchange protocol, where both parties tries to establish a common session key. Let a be public known element such that $a \in R_q$ and χ_β is the discrete Gaussian distribution over R_q with standard deviation β. Now, the key exchange protocol operates as follows:

- First, Alice samples its secret and error term as $s_a, e_a \leftarrow \chi_\beta$ and sends its public key p_a to Bob as $p_a = a.s_a + e_a$.
- Then, Bob samples its secret and error term as $s_b, e_b \leftarrow \chi_\beta$ and computes σ_b as $\sigma_b = p_a.s_b = (a.s_a + e_a).s_b = a.s_a.s_b + e_a.s_b$. Finally, Bob computes computes p_b as $p_b = a.s_b + e_b$ and send it to Alice.
- Alice after receiving p_b computes σ_a as $\sigma_a = p_b.s_a = (a.s_b + e_b).s_a = a.s_a.s_b + e_b.s_a$.

From the above steps, note that σ_a is quite close to σ_b but they are not equal. So, additional efforts have to be made to make σ_a equal to σ_b. This is where reconciliation mechanism has job to be done. So, the main function of reconciliation mechanism is to eliminate the minor difference between σ_a and σ_b values so that both parties involved in key exchange will have same shared secret key or session key. Hence, in the above scenario Bob needs to send additional information about σ_b so that Alice can have same σ_a value. This additional information is the Hint value which is output of Hint function described below.

A reconciliation mechanism broadly comprised of two functions:

- Hint Funtion S(): The Hint function is also known as signal function . The hint function takes x as input, where $x \in R_q$ and output the hint value as $w = S(x)$, where $w \in \{0,1\}^*$.
- Reconciliation function $Rec()$: The Reconciliation function takes σ_a and σ_b along with hint value w as input and output the reconciled value such that $Rec(\sigma_a, w) = Rec(\sigma_b, w)$.

The reconciliation mechanism requires two conditions to be satisfied for its proper functioning. These conditions are as follows:

- Correctness- This condition states that if the difference of the calculated values of σ_a and σ_b are below some threshold value, then σ_a is equal to σ_b. Mathematically, this can be expressed as if $||\sigma_a - \sigma_b|| < thershold\ value$, then $\sigma_a = \sigma_b$. This threshold value depends upon the reconciliation mechanism being employed.
- Security- This condition says that for any element σ such that $\sigma \leftarrow R_q$, the reconciliation function $Rec(\sigma, w)$ is uniformly distributed over $\{0,1\}^*$ (where w is the output of Hint function as $w = S(\sigma)$).

1.2 Research Motivation & Contribution

In post-quantum cryptography (PQC), the lattice-based cryptography is the popular approach to design the public-key cryptographic techniques. These cryptographic techniques include key exchange protocols, digital signatures and encryption techniques. The key exchange protocols are designed using LWE/RLWE problems of lattices and they serves as substitute for Diffie-Hellman-styled key exchange protocols. However, deploying these key exchange protocols in existing Internet standards like Internet Key Exchange (IKE) and TLS makes these protocols insecure when their public/private keys are reused. Key reuse is a common practice that is vastly employed to save computation and communication overhead of the internet standards. Despite its overhead benefits, this practice exposes key exchange protocols to signal leakage attack (SLA) and key mismatch attack (both attacks are discussed in chapter 2). The signal leakage attack is the more serious of the two, as it may be used to compromise nearly every RLWE-based key exchange protocol. Hence, there is need to design an efficient cryptographic key exchange technique that can resist signal leakage attack. Therefore, the main motive of the current research is to propose an efficient key exchange technique that support key reuse in lattice-based cryptography. To achieve the above purpose, the key exchange technique has been proposed for different scenarios. These scenarios are described below as.

Firstly, the crypt-analysis of ideal lattice-based anonymous authentication protocol for mobile devices [4] has been done. This is the first anonymous authentication protocol for mobile devices in lattice-based cryptography. The protocol is designed using the RLWE problem of

lattices and is claimed to be quantum-secure. But after careful analysis, we found that the protocol is vulnerable to signal leakage attack if its public/private keys are reused. Also, due to design flaws, the protocol is susceptible to spoofing attack, manipulation-based attacks, and user anonymity attack. Thus, the Lattice-based Anonymous Password Authenticated Key Exchange (LBA-PAKE) for mobile devices has been proposed to overcome the above security weaknesses. The proposed LBA-PAKE protocol resists the above attacks, in addition to it, it also provides key reuse, anonymity, and perfect forward secrecy features. The formal security analysis of the proposed LBA-PAKE protocol has been done using the widely adopted Real-Or-Random (ROR) model. Finally, the proposed LBA-PAKE protocol and Feng et al.'s protocol have been implemented on the common mobile-server platform for the comparative performance analysis. The experimental results show that the proposed LBA-PAKE protocol is as efficient as Feng et al.'s protocol with an extra security shield.

Next,the crypt-analysis of the Islam's [1] two-party authenticated key agreement (2PAKA) protocol for post-quantum environments has been done. The protocol is based on the RLWE problem and does not include any pre-registration step, hence supports the key exchange between two unknown parties. The protocol employs the implicit authentication of the parties involved and does not use extra cryptographic primitive. Despite the above advantages, it has been proved that the Islam's [1] key agreement protocol (2PAKA) is vulnerable to the improved-signal leakage attack (i-SLA) [13] if its public/private keys are reused. Using i-SLA, the attacker can successfully recover the reused secret key of the honest party by instantiating the utmost q number of key exchange sessions with q number of malformed public keys, which are authenticated by the Certificated Authority (CA). Hence, the modified two-party authenticated key agreement (m-2PAKA) protocol has been proposed to counter the i-SLA attack. The proposed m-2PAKA protocol inherits the basic design of Islam's protocol, with an additional countermeasure to resist the i-SLA attack.

Finally, the three-party key exchange protocols based on RLWE problem has been analyzed. From the analysis, it has been found that all the three-party protocols (Xu et al. [14], Choi et al. [15] and Liu et al. [16]) based on RLWE problem of lattices are vulnerable to signal leakage attack 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, a Simple Lattice-based Three-Party Password Authenticated Key Exchange (SL3PAKE) protocol has been proposed. The proposed SL3PAKE protocol is simple in its design and resists signal leakage attack if its public/private keys are reused. The provable security of the proposed SL3PAKE protocol has been proved using the ROR model. Also, the concrete parameter of choice, implementations, and calculation of communication cost shows that protocol is efficient and practical.

1.3 Thesis Organization

The introduction of the thesis has been presented in chapter 1 and rest is structured as follows:

Chapter 2: Literature Review
This chapter provides a comprehensive literature survey of key exchange protocols designed using LWE/RLWE problems of lattices. The chapter first classifies the key exchange protocols based on the reconciliation mechanism. Further, the security analysis of these key exchange protocols has been provided. In addition, the basic reconciliation schemes have been discussed whose reconciliation mechanisms are adopted by other schemes. Further, the literature survey findings have been described, followed by research gaps and objectives of the research work.

Chapter 3: LBAPAKE: Lattice-based Anonymous Password Authenticated Key Exchange for mobile devices
This chapter first crypt-analyzed the ideal lattice-based anonymous authentication protocol for mobile devices proposed by Feng et al. The cryptanalysis of Feng et al.'s protocol shows that the protocol is vulnerable to signal leakage attack (SLA), spoofing attack, manipulation-based attacks, and user anonymity violation attack. Therefore, we have proposed a new Lattice-based Anonymous Password Authenticated Key Exchange (LBAPAKE) for mobile devices to overcome the above security weaknesses. The proposed protocol resists the above attacks and provides key re-usability, forward secrecy, and anonymity to the participants involved in the key exchange. Also, the formal security of the proposed protocol has been proved using Abdalla et al.'s ROR model. Finally, the comparative performance analysis of the proposed LBAPAKE protocols and Feng et al. protocol has been done.

Chapter 4: Modified two-party authenticated key agreement protocol for post-quantum world
In this chapter, the cryptanalysis of Islam's two-party authenticated key agreement (2PAKA) protocol has been done. It has been found that the protocol is vulnerable to improved-signal leakage attack (i-SLA). Thus, to overcome the above vulnerability, the modified two-party authenticated key agreement protocol has been proposed without changing the original design of Islam's protocol. In addition, the condition of the correctness of the modified 2PAKA has also been computed.

Chapter 5: SL3PAKE: Simple Lattice-based Three-party Password Authenticated Key Exchange for post-quantum world
A Simple Lattice-based Three-party Password Authenticated Key Exchange (SL3PAKE) for the post-quantum world has been proposed in this chapter. The proposed SL3PAKE resists signal leakage attack and thus, supports key reuse. The formal security of the proposed pro-

tocol has been proved using Abdalla et al.'s ROR model. Finally, performance analysis with a concrete choice of parameters has been done.

Chapter 6: Conclusion and Future Scope

This chapter concludes the thesis by highlighting the contributions made using the proposed schemes. Moreover, this chapter provides future directions of the proposed key exchange protocols.

Chapter 2

Literature Review

Post-quantum cryptography is broadly classified into six different families. These families are as follows: lattice-based, code-based, supersingular elliptic curve isogeny, multivariate, hash-based, and symmetric key quantum resistance cryptography. Among these, lattice-based is the leading candidate for public-key post-quantum cryptography. The LWE/RLWE problems of lattices are used to design a wide range of public-key cryptographic techniques, including key exchange protocols, digital signatures, and encryption techniques. The key exchange is the important cryptographic technique for the security of the internet protocols such as Transport Layer Security (TLS), Internet Key Exchange (IKE), Session Initiation Protocol (SIP), etc.

Thus, a comprehensive literature review of key exchange protocols designed using LWE/RLWE problems of lattices has been provided in this chapter. The chapter first classifies the key exchange protocols based on the reconciliation mechanism used. After that, the security analysis of these key exchange protocols has been provided. In addition to it, the complete review and security analysis of the basic reconciliation-based key exchange schemes are given in section 2.3. Also, the literature survey findings have been described, followed by research gaps and objectives of the research work.

2.1 Classification of LWE/RLWE-based key exchange protocols

The taxonomy of key exchange protocols based on LWE/RLWE problem has been shown in Figure 2.1. From the figure, it can be seen that there are two major categories of LWE/RLWE-based key exchange protocols. First category is based on Ding et al.'s [2] reconciliation mechanism and second category is based on Peikert's [3] reconciliation mechanism. Both of these reconciliation mechanism are discussed in detail section 2.3. These two reconciliation mechanisms are the base for all the reconciliation-based key exchange protocols in lattice based cryptography. In this section, all the reconciliation-based key exchange protocols have been classified in two categories. These categories have been discussed below as:

i. **Key exchange protocols based on Ding et al.'s [2] reconciliation mechanism:** Ding et al. [2] is the first one to employ reconciliation mechanism in its key exchange protocol to generate the shared session key. The idea is to propose the Diffie-Hellman variant of the key exchange protocol using LWE/RLWE problem that is secure against the quantum adversary. In this way, the common shared secret key is being generated at both parties naturally as a part of protocol without the use of encryption/decryption. In the proposed protocol, odd modulus and even errors are used while generating LWE and RLWE samples. Also, a hint value (discussed in section 2.3.1) $\sigma_b()$, where $b \in \{0,1\}$ is sent by one party to another party so that both parties agree on same value of session key. This hint value will help to recover the least significant bits (LSB) of the final generated session key. Also, the passive security of the protocol is proved against PPT (Probabilistic Polynomial-Time) adversary, which is based on hardness of LWE and RLWE problem.

In 2015, Zhang et al. [17] proposed the authenticated key exchange protocols using Ding et al.'s reconciliation mechanism. The motivation behind the protocol is to propose the authenticated key exchange protocol using ideal lattices that do not use extra cryptographic primitive like digital signature for authentication and whose security is solely relies on hardness of RLWE problem. The protocol is similar to classical HMQV [18] and OAKE [19] protocol and inherits the nice features of these protocols like implicit authentication, better efficiency. The security of the protocol is proved using BR model [20] with perfect forward secrecy. The protocol uses two different Gaussian distribution to sample RLWE samples and while implementing we find out that approach in this work is a very inefficient way to do lattice-based AKE (Authenticated Key Exchange).

In 2017, another work of Ding et al. [21] designed provably secure password authenticated key exchange (PAKE) using the reconciliation mechanism of Ding et al. The proposed key exchange protocols are the lattice variant of PAK and PPK [[22], [23]] protocols. The proposed protocol assumes that the two parties involved in key exchange have shared password of low entropy beforehand. They will now authenticate each other and generate the shared secret key by using the proposed key exchange protocol. The trick is to do this in such a way that an attacker cannot brute force the password with recorded data. This ensures the password remaining secure in spite of the low entropy because attacker have to talk to one of the parties to try a guess. The security of the protocol is proved using BR model but after carefully analyzing the protocol, we find out that both versions of protocol (PAK & PPK) are vulnerable to signal leakage attack (SLA) (discussed in section 2.2.1). The standard user with malicious intentions can recover the secret key s_s of the server in polynomial time if public/private keys of the server are reused.

Thus, motivated to design key exchange protocol that can resist SLA attack, Gao et al. in 2018, proposed the "practical randomized RLWE based key exchange protocol" [24]. "The protocol inherits the reconciliation mechanism of Ding et al. in its design with the use of additional error term e_p. The scheme introduces two modes of protocol design, one is regular

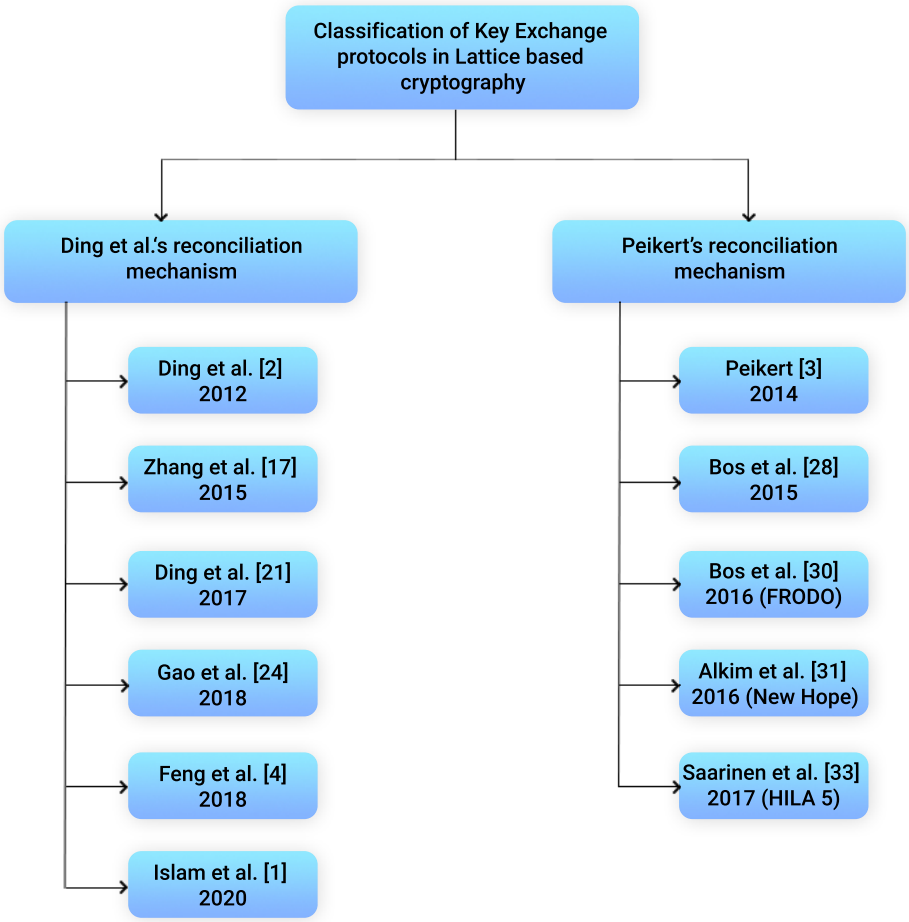


Figure 2.1: Taxonomy of Key Exchange protocols based on LWE/RLWE problems of lattices

mode and the other is key reuse mode. In regular mode, fresh keys are generated every time the protocol is instantiated while key reuse mode allows both the parties to reuse the private/public keys. Due to reuse of public /private keys the computation and communication cost is low in key reuse mode. The practical parameters are chosen in such a way that it offers at least 200-bit of classical and 80-bit of quantum security" [24].

"The error term e_p will ensures the randomness in the generated signal function values and thus, the generated signal values is indistinguishable from uniform random values, therefore, the SLA attack no longer works. The author proved the randomness of the signal value using the NIST statistical test suite [25] for a random number. The result of the test comes positive and the signal value passes all randomness tests on 1,000,000 signal bit strings. These results prove that the output of the signal function is truly random even with reused RLWE keys and therefore, the signal leakage attack is not possible against the proposed protocol" [24]. However, the authors have used a heuristic justification approach (using NIST randomness test) to prove the security of the protocol against signal leakage attack and does not provide provable security of the protocol. The active security of the proposed protocol has also not been analyzed.

In 2018, Feng et al. [4] proposed proposed the first lattice-based anonymous key exchange protocol for mobile devices. The proposed protocol instantiate the reconciliation mechanism of Ding et al. and includes the user registration and mutual authentication phase. Both formal and informal security of the protocol have been provided by the author. The informal security includes security against many of the known attacks and formal security has been done in Random Oracle Model (ROM). The result of the analysis shows that the protocol is secure in client-server environment. But despite of the above security claims, our recent work (Dabra et al. [26]) shows that the protocol is vulnerable to signal leakage attack, spoofing attack, manipulation based attacks and user anonymity violation attack.

Recently, Islam [1] proposed the provably secure two-party authenticated key agreement (2PAKA) protocol using Ding et al.'s reconciliation mechanism. In the proposed protocol, both participating parties uses the public/private keys issues by the Certificate Authority (CA). The protocol has been designed using the intractability assumption of RLWE problem and its provable security is proved using random oracle model (ROM). As public/private keys used by both parties are issued by Certificate Authority so they are being reused. The key reuse is an important feature that saves the computation time and vastly employed in the majority of TLS (Transport Layer Security) connections. But this feature makes this key agreement protocol vulnerable to signal leakage attack (SLA) and thus, adversary can recover the secret key of the honest party by carefully observing the signal function output (also known as hint function).

ii. **Key exchange protocols based on Peikert's [3] reconciliation mechanism :** Peikert proposed another idea of reconciliation mechanism that uses most significant bit (MSB) of the number in $\mathbb{Z}_q$ to generate the reconciliation value instead of least significant bit (LSB) as used in the above Ding et al.'s reconciliation mechanism. Peikert reconciliation mechanism divides $\mathbb{Z}_q$ into four intervals. The two intervals have same signal bits but different reconciled bits if they are not neighboring interval. If signal bit of the secret key is provided to the passive adversary then she cannot decide the region in which the secret key lies. The Peikert also designed key exchange protocol [3] based on the the above reconciliation mechanism. The detailed description of the key exchange protocol along with reconciliation mechanism is given in section 2.3.2. The security of the key exchange protocol has been proved using CK-model in post-specified peer settings [27].

Peikert's error reconciliation mechanism has been adopted by many other key exchange protocols in their work. One of them is the Bos et al.'s scheme [28] that utilizes Peikert's reconciliation mechanism into their key exchange protocol, which is integrated to Transport Layer Security (TLS) protocol. The scheme uses RSA or elliptic curve digital signatures to provide authentication. This work shows that their scheme is provably secure in ACCE (Authenticated and Confidential Channel Establishment) model and also claim that it provides post-quantum forward secrecy. However, in 2017, Gao et al. [29] present a comparison analysis of Ding et al. [2] and Bos et al. [28] key exchange protocols. This work shows that their efficient imple-

mentation of Ding et al.'s key exchange protocol is 11 times faster than Bos et al.'s protocol. This shows that Bos et al. protocol has high computation cost than Ding et al. protocol.

In 2016, Bos et al. [30] proposed the key exchange protocol, termed as Frodo, using LWE problem. The protocol is designed by generalizing the Peikert's reconciliation mechanism to agree on multiple bits rather than on a single bit from coefficients in $\mathbb{Z}_q$. But this will happen at the cost of error term value, which should be kept small. For larger error tolerance, Alkim et al. [31] proposed the key exchange protocol, termed as new hope. The proposed protocol extends the Peikert's reconciliation mechanism and uses the multiple-bit signal value instead of single-bit and extract a single bit from four coefficients. The proposed protocol chooses good parameter sizes and error distribution, specifically, the centered binomial distribution ψ_k of parameter k=16 is utilized instead of Gaussian distribution for sampling the secret and error terms. The security of the proposed scheme is analyzed similar to LWE based schemes, as there is no known attack that makes use of ring structure. After analyzing different LWE attacks, the post-quantum security of the protocol has been claimed to be 128 bits. The scheme received much attention when Google utilized the scheme for the security of google chrome web browser [32] and launched the test version of the google chrome that is quantum secure.

In 2017, Saarinen [33] introduced the efficient reconciliation mechanism by introducing the concept of SafeBits based on Peikert's [3] error reconciliation mechanism. The protocol in [33], which is also known as HILA5, employs an error correction method that can be implemented without branches or table lookups on secret data and thus provides defense against side-channel attacks. The protocol does not need a randomized smoothing function (also known as a randomized doubling function in Peikert's scheme [3]) to produce unbiased keys as SafeBits technique produces non-biased secrets. The author claims that the scheme is as efficient as new hope [31] scheme with shorter messages and decryption failure rate falls to 2^{-128} as compared to 2^{-60} in original new hope scheme. Due to low decryption failure rate, the scheme is suitable for both key exchange protocols and public key encryption. As the scheme is based on existing Peikert's error reconciliation technique, therefore the protocol is categorized under Peikert's reconciliation mechanism. Further, the SafeBits algorithm introduced in this work can be utilized to increase the efficiency of the existing key exchange protocols based on Peikert's reconciliation mechanism. The complete summary of the above-discussed key exchange protocols are tabulated in Table 2.1.

Table 2.1: Summary of key exchange protocols designed using LWE/RLWE problems

Schemes	**Problem**	**Year**	**Proposed Scheme**	**KEM/PKE**	**Message-Pass**	**Security Justification**	**Reconciliation Mechanism used**
Ding et al. [2]	LWE & RLWE	2012	A Simple Provably Secure Key Exchange Scheme Based on the LWE	-	2-PASS	Provable passive security against PPT adversary	Ding et al.
Peikert [3]	RLWE	2014	Lattice Cryptography for the Internet	IND-CPA	3-PASS	Scheme is proved to be SK-Secure	Peikert
BCNS [28]	RLWE	2015	Post-quantum key exchange for the TLS protocol from RLWE	IND-CPA	4-PASS	Proved using ACCE model	Peikert
Zhang et al. [17]	RLWE	2015	Authenticated Key Exchange from Ideal Lattices	-	2-PASS	Proved using BR-model with perfect forward secrecy	Ding et al.
FRODO [30]	LWE	2016	Frodo: Take off the ring! Practical, Quantum-Secure Key Exchange from LWE	IND-CPA	2-PASS	Proved using ACCE model	Peikert

New Hope [31]	RLWE	2016	Post-quantum key exchange-a new hope	-	2-PASS	Brute force the possibility of all the quantum attacks	Peikert
Ding et al. [21]	RLWE	2017	Provably Secure PAKE Based on RLWE for the Post-Quantum World	-	3-PASS/2-PASS	Proved using BR Model	Ding et al.
Saarinen [33]	RLWE	2017	HILA5:On Reliability, Reconciliation and error correction for RLWE encryption	IND-CCA	2-PASS	No provable security	Peikert
Gao et al. [24]	RLWE	2018	Practical Randomized RLWE-Based Key Exchange Against Signal Leakage Attack	-	2-PASS	Heuristic Justification	Ding et al.
Feng et al. [4]	RLWE	2019	Ideal lattice-based anonymous authentication protocol for mobile devices	-	3-PASS	Proved in Random Oracle model (ROM)	Ding et al.
Islam [1]	RLWE	2020	Provably secure two-party authenticated key-agreement protocol for post-quantum environments	-	4-PASS	Proved in Random Oracle model (ROM)	Ding et al.

2.2 Security Analysis of LWE/RLWE-based key exchange protocols

In this section, we will classify the different types of attacks against the LWE/RLWE-based key exchange protocols. The taxonomy of attacks is shown in Figure 2.2. From the figure, it can be seen that the attacks are classified in two branches. First branch is key reuse attacks and other branch is small field attack. The key reuse branch is further divided into signal leakage attack and key mismatch attack. The signal leakage attack node further consist of nodes of the specific signal leakage attack and same is the case with the key mismatch attack. Finally, leaf nodes are the key exchange protocols that are vulnerable to these attacks. Same scenario is adopted for small field attack. The taxonomy of attacks has been discussed below.

