Quantum-Safe Key Management System for Cloud Computing Using a Hybrid Framework

By

Mariya Amin
(Registration No: 00000328960)

Department of Information Security

Military College of signals

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

(2024)

Quantum-Safe Key Management System for Cloud Computing Using a Hybrid Framework

By

Mariya Amin

(Registration No: 00000328960)

A thesis submitted to the National University of Science and Technology, Islamabad,

in partial fulfillment of the requirements for the degree of

Masters in

Information Security

Supervisor: Assoc. Prof. Dr. Shahzaib Tahir

Co-Supervisor: Asst. Prof. Dr. Fawad

Military College of Signals

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

(2024)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS Thesis written by **Ns Mariya Amin**, Registration No. **00000328960**, of **Military College of Signals, NUST** has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations/MS Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members and local evaluators of the scholar have also been incorporated in the said thesis.

Signature: [illegible]

Name of Supervisor: **Assoc. Prof. Dr. Shahzaib Tahir**

Date: 28/10/24

Signature (HOD): [illegible]

Date: 28/10/24

Signature (Dean/Principal) [illegible]

Date: 28/10/24

Brig
Dean, MCS (NUST)
(Asif Masood, Phd)

Form TH-4

NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY

MASTER THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by **NS Mariya Amin, MSIS-19** Regn. No. **00000328960** Titled: **"Quantum-Safe Key Management System for Cloud Computing Using a Hybrid Framework** be accepted in partial fulfillment of the requirements for the award of **MS Information Security** degree.

Examination Committee Members

1. Name: **Asst. Prof. Dr. Fawad Khan** Signature: for [illegible]

2. Name: **Major Bilal Ahmed** Signature: [illegible]

Supervisor's Name: **Assoc. Prof. Dr. Shahzaib Tahir** Signature: [illegible]

Date:

[illegible]
HoD
Information Security
Military College of Sigs
Head of Department

28/10/24
Date

COUNTERSIGNED

Date: 28/10/24

[illegible]
Brig
Dean, MCS (NUST)
Asif Masood, Phd)
Dean

CERTIFICATE OF APPROVAL

This is to certify that the research work presented in this thesis, entitled **"Quantum- Safe Key Management System for Cloud Computing Using a Hybrid Frame- work"** was conducted by **Ms. Mariya Amin** under supervision of **Assoc. Prof. Dr. Shahzaib Tahir**. No part of this thesis has been submitted anywhere else for any other degree. This thesis is submitted to the **Information Security** Department in partial fulfillment of the requirements for the degree of **Master of Science** in Field of **Information security**, Department of Military College of Signals, National University of Sciences and Technology, Islamabad.

Student Name: Mariya Amin Signature: [illegible]

Examination Committee:

a) External Examiner 1: Asst. Prof. Dr. Fawad Khan Signature: [illegible]

(Designation and Office Address)

____

b) External Examiner 2: Major Bilal Ahmed Signature: [illegible]

(Designation and Office Address)

____

Name of Supervisor: Assoc. Prof. Dr. Shahzaib Tahir Signature: [illegible]

Name of Dean/HOD: Assoc. Prof. M. Faisal Amjad Signature: [illegible]

HoD
Information Security
Military College of Sigs

28/10/21

AUTHOR'S DECLARATION

I **Mariya Amin** hereby state that my MS thesis titled **"Quantum-Safe Key Management System for Cloud Computing Using a Hybrid Framework"** is my own work and has not been submitted previously by me for taking any degree from National University of Sciences and Technology, Islamabad or anywhere else in the country/ world.

At any time if my statement is found to be incorrect even after I graduate, the university has the right to withdraw my MS degree.

Student Signature: [illegible]

Name: Mariya Amin

Date: 28/10/24

PLAGIARISM UNDERTAKING

I solemnly declare that research work presented in the thesis titled **"Quantum-Safe Key Management System for Cloud Computing Using a Hybrid Framework"** is solely my research work with no significant contribution from any other person. Small contribution/ help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and National University of Sciences and Technology (NUST), towards plagiarism. Therefore, I, as an author of the above titled thesis, declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS degree, the University reserves the rights to withdraw/revoke my MS degree and that HEC and NUST, has the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized thesis.

Student Signature: [illegible]

Name: Mariya Amin

Date: 28/10/24

DEDICATION

I dedicated this thesis to my Family, Teachers and Friends for their sincere love, support and continuous encouragement.

ACKNOWLEDGEMENT

Praise and glory be to the Glorious and Merciful Almighty Allah. I am very grateful to Almighty Allah for blessing me with the ability and confidence to pursue this research, despite the hindrances that tried to shake my courage. I am deeply thankful to my family, especially my parents, for being the driving force behind me and encouraging me to grow further in life. I extend my hearty vote of thanks to my Supervisor, Assoc. Prof. Dr. Shahzaib Tahir, who supervised me in such a polite, humble, and helpful manner throughout this research. His facilitation and counseling have been an invaluable source of support, and I will continue to value them in the coming years. I am also deeply thankful to my esteemed committee members, Asst. Prof. Dr. Fawad Khan and Major Bilal Ahmed, for their insightful feedback.