2.2.1 Key Reuse attacks

The key reuse is the common practice in widely used Internet standards. One of them is TLS 1.3 (Transport Layer Security), where keys are reused in pre-shared key (PSK) mode. Recently, there has been series of work which shows that the key reuse in the reconciliation based key exchange protocols make them vulnerable to different types of attacks. These attacks are broadly classified into two categories as : signal leakage attack (SLA) and key mismatch attack. In signal leakage attack (SLA), the adversary will observe the signal function output to recover the reused secret key of the honest party. On the other hand, in key mismatch attack the attacker has an approach of querying the honest party number of times and then tries to recover the secret key based on the match or mismatch of the final shared key. These two types of attacks is discussed below.

i. **Signal Leakage Attack (SLA):** As shown in Figure 2.2, the SLA attack is broadly divided into signal attack by Ding et al. [34], improved signal leakage attack by Ding et al. [13] and signal leakage attack by Liu et al. [35]. The three SLA attack uses different approaches to recover the secret key of honest party by observing the signal function output. Firstly, Ding et al. [34] proposed the signal leakage attack against RLWE based key exchange protocols that reuses the public/private key pair. In this attack, the attacker will instantiate the multiple sessions with the honest party and analyze the output of the signal function to derive the private key of the honest party. The attack will also work against the schemes where the final shared keys have been derived from LSB (Least Significant Bits) of approximately same keys computed by both parties. Therefore, the attack will work against the Ding et al. [2] key agreement protocol. The complexity of SLA attack to recover the private key s of the honest party is $2 * q$ (where q is an odd prime number as described in notation Table 1). The attack has been successful against the basic Ding et al. [2] protocol, Ding et al. [21] protocol, Feng et al. [4] protocol and recently, proposed Islam [1] key exchange protocol.

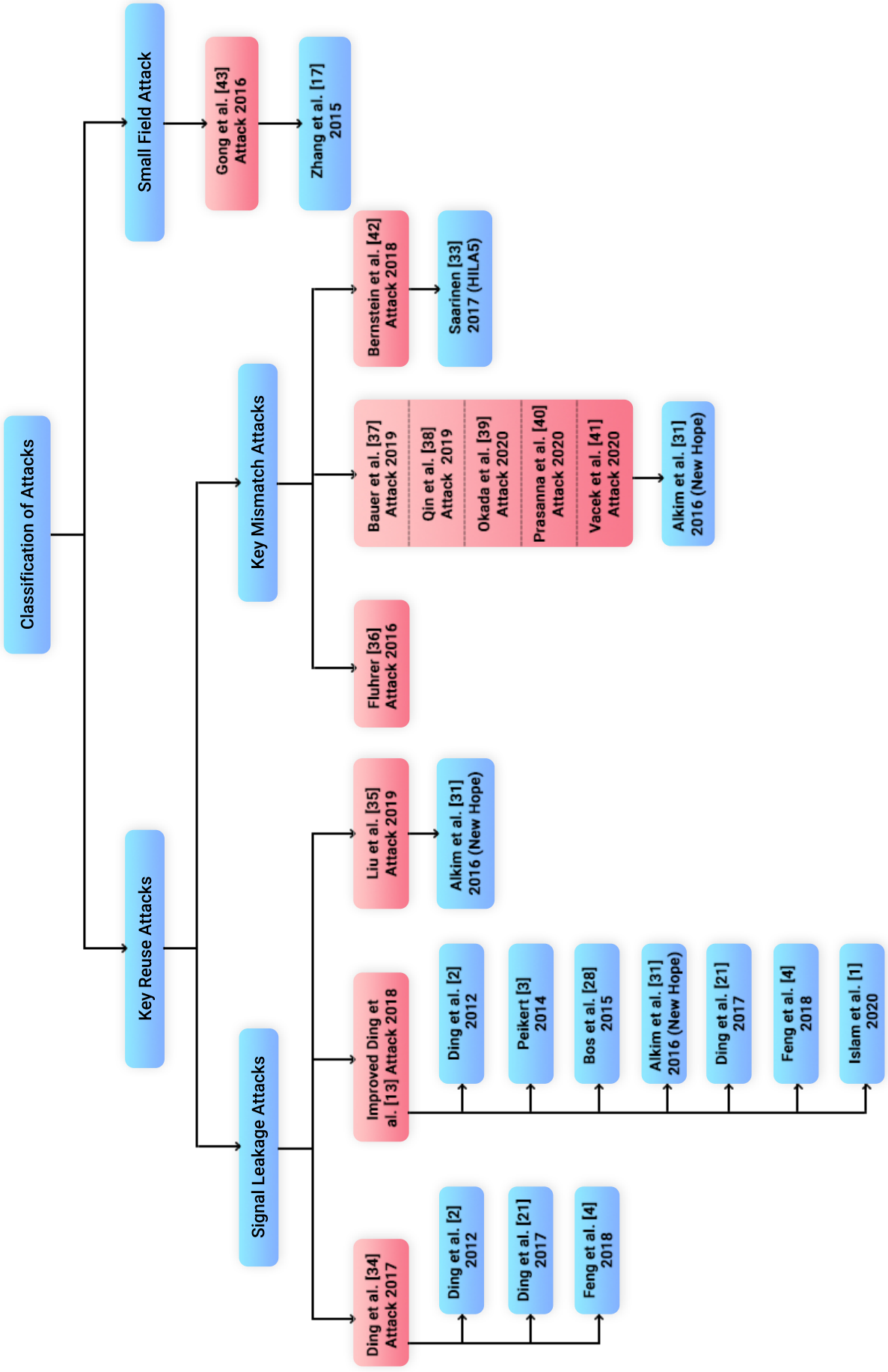


Figure 2.2: Taxonomy of Attacks against Key Exchange protocols designed using LWE/RLWE problems

In another work of Ding et al. [13], it has been shown that the SLA attack can be mounted efficiently with less number of queries. The authors in [13] suggest that it is not necessary to vary the constant k in the malformed public key of adversary looping from 0 to $q-1$ (as done in original SLA [34]) and there are many iterations of k that can be skipped. Thus, the fewer number of queries are required to recover the private key s of the honest party. These fewer number of queries are utmost $q+c$ (where c is a constant and q is an odd prime number as described in notation Table 1) and thus, the complexity of the SLA attack is reduced to $q+c$, in comparison to $2q$ in the original SLA attack. In section 7 of [13], authors considered two cases of basic Ding et al.'s protocol [2] to show how a fewer number of queries can successfully recover the private key of the honest party. The first case is the simplified one where the error term g_B is not added by the honest party during the computation of its shared key K_B. The adversary chooses its private key s_A to be 0 and the corresponding public key is a constant term i.e. $p_A = k$. The second case is the complicated one, where error term g_B is added by the honest party during its shared key computation K_B, and also, adversary deviates slightly from the original protocol by choosing its error term $e_A = 1$ and private key s_A according to the error distribution χ_β. Thus, the public key of the attacker is of the form $p_A = as_A + k$, where $a \in R_q$. The authors show that the complexity of attack for the simplified case is $\frac{q}{2}+4$ and for the complicated case is $q+c$.

The key exchange protocols which are vulnerable to the above Ding et al. [34] signal leakage attack are also vulnerable to this improved attack. In addition to it, this work also shows that the proposed signal leakage attack can be extended to Peikert key exchange protocol [3] and the schemes that use the reconciliation mechanism of Peikert like Bos et al. [28] and Alkim et al. [31], also known as new hope are vulnerable to signal leakage attack (see section 6 of [13]). Lastly, Liu et al. [35] in 2019, proposed a new signal leakage attack against the new hope [31] protocol. The author emphasize the fact that the reconciliation mechanism of new hope is more complex than Ding et al. [2] reconciliation mechanism and thus, the Ding et al. [34] signal leakage attack cannot be directly applied against new hope key exchange protocol. Therefore, in the proposed work a new method is introduced that will use special property of new hope signal function to recover the reused secret key of the honest party. The experimental results of the proposed attack [35] verifies the correctness of the attack.

ii. **Key Mismatch Attack:** In 2016, Fluhrer [36] first proposed the key mismatch attack against the RLWE-based key exchange protocols that reuses the public/private key pair. The attack works against the protocols where final shared keys have been generated from MSB (Most Significant Bits) of the approximately equal keys computed by both parties. Taking idea from this attack, many different versions of key mismatch attack have been proposed which are shown in Figure 2.2.

From the Figure, it can be seen that there are five mismatch attacks that have been proposed against the popular new hope key exchange protocol. These are key mismatch attack by Bauer

et al. [37] in 2019, Qin et al. [38] in 2019, Okada et al. [39] in 2020, Prasanna et al. [40] in 2020 and recently proposed by Vacek et al. [41]. In the first four attacks i.e by Bauer et al. [37], Qin et al. [38], Okada et al. [39] and Prasanna et al. [40], the adversary has access to key mismatch oracle and she is looking to find the specific output patterns. These patterns will help the adversary to recover the secret key of the honest party. In these attacks, the adversary act as as one of the participating party in key exchange protocol i.e. may be Alice or Bob and will require more than 26000 queries to key mismatch oracle. Recently, Vacek et al. [41] proposed more efficient attack against the new hope key exchange protocol. The attack will recover the reused secret key of the honest party with 100% probability with less than 3200 queries on average. In 2018, Bernstein et al. [42] proposed another key mismatch attack against HILA5 [33] technique. The proposed attack proves that HILA5 is not CCA secure and an adversary can recover the secret key of the honest party by sending multiple encapsulation messages and using the replies of honest party to determine whether her decapsulated shared secret matches a certain guess or not.

2.2.2 Small Field Attack (SFA)

This attack was proposed by Gong et al. [43] and it uses the CRT(Chinese Remainder Theorem) basis of R_q to recover every CRT coefficient of the secret key s, corresponding to a public key $p = as + e$. The attack is pointed out by Gong et al. [43] against the one-pass π_1 variant of Zhang et al.'s scheme [17]. Gong et al. [43] has also questioned the security claim (claim 16) of the two-pass π_2 variant of the Zhang et al.'s scheme [17]. The SFA attack allows the adversary to recover the static key of the honest user after issuing a set of random looking session queries to the honest party in one-pass π_1 variant of Zhang et al.'s scheme. The random looking queries make it hard for the honest user to prevent or even detect the attack.

SFA attack will work in the scenario where both static private keys and ephemeral private keys are mixed in generating the key material from which the session key is derived as in case of one-pass π_1 variant of Zhang et al.'s scheme. According to Gong et al. [43], SFA does not work in schemes where only one private key, either the static private key or the ephemeral private key, is utilized for session key generation; the attack is applicable when both, the static private key and the ephemeral private key, are utilized for session key generation. Therefore, this attack does not work in schemes where only ephemeral private keys are involved in the session key generation. Moreover, SFA also requires the adversary to register its public keys on behalf of the dishonest user. This is practically feasible situation and by not allowing an adversary to register public keys on behalf of dishonest user leads to weak and unrealistic security model. Also, the traditional mechanism for proof-of-knowledge (POK) or proof-of-possession (POP) of the private key does not prevent SFA attack as the adversary knows the private key corresponding to the registered public key. In Table 2.3, the comparative security analysis of key exchange protocols has been done using the above discussed attacks.

Table 2.3: Comparative Security Analysis between key exchange protocols and various attacks

Attacks	**Ding et al. [2] 2012**	**Peikert [3] 2014**	**Zhang et al. [17] 2015**	**Bos et al. [28] 2015**	**Bos et al. [30] 2016**	**Alkim et al. [31] 2016 (New Hope)**	**Saarinen [33] 2017 (HILA5)**	**Ding et al. [21] 2017**	**Gao et al. [24] 2018**	**Feng et al. [4] 2019**	**Islam [1] 2020**
Ding et al.'s [34] Signal Leakage Attack	√	√	√	√	√	√	×	√	×	√	×
Improved Signal Leakage Attack [13]	√	√	√	√	√	√	×	√	×	√	√
Liu et al.'s [35] Signal Leakage Attack	×	×	×	×	×	√	×	×	×	×	×
Small Field Attack [43]	×	×	√	×	×	×	×	×	×	×	×
Bauer et al.'s [37] Key Mismatch Attack	×	×	×	×	×	√	×	×	×	×	×
Okada et al.'s [39] Key Mismatch Attack	×	×	×	×	×	√	×	×	×	×	×

Qin et al.'s [38] Key Mismatch Attack	×	×	×	×	×	√	×	×	×	×	×
Prasanna et al.'s [40] Key Mismatch Attack	×	×	×	×	×	√	×	×	×	×	×
Vacek et al.'s [41] Key Mismatch Attack	×	×	×	×	×	√	×	×	×	×	×
Bernstein et al.'s [42] Key Mismatch Attack	×	×	×	×	×	×	√	×	×	×	×

2.3 Review and Security Analysis of the basic reconciliation based Key Exchange Schemes using Lattice-based Cryptography

From section 2.1, it is apparent that the key exchange schemes proposed by Ding et al. [2] and Peikert [3] are the basic instantaneous of the reconciliation mechanism. Schemes in [17,21,24] and [28,30,31] inherit the reconciliation mechanism of Ding et al.'s [2] and Peikert's scheme [3] respectively. In the forthcoming subsections, a detailed review and analysis of the Ding et al. [2] and Peikert's [3] key exchange scheme has been presented.

2.3.1 Review of Ding et al.'s Scheme [2]

The idea behind the design of Ding et al. 's scheme is to propose the quantum variant of the Diffie-Hellman key exchange protocol, which is simple as well as elegant in its design. Therefore, Ding et al. [2] in 2012, proposed a simple provably secure key exchange scheme based on LWE problem and later extended the protocol to RLWE problem. The key idea is to share a secret given by the values of a bilinear function of two elements x and y via the bilinear form

$$Q(x,y) = x \times m \times y \tag{2.1}$$

where x and y are sampled from distribution ψ, ψ is discrete Gaussian distribution over R_q with small standard deviation say β, m is a fixed element of R_q and a small error is introduced to make the system secure. The reconciliation mechasim introduces the notion of robust extractor so that both parties will have a common secret session key. The communicating parties would compute two very close values in R_q and then one party additionally sends a signal of his value to the other so that both the parties would share the same secret key. The protocol in RLWE settings is described as follows.

Let Alice and Bob be the two parties. Now, the reconciliation mechanism of the scheme includes the Signal function σ that uses the Hint algorithm S() to generate the output and robust extractor. These are defined as follows

For prime q such that $q > 2$, the signal functions $\sigma_0(x)$ and $\sigma_1(x)$ are given by

$$\sigma_0(x) = \left\{ \begin{array}{ll} 0 & if\ x \in [-\lfloor\frac{q}{4}\rfloor, \lfloor\frac{q}{4}\rfloor] \\ 1 & \text{otherwise} \end{array} \right\};$$

$$\sigma_1(x) = \left\{ \begin{array}{ll} 0 & if\ x \in [-\lfloor\frac{q}{4}\rfloor + 1, \lfloor\frac{q}{4}\rfloor + 1] \\ 1 & \text{otherwise} \end{array} \right\}$$

The robust extractor is defined as

$$E(x,\sigma) = (x + \sigma.\frac{q-1}{2}\ mod\ q) \bmod 2$$

The Hint function S(), is defined as: For any $y \in \mathbb{Z}_q$, $S(y) = \sigma_b(y)$ where $b \xleftarrow{\$} \{0,1\}$, here \$ denotes randomly chosen from $\{0,1\}$. The parameters for the key exchange protocol are n, q, m, ψ which are defined as follows:

- Let n be the security parameter such that $n = 2^k$, where k is any positive integer.
- q be any prime integer such that $q > 2$.
- m is fixed element in ring R_q, where $R_q = \frac{\mathbb{Z}[x]}{< f(x) >}$, where $f(x) = x^n + 1$.
- ψ is a discrete Gaussian Distribution over $\mathbb{Z}^n$ with small standard deviation say β.

The complete key exchange protocol based on RLWE problem is shown in Figure 2.3 and described as follows. First the public parameters n, q, ψ, β are generated and a fixed element m is sampled from ring R_q. After this, the following communication occurs between Alice and Bob

- Alice first chooses a secret key s_a by sampling from discrete Gaussian distribution ψ as $s_a \leftarrow Sample(\psi)$ and computes $p_a = ms_a + 2e_a$ where error e_a is also sampled from ψ as $e_a \leftarrow Sample(\psi)$. Alice then sends the computed p_a to Bob.
- Bob on receiving p_a, generates the secret key s_b as $s_b \leftarrow Sample(\psi)$ and computes $k_b = p_a + 2e_{b1}$ where error $e_{b1} \leftarrow Sample(\psi)$. Then, Bob computes the signal function output with the help of Hint function S() as $\sigma \leftarrow S(k_b)$. He then samples error term e_b as $e_b \leftarrow Sample(\psi)$ and computes $p_b = ms_b + 2e_b$. Finally, Bob sends (p_b, σ) to Alice and obtains the secret session key as $sk_b = E(k_b, \sigma)$.
- On receiving (p_b, σ) from Bob, Alice samples error e_{a1} as $e_{a1} \leftarrow Sample(\psi)$ and computes $k_a = s_a p_b + 2e_{a1}$. At the end, Alice computes her secret session key as $sk_b = E(k_a, \sigma)$.

For the correctness of the scheme i.e. for $sk_a = sk_b$, the necessary condition to hold are

$$8n\beta^2 \leq \frac{q}{4} - 2 \tag{2.2}$$

As seen from key exchange scheme (Figure 2.3), "Bob has to send a signal to Alice to indicate that the value of k_b is in the range close to $[-\lfloor\frac{q}{4}\rfloor, \lfloor\frac{q}{4}\rfloor]$ or not. This will make sure that the error terms in k_a *and* k_b do not give different outputs to generate a different secret session key. The main drawback of sending the signal value is the leakage of information to the adversary i.e. the information of the value of $s_a \times m \times s_b$ lies close to $[-\lfloor\frac{q}{4}\rfloor, \lfloor\frac{q}{4}\rfloor]$ or away

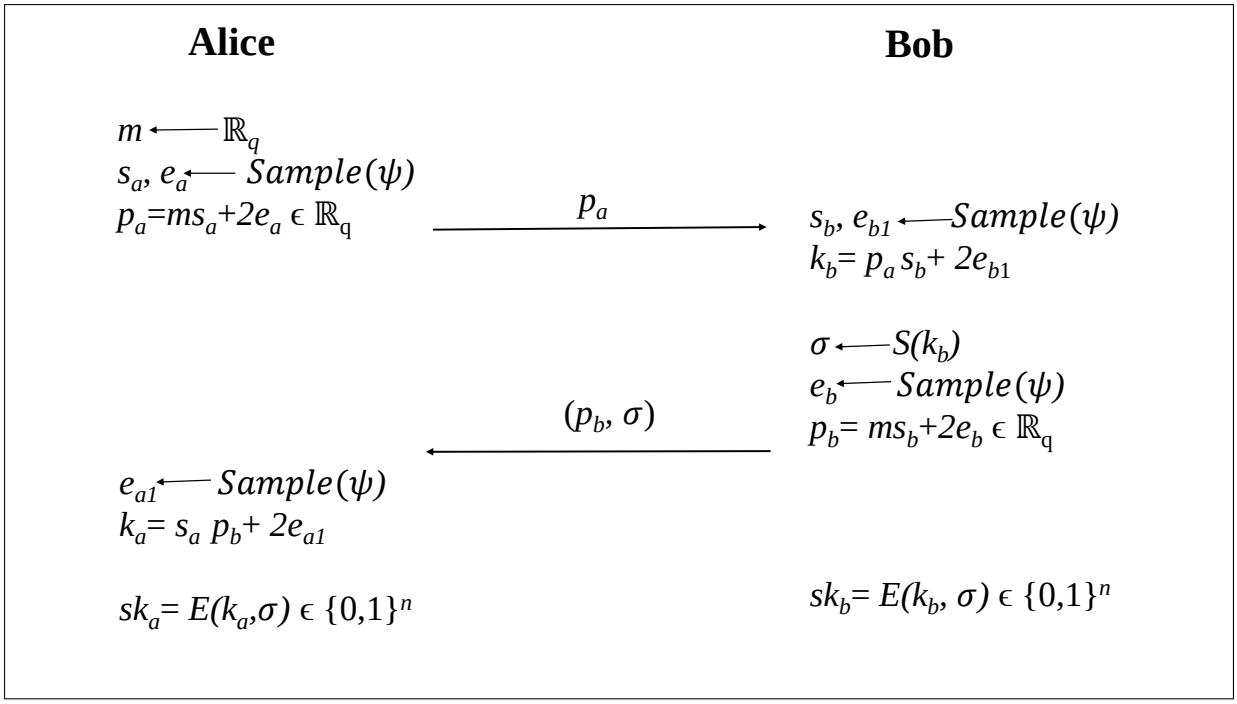


Figure 2.3: Key Exchange Scheme proposed by Ding et al. [2]

from this range get leaked. However, there will be no harm to the security of scheme as it can be showed that k_b is pseudo-random in R_q and the additional modulo 2 operation makes the secret session key uniform in R_2. Authors have compared the proposed scheme with key exchange schemes based on public key encryption dependent on RLWE problem" [2]. The two RLWE based schemes considered for comparison are Lyubashevsky et al.'s scheme [8] and NTRU variant by Stehle et al. scheme [44] for n-bit secrecy. It had been found that the proposed scheme performed better among the other schemes with public key size of $n \log q$, communication complexity of $(2n \log q + n)$ and computation complexity(given by the number of multiplication operation in the ring R_q) of 4.

2.3.1.1 Security Analysis of Ding et al.'s Scheme

The Ding et al.'s scheme is an attempt to provide the Diffie-Hellman version of key exchange protocol in post-quantum cryptography. Like the classical Diffie-Hellman counterpart, the proposed scheme does not support authentication, consistency and identity protection features. Moreover, the security of the scheme depends on the core hardness of RLWE problem and has proved to be secure against PPT adversary. The theorem stating the security of the scheme against PPT adversary is as follows

Theorem: The key exchange scheme proposed above is secure against PPT adversaries, if the RLWE assumption holds.

Despite the above security claims, recently proposed signal leakage attack [34] shows that the scheme is vulnerable to the leakage of signal function if RLWE keys are reused in the key ex-

change. The idea of this attack comes from the cryptanalysis of ring-LWE based key exchange with key share reuse by Fluhrer [36] in 2016. In the proposed attack, the attacker initiates multiple sessions with the honest party and analyzes the output of the signal function. By carefully framing the multiple queries against the honest party, the attacker can successfully recover the private key of the honest party. Finally, the introducers of leakage attack (Ding et al. [34]) conclude that the attack can also be possible against recently proposed New Hope scheme [31]. Therefore, the Ding et al.'s scheme [2] is not fit for use with existing Internet standard protocols like TLS where key reuse is a common practice to save the computational cost.

2.3.2 Review of Peikert Scheme [3]

As discussed in section 1.1.2, authentication [45] and consistency [46] are the core basic requirements of a key exchange protocol to provide a secure channel. One of the solutions to fulfill these two requirements is to use the digital signature as a means for authentication and MACing the identities of the parties involved to ensure consistency. SIGMA family [46] of key exchange protocols adopted the above approach of signature and MAC (SIGn-and-MAc) to design the authenticated Diffie-Hellman type key exchange protocols [47, 48]. The work in [46] explains the underlying design process and the various subtleties related to the design of the key exchange protocol of SIGMA family. According to authors in [46], different variants of SIGMA protocol (variants as given in section 5 of [46]) follow the common design core. While the above analysis of SIGMA protocols is an informal one, the formal analysis of SIGMA protocols is given in the work of Canetti and Krawczyk [27], where the security of SIGMA variants is proved using the complexity-theoretic model of security. Building on the same design process of SIGMA, Peikert [3] in 2014 proposed a key exchange protocol based on RLWE problem in lattice-based cryptography. He first proposed a Key Encapsulation Mechanism (KEM) for lattice-based cryptography which is IND-CPA (Indistinguishability under Chosen Plaintext Attack) secure and later uses this KEM to design an authenticated key exchange (AKE) protocol based on SIGMA model. The Key Encapsulation Mechanism (KEM) and authenticated key exchange (AKE) protocol of Piekert have described below.

i. **Key Encapsulation Mechanism (KEM)**: The KEM proposed by Peikert is based on RLWE problem which is proved to be secure by Regev et al. [8] against quantum attacks. One of the attractions of the KEM is its reconciliation technique characterized by low bandwidth signal function. This property of KEM is attained by replacing one of the two ring elements, say a polynomial of degree n, in the ciphertext with a binary string of dimension n. This replacement reduces the cipher-text length by a factor of two as compared to the previous schemes [[8,49]]. According to Peikert's [3], in previous schemes [[8, 49]], encryption is one of the alternatives

for asymmetric key encapsulation[1] but KEM is more beneficial as the key K is generated as an output of the sender's encapsulation algorithm that can run on the receiver's public key alone.

The parameters for KEM are as follows:

- Let m is a security parameter such that $m = 2^k$, where k is any positive integer.
- n is the degree of the minimal polynomial $x^n + 1$ of *'a'* where *'a'* is a fixed element of ring R_q and $n = \phi(m) = \frac{m}{2}$ for the above described m.
- q is an odd prime integer such that $gcd(q,m) = 1$ *or* $q \equiv 1 (mod\ m)$
- $R_q = \mathbb{Z}/q\mathbb{Z}[\zeta_m] \cong \frac{\mathbb{Z}_q[x]}{<\Phi_m(x)>}$
  Here, ζ_m is the primitive m^{th} root of unity and it is the root of $\Phi_m(x)$, where $\Phi_m(x)$ is the m^{th} cyclotomic polynomial $x^n + 1$ of degree n.
- ψ is the discretized error distribution over $\mathbb{Z}^n$.

In RLWE setting, it is noticeable that the modulus q is an odd prime number, therefore the whole KEM is described taking q as an odd prime. We have omitted the case when q takes even value, the case is trivial one and can be refereed from [3] for more information.

Table 2.5: Key Generation Algorithm

Key Generation
Input: (m, q, a, ψ)
Output: Private key s_s and public key pk_s
$s_s, e \leftarrow Sample(\psi)$
$pk_s = as_s + e$ where $pk_s \in R_q$

Table 2.5, 2.6 and 2.7 describe the key generation, key encapsulation and key decapsulation algorithm respectively in Peikert's scheme [3]. Here the subscript 's' denotes the sender's (initiator or first party) parameters while subscript 'r' denotes the receiver's (responder or second party) parameters. Sample(ψ) is the function that is used to sample the parameters from the discretized error distribution ψ *over* R_q. The key generation procedure takes the parameters m, q, a, ψ as input and outputs the secret key s_s and public key pk_s of the sender. The secret key and error term e are sampled from discrete Gaussian distribution ψ and combined with the fixed element 'a' of ring R_q to generate the public key pk_s of the sender.

[1]Asymmetric Key Encapsulation is different from KEM. It is also known as key transport technique where sender transmits a random chosen cryptographic key K encrypted with receiver's public key and therefore, the key can be only decrypted by the intended receiver. This key K is further used by symmetric algorithms for encryption/decryption of bulk data.

Table 2.6: Key Encapsulation Algorithm

KEM.Encapsulation
Input: public key pk_s
Output: encapsulation c, shared secret key sk_r
$s_r, e_1, e_2 \leftarrow Sample(\psi)$
$pk_r = as_r + e_1$
$v = pk_s s_r + e_2$
$\bar{v} \leftarrow dbl(v)$
$c = (pk_r, <\bar{v}>_{2q,2}) \in R_q \times R_2$
$sk_r = \lfloor \bar{v} \rceil_{2q,2} \in R_2$

The encapsulation algorithm shown in Table 2.6 takes public key pk_s of the sender as input to output the shared secret key sk_r of the receiver and encapsulated ciphertext c. The shared secret key will belong R_2 and ciphertext will lie in $R_q \times R_2$.

Table 2.7: Key Decapsulation Algorithm

KEM.Decapsulation
Input: encapsulated ciphertext c, private key s_s
Output: shared secret key sk_s
$c = (pk_r, <\bar{v}>_{2q,2})$
$d = <\bar{v}>_{2q,2}$
$w = pk_r s_s$
$sk_s = rec(2w, d) \in R_2$

The decapsulation algorithm as shown in Table 2.7 takes the ciphertext c and private key s_s as input to output the shared secret key sk_s. It has been asserted that the above KEM will generate a common secret key at both, the sender and the receiver i.e. $sk_s = sk_r$. By virtue of the smart reconciliation proposed by Peikert, two parties agree upon a common secret key instead of approximate and different values of the secret keys at both the ends. The parameter w of rec() function in decapsulation algorithm is approximately equal to that of v of encapsulation algorithm as:
$w = pk_r.s_s = (as_r + e_1)s_s = s_s as_r + s_s e_1 \simeq s_s as_r + s_r s_s + e_2 \simeq pk_r s_r + e_2 = v$

The definitions of the functions used in the instantaneous of the Peikert's KEM are as follows: As mentioned in note earlier, we will discuss all the functions related to KEM for odd q i.e. odd modulus. The randomness in the output of modulus rounding function (defined below) depends on the value of q. If q is an even integer, then the output of the modulus rounding function is unbiased and is completely random. On the other hand, for the odd q its output is biased and is, therefore, not suitable for cryptographic operations. To avoid this

bias, Peikert scaled up the parameters using a Randomized Doubling Function dbl() (described below) so that computations can be conducted modulo $2q$; this change introduces small amount of extra randomness in the mechanism.

- **Modulus Rounding Function**
  The modulus rounding function for odd q is defined as $\lfloor . \rceil_{2q,2} : \mathbb{Z}_{2q} \to \mathbb{Z}_2$ such that
  $\lfloor y \rceil_{2q,2} = \left\lfloor \frac{2}{2q} . y \right\rceil (mod\ 2) = \left\lfloor \frac{1}{q} . y \right\rceil (mod\ 2)$

- **Randomized Doubling Function dbl()**
  The randomized doubling function is given by $dbl : \mathbb{Z}_q \to \mathbb{Z}_{2q}$ such that for $y \in \mathbb{Z}_q$, $dbl(y) = \bar{y} = (2y - e^{'})\ mod\ 2q \in \mathbb{Z}_{2q}$ for some random error $e^{'} \in \mathbb{Z}$ to be uniformly random over modulo 2 and is independent of y. Here error $e^{'}$ is sampled from the set {-1, 0, 1} with probabilities of {-1, 0, 1} as *p(-1)=p(1)*=1/4 and *p(0)*=1/2. Another alternative of randomized doubling function *dbl()* is randomized rounding function (RandomizedRound()) given by Vikram singh [50] while instantiating the Peikert's scheme. This randomized rounding procedure produces the similar effect as that of randomized doubling function *dbl()* without additional computational overhead of mapping all elements of $\mathbb{Z}_q$ into $\mathbb{Z}_{2q}$. Therefore, Randomized rounding function is more efficient than randomized doubling function dbl().

- **Cross Rounding Function**
  Let us define the disjoint interval for odd modulus q as
  $I_0 = \{0, 1, ..., \left\lfloor \frac{2q}{4} \right\rceil - 1\}\ mod\ 2q = \{0, 1, ..., \left\lfloor \frac{q}{2} \right\rceil - 1\}\ mod\ 2q$
  $I_1 = \{-\left\lfloor \frac{2q}{4} \right\rceil, ..., -1\}\ mod\ 2q = \{-\left\lfloor \frac{q}{2} \right\rceil,$
  $..., -1\}\ mod\ 2q$
  The cross rounding function is defined as $< . >_{2q,2}$: $\mathbb{Z}_{2q} \to \mathbb{Z}_2$
  $$< v >_{2q,2} = \left\{ \begin{array}{ll} 0 & if\ v \in I_0 \cup (I_0 + q) \\ \\ 1 & if\ v \in I_1 \cup (I_0 + q) \end{array} \right\}$$

- **Reconciliation function**
  Defining the disjoint interval for odd modulus q as
  $I_0 = \{0, 1, ..., \left\lfloor \frac{2q}{4} \right\rceil - 1\}\ mod\ 2q = \{0, 1, ..., \left\lfloor \frac{q}{2} \right\rceil - 1\}\ mod\ 2q$
  $I_1 = \{-\left\lfloor \frac{2q}{4} \right\rceil, ..., -1\}\ mod\ 2q = \{-\left\lfloor \frac{q}{2} \right\rceil, ...,$
  $-1\}\ mod\ 2q$
  Let $E = [\frac{-2q}{8}, \frac{2q}{8}) \cap \mathbb{Z} = [\frac{-q}{4}, \frac{q}{4}) \cap \mathbb{Z}$ *and* $b \in \{0, 1\}$
  Now, before applying reconciliation function *rec()*,the parameters are scaled up using the

randomized doubling function *dbl()* as
$2w$=*dbl(w)*
Then, the reconciliation function is applied as
$rec : \mathbb{Z}_{2q} \times \mathbb{Z}_2 \rightarrow \mathbb{Z}_2$

$$rec(2w,b)= \left\{ \begin{array}{ll} 0 & if\ v \in I_b + E\ (mod\ 2q) \\ 1 & \text{otherwise} \end{array} \right\}$$

The above-discussed reconciliation function is used to compute the shared secret key sk_s, where the value of b provides reconciliation information to the *rec()* function.

The KEM's algorithm corresponding to key generation, encapsulation and decapsulation are shown in Table 2.5, 2.6, 2.7 respectively. Peikert's uses this KEM to design the SK-secure Authenticated Key Exchange (AKE) derived from CK-model [27] in post-specified peer setting. The complete AKE has been explained below.

ii. **SIGN-and-MAC Authenticated Key Exchange (SMAKE) :** The family of key exchange protocol that uses SIGN-and-MAC approach was proposed by Krawczyk [46]. Based on the same idea, Peikert [3] proposed the key exchange protocol with the replacement of Diffie-Hellman key agreement in SIGMA model by the above described KEM. This replacement is permissible as the core functionality of Diffie-Hellman key exchange is to generate the common shared secret key without providing additional security features like authentication, consistency in the key exchange protocol. The security of the IND-CPA secure KEM scheme is dependent on the well studied RLWE problem [8]. The successful execution of the key exchange protocol will output the session key k_0 at both the parties.
The parameters for the SMAKE key exchange protocol are given by

- Key encapsulation Mechanism (KEM) with key space κ
- Pseudorandom function $PRF : \kappa \times \{0,1\} \rightarrow \kappa^{'}$
- Digital Signature Scheme SIG with long term signing key x
- Message Authentication Code scheme with key space of $\kappa^{'}$ and message space of $\{0,1\}^*$

The description SMAKE key exchange scheme shown in Figure 2.4 is as follows:

(a) **Start Message (S $\rightarrow$ R): (sid, pk_s)**
The sender S is also known as the initiator or the first party with its identity as ID_s. Sender S instantiates the protocol by generating session identifier (*sid*), which must be distinct from all the previously chosen session identifiers in previous sessions instantiated with identity ID_s. Then, the KEM Key generation algorithm (as described in Table 2.5) is invoked with input parameters m, q, a, ψ to output the ephemeral private key s_s and

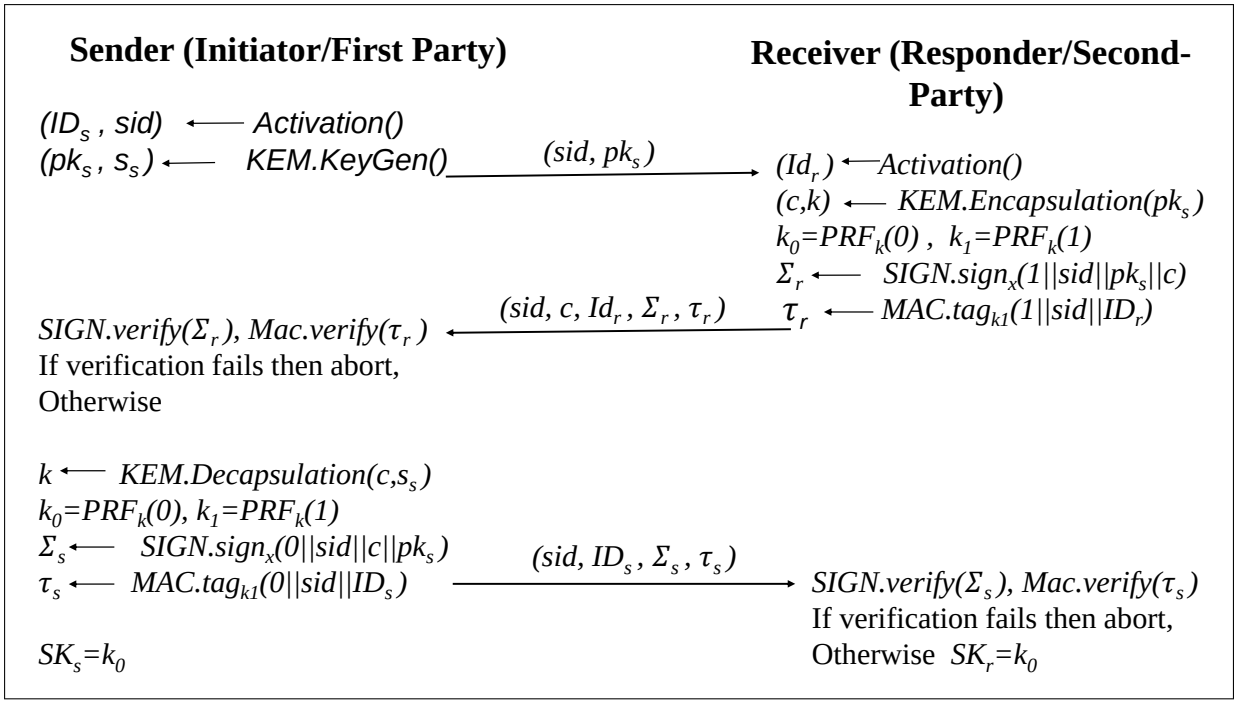


Figure 2.4: SIGN-and-MAC Authenticated Key Exchange Scheme (SMAKE) [3]

public key pk_s of the first party. After generating private and public key pair, the first party (sender) will send the pair of session identifier sid and public key pk_s to the second party (receiver).

(b) **Response Message** $R \rightarrow S : (sid, c, ID_r, \sum_r, \tau_r)$
The receiver R with identity ID_r is known as the responder or the second party. R receives the start message (sid, pk_s) and verifies whether the session identifier *sid* sent by S having identity ID_s was used earlier at receiver with identity ID_r. If *sid* is never used earlier then R activates the session *sid* as responder. After that, the KEM encapsulation algorithm (as described in Table 2.6) is invoked using public key pk_s of S to output the secret key k and encapsulated ciphertext c. This secret key is used to derive the shared session key k_0 using the pseudorandom function as $k_0 = PRF_k(0)$. The pseudorandom function is also used to generate the key for MAC.tag() function as $k_1 = PRF_{k1}(1)$ and then R compute the MAC.tag() as $\tau_r = MAC.tag_{k1}(1||sid$
$||ID_r)$. Also, there is a long term signing key x to compute the digital signature as $\sum_r = SIGN.sign_x(1||sid||pk_s||c)$. R then sends the above computed information i.e. $(sid, c, ID_r, \sum_r, \tau_r)$ to S.

(c) **Finish Message** $(S \rightarrow R) : (sid, ID_s, \sum_s, \tau_s)$
When S receives the response message from R, it verifies the digital signature and the MAC tag and if any of these fails to match, the connection is aborted. Otherwise, S calls the decapsulation algorithm (as described in table 2.7) on encapsulated ciphertext

c using the secret key s_s to output the shared secret key k. This shared secret key k is used to generate the shared session key k_0 and k_1 as described in above paragraph using pseudorandom function PRF(). Similarly, digital signature and MAC tag is computed to send in finish message as
$(sid, ID_s, \sum_s, \tau_s)$.

(d) **Completion at receiver**
After receiving the finish message from S the receiver R verifies the digital signature and MAC tag and if any of these verification fails, the connection is aborted. Otherwise, both the parties have the common shared session key k_0 in the key space $\kappa^{'}$ at the completion of the protocol.

2.3.2.1 Security Analysis of Peikert's Scheme

The security of the Peikert key exchange protocol is proved in post-specified peer settings using the notion of SK-security (See details of post-specified peer setting with SK-security in [27]). The SK-security requires two properties to hold by a key exchange protocol. These properties are as follows: (1) When two uncorrupted parties with identity ID_s and ID_r complete a key exchange protocol with the matching sessions[2] , then both the parties should output the same session key. This property ensures the correctness of the protocol. (2) There is no polynomial time attacker that can distinguish the real response from a random response using a test session query. The second property will ensure the secrecy of the key exchange protocol. The theorem proving the SK-security of Peikert's key exchange scheme is as follows

Theorem: Assuming that digital signature, MAC are unforged under chosen message attack (CMA), PRF is secure pseudorandom function and KEM is IND-CPA secure, then the proposed key exchange protocol is SK-secure in the post-specified peer model of [27].

The protocol inherits the properties of authentication [51, 52] and consistency from the Krawczyk's SIGMA family of protocols [46]. The digital signature is used for providing the authentication while MAC is used for providing the consistency among the parties involved in the key exchange. The SK-security of the scheme relies on the security of KEM, which is assumed to be IND-CPA secure in the above-specified theorem. Bos et al. [28] instantiate the Peikert's scheme by integrating with TLS protocol and recently, it has been found that the scheme is vulnerable to signal leakage attack proposed by Ding et al. [34]. In the resumption mode of TLS protocol where the private/public keys are reused, the attacker can successfully recover the secret keys of the honest user by initiating multiple sessions with it. The active

[2]Details of matching session and test session are given in section 2.1 of [27]

secure KEM proposed by Peikert is resistant to this attack by use of Fujisaki-Okamoto transformation [53] but this transformation is computationally very costly. Thus, a new approach is required to resist this attack that should be computationally efficient.

2.4 Literature survey findings

In this section, we provide the insights of this review work. The following are the implications of this work:

- There are two basic instantaneous of the reconciliation technique proposed by Ding et al. [2] and Peikert [3]. All the reconciliation based key exchange protocols proposed in lattice cryptography adopted the reconciliation technique of Ding et al. or Peikert.
- The reconciliation technique of Ding et al. [2] is more efficient than Peikert's reconciliation technique.
- While the schemes like that of Zhang [17] adopted the reconciliation technique of Ding et al. [2] to propose the secure key exchange and Bos et al. [28] adopted the Peikert reconciliation technique [3]. Likewise their are others schemes which adopted the reconciliation technique of the above two schemes.
- It has been found that all these schemes are vulnerable to different types of attacks like Zhang et al. [17] scheme is vulnerable to small field attack [43] and Bos et al. [28] scheme with public/private key reuse is vulnerable to signal leakage attack [34]. Both of these attacks allow an attacker to obtain the secret session key shared/agreed between the honest parties.
- The cause of signal leakage attack is the reuse of public/private keys. The adversary will exploit this vulnerability by sending manipulated queries to the honest user and then analyzing the signal changes to extract the value of secret s of the honest party.
- The signal leakage attack can also be adapted to New Hope [31] scheme, whose reconciliation mechanism is extension of Peikert's error reconciliation mechanism.
- The basic scheme given by Ding et al. [2] suffers from signal leakage attack. Same is true for the basic scheme given by Peikert [3]. The only scheme available is given by Zhang et al. [17]; it resists signal leakage attack but is susceptible to small field attack. Since the design of any reconciliation based key exchange scheme is inherited from these basic schemes (so far from schemes in [2,3]), all the proposed reconciliation schemes are bad omened with these two attacks. Very recently proposed Gao et al.'s scheme [24] is successful in combating both of these attacks but it lacks any kind of security proof. Instead of any security model the author used the randomness test of NIST statistical test suite [25] in support of the security of their scheme.

- Security of all the reconciliation based key exchange schemes in lattice-based cryptography has been justified using classical security models such as Random Oracle Model (ROM) is used by Ding et al. [21]. However, proofs from classical security models are inefficient in establishing/evaluating the security of the key exchange schemes designed for quantum world against quantum adversary. For example, Ding et al. [21] poses to evaluate the security of their proposed scheme in Quantum Oracle Model(QROM) in the future scope of their work.

2.5 Research Gaps

After the comprehensive literature survey, the following research gaps have been identified:

- **The cryptographic schemes give only theoretical explanation of the schemes in post-quantum setting without specifying the complete details:** Some of the proposed cryptographic schemes, presented the theoretical idea of their work and did not give the complete details of the parameter sizes along with hashing technique used. The detail description is required to validate the correctness and security of the proposed scheme [17].

- **Lack of comparison between different quantum resistive cryptographic schemes:** The lattice based cryptography is used to develop the public-key cryptographic schemes like key exchange protocols, digital signatures and encryption techniques. For selecting an efficient cryptographic scheme for quantum computing, there is need to compare the different cryptographic schemes on common basis [21].

- **The key exchange protocols that are designed using RLWE problem are not secure if their public/private keys are reused:** Key reuse is an important feature that saves the computation and communication overhead and is vastly employed in a majority of TLS connections. Despite its overhead benefit, this feature generate the security issue for the RLWE-based key exchange protocols. These key exchange protocols are subject to signal leakage and key mismatch attacks due to key reuse. The signal leakage attack is the most serious of these, and most RLWE-based key exchange protocols are vulnerable to it [13].

- **Flaw in design methodology for key exchange protocols:** The design of a key exchange protocol is the major issue that decides its efficiency and security. It has been found that a design flaw can lead to an inefficient scheme [17] with different security vulnerabilities. So, there is a need for better design models for a key exchange scheme for different applications. Different design models are prevalent in modern cryptography that are used to design lattice-based key exchange schemes, like SIGMA model [46], CK-model [54], BR-model [20] etc. Now, adopting these design models for lattice-based key exchange schemes and proving their security is an open problem.

- **Absence of provable security of the key exchange protocols:** The security analysis is one of the crucial aspects of designing a key exchange protocol that provides a secure channel. Particularly, instead of heuristic justifications [24], researchers prefer provable security to prove the security of the scheme.
- **Lack of standardization of the key exchange protocols:** There is no standardization of key exchange protocols, so as to find the key lengths of the schemes that will provide particular level of security [55].

2.6 Objectives of the Research Work

After analysis of existing proposals, and the research gaps following objectives are proposed in the proposal.

1. To study and explore the existing cryptographic schemes used in quantum computing.
2. To design and propose an efficient cryptographic technique.
3. To validate the proposed technique using different types of attack.

2.7 Summary

This chapter presents the detailed analysis and classification of LWE/RLWE-based key exchange schemes in lattice-based cryptography. From the detailed analysis, it has been found that all the key exchange schemes are vulnerable to either signal leakage attack, key mismatch attack, or small field attack. Among these, signal leakage attack is the most severe to which almost all the schemes are affected. Therefore, there is a need to design an efficient key exchange technique that can resist signal leakage attack. So, in the next chapter the lattice-based anonymous password authenticated key exchange (LBA-PAKE) for mobile devices has been proposed. The proposed protocol would be able to resist signal leakage attack and will provide key reuse, anonymity, and perfect forward secrecy.

Chapter 3

LBA-PAKE: Lattice-based Anonymous Password Authenticated Key Exchange for mobile devices

The recent advancement in quantum computers generates the threat alarm of breaking the security of key exchange protocols which is based on discrete logarithmic or prime factorization problem in polynomial time. Hence, motivated to develop the key exchange protocol that is secure in the post-quantum era, Feng et al. proposed an anonymous authenticated key exchange protocol for mobile devices in a post-quantum world. Although the protocol is simple, elegant, and efficient for mobile devices, but it is vulnerable to different types of attacks.

Therefore, in this chapter the cryptanalysis of Feng et al.'s protocol has been performed and it has been proved that the protocol is vulnerable to signal leakage attack, spoofing attack, manipulation-based attacks, and user anonymity violation attack. Further, to overcome the above security weaknesses, the LBA-PAKE protocol for mobile devices has been proposed.

The proposed LBA-PAKE protocol resists the above attacks, in addition to it, it also provides key reuse, anonymity, and perfect forward secrecy features. Also, the formal security analysis of the proposed LBA-PAKE protocol has been done using the widely adopted Real-Or-Random (ROR) model. In the end, the proposed LBA-PAKE protocol and Feng et al.'s protocol have been implemented on the common mobile-server platform for the comparative performance analysis. The experimental results show that the proposed LBA-PAKE protocol is as efficient as Feng et al.'s protocol with an extra shield of security.

3.1 Background

In 2015, Kirkwood et al. [56] first revealed that the RLWE based reconciliation schemes, that reuses the public/private key pairs may leak the private key of the honest party to the adversary. Further, they suggested that, to secure the schemes against this vulnerability, there

is a dire need for public-key validation. The authors do not know the direct way of public key validation in lattice-based cryptography and, thus, suggested the indirect key validation through Fujisaki and Okamoto transformation [53]. Although this work warns the users against the reuse of public/private key pairs but they do not provide a concrete description of how to exploit this vulnerability. In 2016, Fluhrer [36] showed how RLWE based key exchange protocols that reuse public/private keys can be broken by the information derived from the two-party agreement. The idea behind the attack is to derive information about the private key of the honest party from the match or mismatch of the final shared key. However, the attack will not work against the key agreement protocols where the final shared keys have been derived from LSB (Least Significant Bits) of the approximately equal keys computed by both parties. Thus, the attack does not work against Ding et al.'s [2] key agreement protocol.

Taking ideas from the above attacks, Ding et al. [34] in 2017 proposed another attack against the RLWE-based reconciliation schemes. The attack is termed as Signal Leakage Attack (SLA). Similar to the above attacks, in SLA attack, an attacker will take advantage of reused public/private key pairs to breach the security of the scheme. The attacker will instantiate the multiple sessions with the honest party and will analyze the output of the signal function to derive the secret key, say, s of the honest party. It has been shown that signal function tends to leak the information that might help to recover the secret key s of the honest party. The attack will also work against the schemes where the final shared keys have been derived from LSB (Least Significant Bits) of the approximately equal keys computed by both parties. Therefore, the attack will work against the Ding et al. [2] key agreement protocol. They also mention that the complexity of the SLA attack to find the secret key s of the honest party is $2 * q$ (where q is an odd prime number as described in Table 3.1). As per the related work discussed above, the recently proposed Feng et al.'s [4] ideal lattice-based anonymous authentication protocol for mobile devices is vulnerable to the signal leakage attack (SLA), in addition to it, it is also suffering from spoofing attack, manipulation based attacks, and user anonymity violation attack. Hence, in this work, an RLWE-based key exchange protocol for mobile devices, termed as LBA-PAKE has been proposed that is secure against all of the above-discussed attacks.