Contents

List of Tables

List of Figures

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

SVP	Shortest Vector Problem
DKMS	Distributed Key Management System
SSS	Shamir Secret Sharing Scheme
LBC	Lattice Based Cryptography
LBSSS	Lattice Based Shamir Secret Sharing Scheme
NIST	National Institute of Standards and Technology
LWE	Learning with Error
QSC	Quantum Secure Cryptography
CVP	Closest Vector Problem
CSP	Cloud Service Provider
DEK	Data Encryption Key
SSH	Secure Shell/Secure Socket Shell
mTLS	Mutual Transport Layer Security
ABE	Attribute Based Encryption
IBE	Identity Based Encryption
PKC	Public Key Cryptography
ECC	Elliptic Curve Cryptography
PQC	Post Quantum Cryptography
$\mathbb{Z}^n$	N-dimensional integer lattice/set of integers modulo n
$\Lambda/\mathcal{L}$	Lattice structure
$\mathbb{R}^n$	N-dimensional Euclidean space over the real numbers
$\lVert \cdot \rVert$	Norm of vector
γ	Approximation factor or the Hermite constant

Abstract

Advantages of Cloud Computing include its flexibility, scalability, cost savings, and around-the-clock availability with the latest features. It's a cloud-based remote access paradigm that allows users to access computer resources like servers, storage, and apps. Storage service, a key offering in cloud computing, provides secure and backed-up data across the globe, making it ideal for client organizations. However, cloud environments are often multi-tenant, raising concerns about data security, as both internal and external attacks can target confidential data stored in the cloud.Cloud computing providers do not guarantee complete data security, so client organizations prefer to secure data at their end by encrypting it before storing it in the cloud, requiring them to manage the encryption and decryption keys. A Key Management System (KMS) is essential for managing these keys and their associated metadata, including generation, distribution, storage, backup, recovery, and destruction. However, security issues like data confidentiality and unauthorized user access may arise.In my thesis, I have worked on a distributed Key Management System (KMS), including premise-based and cloud-based KMS, for managing cryptographic keys. My research combines traditional cryptographic algorithms with quantum cryptography primitive (Lattices)to secure keys against classical as well as quantum attacks while minimizing security concerns related to cloud-based KMS.

Keywords: Lattice Based Cryptography, Distributed KMS, Insider threats, Shortest Vector Problem, Quantum cryptography, Shamir Secret Sharing.

Chapter 1

INTRODUCTION

1.1 Background and Motivation For the Research

[23]There are a few concerns to consider when adopting the cloud paradigm. One of these is information safety, which is an essential part of the quality of service. The issue with security in public clouds is that due to the cloud's multi-tenant nature, where users share resources with many others, it's harder to keep things confidential.[26] Traditional security methods like encryption, digital signatures, and access control may not be enough to protect information in public clouds because of some limitations. IDC predicts [52] that by 2025, 59% of the world's data will be stored in public clouds. With more devices and people accessing the cloud, the chances of a data breach increase. When control of data is given to the cloud, more people can access it, making a breach more likely. With more parties, devices, and applications relying on the cloud than ever before, data is exposed to a greater risk of being compromised at various points of access. As more people gain access to the data, data breaches are more likely as a result of businesses giving the cloud access over their data.[41] The main challenge of cloud storage is ensuring that data owners can maintain control and management of outsourced data, as this responsibility often lies with service providers or third parties.[44] Verizon's 2024 Data Breach Investigation Report states that third-party providers, including hosting partners, software supply chains, and data custodians such cloud computing, were involved in 15 percent of breaches.

[19] Shor presented a quantum technique for tackling 'discrete logarithm' and 'integer factorization' issues in polynomial time, posing a challenge to the security of traditional public key cryptosystems. Consequently, Cloud-based Key Management Systems (KMS) have become crucial as they are also prone to distinct security challenges, notably concerning key storage vulnerabilities from insider attacks.Therefore, there's a need to build such KMS which can reduce these security concerns and also regain the trust of client organizations providing them with a partial role in key management.Based on the

examination of prior research conducted on cloud-based key management systems, there arises a necessity to construct key management systems that not only eleminate reliance on a single entity for key generation and administration but also thwart key hijacking and attacks initiated by quantum computers. This aim can be achieved by employing algorithms that exhibit resilience against threats related to quantum computing.

The proposed approach allows organizations to leverage the benefits of cloud-based key management while also maintaining control over certain critical aspects of key management within their own environment.This architecture also provides a balance between the convenience and scalability of cloud services and the security and control offered by on-premise solutions.

The proposed KMS uses hybrid approach by incorporating classical algorithm (Shamir Secret Sharing Scheme) with post-quantum primitive (Lattices) to improve the security of KMS. Post-quantum primitive is based on lattices whose security is believed to be hard to break even by using quantum computing.[19] Based on the difficulties of lattice problems such Ring Learning with Error (LWE), Closest Vector Problem (CVP), and Shortest Vector Problem (SVP), lattice-based cryptosystems are provably secure in the worst scenario.