3.2 Weakness of Feng et al.'s [4] protocol

Table 3.1 depicts the notations of Feng et al.'s lattice-based anonymous authenticated key exchange protocol. The user registration and mutual authentication phase are given in Table 3.2 and 3.3 respectively. For a detailed description of the protocol, refer to the article [4]. This section presents the cryptanalysis of Feng et al.'s protocol. Thorough cryptanalysis of Feng et al.'s protocol suggests that the protocol is vulnerable against signal leakage attack (SLA), spoofing attack, manipulation-based attacks which includes Trojan horse attack and device stolen attack and user anonymity violation attack. The detailed description of the above attacks is given below.

Table 3.1: Feng et al.'s protocol parameters

Feng et al.'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 postive 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
p_i: master public key of server SP_j s.t. $p_i = a.s + 2.e$
where $s, e \leftarrow \chi_\beta$
s: master secret key of server SP_j
$A \mid\mid B$: String A concatenated with string B
$H_0 : \{0,1\}^* \rightarrow \{0,1\}^l$ Secure hash function that
output the fixed length string of length l
$\{n, q, \chi_\beta, a, p_i, H_0()\}$: System or Public parameters
s: Confidential secret key of the server

Table 3.2: Feng et al.'s mobile user registration phase

Mobile User MU_i	Server SP_j
Choose id_i	
$\xrightarrow{\{id_i\}}$	
	$s_i = H_0(id_i \mid\mid s)$
	Generate pw_i^*
	$d_i^* = s_i \oplus H_0(id_i \mid\mid pw_i^*)$
	$\xleftarrow{\{pw_i^*, d_i^*\}}$
Choose pw_i	
$d_i = d_i^* \oplus H_0(id_i \mid\mid pw_i^*) \oplus H_0(id_i \mid\mid pw_i)$	
$v_i = H_0(id_i \mid\mid pw_i \mid\mid d_i)$	
Stores $\{d_i, v_i\}$ into his/her mobile device	

3.2.1 Signal leakage attack (SLA)

While applying SLA to Feng et al.'s key exchange protocol, one can observe from Table 3.1, there is a master public key p_i of the server SP_j such that $p_i = a.s + 2.e$ where a is random element in R_q and s, e are random sample from discrete Gaussian distribution χ_β. The master public key p_i has corresponding master secret key s and both are reused by the server in each instantaneous of the protocol.

Table 3.3: Feng et al.'s mutual authentication phase

Mobile User MU_i	Server SP_j
Inputs id_i and pw_i	
MD_i: Check $v_i \stackrel{?}{=} H_0(id_i\|\|pw_i\|\|d_i)$	
MD_i: Generate $r_i, f_i \leftarrow \chi_\beta$	
MD_i: $x_i = a.r_i + 2.f_i$	
	$\xrightarrow{\{x_i\}}$
	Generate $r_s, f_s \leftarrow \chi_\beta$
	Compute: $x_s = a.r_s + 2.f_s$
	$k_s = x_i.s$
	$w_s = Cha(k_s)$
	$\sigma_s = Mod_2(k_s, w_s)$
	$\alpha_s = H_0(x_i\|\|x_s\|\|w_s\|\|\sigma_s)$
	$\xleftarrow{\{x_s, w_s, \alpha_s\}}$
MD_i: computes $k_s^{'} = r_i.p_i$	
MD_i: $\sigma_s^{'} = Mod_2(k_s^{'}, w_s)$	
MD_i: Check $\alpha_s \stackrel{?}{=} H_0(x_i\|\|x_s\|\|w_s\|\|\sigma_s^{'})$	
MD_i: Compute: $s_i = d_i \oplus h(id_i\|\|pw_i)$	
MD_i: $k_i = x_s.r_i$	
MD_i: $w_i = Cha(k_i)$	
MD_i: $\sigma_i = Mod_2(k_i, w_i)$	
MD_i: $aid_i = id_i \oplus H_0(x_i\|\|w_i\|\|\sigma_i\|\|x_s\|\|w_s\|\|\sigma_s^{'})$	
MD_i: $\alpha_i = H_0(id_i\|\|s_i\|\|aid_i\|\|x_i\|\|w_i\|\|\sigma_i\|\|x_s\|\|w_s\|\|\sigma_s^{'}\|\|\alpha_s)$	
MD_i: $sk_i = H_0(id_i\|\|aid_i\|\|x_i\|\|w_i\|\|\sigma_i\|\|\alpha_i\|\|x_s\|\|w_s\|\|\sigma_s^{'}\|\|\alpha_s)$	
	$\xrightarrow{\{aid_i, w_i, \alpha_i\}}$
	$k_i^{'} = x_i.r_s$
	$\sigma_i^{'} = Mod_2(k_i^{'}, w_i)$
	$id_i = aid_i \oplus H_0(x_i\|\|w_i\|\|\sigma_i^{'}\|\|x_s\|\|w_s\|\|\sigma_s)$
	$s_i = H_0(id_i\|\|s)$
	$\alpha_i^{'} = H_0(id_i\|\|s_i\|\|aid_i\|\|x_i\|\|w_i\|\|\sigma_i^{'}\|\|x_s\|\|w_s\|\|\sigma_s\|\|\alpha_s)$
	Check $\alpha_i^{'} \stackrel{?}{=} \alpha_i$, If not Abort
	Otherwise,
	compute: $sk_s = H_0(id_i\|\|aid_i\|\|x_i\|\|w_i\|\|\sigma_i^{'}\|\|\alpha_i\|\|x_s\|\|w_s\|\|\sigma_s\|\|\alpha_s)$

Theorem 1. The mobile user (MU_i) which acts as an adversary $\mathscr{A}$ can recover the reused master secret key s of the server SP_j with overwhelming probability in polynomial time.

Proof: To carry out the SLA attack, an adversary $\mathscr{A}$ plays the role of valid mobile user (MU_i). The adversary which acts as a valid mobile user MU_i, generates its public key x_i by choosing its secret key $r_i = 0$ and error term $f_i = 1$ in R_q. Thus, the public key x_i is computed as $x_i = a.0 + k.1 = k$. The deviated public key x_i is sent to the server (SP_j) on a public channel. The server on receiving the public key x_i computes k_s as $k_s = x_i.s = k.s$. The k_s is utilized as input of signal function Cha() to generate the signal function output as $w_s = Cha(k_s) =$

$Cha(x_i.s) = Cha(k.s)$. In particular, $w_s[i] = Cha(k_s[i]) = Cha(x_i.s[i]) = Cha(k.s[i])$, where k loops from 0 to $q-1$. Thus, $w_s[i]$ can be used to recover the master secret key $s[i]$ of the server, as the value of signal $w_s[i]$ flips exactly $2*s[i]$ times when k loops from 0 to $q-1$. Therefore, by initiating multiple sessions with server using public key x_i generated by looping k from 0 to $q-1$, adversary MU_i can recover the master secret key s of the server upto $\pm$ sign based on the number of times the signal $w_s[i]$ changes. Further, $\pm$ sign of each of the coefficient of $s[i]$ can be resolved by further initiating q number of sessions with public key $x_i = (x+1)k$, where k loops from 0 to $q-1$. Hence, the total number of key exchange sessions required to recover the secret key s of the server is $2*q$ [34]. The complexity of attack can further be reduced to $q+c$ by using [13] approach, where c is a constant.

In the above approach, the adversary MU_i chooses its public key x_i as $x_i = k$, which is constant polynomial in R_q. The victim SP_j can defend itself against the above attack by verifying the public key x_i of the dishonest party MU_i and checking whether x_i is a constant polynomial in R_q or not. The honest party SP_j will abort the session if the public key of user MU_i is a constant polynomial in R_q. To overcome this, the adversary will modify the above attack to bypass the verification of its public key x_i by generating it in a way that it is indistinguishable to the honest party SP_j. All other steps are same as described in the above simplified SLA attack (refer to section 4.4 of [34] for more details of the extended SLA attack).

3.2.2 Spoofing attack

After carefully analyzing the Feng et al.'s protocol, we found that the security of the scheme is largely dependent on the master secret key s of the server. If the adversary somehow gets the master secret key s of the server, then the adversary can easily masquerade as a legal server to mobile users and thus, leads to the spoofing attack. Thus, we prove the following theorem:

Theorem 2. If the adversary gets the master secret key s of the server SP_j, then the adversary $\mathscr{A}$ can successfully masquerade as a valid service provider SP_j (i.e. as a valid server) to the mobile users.

Proof: Here, the mobile user MU_i and server SP_j act as an honest party, while the adversary $\mathscr{A}$ is the man-in-the-middle between the mobile user MU_i and the legal server SP_j. Also, the adversary $\mathscr{A}$ holds the long term master secret key s of the legal server SP_j and pretends to be valid server SP_j to the mobile user MU_i. The mobile user (MU_i) authenticate the server (SP_j) when MU_i checks that $\alpha_s \stackrel{?}{=} H_0(x_i||x_s||w_s||\sigma_s^{'})$. Now, if the adversary $\mathscr{A}$ can somehow bypass this authentication step, then he can successfully masquerade as a legal server. For this to happen, adversary $\mathscr{A}$ requires to compute the correct α_s. Now, $\alpha_s = H_0(x_i||x_s||w_s||\sigma_s)$. Therefore, the adversary $\mathscr{A}$ requires the parameters $\{x_i, x_s, w_s, \sigma_s\}$. As an active adversary, $\mathscr{A}$ can intercept the x_i, w_s parameter from the network and generate the parameter x_s as x_s^* as follows:

$x_s^* = a.r_s^* + 2f_s^*$, where $r_s^*, f_s^* \leftarrow \chi_\beta$ and lastly, compute the σ_s from (k_s and w_s) as $k_s = x_i * s$, $w_s = Cha(k_s)$ and $\sigma_s = Mod_2(k_s, w_s)$. Here, it is assumed that the master secret key s is known to the adversary $\mathscr{A}$ by SLA attack or any other attack. Therefore, this proves the theorem.

3.2.3 Manipulation-based attacks

Feng et al.'s lattice-based anonymous authentication protocol provides anonymity of the mobile users (MU_i) by authenticating the users locally i.e. at mobile user end through an application (App) installed on the mobile user device. For this, the protocol stores the parameters $\{d_i, v_i\}$ locally on the mobile device during the registration phase (see Table 3.2) and these parameters are responsible for authenticating the mobile users. Now, the idea of storing the sensitive parameters on the mobile device creates a security issue for mobile users. The adversary can bypass the authentication by modifying, delete, or steal the values of these parameters from a mobile device, and thus, the protocol is vulnerable to manipulation-based attacks. Thus, the cost of providing anonymity by authenticating the mobile users locally is very expensive and is at the expense of the security of the protocol. Hence, Feng et al. protocol is vulnerable to two manipulation-based attacks. First one Trojan horse attack and second is the device stolen attack. We discuss them by one by one as follows:

3.2.3.1 Trojan Horse attack

A Trojan horse is a malicious code that looks like a genuine program but it is designed by an adversary to control the honest user device. A Trojan may steal, modify, or send the honest user information back to an adversary. The Feng. et al. protocol stores the parameters $\{d_i, v_i\}$ into the mobile device for authentication of mobile user MU_i locally through the installed mobile App. Nowadays, with much storage and network resources available, mobile users usually install a large number of mobile Apps for their different needs. Some of the Apps has embedded Trojan horse inside it. Therefore, If the mobile user installs this infected App in his/her mobile device with Admin privilege, then malicious Trojan horse code also gets executed with Admin privilege. This Trojan horse code can modify, delete or send these stored parameters $\{d_i, v_i\}$ back to adversary. Thus, the adversary can then use these modified parameters to authenticate itself as a valid mobile user.

3.2.3.2 Device stolen attack

The second manipulation-based attack is the device stolen attack. The adversary can steal the mobile device of the honest mobile user and then, modify the parameters $\{(d_i, v_i)\}$ stored in the mobile device to authenticate itself as a valid mobile user. Thus, an adversary can masquerade as a valid mobile user to the remote server.

3.2.4 User anonymity violation attack

Feng et al.'s protocol aims to conceal the identity of the mobile user MU_i. To achieve this purpose, the protocol does not send the mobile user identity over the public channel. But after carefully analyzing the protocol, we find out that an active adversary can successfully determine the identity of an honest mobile user. For this, an adversary only needs to know the master secret key s of the server and he/she can recover the exact identity of the mobile user MU_i. Thus, we prove the following theorem:

Theorem 3. An active adversary $\mathscr{A}$ having the master secret key s of the server SP_j can recover the exact identity id_i of the honest mobile user MU_i.
Proof: Let $\mathscr{A}$ be an active adversary that intercepts the network traffic between an honest mobile user MU_i and server SP_j. Now, the adversary $\mathscr{A}$ will act as an active man-in-the-middle between the mobile user MU_i and the server SP_j.
The adversary aims to obtain id_i of the mobile user MU_i. Now, the identity of the mobile user is obtained by solving the following equation (see Table 3.3):

$$id_i = aid_i \oplus H_0(x_i||w_i||\sigma_i^{'}||x_s||w_s||\sigma_s) \tag{3.1}$$

To solve the above equation, adversary will first intercept the public key x_i from the mobile user MU_i. The adversary $\mathscr{A}$ will allow this value to pass to the server SP_j as it is. The adversary $\mathscr{A}$ will again intercept the value, $\{x_s, w_s, \alpha_s\}$ sent from the server. $\mathscr{A}$ will pass the w_s as it is to mobile user but modify the value of x_s and corresponding value of α_s. The modified value are labeled with *. Therefore, the adversary $\mathscr{A}$ will modify x_s as x_s^* and calculate x_s^* as:

$$x_s^* = a.r_s^* + 2.f_s^*, such\ that\ r_s^*, f_s^* \leftarrow \chi_\beta \tag{3.2}$$

Therefore, $\alpha_s^* = H_0(x_i||x_s^*||w_s||\sigma_s)$, here σ_s is computed from (k_s and w_s) as $k_s = x_i * s$, $w_s = Cha(k_s)$ and $\sigma_s = Mod_2(k_s, w_s)$. Here, it is assumed that adversary has the master secret key s of the server.

Finally, adversary $\mathscr{A}$ will capture the $\{aid_i, w_i, \alpha_i\}$ sent from mobile user MU_i. From this point of time, we assume that adversary $\mathscr{A}$ has everything to recover the identity id_i of the mobile user MU_i.

Again, from the equation 3.1, it can be seen that adversary $\mathscr{A}$ require $aid_i, x_i, w_i, \sigma_i^{'}, x_s, w_s, \sigma_s$ to solve right hand of equation 3.1 and to recover the exact value of identity id_i of mobile user MU_i. Now, continuing with the above scenario, we can see that adversary $\mathscr{A}$ intercept the value of aid_i, x_i, w_i, w_s from the public channel, while x_s is modified as x_s^* by the adversary. Now, only σ_s and $\sigma_i^{'}$ are required by the adversary to complete the attack.
The parameter $\sigma_i^{'}$ can be recovered by adversary by computing $k_i^{'}$ as $k_i^{'} = x_i.r_s^*$, here r_s^* is the

Table 3.4: LBA-PAKE mobile user registration phase

Mobile User MU_i	Server SP_j
Choose id_i and pw_i	
$\xrightarrow{\{id_i\}}$	
	Generate a random number r_j, pseudo-identity $psid_i$ and pseudo-secret key s^* such that $s^* \leftarrow \chi_\beta$
	$s_i = H_0(id_i \| s^*)$, s^* is server's pseudo-secret key
	Computes $d_i^* = s_i \oplus H_0(id_i \| r_j)$
	$\xleftarrow{\{r_j, d_i^*, psid_i\}}$
Computes $d_i = d_i^* \oplus H_0(id_i \| r_j) \oplus H_0(id_i \| pw_i)$	
Computes $v_i = H_0(id_i \| pw_i \| d_i)$	
Computes $PW_i = H_1(v_i) \in R_q$	
Stores $\{psid_i, d_i\}$ into his/her mobile device	
$\xrightarrow{\{PW_i\}}$	
	Stores $\{psid_i,\ PW_i$ *and* $s^*\}$ in server's secure database

modified secret key generated while calculating modified public key x_s^* as shown in equation 3.2. From the $k_i^{'}$, the $\sigma_i^{'}$ can be easily computed as $\sigma_i^{'} = Mod_2(k_i^{'}, w_i)$, here w_i is already intercepted by the adversary. Now, the only parameter left to be computed is σ_s. The σ_s is already computed above during the computation of α_s^*. Thus, the adversary $\mathscr{A}$ can now recover the exact identity id_i of the mobile user MU_i by using equation 3.1. Thus, this ends the required proof.

3.3 The Proposed Protocol

In this section, the improved protocol is presented, which includes the different phases along with the correctness of the protocol. As discussed in the previous section, Feng et al.'s protocol is vulnerable to different attacks, and signal leakage attack (SLA) is the most severe among all the attacks. The solution of the SLA attack is the public key validation, therefore, the idea of direct public key validation of Ding et al. [57] has been used to design the proposed LBA-PAKE protocol. The proposed LBA-PAKE protocol is described as follows:

3.3.1 Setup Phase

The server SP_j executes the following steps to complete the system initialization:

- SP_j chooses the security parameter n such that n is power of two, and also the value of n is chosen according to sensitivity of the application for which protocol is to be used.
- SP_j chooses an odd prime number q such that $q \ mod \ 2n = 1$, a discrete Gaussian distribution χ_β over R_q with standard deviation β and random element a such that $a \in R_q$.

- SP_j randomly samples $s, e \leftarrow \chi_\beta$ and calculates the master public key as $p_i = a.s + 2e$, where s is the master secret key of the server.

- SP_j selects three one-way hash functions. They are $H_0(), H_1()$ *and* $H_2()$. They are described as follows:

$$H_0 : \{0,1\}^* \rightarrow \{0,1\}^l \tag{3.3}$$

Here, $H_0()$ is the one-way hash function whose input is the string of variable length and output is hash of fixed length say l. Typically, it is implement using SHA-3 family of hash function.

$$H_1 : \{0,1\}^l \rightarrow R_q \tag{3.4}$$

$H_1()$ is the secure one-way hash function whose input is a string of fixed length l and whose output is an element in R_q.

$$H_2 : \{0,1\}^* \rightarrow \chi_\beta \tag{3.5}$$

$H_2()$ is the secure one-way hash function whose input is an element in R_q and output is sampled from χ_β distribution. The $H_2()$ hash function is modeled as a random oracle H_2 (in a similar way as $H_1(x)$ is modeled as random oracle H_1 in [57]) and typically implemented using the combination of $H_0()$ hash function and an algorithm P that samples the element from χ_β. The fixed-length output generated from the $H_0()$ is used as a seed for the P algorithm to generate the random sample from χ_β distribution.

- Finally, Server SP_j publishes the system parameters $\{n, q, \chi_\beta, a, p_i, H_0(), H_1(), H_2()\}$ publicly and keeps master secret key s as confidential.

3.3.2 Registration Phase

After the Setup phase, registration of the mobile users is commenced in the following manner (see Table 3.4). The messages transferred in this phase are through the secure channel.

1. First, the mobile user MU_i chooses his id id_i, password pw_i and sends id_i to the server SP_j.

2. Server SP_j on receiving id_i from MU_i generates the random number r_j along with pseudo-identity $psid_i$ and pseudo-secret key s^* as $s^* \leftarrow \chi_\beta$. Server SP_j will then compute s_i as $s_i = H_0(id_i||s^*)$and $d_i^* = s_i \oplus H_0(id_i||r_j)$. SP_j will send $\{r_j, d_i^*, psid_i\}$ to the MU_i.

3. MU_i after receiving $\{r_j, d_i^*, psid_i\}$ from SP_j computes $d_i = d_i^* \oplus H_0(id_i||r_j) \oplus H_0(id_i||pw_i)$,

$v_i = H_0(id_i||pw_i||d_i)$ and $PW_i = H_1(v_i)$. MU_i will store parameters $\{psid_i, d_i\}$ in his mobile device and send $\{PW_i\}$ to the server SP_j.

4. Finally, the server SP_j will store the parameters $\{psid_i, PW_i, s^*\}$ in server's secure database.

3.3.3 Login & Authentication Phase

In this phase, the mobile user MU_i and server SP_j authenticate each other and establish a common session key (see Table 3.5). All the messages in this phase are transferred in plain-text through the insecure public channel. This phase has been described below as:

1. The mobile user MU_i initiates the session by inputing the identity id_i and password pw_i. The parameters $\{psid_i, d_i\}$ are fetched from the mobile device for further computation of other parameters. The mobile user MU_i computes v_i as $v_i = H_0(id_i||pw_i||d_i)$, $PW_i = H_1(v_i)$ and public key x_i as $x_i = a.r_i + 2.f_i$, where a is the random element in R_q and $r_i, f_i \leftarrow \chi_\beta$. After that MU_i computes x_i^* as $x_i^* = x_i + PW_i$ and sends the parameters $\{psid_i, x_i^*\}$ to the server SP_j.

2. Server SP_j after receiving the parameters $\{psid_i, x_i^*\}$ from the mobile user MU_i, fetch PW_i *and* s^* corresponding to $psid_i$ from the server's secure database. The server SP_j then recovers the public key x_i as $x_i = x_i^* - PW_i$ and computes its ephemeral public key x_s as $x_s = a.r_s + 2.f_s$, where $r_s, f_s \leftarrow \chi_\beta$. SP_j then calculates y, z as $y = H_2(x_i)$, $z = H_2(x_s)$, computes $\bar{x_i} = x_i + a.y$ and k_s as $k_s = \bar{x_i}.(s+z) + 2.f_s^{'}$ such that $f_s^{'} \leftarrow \chi_\beta$. Finally, SP_j computes $w_s = Cha(k_s)$, $\sigma_s = Mod_2(k_s, w_s)$, $\alpha_s = H_0(x_i||x_s||w_s||\sigma_s)$ and send the parameters $\{x_s, w_s, \alpha_s\}$ to the mobile user MU_i.

3. MU_i after receiving the parameters $\{x_s, w_s, \alpha_s\}$ from SP_j, authenticate SP_j and generate shared session key in the following manner: The server MU_i first computes the parameters y, z in same manner as done by SP_j as $y = H_2(x_i), z = H_2(x_s)$. Then, MU_i computes $\bar{p_i} = p_i + a.z$, $k_s^{'} = \bar{p_i}.(r_i + y) + 2.f_i^{'}$ such that $f_i^{'} \leftarrow \chi_\beta$ and $\sigma_s^{'} = Mod_2(k_s^{'}, w_s)$. MU_i now authenticate the server SP_j by checking whether $\alpha_s \stackrel{?}{=} H_0(x_i||x_s||w_s||\sigma_s^{'})$. If it is not equal, then the session is aborted, otherwise MU_i will continue to complete the session. After the successful authentication of SP_j, MU_i computes $s_i = d_i \oplus H_0(id_i||pw_i)$, $k_i = x_s.r_i$, $w_i = Cha(k_i)$, $\sigma_i = Mod_2(k_i, w_i)$, $aid_i = id_i \oplus H_0(x_i||w_i||\sigma_i||x_s||w_s||\sigma_s^{'})$,

   $\alpha_i = H_0(id_i||s_i||aid_i||x_i||w_i||\sigma_i||x_s||w_s||\sigma_s^{'}||\alpha_s)$ and finally, the session key
   $sk_i = H_0(id_i||aid_i||x_i||w_i||\sigma_i||\alpha_i||x_s||w_s||\sigma_s^{'}||\alpha_s)$. The mobile user MU_i sends the parameters $\{aid_i, w_i, \alpha_i\}$ to the server SP_j.

4. SP_j after receiving the parameters $\{aid_i, w_i, \alpha_i\}$ from MU_i, authenticate MU_i and generate the shared session key in the following manner: SP_j computes $k_i^{'} = x_i.r_s$, $\sigma_i^{'} =$

Table 3.5: LBA-PAKE login and authentication phase

Mobile User MU_i		Server SP_j
Inputs id_i and pw_i		
MU_i: Fetch $psid_i, d_i$ from mobile device		
MU_i: computes $v_i = H_0(id_i \| \| pw_i \| \| d_i)$		
MU_i: computes $PW_i = H_1(v_i) \in R_q$		
MU_i: Generate $r_i, f_i \leftarrow \chi_\beta$		
MU_i: $x_i = ar_i + 2f_i$		
MU_i: $x_i^* = x_i + PW_i \in R_q$		
	$\xrightarrow{\{psid_i, x_i^*\}}$	
		Fetch PW_i and s^* corresponding to $psid_i$ from Server's Database
		Compute: $x_i = x_i^* - PW_i$
		Generate $r_s, f_s \leftarrow \chi_\beta$
		Compute: $x_s = ar_s + 2f_s$
		Compute: $y = H_2(x_i),\ z = H_2(x_s)$
		Computes: $\bar{x}_i = x_i + a.y$
		$k_s = \bar{x}_i(s+z) + 2f_s^{'}$, where $f_s^{'} \leftarrow \chi_\beta$
		$w_s = Cha(k_s)$
		$\sigma_s = Mod_2(k_s, w_s)$
		$\alpha_s = H_0(x_i \| \| x_s \| \| w_s \| \| \sigma_s)$
	$\xleftarrow{\{x_s, w_s, \alpha_s\}}$	
MU_i: computes $y = H_2(x_i),\ z = H_2(x_s)$		
MU_i: computes $\bar{p}_i = p_i + a.z$		
MU_i: computes $k_s^{'} = \bar{p}_i.(r_i + y) + 2f_i^{'}$, where $f_i^{'} \leftarrow \chi_\beta$		
MU_i: $\sigma_s^{'} = Mod_2(k_s^{'}, w_s)$		
MU_i: Check $\alpha_s \stackrel{?}{=} H_0(x_i \| \| x_s \| \| w_s \| \| \sigma_s^{'})$		
MU_i: Compute: $s_i = d_i \oplus H_0(id_i \| \| pw_i)$		
MU_i: $k_i = x_s.r_i$		
MU_i: $w_i = Cha(k_i)$		
MU_i: $\sigma_i = Mod_2(k_i, w_i)$		
MU_i: $aid_i = id_i \oplus H_0(x_i \| \| w_i \| \| \sigma_i \| \| x_s \| \| w_s \| \| \sigma_s^{'})$		
MU_i: $\alpha_i = H_0(id_i \| \| s_i \| \| aid_i \| \| x_i \| \| w_i \| \| \sigma_i \| \| x_s \| \| w_s \| \| \sigma_s^{'} \| \| \alpha_s)$		
MU_i: $sk_i = H_0(id_i \| \| aid_i \| \| x_i \| \| w_i \| \| \sigma_i \| \| \alpha_i \| \| x_s \| \| w_s \| \| \sigma_s^{'} \| \| \alpha_s)$		
	$\xrightarrow{\{aid_i, w_i, \alpha_i\}}$	
		$k_i^{'} = x_i.r_s$
		$\sigma_i^{'} = Mod_2(k_i^{'}, w_i)$
		$id_i = aid_i \oplus H_0(x_i \| \| w_i \| \| \sigma_i^{'} \| \| x_s \| \| w_s \| \| \sigma_s)$
		$s_i = H_0(id_i \| \| s^*)$
		$\alpha_i^{'} = H_0(id_i \| \| s_i \| \| aid_i \| \| x_i \| \| w_i \| \| \sigma_i^{'} \| \| x_s \| \| w_s \| \| \sigma_s \| \| \alpha_s)$
		Check $\alpha_i^{'} \stackrel{?}{=} \alpha_i$, If not Abort
		Otherwise,
		compute: $sk_s = H_0(id_i \| \| aid_i \| \| x_i \| \| w_i \| \| \sigma_i^{'} \| \| \alpha_i \| \| x_s \| \| w_s \| \| \sigma_s \| \| \alpha_s)$

$Mod_2(k_i^{'}, w_i)$, $id_i = aid_i \oplus H_0(x_i \| \| w_i \| \| \sigma_i^{'} \| \| x_s \| \| w_s \| \| \sigma_s)$, $s_i = H_0(id_i \| \| s^*)$ and

$\alpha_i^{'} = H_0(id_i||s_i||aid_i||x_i||w_i||\sigma_i^{'}||x_s||w_s||\sigma_s||\alpha_s)$. The server SP_j will authenticate the mobile user MU_i by checking if $\alpha_i^{'} \stackrel{?}{=} \alpha_i$. If it is not equal, SP_j will abort the session, otherwise SP_j will compute the session key sk_s as $sk_s = H_0(id_i||aid_i||x_i||w_i||\sigma_i^{'}||\alpha_i||x_s||w_s||\sigma_s||\alpha_s)$. Here, we assert that $sk_s == sk_i$ i.e. the session key sk_s computed by SP_j is same as the session key sk_i computed by MU_i.

3.3.4 Password Update Phase

If the mobile user MU_i wants to update his password, he can do so by inputting the old password. The whole procedure is shown in Table 3.6. It is worth noting that all the messages transferred in the password update phase are through the secure channel.

3.3.5 Condition for the correctness of the proposed protocol

From the Login & authentication phase of the proposed protocol shown in Table 3.5, it can be see that σ_s and $\sigma_s^{'}$ should be same for the correctness of the protocol. Similarly, σ_i and $\sigma_i^{'}$ should be same so that both parties agree on the same shared session key. The condition for the correctness of the proposed protocol for σ_s and $\sigma_s^{'}$ has been derived and the reader can derive similarly the correctness condition for σ_i and $\sigma_i^{'}$. Both computed conditions for the correctness are same.

The values of σ_s and $\sigma_s^{'}$ are derived from k_s and $k_s^{'}$ respectively.

$$\begin{aligned} k_s &= \bar{x_i}.(s+z) + 2.f_s^{'} \\ k_s &= a.r_i.s + a.r_i.z + a.y.s + a.y.z + \\ &\quad 2f_i.s + 2f_i.z + 2.f_s^{'} \end{aligned} \tag{3.6}$$

$$\begin{aligned} k_s^{'} &= \bar{p_i}.(r_i+y) + 2.f_i^{'} \\ k_s^{'} &= a.s.r_i + a.s.y + a.z.r_i + a.z.y + \\ &\quad 2e.r_i + 2e.y + 2.f_i^{'} \end{aligned} \tag{3.7}$$

From the above system of equation, we get k_s and $k_s^{'}$. Now, subtracting the values of k_s and $k_s^{'}$. Thus, we get

$$\begin{aligned} k_s - k_s^{'} &= 2f_i.s + 2f_i.z + 2.f_s^{'} - 2e.r_i - 2e.y - 2.f_i^{'} \\ &= 2(f_i.s + f_i.z + f_s^{'} - e.r_i - e.y - f_i^{'}) \\ ||k_s - k_s^{'}|| &= 2\{||f_i.s|| + ||f_i.z|| + ||f_s^{'}|| \\ &\quad - ||e.r_i|| - ||e.y|| - ||f_i^{'}||\} \end{aligned} \tag{3.8}$$

Table 3.6: LBA-PAKE password update phase

Mobile User MU_i		Server SP_j
Input id_i and old password pw_i		
Fetch $psid_i$, d_i stored in mobile device		
Compute $v_i = H_0(id_i \| pw_i \| d_i)$		
Compute $PW_i^* = H_1(v_i)$		
	$\xrightarrow{\{psid_i, PW_i^*\}}$	
		For $psid_i$, Fetch PW_i from database
		Check if $PW_i \stackrel{?}{=} PW_i^*$
		If it is not equal, abort the connection.
		Otherwise, send ACK Token
	$\xleftarrow{\{ACK-Token\}}$	
IF ACK-Token is received, then choose new password pw_i^{new}		
Computes $d_i^{new} = d_i \oplus H_0(id_i \| pw_i) \oplus H_0(id_i \| pw_i^{new})$		
Computes $v_i^{new} = H_0(id_i \| pw_i^{new} \| d_i^{new})$		
Compute $PW_i^{new} = H_1(v_i^{new})$		
Stores updated $\{d_i^{new}\}$ in mobile device		
	$\xrightarrow{\{PW_i^{new}\}}$	
		Stores updated PW_i^{new} in server's secure database

Now, Applying lemma 1 and lemma 2. we get,

$$||k_s - k_s^{'}|| < 2(4.\beta^2.n^{3/2} + 2.\beta.\sqrt{n}) \tag{3.9}$$

By referring Lemma 3, we can see that $||u-v|| < ||2.e|| < \frac{q}{4}$. Thus, $||e|| < \frac{q}{8}$. Thus, for the correctness of the protocol:

$$\begin{aligned} \frac{q}{8} &> 2(4.\beta^2.n^{3/2} + 2.\beta.\sqrt{n}) \\ q &> 16(4.\beta^2.n^{3/2} + 2.\beta.\sqrt{n}) \end{aligned} \tag{3.10}$$

Thus, above is the condition for the correctness of the proposed protocol.

3.4 Formal Security of the Proposed protocol

In this section, the formal security of the proposed LBA-PAKE protocol is proved using Abdalla et al.'s [58] Real-Or-Random (ROR) model. First, a brief description of our security model is provided, which is designed by following the ROR model and then, the security of the proposed LBA-PAKE protocol is proved in Theorem 4.

3.4.1 Brief description of security model

In this subsection, the main components of our security model have been described, which is followed from the ROR model [58]. All the security definitions are directly followed from the ROR model. The execute, send, test oracle queries are the same as in the basic ROR model. Additionally, we introduce the CorruptMD(U^i) and CorruptS(S^j) oracle queries for verifying the Perfect Forward Secrecy (PFS) of the proposed LBA-PAKE protocol. These oracle queries are as follows:

- **CorruptMD(U^i):** This oracle query simulates the manipulation based attacks which include mobile device stolen and Trojan Horse attack against the user instance U^i. The query outputs all the sensitive information stored in the mobile device of user U^i to the malicious adversary $\mathscr{A}$.

- **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 a long term secret key but also a secure database has been compromised.

Semantic Security of the session key in our security model: In the ROR model, an adversary can ask Execute, Send and Test oracle queries but not the Reveal query. Indeed, the adversary can ask as many Test queries as he can. In this case, all the Test queries will be answered in accordance with the same value of flipped coin bit b which has been 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 keys.

Let SUCC denotes the event in which adversary correctly guesses the bit b of the flipped coin.

Definition 1: The advantage of an adversary $\mathscr{A}$ in correctly guessing the bit b of the flipped coin i.e. breaking the semantic security of the protocol P is given as:

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

The protocol P is secure authentication and key agreement protocol if $Adv_P^{PAKE}(\mathscr{A})$ is negligible.

Definition 2: As proved in [8], the RLWE problem is hard for the polynomial-time adversary. Thus, the advantage of the polynomial-time adversary ($Adv_{R_q}^{RLWE}$) to solve the RLWE problem is negligible. Hence, $Adv_{R_q}^{RLWE} \leq \varepsilon$, where ε is a sufficiently small value.

3.4.2 Security Proof of the proposed LBA-PAKE protocol:

This sub-section states the theorem that proves the security of the proposed LBA-PAKE protocol by following the above-discussed security model.

Theorem 4: The proposed LBA-PAKE protocol is secure and supports perfect forward secrecy, if the size of password and identity dictionary, the range space of hash function, the size of Gaussian distribution space are sufficiently large enough and the advantage for polynomial-time adversary $\mathscr{A}$ to solve the RLWE problem should be negligible i.e. $Adv_{R_q}^{RLWE} \leq \varepsilon$, where ε is a sufficiently small value.
Now, Let $Adv_P^{PAKE}(\mathscr{A})$ denotes the advantage function for adversary $\mathscr{A}$ to break the semantic security of the proposed LBA-PAKE protocol P, then

$$\begin{aligned} Adv_P^{PAKE}(\mathscr{A}) \leq & \frac{q_H^2}{|Hash|} + \frac{q_G^2}{q^n} \\ & + \frac{(q_s + q_{exe})^2}{|SP_{\chi_\beta}|} + 2.\frac{q_s}{|\mathscr{D}_{pw}| * |\mathscr{D}_{ID}|} \\ & + 2.Adv_{R_q}^{RLWE} \end{aligned} \tag{3.11}$$

where $|Hash|$, q^n, $|SP_{\chi_\beta}|$, $|\mathscr{D}_{pw}|$ *and* $|\mathscr{D}_{ID}|$ are the range space of $H_0()$ hash function, range space of $H_1()$ hash function, range space of Gaussian distribution χ_β, size of password dictionary and size of identity dictionary respectively. Also, $Adv_{R_q}^{RLWE}$ denotes the advantage for the polynomial-time adversary in solving the RLWE problem.

Proof: The proof of the theorem consists of a sequence of games G_i where i=0,1,...,4. For each Game G_i, $SUCC_i$ denote the event wherein the adversary $\mathscr{A}$ succeeds in guessing the bit b in the Game G_i. Here, bit b is generated at the start of the experiment by flipping the coin and kept concealed from the adversary $\mathscr{A}$. This value of bit b is utilized in the sequence of games played, whenever Test() oracle query will be called.

Game G_0: The game G_0 is the real attack by an adversary $\mathscr{A}$ against the proposed protocol P. In this game, the adversary $\mathscr{A}$ will guess the bit b of the flipped coin. According to definition 1, the advantage of the adversary $\mathscr{A}$ in guessing the bit b is:

$$Adv_P^{PAKE}(\mathscr{A}) = 2.Pr[SUCC_0] - 1 \tag{3.12}$$

Also, note that the advantage of the adversary in guessing the bit b is 0 in the above equation (as $Pr[SUCC_0] = \frac{1}{2}$ for a random flip of a coin) due to the rescaling of the probabilities.

Game G_1: The Game G_1 simulates the eavesdropping attack using Execute oracle query $(Execute(C^i, S^j))$. The Execute oracle query $(Execute(C^i, S^j))$ captures the honest execution between a client instance C^i and a server instance S^j. The query outputs the transcript that consists of messages exchanged during the honest execution of the protocol P. In the end, the adversary $\mathscr{A}$ calls the Test Oracle query $(Test(U_i))$ and it has to output the guess $b^{'}$ of bit b, to determine whether the output of Test oracle query is the genuine session key or a random key.

In the proposed protocol, session key is calculated as $sk_i = H_0(id_i||aid_i||x_i||w_i||\sigma_i||\alpha_i||x_s||w_s||\sigma_s^{'}||\alpha_s) = H_0(id_i||aid_i||x_i||w_i||\sigma_i^{'}||\alpha_i||x_s||w_s||\sigma_s||\alpha_s) = sk_s$. In order to derive the session key sk_i or sk_s, the adversary $\mathscr{A}$ requires the parameters $id_i, \sigma_i, \sigma_s^{'}$, as other parameters $\{aid_i, x_i, w_i, \alpha_i, x_s, w_s, \alpha_s\}$ are obtained through Execute oracle query $(Execute(C^i, S^j))$. Now, σ_i is calculated from k_i as $k_i = x_s.r_i$. Thus, adversary $\mathscr{A}$ require secret key r_i to derive σ_i. Similarly, s is required to derive to compute σ_s. Therefore, the probability of wining the Game G_1 by an adversary $\mathscr{A}$ is not increased by the eavesdropping attack. Mathematically, this can be expressed as :

$$Pr[SUCC_0] = Pr[SUCC_1] \tag{3.13}$$

Game G_2: This game is modeled as an active attack using a hash, send and execute oracle queries. The game simulates the real-time hash functions and considers the collisions of hash values. The proposed protocol consist of three hash functions $H_0(), H_1()$ *and* $H_2()$. The input parameter of $H_2()$ oracle are available on a public channel and can be intercepted by Execute oracle query $(Execute(C^i, S^j))$. Thus, adversary $\mathscr{A}$ is interested in the collisions of $H_0()$ *and* $H_1()$ oracles. For this purpose, $\mathscr{A}$ calls adaptively the Hash oracle queries in order to identify the hash collisions. Therefore, using the Birthday paradox, the probability of collision using Hash oracle query is $\leq \frac{q_H^2}{2 * |Hash|} + \frac{q_G^2}{2 * q^n}$. Here, q_H and q_G are the number of hash queries corresponding to hash $H_0()$ and $H_1()$ respectively, While $|Hash|$ and q^n is the range space of $H_0()$ and $H_1()$ hash function respectively. The range space of $H_1()$ is q^n because it outputs the polynomial of degree n, where each coefficient of the polynomial can have q possible values (Here q is an odd prime number as described in setup phase).

The probability of collisions of the parameters $\{x_i^*, x_s\}$ using send oracle query $Send(U^i, m)$ and execute oracle query $(Execute(C^i, S^j))$ is also given using Birthday paradox and is utmost $\frac{(q_s + q_{exe})^2}{2 * |SP_{\chi_\beta}|}$. Here, q_s and q_{exe} are the number of send and execute queries, while $|SP_{\chi_\beta}|$ is the range space of discrete Gaussian distribution χ_β. The reason for this value of probability is because the computation of the parameters $\{x_i^*, x_s\}$ involves the secret key value, error term r_i, f_i and r_s, f_s respectively as $x_i = a.r_i + 2.f_i$ and $x_s = a.r_s + 2.f_s$, which is sampled from discrete Gaussian distribution χ_β. Therefore, we have

$$|Pr[SUCC_1] - Pr[SUCC_2]| \leq \frac{q_H^2}{2 * |Hash|} + \frac{q_G^2}{2 * q^n} + \frac{(q_s + q_{exe})^2}{2 * |SP_{\chi_\beta}|} \tag{3.14}$$

Game G_3: This game simulates the corrupt mobile device oracle query $CorruptMD(U^i)$ and models the manipulation based attacks. The manipulation based attacks consist of controlling the device by an adversary either by injecting the Trojan horse or either stealing the mobile device. By either way, Adversary $\mathscr{A}$ gets access of the stored information on a mobile device and he can use that information to bypass the authentication.

In the proposed protocol, the adversary $\mathscr{A}$ gets the parameters $\{psid_i, d_i\}$ by CoruptMD oracle query. Now, the adversary can make an online guess of id and password combination, eliminating one possibility with each guess. Therefore, each send query will eliminate one possible combination of id and password. Hence, mathematically the probability of an adversary $\mathscr{A}$ in guessing id and password combination correctly is given by :

$$|Pr[SUCC_2] - Pr[SUCC_3]| \leq \frac{q_s}{|\mathscr{D}_{pw}| * |\mathscr{D}_{ID}|} \tag{3.15}$$

Here, q_s is the number of send queries, $\mathscr{D}_{pw}$, and $\mathscr{D}_{id}$ are the size of the password and identity dictionary respectively. However, the number of wrong identity and password combination inputs should be limited by the server.

Game G_4: This game simulates the corrupt server oracle query $CorruptS(S^j)$ and models the attack in which the server S gets fully compromised by an adversary $\mathscr{A}$. This game captures the notion of perfect forward secrecy of the proposed protocol. In this game, adversary $\mathscr{A}$ gets the stored parameters of the mobile users in the server's secure database and also, the long term master secret key s of the server. The adversary $\mathscr{A}$ aims to generate the past session key from the past intercepted parameters values. The adversary $\mathscr{A}$ gets the parameters $\{psid_i, PW_i, s, s^*\}$ pertaining to all mobile users by $CorruptS(S^j)$ oracle query. Now, the adversary will get past intercepted parameter's values by execute oracle query $(Execute(C^i, S^j))$. The parameters are $\{psid_i, x_i^*\}$, $\{x_s, w_s, \alpha_s\}$ and $\{aid_i, w_i, \alpha_i\}$. The adversary can compute x_i from x_i^* as $x_i = x_i^* - PW_i$, as adversary has both x_i^* and PW_i pertaining to any mobile user. Now, the adversary task is to fetch s and r_s to generate σ_s and $\sigma_i^{'}$ respectively and correspondingly the session key $sk_s = H_0(id_i||aid_i||x_i||w_i||\sigma_i^{'}||\alpha_i||x_s||w_s||\sigma_s||\alpha_s)$ (all other parameters are already intercepted by the adversary through execute oracle query $(Execute(C^i, S^j))$). The value s is provided to adversary $\mathscr{A}$ through corrupt server $CorruptS(S)$ oracle query and the task of computing r_s from x_s are same as solving the RLWE problem, which is hard as stated in definition 2.

Therefore, we have following result:

$$|Pr[SUCC_3] - Pr[SUCC_4]| \leq Adv_{R_q}^{RLWE} \tag{3.16}$$

All the games are executed until now. If the adversary has tried all possible ways to break the security of the protocol but he is not successful, then the last way to win is to call the Test oracle query $Test(U^i)$ and guess the bit b produced by that query. Therefore, the probability of winning the game G_4 is:

$$Pr[SUCC_4] = \frac{1}{2} \tag{3.17}$$

Using the equation 3.12, we obtain

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

Using triangular inequality, we solve the equation as

$$\begin{aligned} Adv_P^{PAKE}(\mathscr{A}) \leq 2.(&|Pr[SUCC_0] - Pr[SUCC_2]| \\ &+ |Pr[SUCC_2] - Pr[SUCC_3]| \\ &+ |Pr[SUCC_3] - Pr[SUCC_4]|) \end{aligned} \tag{3.18}$$

Using Equation 3.13, we get the following equation:

$$\begin{aligned} Adv_P^{PAKE}(\mathscr{A}) \leq 2.(&|Pr[SUCC_1] - Pr[SUCC_2]| \\ &+ |Pr[SUCC_2] - Pr[SUCC_3]| \\ &+ |Pr[SUCC_3] - Pr[SUCC_4]|) \end{aligned} \tag{3.19}$$

Putting values from equation (3.14, 3.15 and 3.16) we get ,

$$\begin{aligned} Adv_P^{PAKE}(\mathscr{A}) \leq 2.(&\frac{q_H^2}{2 * |Hash|} + \frac{q_G^2}{2 * q^n} \\ &+ \frac{(q_s + q_{exe})^2}{2 * |SP_{\chi_\beta}|} + \frac{q_s}{|\mathscr{D}_{pw}| * |\mathscr{D}_{ID}|} \\ &+ Adv_{R_q}^{RLWE}) \end{aligned}$$

$$Adv_P^{PAKE}(\mathscr{A}) \leq \frac{q_H^2}{|Hash|} + \frac{q_G^2}{q^n} + \frac{(q_s+q_{exe})^2}{|SP_{\chi_\beta}|} + 2.\frac{q_s}{|\mathscr{D}_{pw}| * |\mathscr{D}_{ID}|} + 2.Adv_{R_q}^{RLWE} \tag{3.20}$$

Hence, proving the Theorem 4.

3.5 Performance Analysis

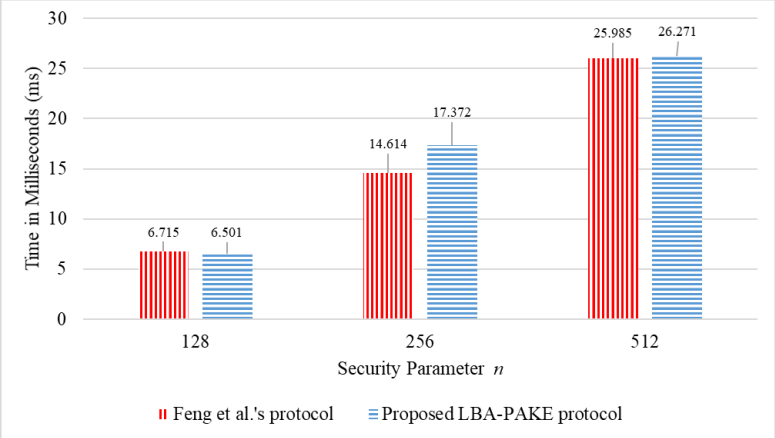
In this section, firstly, suitable parameters are chosen for the proposed LBA-PAKE protocol. Then, classical and quantum security levels are analyzed for the parameters of choice. Further, the proposed LBA-PAKE and Feng et al.'s protocol have been implemented for $\{512, 256, 128\}$ values of security parameter n, where n is the degree of any polynomial $p(x)$ such that $p(x) \in R_q$. Finally, the communication cost of the proposed LBA-PAKE protocol has been computed.

3.5.1 Parameters chosen, computation & communication overheads

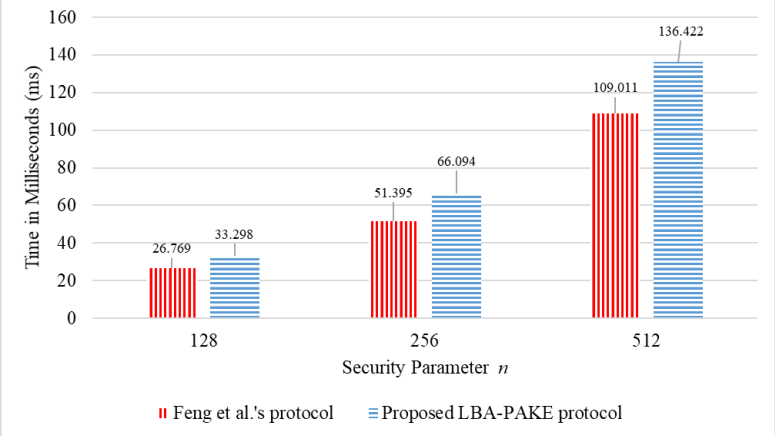
The following parameters has been chosen for calculating the computation and communication overhead of the proposed LBA-PAKE protocol.

The security parameter n is of power of two and is of $\{512, 256, 128\}$ values, the discrete Gaussian distribution χ_β for sampling secret and error terms is of standard deviation $\beta = \frac{8}{\sqrt{2*\pi}} \simeq 3.192$. For the correctness of the protocol $q > 16(4.\beta^2.n^{3/2} + 2.\beta.\sqrt{n})$, therefore for n equals to 512 and standard deviation of $\beta = \frac{8}{\sqrt{2*\pi}} \simeq 3.192$ of χ_β distribution, the calculated value of odd prime q is 7557773.

With the above values of the parameters, we analyze the classical security of the proposed LBA-PAKE protocol using the LWE estimator [59]. LWE estimator estimates the level of security provided with the particular set of the parameter's values for both LWE and RLWE based cryptosystem. The idea is to compute the attack complexity of the meet-in-the-middle, decoding, reducing BDD to unique-SVP, BKW, lattice reduction, and exhaustive search attack. From the given set of parameter's values, the LWE estimator outputs the time and space complexity of these attacks. The snapshot of the sage code to estimate the classical security level of the proposed LBA-PAKE protocol is given in Figure 3.2.



(a) Server-side computation cost



(b) Client-side (Mobile) computation cost

Figure 3.1: Comparison of key exchange timings of Feng et al.'s protocol & proposed LBA-PAKE protocol

```
In [1]: load("estimator.py")

In [19]: n, alpha,q=512,alphaf(8,7557773),7557773

In [20]: set_verbose(1)

In [ ]: _=estimate_lwe(n,alpha,q,skip=["arora-gb"])
```

Figure 3.2: Snapshot of sage code using LWE estimator

The output of the above sage code shows that our parameters of choice will provide at least 100-bit of classical security.

For the estimation of quantum security, the idea of NewHope key exchange protocol [31] has been followed to calculate the quantum security level of the proposed LBA-PAKE protocol. "The idea is to compute the computation complexity of primal and dual attacks for solving the underlying lattice problem on a quantum computer. The estimation is pessimistic as the underlying lattice problem considered is only core SVP and actual complexity is much higher

than the result of the estimation" [31]. Hence, the results of the estimation for the above parameters of choice shows that the proposed LBA-PAKE protocol offers at least 75-bits of quantum security.

The notations for the other calculated parameters are as follows:

- T_{samp} depicts the average time elapsed for sampling from the discrete Gaussian distribution χ_β.
- T_{rmul} shows the average time elapsed for the multiplication of two polynomial elements a, b where $a \in R_q$ and b is sampled from discrete Gaussian distribution χ_β as $b \leftarrow \chi_\beta$.
- T_{h0} shows the average time elapsed for computing the $H_0()$ function (specifically sha3-224() function).
- T_{h1}: depicts the average time elapsed for computing the $H_1()$ function.
- T_{h2}: stands for the average time elapsed for computing the $H_2()$ function.
- T_{Cha} shows the average time elapsed for computing the $Cha()$ function.
- T_{mod} depicts the average time elapsed for computing the $Mod_2()$ function.

The system configurations used for the implementation of the proposed LBA-PAKE and Feng et al.'s protocol are as follows: The mobile user uses the Redmi 5 smartphone with Octa-core Max 1.80 GHz CPU and 2GB RAM on the Android version-7.1.2 N2G47H. The server is running on Intel(R)Core(TM)i5-7200U processor with 2.50 GHz frequency and 8 GB RAM. The message digest is generated using SHA3-224() hash function as $H_0()$ output. The running time of the above-discussed parameters are listed in Table 5.2.

The comparison of the computational overhead of the proposed LBA-PAKE and Feng et al.'s protocol has been presented in Fig. 3.1. The Figure 3.1a depicts the server-side computation cost while Figure 3.1b shows the client-side (mobile) computation cost. From the Figure 3.1a and 3.1b, it can be seen that the computation cost of the proposed LBA-PAKE protocol is slightly more (few milliseconds) than the Feng et al.'s protocol. The reason for this extra computational cost is the provision in the proposed LBA-PAKE protocol to resist the signal leakage attack (SLA). As discussed in above sections, Feng et al.'s protocol is vulnerable to SLA attack and the attacker can successfully recover the master secret key s of the server. Thus, to resist the SLA attack, the proposed LBA-PAKE protocol introduces extra one-way hash function $H_2()$. The idea behind the SLA attack is the manipulation of the public key by the mobile user to obtain the master secret key s of the server. Now, the $H_2()$ hash function keeps a check on the public key x_i of the mobile user and ensures that it does not deviate from the protocol. However, this will add to computation cost, as seen from the Table 3.5, the computation of $y = H_2(x_i)$, $z = H_2(x_s)$ and $\bar{x_i}$ on server-side and $y = H_2(x_i)$, $z = H_2(x_s)$ and $\bar{p_i}$ on mobile-side.

Table 3.7: Running time of cryptographic operations (in microseconds) for $n = 512$, $\beta = 3.192$ and $q = 7557773$

Cryptographic Operation	**Running Time of Server**	**Running Time of Mobile User**
T_{samp}	48.52	150.61
T_{rmul}	496.59	1518.764
T_{h0}	2253.79	10915.585
T_{h1}	127.96	281.282
T_{h2}	1612.21	6562.526
T_{Cha}	1845.08	8512.699
T_{mod}	756.34	5018.214

This extra computation cost contributes to secure the proposed LBA-PAKE protocol against the SLA attack. Thus, the proposed LBA-PAKE protocol is more secure than Feng et al.'s protocol with approximately the same computational overhead. Also, one can see from Table 5.2 that the individual operations on a mobile device take more time than on the server. This behavior is quite obvious as the server is computationally more powerful than the mobile device.

Further, the proposed protocol has three rounds of communication between mobile client and server. Hence, to compute the communication complexity, we calculate the bit size of the each parameter exchanged between mobile client and server. The parameters exchanged between the mobile client and server in three rounds of communication are $\{psid_i, x_i^*\}$, $\{x_s, w_s, \alpha_s\}$ and lastly, $\{aid_i, w_i, \alpha_i\}$. The bit size of $psid_i$ is of 8 bits, x_i^*, x_s is $n * \log_2(q)$ bits each, w_s, w_i is n bits each and $\alpha_s, \alpha_i, aid_i$ is 224 bits each (as it is output of sha3_224() function). Thus, combing together the overall communication complexity is $2n\log_2(q) + 2n + 3 * 224 + 8$=$2n\log_2(q) + 2n + 680$ bits.

3.6 Summary

In this chapter, the Feng et al.'s anonymous authentication key exchange protocol for mobile devices has been crypt-analyzed. From the crypt-analysis, it has been found that the Feng et al.'s protocol has many security risks. Thus, a new protocol named LBA-PAKE has been proposed, which can resist the signal leakage attack (SLA) and other attacks, to which Feng et al.'s protocol is vulnerable. The idea to prevent the SLA attack is to verify the public keys using the direct validation technique of Ding et al. [57]. The formal security analysis has been done using Abdalla et al.'s ROR model. The implementation parameters of choice offer 100 bits of classical and 75 bits of quantum security. The computation timings also validate the proposed protocol in terms of efficiency for the mobile client-server environment.

Chapter 4

Modified two-party authenticated key agreement protocol for post-quantum world

The previous chapter expatiated the impact of signal leakage attack (SLA) on the security of Feng et al.'s key exchange protocol. Thus, the LBA-PAKE protocol for mobile devices has been proposed. The proposed protocol resists the SLA attack and also provides key reuse, anonymity, and perfect forward secrecy features.

In this chapter, it has been shown that Islam's two-party key agreement is vulnerable to the modified version of the signal leakage attack (SLA), which is also termed as improved-signal leakage attack (i-SLA). Therefore, the modified two-party authenticated key agreement (m-2PAKA) protocol for post-quantum world has been proposed. This protocol is the improvement of Islam's provably secure two-party authenticated key agreement (2PAKA) protocol. Also, the condition for correctness of the modified protocol has been calculated.

4.1 Background

In 2018, Ding et al. [13] showed that the SLA attack can be mounted efficiently with less number of queries. The authors in [13] suggest that it is not necessary to vary the constant k in the malformed public key of adversary looping from 0 to $q-1$ (as done in original SLA [34]) and there are many iterations of k that can be skipped. Thus, the fewer number of queries are required to recover the private key s of the honest party. These fewer number of queries are utmost $q+c$ (where c is a constant and q is an odd prime number) and thus, the complexity of the SLA attack is reduced to $q+c$, in comparison to $2q$ in the original SLA attack.

The above version of the SLA attack is termed as improved-signal leakage attack (i-SLA). In this work, it has been shown that the Islam's [1] key agreement protocol (2PAKA) is vulnerable to the improved-signal leakage attack (i-SLA) [13]. Using i-SLA, the attacker can

successfully recover the reused secret of the honest party by instantiating utmost q number of key exchange sessions with the honest party using q number of malformed public keys, which are authenticated by the Certificated Authority (CA). The countermeasure to defend the protocol against this attack has been provided without changing the original design of the protocol.

4.2 Cryptanalysis of Islam's [1] two-party authenticated key agreement protocol (2PAKA)

This section has been reserved for the cryptanalysis of Islam's [1] two-party authenticated key agreement (2PAKA) protocol. The summary of the notations of the Islam's protocol is given in and Table 4.1 respectively. The complete key agreement protocol of Islam is given in Table 4.2. As discussed above, Islam's [1] key agreement protocol (2PAKA) is vulnerable to improved-signal leakage attack (i-SLA) [13]. Hence, the method to exploit the Islam's [1] key agreement protocol has been described using the i-SLA attack.

Table 4.1: Islam's protocol parameters

Islam.'s notations of parameters
n: Positive security parameter s.t. $q \ mod \ 2n = 1$
χ_β: Discrete Gaussian Distribution over R_q with standard deviation β
a: random element in R_q
$x \leftarrow \chi_\beta$: x is sampled from R_q unifromly at random according to χ_β
CA: Certificate Authority
$U_i \vert U_j$: Initiator (User)|Responder (User)
$id_i \vert id_j$: Identity of U_i|Identity of U_j
s_i: Long term private key of U_i
p_i: Long term public key of U_i s.t. $p_i = a.s_i + 2.e_i$
where $s_i, e_i \leftarrow \chi_\beta$
s_j: Long term private key of U_j
p_j: Long term public key of U_j s.t. $p_j = a.s_j + 2.e_j$
where $s_j, e_j \leftarrow \chi_\beta$
Region E: $E = \{-\left\lfloor \frac{q}{4} \right\rfloor, ..., \left\lfloor \frac{q}{4} \right\rceil\}$
E^c: Complement of E
$Cha(.)$: Characteristic Function $Cha()$ is complement of E such that
$Cha(v) = 0$ if $v \in E$ and 1 otherwise
$Mod_2(.)$: $Mod_2(v,w) = (v + w.\frac{q-1}{2}) \ mod \ q \ mod \ 2$
$H_l(.)$: Secure hash function that output the fixed length string
$H_l : \{0,1\}^* \rightarrow \{0,1\}^l, l = 1,2,3$
$\{n, q, \chi_\beta, a, p_i, p_j, H_l()\}$: System or Public parameters

Table 4.2: Islam's [1] two-party authenticated key agreement protocol (2PAKA)

User U_i (Initiator)	User U_j (Responder)
Sample $r_i, f_i \leftarrow \chi_\beta$	Sample $r_j, f_j \leftarrow \chi_\beta$
Compute $x_i = ar_i + 2f_i$	Compute $x_j = ar_j + 2f_j$
Compute $t_{i1} = s_i.p_j$	Compute $t_{j1} = s_j.p_i$
Compute $w_{i1} = Cha(t_{i1})$	Compute $w_{j1} = Cha(t_{j1})$
Compute $\sigma_{i1} = Mod_2(t_{i1}, w_{i1})$	Compute $\sigma_{j1} = Mod_2(t_{j1}, w_{j1})$
Compute $\alpha_{i1} = H_1(id_i \|\| x_i \|\| \sigma_{i1})$	Compute $\alpha_{j1} = H_1(id_j \|\| x_j \|\| \sigma_{j1})$
$\xrightarrow{\{id_i, x_i, w_{i1}, \alpha_{i1}\}}$	
$\xleftarrow{\{id_j, x_j, w_{j1}, \alpha_{j1}\}}$	
Compute $\sigma'_{j1} = Mod_2(t_{i1}, w_{j1}), \alpha'_{j1} = H_1(id_j \|\| x_j \|\| \sigma'_{j1})$	Compute $\sigma'_{i1} = Mod_2(t_{j1}, w_{i1}), \alpha'_{i1} = H_1(id_i \|\| x_i \|\| \sigma'_{i1})$
if $\alpha_{j1} \neq \alpha'_{j1}$, *Abort*	if $\alpha_{i1} \neq \alpha'_{i1}$, *Abort*
Else, Compute $t_{i2} = r_i.x_j$, $w_{i2} = Cha(t_{i2})$	Else, Compute $t_{j2} = r_j.x_i$, $w_{j2} = Cha(t_{j2})$
Compute $\sigma_{i2} = Mod_2(t_{i2}, w_{i2}), \alpha_{i2} = H_2(id_i \|\| x_i \|\| x_j \|\| \sigma_{i2})$	Compute $\sigma_{j2} = Mod_2(t_{j2}, w_{j2}), \alpha_{j2} = H_2(id_j \|\| x_j \|\| x_i \|\| \sigma_{j2})$
$\xrightarrow{\{id_i, w_{i2}, \alpha_{i2}\}}$	
$\xleftarrow{\{id_j, w_{j2}, \alpha_{j2}\}}$	
Compute $\sigma'_{j2} = Mod_2(t_{i2}, w_{j2}), \alpha'_{j2} = H_2(id_j \|\| x_j \|\| x_i \|\| \sigma'_{j2})$	Compute $\sigma'_{i2} = Mod_2(t_{j2}, w_{i2}), \alpha'_{i2} = H_2(id_i \|\| x_i \|\| x_j \|\| \sigma'_{i2})$
if $\alpha_{j2} \neq \alpha'_{j2}$, *Abort*	if $\alpha_{i2} \neq \alpha'_{i2}$, *Abort*
Else, Compute $sid = (id_i \|\| id_j \|\| x_i \|\| x_j \|\| w_{i1} \|\| w_{j1} \|\| \alpha_{i1} \|\| \alpha_{j1})$	Else, Compute $sid = (id_i \|\| id_j \|\| x_i \|\| x_j \|\| w_{i1} \|\| w_{j1} \|\| \alpha_{i1} \|\| \alpha_{j1})$
Compute session key $sk = H_3(sid \|\| \sigma_{i1} \|\| \sigma_{i2} \|\| \sigma'_{j1} \|\| \sigma'_{j2})$	Compute session key $sk = H_3(sid \|\| \sigma'_{i1} \|\| \sigma'_{i2} \|\| \sigma_{j1} \|\| \sigma_{j2})$

4.2.1 improved-Signal Leakage Attack (i-SLA) on [1]

While applying i-SLA to Islam's key agreement protocol, one can observe from Table 4.1 that, p_i, s_i are the long term public and private key respectively of user U_i. Similarly, p_j, s_j are the long term public and private key respectively of user U_j. Also, these long term public keys i.e. p_i and p_j are authenticated by a certificate authority (CA) before using them. Using old SLA [34], the adversary require $2q$ number of public keys authenticated by CA to recover the private key of the honest party (where q is an odd prime number). Using i-SLA, the adversary can recover the private key of the honest party with utmost q number of public keys that are authenticated by CA.

Let us consider a case, where the user U_i is an adversary and user U_j is an honest party that reuses its long term key pair (p_j, s_j). The goal of adversary U_i is to recover the long term private key s_j of the honest party U_j. To fulfill its goal, the adversary U_i will instantiate multiple session with the user U_j while varying its public key p_i in each session in such a manner to get some information about the private key s_j of user U_j in each of the session instantiated. Here, we assume that adversary has q number of authenticated public keys by CA, where q is an odd prime. Now, we claim the following statement:

Claim 1: The user U_i which act as an adversary can recover the long term private key s_j of the honest user U_j by instantiating utmost q number of key exchange sessions with q number of malformed public keys, that are authenticated by certificate authority (CA) using i-SLA.

Proof: In this scenario, it is assumed that the user U_i act as an adversary while user U_j act as an honest user. From Table 4.2, it can be seen that the signal value w_{j1} may leak the long term secret key s_j of the honest user U_j to adversary U_i using signal leakage attack. Now, using old signal leakage attack [34], the adversary U_i counts the number of times the signal bit $w_{j1} = Cha(t_{j1})$ changes for each coefficient of t_{j1}, for varying k across all the values of $\mathbb{Z}_q$ in the public key $p_i = a.s_i + ke_i$ of the adversary U_i (here k is a constant whose value varies from 0 to $q-1$). To obtain the precise value of the secret key s_j, this attack requires $2q$ number of queries to party U_j. (refer to [34] for detailed description).

In 2018, Ding et al. [13] proposed the more efficient signal leakage attack (which is termed as i-SLA in this article) than the old signal leakage attack [2]. "The attack emphasize the fact that it is not necessary to vary the value of k through all the values of $\mathbb{Z}_q$ in the public key $(p_i = a.s_i + k.e_i)$ of the adversary. This is due to the reason that for each coefficient of secret key $s_j[i]$, as k varies from 0 to q-1, the value of t_{j1} changes in multiples of $s_j[i]$. Thus, depending of the value of $s_j[i]$, the period of signal change varies and this can be used to perform the attack more efficiently" [13]. Here, we consider two different cases to show the successful launch of the i-SLA attack against the Islam's [1] two-party authenticated key agreement protocol (2PAKA).

Case 1: First, we'll look at a simple scenario. Here, the adversary U_i will first generate its public keys $p_i = a.s_i + k.e_i$ by choosing the secret key $s_i = 0$ and $e_i = 1$ in R_q. Thus, the public key of the adversary U_i becomes $p_i = a.0 + k.1 = k$. Now, it has been observed that the first value of k where the signal bit w_{j1} flips gives the value of $s_j[i]$ upto $\pm$ sign. This is due to the fact that the first flip of signal bits happens when k changes from $\left\lfloor \frac{q}{4s_B[i]} \right\rceil$ to $\left\lfloor \frac{q}{4s_B[i]} \right\rceil + 1$ by the definition of region E, E^c and signal function $Cha(.)$ (see Table 4.1 for the definition of E, E^c *and* $Cha(.)$). Also, the adversary U_i will recover each coefficient i of $s_j[i]$ at once by varying k from 0 to $\frac{q}{4} + 2$. This is due to the reason that the smaller values of $s_j[i]$ needs more number of queries for counting the first signal change e.g. $s_j[i] = \pm 1$ needs $\frac{q}{4} + 2$ queries for first signal change, $s_j[i] = \pm 2$ needs $\frac{q}{4} + 1$ queries for first signal change and so on. Thus, using $\frac{q}{4} + 2$ queries, the adversary U_i can recover the value of secret key $s_j[i]$ upto $\pm$ sign. Again, using $\frac{q}{4} + 2$ number of queries the adversary U_i with public key $p_i = (1+x)k$ can resolve the ambiguity of $\pm$ sign. The idea to resolve the ambiguity of $\pm$ is same as that of old SLA [34] (refer section 4.2 of [34]) but the value of k in the public key $(p_i = (1+x)k)$ of the adversary U_i is not varied through all the values of $\mathbb{Z}_q$. Instead, the adversary will find the first value of k where the signal bit $w_{j1}[i]$ flips (which is $\frac{q}{4} + 2$ in worst case i.e. when $s[j] = \pm 1$ as described above) and hence, recover the coefficient values of $s_j[0] - s_j[n-1]$, $s_j[1] + s_j[2], ..., s_j[n-2] + s_j[n-1]$ upto $\pm$ sign. Now, with additional information about these coefficients, the adversary U_i can determine if each pair of coefficients $s_j[m], s_j[n]$ have equal or opposite sign and therefore, the

ambiguity of $\pm$ sign can be resolved using additional $\frac{q}{4}+2$ number of queries. Lastly, the adversary U_i can recover the secret key s_j of the honest user U_j with $2.(\frac{q}{4}+2) = \frac{q}{2}+4$ number of queries and thus reducing the query complexity of i-SLA attack by factor of $\frac{1}{4}$ as compared to old SLA attack [34].

In the above approach, the adversary U_i chooses its public key p_i as $p_i = k$, which is a constant polynomial in R_q. The victim U_j can defend itself against the above attack by verifying the public key p_i of the dishonest party U_i and checking whether p_i is a constant polynomial in R_q or not. The honest party U_j will abort the session if the public key of user U_i is a constant polynomial in R_q. To overcome this, the adversary will modify the above attack to bypass the verification of its public key p_i by generating it in way that it is indistinguishable to the honest party U_j. This approach is accomplished below in case 2 as:

Case 2: This is the second case of the protocol where the public key p_i of the adversary U_i is computed as $p_i = a.s_i + k.e_i$, where s_i is sampled from the error distribution χ_β, $e_i = 1$ in R_q, a is random element in R_q and k is a constant. Thus, the public key p_i of the adversary U_i becomes $p_i = a.s_i + k.1 = a.s_i + k$. At honest user side U_j, the t_{j1} is computed as $t_{j1} = a.s_i.s_j + k.s_j$and the corresponding signal function $w_{j1}$is computed as $w_{j1} = Cha(a.s_i.s_j + k.s_j)$. Now in this case, we cannot use the first k value where the signal w_{j1} flips to determine the value of s_j as done in case 1. This is due to the reason that for every coefficient i, a constant value of $a.s_i.s_j[i]$ is added to $k.s_j[i]$ while computing t_{j1} and this value of $a.s_i.s_j[i]$ is unknown to the adversary U_i. Hence, to count the number of signal changes, the adversary U_i first varies k from 0 to positive values of k until the first signal change w_{j1} happens. Let m_1 be the value where the adversary U_i notice the first signal change in w_{j1}. After that, attacker U_i vary k for negative values and notice the first signal change in w_{j1}. Let m_2 be the k value for first signal change in negative direction. Now, the value of $(m_1 - m_2)$ is the span of region E or E^c (see Table 4.1) in multiples of $s_j[i]$. Hence, the value $\left\lfloor \frac{q}{2.(m_1-m_2)} \right\rceil$ reveals the value of $s_j[i]$ upto $\pm$ sign as the period of the signal change is $(m_1 - m_2)$. Thus, by instantiating $\frac{q}{2}$ number of sessions with $\frac{q}{2}$ number of public keys, the adversary U_i can recover the private key $s_j[i]$ of the honest user U_j upto $\pm$ sign. For recovering the exact value of $s_j[i]$, additional $\frac{q}{2}$ number of public keys of the form $(1+x)p_i$ is required to instantiate the $\frac{q}{2}$ number of sessions. Thus, with utmost q number of sessions using q number of public keys, that are authenticated by CA, the adversary U_i can recover the long term private key s_j of the honest user U_j. Thus, this completes the proof.

Now, in another scenario if the user U_i is the honest party and the user U_j is an adversary, then user U_j can also recover the long term secret key s_i of the user U_i. Thus, we claim the following statement:

Claim 2: The user U_j which act as an adversary can recover the long term private key s_i of

Table 4.3: Modified two-party authenticated key agreement (m-2PAKA) protocol

User U_i (Initiator)	User U_j (Responder)
Sample $r_i, f_i \leftarrow \chi_\beta$	Sample $r_j, f_j \leftarrow \chi_\beta$
Compute $x_i = ar_i + 2f_i$	Compute $x_j = ar_j + 2f_j$
Compute $\boxed{m = H_0(p_i), n = H_0(p_j)}$	Compute $\boxed{m = H_0(p_i), n = H_0(p_j)}$
Compute $\boxed{\bar{p_j} = p_j + a.n + 2.f_{i0}}$, where $f_{i0} \leftarrow \chi_\beta$	Compute $\boxed{\bar{p_i} = p_i + a.m + 2.f_{j0}}$, where $f_{j0} \leftarrow \chi_\beta$
Compute $\boxed{t_{i1} = \bar{p_j}.(s_i + m) + 2.f_{i1}}$, where $f_{i1} \leftarrow \chi_\beta$	Compute $\boxed{t_{j1} = \bar{p_i}.(s_j + n) + 2.f_{j1}}$, where $f_{j1} \leftarrow \chi_\beta$
$t_{i1} = (p_j + a.n + 2.f_{i0}).(s_i + m) + 2.f_{i1}$	$t_{j1} = (p_i + a.m + 2.f_{j0}).(s_j + n) + 2.f_{j1}$
$t_{i1} = p_j.s_i + m.p_j + a.n.m + n.p_i + 2.s_i.f_{i0} + 2.m.f_{i0}$ $+2.f_{i1} - 2.n.e_i$	$t_{j1} = p_i.s_j + n.p_i + a.m.n + m.p_j + 2.s_j.f_{j0} + 2.n.f_{j0}$ $+2.f_{j1} - 2.m.e_j$
$t_{i1} = p_j.s_i + \Delta + 2.s_i.f_{i0} + 2.m.f_{i0} + 2.f_{i1} - 2.n.e_i$	$t_{j1} = p_i.s_j + \Delta + 2.s_j.f_{j0} + 2.n.f_{j0} + 2.f_{j1} - 2.m.e_j$
Here $\Delta = m.p_j + a.n.m + n.p_i$	Here $\Delta = n.p_i + a.m.n + m.p_j$
Check if $\Delta[i] \neq 0$ AND $\|\|\Delta\|\|$ is large enough	Check if $\Delta[i] \neq 0$ AND $\|\|\Delta\|\|$ is large enough
Otherwise Abort the session.	Otherwise Abort the session.
Compute $w_{i1} = Cha(t_{i1})$	Compute $w_{j1} = Cha(t_{j1})$
Compute $\sigma_{i1} = Mod_2(t_{i1}, w_{i1})$	Compute $\sigma_{j1} = Mod_2(t_{j1}, w_{j1})$
Compute $\alpha_{i1} = H_1(id_i \|\| x_i \|\| \sigma_{i1})$	Compute $\alpha_{j1} = H_1(id_j \|\| x_j \|\| \sigma_{j1})$
$\xrightarrow{\{id_i, x_i, w_{i1}, \alpha_{i1}\}}$	
	$\xleftarrow{\{id_j, x_j, w_{j1}, \alpha_{j1}\}}$
Compute $\sigma'_{j1} = Mod_2(t_{i1}, w_{j1}), \alpha'_{j1} = H_1(id_j \|\| x_j \|\| \sigma'_{j1})$	Compute $\sigma'_{i1} = Mod_2(t_{j1}, w_{i1}), \alpha'_{i1} = H_1(id_i \|\| x_i \|\| \sigma'_{i1})$
if $\alpha_{j1} \neq \alpha'_{j1}$, *Abort*	if $\alpha_{i1} \neq \alpha'_{i1}$, *Abort*
Else, Compute $t_{i2} = r_i.x_j$, $w_{i2} = Cha(t_{i2})$	Else, Compute $t_{j2} = r_j.x_i$, $w_{j2} = Cha(t_{j2})$
Compute $\sigma_{i2} = Mod_2(t_{i2}, w_{i2}), \alpha_{i2} = H_2(id_i \|\| x_i \|\| x_j \|\| \sigma_{i2})$	Compute $\sigma_{j2} = Mod_2(t_{j2}, w_{j2}), \alpha_{j2} = H_2(id_j \|\| x_j \|\| x_i \|\| \sigma_{j2})$
$\xrightarrow{\{id_i, w_{i2}, \alpha_{i2}\}}$	
	$\xleftarrow{\{id_j, w_{j2}, \alpha_{j2}\}}$
Compute $\sigma'_{j2} = Mod_2(t_{i2}, w_{j2}), \alpha'_{j2} = H_2(id_j \|\| x_j \|\| x_i \|\| \sigma'_{j2})$	Compute $\sigma'_{i2} = Mod_2(t_{j2}, w_{i2}), \alpha'_{i2} = H_2(id_i \|\| x_i \|\| x_j \|\| \sigma'_{i2})$
if $\alpha_{j2} \neq \alpha'_{j2}$, *Abort*	if $\alpha_{i2} \neq \alpha'_{i2}$, *Abort*
Else, Compute $sid = (id_i \|\| id_j \|\| x_i \|\| x_j \|\| w_{i1} \|\| w_{j1} \|\| \alpha_{i1} \|\| \alpha_{j1})$	Else, Compute $sid = (id_i \|\| id_j \|\| x_i \|\| x_j \|\| w_{i1} \|\| w_{j1} \|\| \alpha_{i1} \|\| \alpha_{j1})$
Compute session key $sk = H_3(sid \|\| \sigma_{i1} \|\| \sigma_{i2} \|\| \sigma'_{j1} \|\| \sigma'_{j2})$	Compute session key $sk = H_3(sid \|\| \sigma'_{i1} \|\| \sigma'_{i2} \|\| \sigma_{j1} \|\| \sigma_{j2})$

the honest user U_i by instantiating utmost q number of key exchange sessions with q number of malformed public keys, that are authenticated by certificate authority (CA) using i-SLA.

Proof: The proof of this claim can be derived similarly as that of claim 1.

4.3 Countermeasure

In this section, we provide countermeasure to improve the Islam's two-party authenticated key agreement protocol (2PAKA) so that it can resist the improved-signal leakage attack (i-SLA). The cause of both SLA and its modified version i-SLA is the deviation from the protocol by an active adversary (which acts as one of the party in key exchange) and generating the public key in a way to recover the long term (or reused) secret key of the honest party. Thus, to resist this attack there is need of public key validation technique so that adversary (acting as one

of the legitimate party) cannot generate the public key that may help to leak information about private key of the honest party (acts as other legitimate party). In 2018, Ding et al. [57] proposed a technique for direct public key validation in RLWE-based zero-knowledge authentication protocol. Our countermeasure is based on the trick used in this work to validate the RLWE-based public/private key pairs. The trick is to force each of the party involved in key agreement protocol to behave honestly by hashing their public keys with a hash function that is modeled as a random oracle and whose output are sampled from the error distribution χ_β. The patched Islam's key agreement protocol with applied modifications that are boxed out are shown in Table 4.3.

As seen from Table 4.3, the user U_j prepossesses the public key p_i of the user U_i as $\bar{p}_i$, where $\bar{p}_i$ is computed as follows:

$$\begin{aligned} \bar{p}_i &= p_i + a.H_0(p_i) + 2.f_{j0} \; where \; f_{j0} \leftarrow \chi_\beta \\ &= p_i + a.m + 2.f_{j0} \end{aligned} \tag{4.1}$$

As discussed above, the hash function $H_0()$ is modeled as a random oracle whose output are sampled from χ_β. Typically, $H_0()$ is implemented using combination of SHA3 like hash function and discrete Gaussian sampler. Mathematically, it can be represented as:

$$\begin{aligned} H_0(x) &= H_{02}(H_{01}(x)), \; where \; x \in R_q \\ H_0(x) &= H_{02}(seed), \; where \; seed = H_{01}(x) \end{aligned}$$

In the above equation, the H_{01} is a typical hash function of SHA3 series whose input x is the element in R_q and it will generate the fixed length output string (also termed as seed). It is implemented by concatenating the coefficients of the input element x, where $x \in R_q$. The fixed length output string generated from $H_{01}()$ is feed to $H_{02}()$ function as input. The $H_{02}()$ is a discrete Gaussian sampler that generates the random polynomial of degree n whose coefficients are sampled from discrete Gaussian distribution χ_β. The $H_{02}()$ function is represented as

$$\begin{aligned} H_{02} &= DiscreteGaussianSampler(n, q, \beta, H_{01}(x)) \\ &\quad where \; x \in R_q \\ H_{02} &= DiscreteGaussianSampler(n, q, \beta, seed) \\ &\quad where \; seed = H_{01}(x) \end{aligned}$$

Here, the parameters n, q, β are security parameter, an odd prime number and standard deviation of discrete Gaussian distribution χ_β respectively.

Now, the user U_j will generate t_{j1} key using preprocessed public key $\bar{p}_i$ as follows:

$$\begin{aligned} t_{j1} &= \bar{p}_i.(s_j + n) + 2.f_{j1},\ where\ f_{j1} \leftarrow \chi_\beta \\ &= a.s_i.s_j + a.n.s_i + a.m.s_j + a.m.n + 2.e_i.s_j \\ &\quad + 2.e_i.n + 2.s_j.f_{j0} + 2.n.f_{j0} + 2.f_{j1} \end{aligned} \tag{4.2}$$

From the above equation, we can see that the adversary acts as user U_i do not get any information about the private key s_j of the user U_j from the computed signal function output $w_{j1} = Cha(t_{j1})$. This is due to the reason that m, n and f_{j0} are parameters used in the computation of t_{j1} and the values of the coefficient of m, n and f_{j0} are randomly chosen and uniformly distributed over discrete Gaussian distribution χ_β.

Again, the user U_i prepossesses the public key p_j of the user U_j to force its honest behavior. The computation on user side U_i is as follows:

$$\begin{aligned} \bar{p}_j &= p_j + a.H_0(p_j) + 2.f_{i0}\ where\ f_{i0} \leftarrow \chi_\beta \\ &= p_j + a.n + 2.f_{i0} \end{aligned} \tag{4.3}$$

The t_{i1} is computed as:

$$\begin{aligned} t_{i1} &= \bar{p}_j.(s_i + m) + 2.f_{i1},\ where\ f_{i1} \leftarrow \chi_\beta \\ &= a.s_i.s_j + a.n.s_i + a.m.s_j + a.m.n + 2.e_j.s_i \\ &\quad + 2.e_j.m + 2.s_i.f_{i0} + 2.m.f_{i0} + 2.f_{i1} \end{aligned} \tag{4.4}$$

Similarly, we can see that the adversary act as user U_j does not get any information about the private key s_i of the user U_i from the computed signal function output $w_{i1} = Cha(t_{i1})$. This is due to the reason that n, m and f_{i0} are parameters used in the computation of t_{i1} and the values of the coefficient of n, m and f_{i0} are randomly chosen and uniformly distributed over discrete Gaussian distribution χ_β. Thus, the above modification to Islam's key agreement protocol make it secure against improved-signal leakage attack.

Further, due to above modifications to Islam's key agreement protocol, there is change in the correctness condition of the modified protocol. The modified condition for correctness is described below.

4.3.1 Condition for correctness of the modified protocol :

From the modified Islam's protocol (see Table 4.3), it can be seen that for the correctness of the protocol following condition should be satisfied:
$\sigma_{i1} = \sigma'_{i1}$, $\sigma_{j1} = \sigma'_{j1}$, $\sigma_{i2} = \sigma'_{i2}$ and $\sigma_{j2} = \sigma'_{j2}$.
Now, it can be observed that in computation of σ_{i1}, σ'_{i1}, σ_{j1} and σ'_{j1}, the preprocessed public keys $\bar{p}_i$ and $\bar{p}_j$ are used (during the computation of t_{i1} and t_{j1} which are used in computation

of σ_{i1}, $\sigma_{i1}^{'}$, σ_{j1} and $\sigma_{j1}^{'}$). Due to this preprocessing of public keys, the computation of σ_{i1}, $\sigma_{i1}^{'}$, σ_{j1} and $\sigma_{j1}^{'}$ consist of extra terms in its computation than the computation of σ_{i2}, $\sigma_{i2}^{'}$, σ_{j2} and $\sigma_{j2}^{'}$. Thus, for $\sigma_{i1} = \sigma_{i1}^{'}$ and $\sigma_{j1} = \sigma_{j1}^{'}$, the correctness condition require larger error tolerance than for the $\sigma_{i2} = \sigma_{i2}^{'}$ and $\sigma_{j2} = \sigma_{j2}^{'}$. Thus, we derive the condition for $\sigma_{i1} = \sigma_{i1}^{'}$ and $\sigma_{j1} = \sigma_{j1}^{'}$ and it will also satisfies the correctness condition for $\sigma_{i2} = \sigma_{i2}^{'}$ and $\sigma_{j2} = \sigma_{j2}^{'}$.

Now, trivially it can be seen that the correctness condition for $\sigma_{i1} = \sigma_{i1}^{'}$ and $\sigma_{j1} = \sigma_{j1}^{'}$ are same (as number of terms in computation of t_{i1} and t_{j1} are same, which are further used in computation of σ_{i1}, $\sigma_{i1}^{'}$, σ_{j1} and $\sigma_{j1}^{'}$). So we will derive the correctness condition for $\sigma_{i1} = \sigma_{i1}^{'}$ and readers can derive similarly the correctness condition for $\sigma_{j1} = \sigma_{j1}^{'}$.
The values of σ_{i1} and $\sigma_{i1}^{'}$ are computed from t_{i1} and t_{j1} receptively. The t_{i1} and t_{j1} are computed as:

$$\begin{aligned} t_{i1} &= \bar{p}_j.(s_i+m)+2.f_{i1},\ where\ f_{i1} \leftarrow \chi_\beta \\ &= a.s_i.s_j + a.n.s_i + a.m.s_j + a.m.n + 2.e_j.s_i \\ &\quad + 2.e_j.m + 2.s_i.f_{i0} + 2.m.f_{i0} + 2.f_{i1} \end{aligned} \tag{4.5}$$

$$\begin{aligned} t_{j1} &= \bar{p}_i.(s_j+n)+2.f_{j1},\ where\ f_{j1} \leftarrow \chi_\beta \\ &= a.s_i.s_j + a.n.s_i + a.m.s_j + a.m.n + 2.e_i.s_j \\ &\quad + 2.e_i.n + 2.s_j.f_{j0} + 2.n.f_{j0} + 2.f_{j1} \end{aligned} \tag{4.6}$$

Now, subtracting the equation 4.5 and equation 4.6, we get

$$\begin{aligned} t_{i1} - t_{j1} &= 2.e_j.s_i + 2.e_j.m + 2.s_i.f_{i0} + 2.m.f_{i0} \\ &\quad + 2.f_{i1} - 2.e_i.s_j - 2.e_i.n - 2.s_j.f_{j0} \\ &\quad - 2.n.f_{j0} - 2.f_{j1} \end{aligned} \tag{4.7}$$

$$\begin{aligned} ||t_{i1} - t_{j1}|| &= 2.\{||e_j.s_i|| + ||e_j.m|| + ||s_i.f_{i0}|| \\ &\quad + ||m.f_{i0}|| + ||f_{i1}|| - ||e_i.s_j|| \\ &\quad - ||e_i.n|| - ||s_j.f_{j0}|| - ||n.f_{j0}|| \\ &\quad - ||f_{j1}||\} \end{aligned} \tag{4.8}$$

Now applying Lemma 1 and Lemma 2 of [1] to the above equation, we get

$$\begin{aligned} ||t_{i1} - t_{j1}|| &\leq 2(8\beta^2.n^{3/2} + 2.\beta\sqrt{n}) \\ &\leq (16\beta^2.n^{3/2} + 4.\beta\sqrt{n}) \end{aligned} \tag{4.9}$$

Applying Lemma 4 of [1] to above equation, we get

$$\begin{aligned} \frac{q}{4} &> 2(8\beta^2.n^{3/2} + 2.\beta\sqrt{n}) \\ q &> 8(8\beta^2.n^{3/2} + 2.\beta\sqrt{n}) \\ q &> 16(4\beta^2.n^{3/2} + \beta\sqrt{n}) \end{aligned} \tag{4.10}$$

Thus, above is the condition for correctness of the modified protocol. This implies that with $q > 16(4\beta^2.n^{3/2} + \beta\sqrt{n})$ as an odd prime number, the party U_i and U_j will generate the same session *sk* by following the modified Islam's protocol.

4.4 Summary

This chapter shows that Islam's two-party key agreement is vulnerable to the improved-signal leakage attack (i-SLA). Using i-SLA, the attacker can successfully recover the long-term private key of the honest user by instantiating the utmost q number of key exchange sessions with the honest user using q number of malformed public keys. To overcome the attack, we provide a countermeasure to the existing Islam protocol and hence, the modified two-party key agreement protocol. The correctness condition of the modified two-party key agreement protocol has also been provided.

Chapter 5

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

The RLWE-based three-party key exchange (3PAKE) protocols are vulnerable to signal leakage attacks if their public/private keys are reused. Also, the design of the RLWE-based 3PAKE protocols are pretty complex, thus making these protocols highly inefficient. Therefore, in this chapter a simple lattice-based three-party password authenticated key exchange (SL3PAKE) protocol has been proposed. The protocol is simple in its design and resists signal leakage attack if its public/private keys are reused. Further, the provable security of the proposed protocol has been proved using Abdalla et al's ROR model.

Lastly, the comparative analysis based on communication overhead among the proposed SL3PAKE and other three-party protocols has been presented. In the analysis, it has been shown that the proposed SL3PAKE protocol has much less communication overhead and fewer communication rounds than the other three-party protocols. Thus, the proposed SL3PAKE protocol is more efficient than the other three-party protocols for the post-quantum world.

5.1 Background

Motivated to propose the 3PAKE that can resist quantum-attacks, Xu et al. [14] in 2017, proposed the first three-party password-authenticated key exchange (3PAKE) using the RLWE problem [8] of lattices. The security of the protocol is proved in Random Oracle Model (ROM). Further, the implementation of the protocol has been done using the LatticeCrypto library, and it is claimed to be efficient for practical applications. However, despite the above claims, Choi et al. [15] in 2018, find the Xu et al.'s protocol complicated and is not computationally efficient.

Further, it proposes a new three-party key exchange protocol named "AtLast." AtLast is a simple three-party PAKE protocol based on the RLWE problem of lattices. The protocol in-

herits the design of Ding et al.'s [21] RLWE-PPK protocol to three-party settings by using the generic approach of Abdalla et al. [58]. Further, the protocol is claimed to be quantum-secure and resist various known attacks. Apart from the above claims, we find that the proposed protocol is prone to signal leakage attack [34] if its public/private keys are reused. In 2019, Liu et al. [16] proposed the provably secure three-party password-authenticated Key Exchange (3PAKE) using the RLWE problem. The protocol's security is only dependent on the hardness of the RLWE problem and is independent of any external cryptographic primitives. The provable security of the protocol is proved using the modified Bellare et al.'s [60, 61] model. But we find that the protocol's design gets extra complicated to provide the authentication of the participating party. Also, the protocol is vulnerable to modified signal leakage attacks (see section 6 of [13]), if its public/private keys are reused. Therefore, we have proposed the simple lattice-based three-party password-authenticated key exchange (SL3PAKE). As from the name of the protocol, the protocol is simple in its design and resists signal leakage attacks if its public/private keys are reused.

5.2 Proposed SL3PAKE Protocol

In this section, the proposed SL3PAKE protocol has been described along with the correctness condition of the proposed protocol.

5.2.1 Description of the protocol

Let q be an odd prime number such that $q = 1 \ mod \ 2n$ and a be random element such that $a \in R_q$. It is assumed that all the parties involved in the key exchange are aware of the element a and identity ID_S of the server . Furthermore, the hashes of the participants' passwords are saved in the server's database in advance via some secure channels. Let χ_β be a discrete Gaussian distribution with a standard deviation of β. Let $h_0(), h_1() \ and \ h_2()$ be the hash functions such that

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

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

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

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.

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

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 SL3PAKE protocol is based on

Table 5.1: Simple Lattice-based Three-party Password Authenticated Key Exchange

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 abort, else 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 abort, else continue
		$x_S = a.s_S + 2e_S,\ where\ s_S, e_S \leftarrow \chi_\beta$
		$c_A = x_B^{'}.s_S + 2.f_{S4},\ where\ f_{S4} \leftarrow \chi_\beta$
		$c_B = x_A^{'}.s_S + 2.f_{S5},\ where\ f_{S5} \leftarrow \chi_\beta$
		$m = h_2(ID_S, ID_A, x_s, x_A^{'})$
		$n = h_2(ID_S, ID_B, x_s, x_B^{'})$
		$k_{SA} = (x_A^{'}.s_S + 2.m).m + 2.f_{S1}$
		$w_{SA} = Cha(k_{SA})$
		$\sigma_{SA} = Mod_2(k_{SA}, w_{SA})$
		$\alpha_{SA} = h_1(ID_A, ID_B, ID_S, c_A, x_A^{'}, \sigma_{SA})$
		$k_{SB} = (x_B^{'}.s_S + 2.n).n + 2.f_{S3}$
		$w_{SB} = Cha(k_{SB})$
		$\sigma_{SB} = Mod_2(k_{SB}, w_{SB})$
		$\alpha_{SB} = h_1(ID_A, ID_B, ID_S, c_B, x_B^{'}, \sigma_{SB})$
		$\xleftarrow{\{c_A, c_B, x_S, w_{SA}, w_{SB}, \alpha_{SA}, \alpha_{SB}\}}$
	$n = h_2(ID_S, ID_B, x_s, x_B)$	
	$k_{BS} = (x_S.s_B + 2.n).n + 2.f_{B1}$	
	$\sigma_{BS} = Mod_2(k_{BS}, w_{SB})$	
	$\alpha_{BS} = h_1(ID_A, ID_B, ID_S, c_B, x_B, \sigma_{BS})$	
	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		
$v_{AB} = c_A.s_A + 2.f_{A2},\ where\ f_{A2} \leftarrow \chi_\beta$		
$\sigma_{AB} = Mod_2(v_{AB}, w_{BA})$		
$sk_{AB} = h_1(ID_A, ID_B, ID_S, x_A^*, x_B^*, \sigma_{AB})$		

Ding et al.'s [2] error reconciliation mechanism. The proposed protocol presented in Table 5.1 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 $h_1(ID_A, ID_S, x_A', x_A^*)$. Then, it will check whether $h_{AS} \stackrel{?}{=} h_1(ID_A, ID_S, x_A', x_A^*)$, if it is not equal then server S will abort the session otherwise it will continue. Similarly, server S will authenticate client B by computing $x_B' = x_B^* - h_0(pw_B)$ and $h_1(ID_B, ID_S, x_B', x_B^*)$. Then, it will check whether $h_{BS} \stackrel{?}{=} h_1(ID_B, ID_S, x_B', x_B^*)$, if it's not equal then server S will abort the session otherwise it'll continue.

4. After the successful authentication of client A and client B, the 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's not equal then server S will abort the session otherwise it'll 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's not equal then server S will abort the session otherwise it'll 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 $sk_{AB} == sk_{BA}$.

5.2.2 Condition for the correctness of the proposed SL3PAKE protocol

From the SL3PAKE protocol shown in Table 5.1, it can be seen that for the correctness of the protocol $\sigma_{AB} = \sigma_{BA}, \sigma_{SA} = \sigma_{AS}\ and\ \sigma_{BS} = \sigma_{SB}$. Now, trivially it can be seen that the computation of σ_{AB} and σ_{BA} includes extra terms as compared to computation of $\sigma_{AS}, \sigma_{SA}, \sigma_{BS}\ and\ \sigma_{SB}$. Thus, the correctness condition for $\sigma_{AB} = \sigma_{BA}$ requires larger error tolerance than for $\sigma_{AS} = \sigma_{SA}$ and $\sigma_{BS} = \sigma_{SB}$. Therefore, the condition of correctness for $\sigma_{AB} = \sigma_{BA}$ has been derived and this correctness condition is also true for $\sigma_{AS} = \sigma_{SA}$ and $\sigma_{BS} = \sigma_{SB}$. Now, the value of σ_{AB} and σ_{BA} are computed from v_{AB} and v_{BA} respectively. The v_{AB} and v_{BA} are computed as:

$$\begin{aligned} v_{AB} &= c_A.s_A + 2.f_{A2},\ where\ f_{A2} \leftarrow \chi_\beta \\ &= a.s_A.s_B.s_S + 2.e_B.s_S.s_A + 2.s_A.f_{S4} + 2.f_{A2} \end{aligned} \tag{5.1}$$

$$\begin{aligned} v_{BA} &= c_B.s_B + 2.f_{B2},\ where\ f_{B2} \leftarrow \chi_\beta \\ &= a.s_A.s_B.s_S + 2.e_A.s_S.s_B + 2.s_B.f_{S5} + 2.f_{B2} \end{aligned} \tag{5.2}$$

Subtracting equation 5.1 and 5.2, we get

$$\begin{aligned} v_{BA} - v_{AB} = {} & 2.e_A.s_S.s_B + 2.s_B.f_{S5} + 2.f_{B2} \\ & - 2.e_B.s_S.s_A - 2.s_A.f_{S4} - 2.f_{A2} \end{aligned} \tag{5.3}$$

$$\begin{aligned} ||v_{BA} - v_{AB}|| = 2.\{&||e_A.s_S.s_B|| + ||s_B.f_{S5}|| + ||f_{B2}|| \\ &- ||e_B.s_S.s_A|| - ||s_A.f_{S4}|| - ||f_{A2}||\} \end{aligned} \tag{5.4}$$

Applying Lemma 1 and Lemma 2, we get

$$\begin{aligned} ||v_{BA} - v_{AB}|| &< 2.(2.\beta^3.\sqrt[5]{n} + 2.\beta^2.\sqrt[3]{n} + 2.\beta.\sqrt{n}) \\ &< 4.(\beta^3.\sqrt[5]{n} + \beta^2.\sqrt[3]{n} + \beta.\sqrt{n}) \\ &< 4.\beta.\sqrt{n}(\beta^2.n^2 + \beta.n + 1) \end{aligned} \tag{5.5}$$

Now, applying Lemma 3 to above equation, we get

$$\begin{aligned} \frac{q}{8} &> 4.\beta.\sqrt{n}(\beta^2.n^2 + \beta.n + 1) \\ q &> 32.\beta.\sqrt{n}(\beta^2.n^2 + \beta.n + 1) \end{aligned} \tag{5.6}$$

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

5.3 Formal Security of the Proposed SL3PAKE protocol

Using the ROR model, this section has presented the formal security of the proposed SL3PAKE protocol.

Let SUCC denote the event of an adversary $\mathscr{A}$ correctly predicting the bit b of a flipped coin.

Definition 1: The advantage of an adversary $\mathscr{A}$ correctly predicting the bit b of the flipped coin is as follows::

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

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

Definition 2: As proved by Lyubashevsky et al. [8], an adversary $\mathscr{A}$ 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

$$Adv_{R_q}^{RLWE} \leq \varepsilon$$

, where ε is a low number.

Theorem 1: For a huge password and identity dictionary, as well as a wide range of hash algorithms $(h_0(), h_1(), h_2())$, large size of discrete Gaussian distribution space and small advantage of solving RLWE problem i.e. $Adv_{R_q}^{RLWE} \leq \varepsilon$, the proposed SL3PAKE protocol is a secure key exchange protocol.
Now, Let $Adv_P^{3PAKE}(\mathscr{A})$ represents the advantage of breaking the semantic security of the proposed SL3PAKE protocol P by an adversary $\mathscr{A}$, then

$$\begin{aligned} Adv_P^{3PAKE}(\mathscr{A}) \leq & \frac{q_{h0}^2}{q^n} + \frac{q_{h1}^2}{|RS_{h1}|} + \frac{(q_s + q_{exe})^2}{|RS_{\chi_\beta}|} \\ & + \frac{2.q_s}{|D_{IDA}| * |D_{pwA}| * |D_{IDB}| * |D_{pwB}|} \\ & + 2.Adv_{R_q}^{RLWE} \end{aligned} \tag{5.7}$$

where $q^n, |RS_{h1}|, |RS_{\chi_\beta}|, |D_{pwA}|/|D_{IDA}|, |D_{pwA}|/|D_{IDA}|$ are the $h_0()$ and $h_1()$ range spaces, Gaussian distribution range space $\chi\beta$, 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.

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 $\mathscr{A}$.

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

$$Adv_P^{3PAKE}(\mathscr{A}) = |2.Pr[SUCC_0] - 1| \tag{5.8}$$

Game G_1: The game G_1 simulate the eavesdropping attack using the execute oracle query (Execute(U_i, S_i)). Finally, adversary $\mathscr{A}$ 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 ID_S, σ_{AB}, and 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 $\mathscr{A}$ requires the parameters s_A, s_B, and s_S to drive σ_{AB}, and σ_{BA}. As a result, the eavesdropping attack has no effect on the chance of an adversary $\mathscr{A}$ winning the game. So,

$$Pr[SUCC_0] = Pr[SUCC_1] \tag{5.9}$$

Game G_2: An active attack using the send, execute, and hash oracle queries are simulated in the game G_2. This game attempts to identify collisions of hash values by simulating real-time hash functions. The three hash functions $h_0(), h_1()$, and $h_2()$ have been utilized in the proposed SL3PAKE protocol. Let q_{h0} and q_{h1} be the number of hash queries corresponding to hash function $h_0()$ *and* $h_1()$ respectively. The corresponding range space of $h_0()$ *and* $h_1()$ are q^n (where q is an odd prime number) and $|RS_{h1}|$ receptively. Now, as a result of the Birthday paradox, the risk of hash values colliding is $\leq \frac{q_{h0}^2}{2.q^n} + \frac{q_{h1}^2}{2.RS_{h1}}$. Also, the probability of collision of then parameters x_A^*, x_B^*, x_S by using send and execute oracle query is $\leq \frac{(q_s+q_{exe})^2}{2.|RS_{\chi_\beta}|}$ (using Birthday paradox), where 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

$$|Pr[SUCC_1]] - |Pr[SUCC_2]| \leq \frac{q_{h0}^2}{2.q^n} + \frac{q_{h1}^2}{2.|RS_{h1}|} + \frac{(q_s+q_{exe})^2}{2.|RS_{\chi_\beta}|} \tag{5.10}$$

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 will rule out one possible id and password combination. Mathematically, it can be represented as

$$|Pr[SUCC_2] - Pr[SUCC_3]| \leq \frac{q_s}{|D_{IDA}| * |D_{pwA}| * |D_{IDB}| * |D_{pwB}|} \tag{5.11}$$

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

Game G_4: The game G_4 simulates the corrupt server (CorruptS_j) oracle query, where adversary gets the long-term secret of a server S and the stored parameters of client A and B in the server's database. Now, the goal of an adversary to generate the past session key from the values of the intercepted parameters through execute oracle query. Thus, the adversary will get 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 $\mathscr{A}$ 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

$$|Pr[SUCC_3]] - [Pr[SUCC_4]| \leq Adv_{\mathbf{R}_q}^{RLWE} \tag{5.12}$$

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

$$Pr[SUCC_4] = \frac{1}{2} \tag{5.13}$$

From (5.8), we have

$$\begin{aligned} Adv_P^{3PAKE}(\mathscr{A}) &= 2.|Pr[SUCC_0] - \frac{1}{2}| \\ &= 2.|Pr[SUCC_0] - Pr[SUCC_4]| \end{aligned} \tag{5.14}$$

By using triangular inequality and using equation (5.9), we get

$$\begin{aligned} Adv_P^{3PAKE}(\mathscr{A}) \leq 2.(&|Pr[SUCC_1] - Pr[SUCC_2]| \\ &+ |Pr[SUCC_2] - Pr[SUCC_3]| \\ &+ |Pr[SUCC_3] - Pr[SUCC_4]|) \end{aligned} \tag{5.15}$$

putting the values from (5.10) - (5.13), we get:

$$\begin{aligned} Adv_P^{3PAKE}(\mathscr{A}) \leq 2.(&\frac{q_{h0}^2}{2.q^n} + \frac{q_{h1}^2}{2.|RS_{h1}|} + \frac{(q_s + q_{exe})^2}{|2.RS_{\chi_\beta}|} \\ &+ \frac{q_s}{|D_{IDA}| * |D_{pwA}| * |D_{IDB}| * |D_{pwB}|} \\ &+ Adv_{R_q}^{RLWE}) \end{aligned} \tag{5.16}$$

$$\begin{aligned} Adv_P^{3PAKE}(\mathscr{A}) \leq &\frac{q_{h0}^2}{q^n} + \frac{q_{h1}^2}{|RS_{h1}|} + \frac{(q_s + q_{exe})^2}{|RS_{\chi_\beta}|} \\ &+ \frac{2.q_s}{|D_{IDA}| * |D_{pwA}| * |D_{IDB}| * |D_{pwB}|} \\ &+ 2.Adv_{R_q}^{RLWE} \end{aligned} \tag{5.17}$$

hence proved.

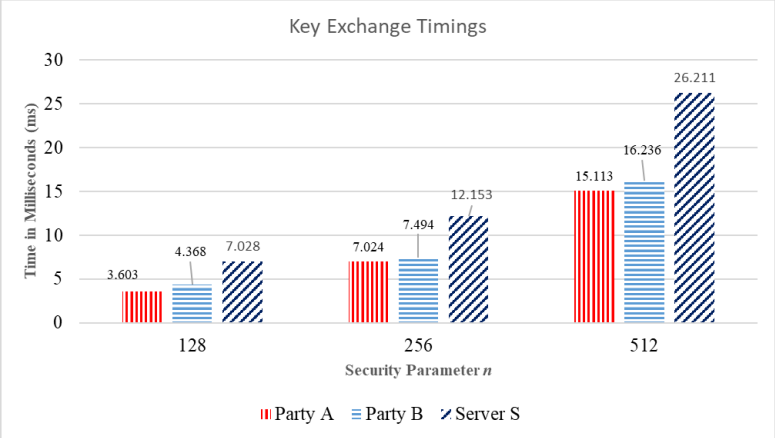


Figure 5.1: Key exchange timings of the proposed SL3PAKE protocol

5.4 Concrete parameters & implementation of SL3PAKE

In this section, the concrete choices of parameters and the performance analysis of the proposed SL3PAKE protocol have been presented. The performance analysis has been done with {512, 256, 128} values of security parameter n. Also, the comparative analysis based on communication cost among the proposed SL3PAKE protocol and the other three party protocols has been done.

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

As stated above, the chosen values of n are {512, 256, 128} such that $n = 2^k$, where k is any positive integer. The χ_β is a discrete Gaussian distribution with standard deviation $\beta = \frac{8}{2\sqrt{2\pi}} \approx 1.596$. From section 5.2.2, the condition for the correctness of the proposed protocol is $q > 32.\beta.\sqrt{n}(\beta^2.n^2 + \beta.n + 1)$. 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.

Now, with the above parameters of choice, the "classical security level of the proposed SL3PAKE protocol has been computed using LWE estimator. LWE estimator indicates the level of security provided by the particular sets of parameters in the LWE/RLWE based cryptosystem. This work is accomplished by calculating the attack complexities of the decoding, BKW, lattice reduction, meet-in-the-middle, reducing BDD to unique-SVP, and exhaustive search attacks. The LWE estimator estimates both the time and space complexities of the above attacks" [59]. The sage commands screenshot to calculate the classical security level of SL3PAKE protocol using LWE estimator has been shown in Figure 5.2.

Table 5.2: Cryptographic operations running time (in milliseconds) for $\beta = 1.5965$ and $q = 1931502101$

Cryptographic 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

The output of sage commands presented in Figure 5.2 shows that the proposed SL3PAKE provides 55-bit of classical security with the above parameter of choice.

```
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"])
```

Figure 5.2: Sage commands to calculate classical security level using LWE estimator

The representation of various cryptographic operations are:

- T_{samp} shows the average time taken to sample an element x from the discrete Gaussian distribution χ_β.
- T_{rmul} represents the average time spent for the multiplication of two polynomial elements c, d where $c \in R_q$ and d sampled 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 configurations for the performance analysis of the proposed SL3PAKE are as follows: The client and server run on Intel(R)Core(TM)i5-7200U processor with 8 GB RAM. The hash

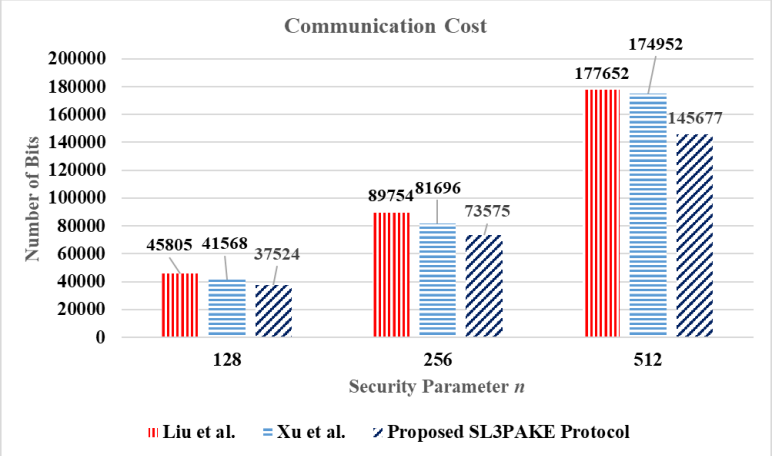


Figure 5.3: Comparative Analysis based on Communication Cost

function $h_1()$ output has been produced using the SHA3-224() hash function. The computational time of the cryptographic operations are shown in Table 5.2.

The average key exchange timings of Party A, Party B, and Server S for the proposed SL3PAKE protocol have been presented in Figure 5.1. From the Figure, it can be seen that the proposed SL3PAKE protocol is more server computational intensive rather than user computational intensive. This behavior is quite favorable as, in most 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.

Finally, the communication cost of the proposed SL3PAKE protocol has been computed. From the Table 5.1, it can be seen that the SL3PAKE protocol has four rounds of communications 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 $9n\log_2(q) + 4n + 6 \times 224 + 4 \times 32 = 9n\log_2(q) + 4n + 1472$ bits.

Similarly, the communication complexity of Xu et al. [14] and Liu et al. [16] are $10n\log_2(q) + 5n + 1440$ and $11n\log_2(q) + 4n + 1856$ bits receptively. Now, with the above communication complexities of the proposed protocol and other two protocols, the comparative analysis based on communication cost has been done and presented in Figure 5.3 for $n = \{128, 256, 512\}$ *and* $q =$ 1931502101. From the Figure 5.3, it can be seen that the communication cost of the proposed SL3PAKE protocol is much less than the Xu et al.'s [14] and Liu et al.'s [16] protocol. Thus, the proposed SL3PAKE protocol is more efficient than the other three-party protocol. Besides, the proposed SL3PAKE protocol only needs four communication rounds while Xu et al.'s [14] protocol needs six communication rounds, and Liu et al.'s [16] needs seven communication

rounds.

5.5 Summary

In this chapter, a new 3PAKE protocol termed as SL3PAKE has been proposed. The protocol is simple and resists signal leakage attacks if its public/private keys are reused. Furthermore, the provable security of the protocol has been proved in the ROR model. Again, the concrete choice of parameters and implementation result shows that the protocol is efficient and favorable for client/server scenarios. Finally, the communication cost of the protocol has been calculated. From the computation and communication calculation, it can be seen that the proposed SL3PAKE protocol is efficient to be used in real-world applications.

Chapter 6

Conclusion and Future Scope

This Chapter presents the summary of the research work and Future scope. Some significant findings of research works are also pointed here. Section 6.1 introduces the conclusion of the research work and Section 6.2 discusses the Future scope.

6.1 Conclusion

This thesis introduces the key exchange protocols designed using LWE/RLWE of lattices. These key exchange protocols claim to resist classical and quantum attacks. Moreover, the detailed literature survey related to LWE/RLWE-based key exchange protocols has been discussed. From the literature survey, it has been found that the LWE/RLWE-based key exchange protocols have security vulnerability if their public/private keys are reused. Due to key reuse, these key exchange protocols are vulnerable to SLA and key mismatch attack. The signal leakage attack (SLA) is the most severe among the two attacks, and almost all the RLWE-based key exchange protocols are vulnerable to this attack. Therefore, the main motive of this work is to design an efficient key exchange technique that will resist the signal leakage attack if its public/private is reused. To achieve the above objective, the RLWE-based key exchange protocols have been proposed for different scenarios. These scenarios are discussed bellows as.

In the first type, the Lattice-based Anonymous Password Authenticated Key Exchange protocol for mobile devices (LBA-PAKE) has been proposed. The proposed protocol resists signal leakage attack (SLA) if its public/private keys are reused. The idea to prevent the SLA attack is to verify the public keys using the direct validation technique of Ding et al. [57]. The formal security analysis has been done using Abdalla et al.'s ROR model. The implementation parameters of choice offer 100 bits of classical and 75 bits of quantum security. The computation timings also validate the proposed protocol in terms of efficiency for the mobile client-server environment.

In the second type, the modified two-party authenticated key agreement (m-2PAKA) protocol has been proposed for post-quantum world. This protocol is the improvement of Islam's

provably secure two-party authenticated key agreement (2PAKA) protocol. By the cryptanalysis of Islam's 2PAKA protocols, it has been found that the protocol is vulnerable to improved-signal leakage attack (i-SLA) if its public/private keys are reused. Hence, the modified two-party authenticated key agreement (m-2PAKA) protocol has been proposed to counter the i-SLA attack. The proposed m-2PAKA protocol inherits the basic design of Islam's protocol, with an additional countermeasure to resist the i-SLA attack.

In the third type, a Simple Lattice-based Three-party Password Authenticated Key Exchange (SL3PAKE) protocol has been proposed for the post-quantum world. The protocol is simple and resists signal leakage attacks if its public/private keys are reused. Furthermore, the provable security of the protocol has been proved in the ROR model. Again, the concrete parameters and implementation results show that the protocol is efficient and favorable for client/server scenarios. Finally, the comparative analysis based on communication cost has been done. From the analysis, it can be seen that the proposed SL3PAKE protocol is efficient to be used in real-world applications.

6.2 Future Scope

The above key exchange protocols constitute a step towards proposing provably secure key exchange protocols in lattice-based cryptography. The future of the quantum secure key exchange schemes can be seen in providing security to the modern smart infrastructure. This will include the modern infrastructure like smart grid, smart city, smart healthcare, smart traffic etc. It will not be an exaggeration to say that quantum secure key exchange schemes will replace the existing key exchange schemes everywhere, leading to a quantum secure post-quantum world.

References

[1] SK Hafizul Islam. Provably secure two-party authenticated key agreement protocol for post-quantum environments. *Journal of Information Security and Applications*, 52:102468, 2020.

[2] Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. *IACR Cryptology EPrint Archive*, 2012:688, 2012.

[3] Chris Peikert. Lattice cryptography for the internet. In *International Workshop on Post-Quantum Cryptography*, pages 197–219. Springer, 2014.

[4] Qi Feng, Debiao He, Sherali Zeadally, Neeraj Kumar, and Kaitai Liang. Ideal lattice-based anonymous authentication protocol for mobile devices. *IEEE Systems Journal*, 13(3):2775–2785, 2018.

[5] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[6] Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*, 1996.

[7] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[8] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer, 2010.

[9] Aparna Kumari and Sudeep Tanwar. A secure data analytics scheme for multimedia communication in a decentralized smart grid. *Multimedia Tools and Applications*, pages 1–26, 2021.

[10] Muzafer Saračević, Saša Adamović, Nemanja Maček, Mohamed Elhoseny, and Shahenda Sarhan. Cryptographic keys exchange model for smart city applications. *IET Intelligent Transport Systems*, 14(11):1456–1464, 2020.

[11] MK Bhuyan, Debanga Raj Neog, and Mithun Kumar Kar. Fingertip detection for hand pose recognition. *International Journal on Computer Science and Engineering*, 4(3):501, 2012.

[12] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[13] Jintai Ding, Scott Fluhrer, and Saraswathy Rv. Complete attack on rlwe key exchange with reused keys, without signal leakage. In *Australasian Conference on Information Security and Privacy*, pages 467–486. Springer, 2018.

[14] Dongqing Xu, Debiao He, Kim-Kwang Raymond Choo, and Jianhua Chen. Provably secure three-party password authenticated key exchange protocol based on ring learning with error. *IACR Cryptol. ePrint Arch.*, 2017:360, 2017.

[15] Rakyong Choi, Hyeongcheol An, and Kwangjo Kim. 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.

[16] Chao Liu, Zhongxiang Zheng, Keting Jia, and Qidi You. Provably secure three-party password-based authenticated key exchange from rlwe. In *International Conference on Information Security Practice and Experience*, pages 56–72. Springer, 2019.

[17] Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. Authenticated key exchange from ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 719–751. Springer, 2015.

[18] Hugo Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In *Annual International Cryptology Conference*, pages 546–566. Springer, 2005.

[19] Andrew Chi-Chih Yao and Yunlei Zhao. Oake: a new family of implicitly authenticated diffie-hellman protocols. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1113–1128. ACM, 2013.

[20] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Annual international cryptology conference*, pages 232–249. Springer, 1993.

[21] Jintai Ding, Saed Alsayigh, Jean Lancrenon, RV Saraswathy, and Michael Snook. Provably secure password authenticated key exchange based on rlwe for the post-quantum world. In *Cryptographers' Track at the RSA Conference*, pages 183–204. Springer, 2017.

[22] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 156–171. Springer, 2000.

[23] Philip MacKenzie. The pak suite: Protocols for password-authenticated key exchange. *Contributions to IEEE P*, 1363(2), 2002.

[24] Xinwei Gao, Jintai Ding, Lin Li, and Jiqiang Liu. Practical randomized rlwe-based key exchange against signal leakage attack. *IEEE Transactions on Computers*, (1):1–1, 2018.

[25] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, and Elaine Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Booz-Allen and Hamilton Inc Mclean Va, 2001.

[26] Vivek Dabra, Anju Bala, and Saru Kumari. Lba-pake: Lattice-based anonymous password authenticated key exchange for mobile devices. *IEEE Systems Journal*, 2020.

[27] Ran Canetti and Hugo Krawczyk. Security analysis of ike's signature-based key-exchange protocol. In *Annual International Cryptology Conference*, pages 143–161. Springer, 2002.

[28] Joppe W Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 553–570. IEEE, 2015.

[29] Xinwei Gao, Jintai Ding, RV Saraswathy, Lin Li, and Jiqiang Liu. Comparison analysis and efficient implementation of reconciliation-based rlwe key exchange protocol. Technical report, Cryptology ePrint Archive, Report 2017/1178, 2017. http://eprint. iacr. org . . . , 2017.

[30] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. 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*, pages 1006–1018. ACM, 2016.

[31] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange-a new hope. In *USENIX Security Symposium*, volume 2016, 2016.

[32] Experimenting with post-quantum cryptography. `https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html`. Last accessed: 2020-07-14.

[33] Markku-Juhani O Saarinen. Hila5: On reliability, reconciliation, and error correction for ring-lwe encryption. In *International Conference on Selected Areas in Cryptography*, pages 192–212. Springer, 2017.

[34] Jintai Ding, Saed Alsayigh, RV Saraswathy, Scott Fluhrer, and Xiaodong Lin. Leakage of signal function with reused keys in rlwe key exchange. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[35] Chao Liu, Zhongxiang Zheng, and Guangnan Zou. Key reuse attack on newhope key exchange protocol. In *International Conference on Information Security and Cryptology*, pages 163–176. Springer, 2018.

[36] Scott R Fluhrer. Cryptanalysis of ring-lwe based key exchange with key share reuse. *IACR Cryptology ePrint Archive*, 2016:85, 2016.

[37] Aurélie Bauer, Henri Gilbert, Guénaël Renault, and Mélissa Rossi. Assessment of the key-reuse resilience of newhope. In *Cryptographers' Track at the RSA Conference*, pages 272–292. Springer, 2019.

[38] Yue Qin, Chi Cheng, and Jintai Ding. A complete and optimized key mismatch attack on nist candidate newhope. In *European Symposium on Research in Computer Security*, pages 504–520. Springer, 2019.

[39] Satoshi Okada, Yuntao Wang, and Tsuyoshi Takagi. Improving key mismatch attack on newhope with fewer queries. *IACR Cryptol. ePrint Arch.*, 2020:585, 2020.

[40] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based pke and kems. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 307–335, 2020.

[41] Jan Vacek and Jan Václavek. Key mismatch attack on newhope revisited.

[42] Daniel J Bernstein, Leon Groot Bruinderink, Tanja Lange, and Lorenz Panny. Hila5 pindakaas: on the cca security of lattice-based encryption with error correction. In *International Conference on Cryptology in Africa*, pages 203–216. Springer, 2018.

[43] Boru Gong and Yunlei Zhao. Small field attack, and revisiting rlwe-based authenticated key exchange from eurocrypt'15. *IACR Cryptology ePrint Archive*, 2016:913, 2016.

[44] Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 27–47. Springer, 2011.

[45] Muhammad Khurram Khan, Saru Kumari, and Pitam Singh. Cryptanalysis of an'efficient-strong authentiction protocol (e-sap) for healthcare applications using wireless medical sensor networks'. *KSII Transactions on Internet and Information Systems (TIIS)*, 7(5):967–979, 2013.

[46] Hugo Krawczyk. Sigma: The 'sign-and-mac'approach to authenticated diffie-hellman and its use in the ike protocols. In *Annual International Cryptology Conference*, pages 400–425. Springer, 2003.

[47] R Madhusudhan et al. A secure and lightweight authentication scheme for roaming service in global mobile networks. *Journal of information security and applications*, 38:96–110, 2018.

[48] Lu Zhou, Xiong Li, Kuo-Hui Yeh, Chunhua Su, and Wayne Chiu. Lightweight iot-based authentication scheme in cloud computing circumstance. *Future generation computer systems*, 91:244–251, 2019.

[49] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 35–54. Springer, 2013.

[50] Vikram Singh. A practical key exchange for the internet using lattice cryptography. *IACR Cryptology ePrint Archive*, 2015:138, 2015.

[51] Mohammad Wazid, Ashok Kumar Das, Rasheed Hussain, Giancarlo Succi, and Joel JPC Rodrigues. Authentication in cloud-driven iot-based big data environment: Survey and outlook. *Journal of systems architecture*, 97:185–196, 2019.

[52] Wiem Tounsi, Joaquin Garcia-Alfaro, Nora Cuppens-Boulahia, and Frédéric Cuppens. Securing the communications of home health care systems based on rfid sensor networks. In *2010 8th Annual Communication Networks and Services Research Conference*, pages 284–291. IEEE, 2010.

[53] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Annual International Cryptology Conference*, pages 537–554. Springer, 1999.

[54] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 453–474. Springer, 2001.

[55] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016.

[56] Daniel Kirkwood, Bradley C Lackey, John McVey, Mark Motley, Jerome A Solinas, and David Tuller. Failure is not an option: standardization issues for post-quantum key agreement. In *Talk at NIST workshop on Cybersecurity in a Post-Quantum World: http://www. nist. gov/itl/csd/ct/post-quantum-crypto-workshop-2015. cfm*, volume 2, 2015.

[57] Jintai Ding, RV Saraswathy, Saed Alsayigh, and Crystal Clough. How to validate the secret of a ring learning with errors (rlwe) key. *IACR Cryptology ePrint Archive*, 2018:81, 2018.

[58] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In *International Workshop on Public Key Cryptography*, pages 65–84. Springer, 2005.

[59] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[60] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: the three party case. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 57–66, 1995.

[61] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *International conference on the theory and applications of cryptographic techniques*, pages 139–155. Springer, 2000.