By using lattice-based techniques into the Shamir's Secret Sharing scheme, we can enhance its resistance to potential future quantum attacks. Since, the Shamir Secret Sharing Scheme (SSS) depends on the hardness of solving particular mathematical problems—like the challenge of reconstructing the original polynomial from a subset of points—there is a chance that these these problems could be solved more quickly using quantum computers, lowering the scheme's security margin.

1.2 Problem Statement and Research Objectives

Centralizing key storage and generation in a single entity creates a high-value target for attackers. If this entity is compromised, all keys managed by the KMS could be at risk. It also increases the risk of insider threats as an insider with malicious intent could abuse their access to compromise keys, manipulate key generation processes, or even leak sensitive information. The reason why Shamir's Secret Sharing is considered to be secure because linear equations over a finite field are difficult to solve for classical computers. However, with the use of quantum algorithms and a quantum oracle, a quantum computer could solve these systems efficiently, potentially allowing a quantum adversary to recover the secret from a smaller subset of shares. In Shamir Secret Sharing scheme(SSS) generation of shares and the polynomial coefficients involve randomness and it is these random values that makes the scheme's security because of its unpredictability. If an attacker can predict the random values used in generating shares, they might be able to gain information about the secret.

The main objectives of the thesis are as follows:

- Develop a quantum secure distributed key management approach instead of relying on a centralized system.
- To reduce the concentration of keys in a single location or entity as well as dependency on the central entity.
- To ensure users' participation in key generation as well as Key management process.
- Devising a more robust cryptographic scheme to prevent attacks from quantum computers by incorporating lattice-based cryptographic primitive into Shamir Secret Sharing (SSS) scheme to enhance its resistance to potential future quantum attacks.

1.3 Overview of the Proposed Key Management System (KMS) using Lattice Shamir Secret Sharing Scheme

Key Management systems are not only responsible to manage the cryptographic keys used for the encryption and decryption of the data to be stored on the cloud and ensuring security of those keys from adversaries, be it insider or external.The proposed DKMS used improved version on Lattice Based Shamir Secret Sharing (LBSS) to enhance robustness of traditional SSS scheme for the shares residing at the key storage. DKMS employes strong cryptographic protocol such as mTLS to establish authenticated and encrypted communication channels between the on-premise and cloud- based KMS that prevents unauthorized entities from participating in the key transfer process.

1.3.1 Thesis Organization

The following is the arrangement of the thesis: The research is briefly summarized in **Chapter 1**, which includes objjetives and motivation of research, a brief description, background, and details about the proposed Key Management System (KMS) that uses the Lattice Shamir Secret Sharing Scheme (LBSSS). A literature review of cloud-based key management systems, security vulnerabilities, and quantum-resilient cryptography is provided in **Chapter 2**. **Chapter 3** ensures the theoretical groundwork, discussing Shamir Secret Sharing, lattice-based cryptography, and how these are integrated into LBSSS. **Chapter 4** describes the proposed DKMS, focusing on design and security considerations. **Chapter 5** compares Lattice Based Shamir Secret Sharing Scheme (LBSSS) with classical schemes and evaluates its security against insider and quantum threats. **Chapter 6** covers research results, deployment considerations, and future improvements. **Chapter 7** summarizes the research work, while **Chapter 8** brings the thesis to a conclusion..

Chapter 2

LITERATURE REVIEW

This chapter explores and reviews the existing literature on key management approaches in cloud environments. It critically examines the limitations of current Key Management Systems (KMSs) concerning key security, particularly in the context of insider threats and attacks. The chapter provides a evaluation of various existing schemes and protocols, discussing their advantages and disadvantages, along with a comprehensive analysis of their security and performance characteristics.

2.1 Survey of Existing Cloud-Based Key Management Systems

[7] Relies on the trustworthiness of the adapter (a separate entity from the cloud provider)and requires complex co-ordination and communication among the entities that share the parts of the encryption key, which can introduce additional overhead and latency. [9]Uses verifiable secret sharing schemes to distribute and manage cryptographic keys in a cloud environment by depend on main instance(dealer) at cloud side for reconstructing and holding the keys temporarily. Also, the proposed scheme is complex and computationally unfeasible for handling many users' keys because the shares are dispersed over several servers, resulting to latency.

[16] Uses a combination of symmetric and asymmetric encryption techniques to secure the data and keys in the cloud by relying on a third party Key Management Server (KMS) for the generation and distribution of encryption keys which may lead to insider attacks.

[24] The hyperelliptic curve cryptosystem-based approach partitions the key matrix into several sub-matrices based on user identity, hence resolving the large-scale key management and storage difficulties in cloud storage. [37] Relies on multi-cloud deployment to ensure the security of keys by distributing the data and keys among

multiple clouds which leads to the computation and latency problems. This framework enables users to store only mandatory key fragments and make use of encryption key as optional key fragments (private keys) are scattered throughout the user's mobile device's storage. Proposed framework does not support scalability and is not suitable for large number of users and also doesn't serve the purpose of being Cloud KMS. [3] describes a distributed key management system for single-tenant environments that manages keys particular to each user and splits the key into shares and then combines keys on runtime using XOR operation. Scheme works by fully trusting the Tenant Admin for splitting and combining the keys. [34]Distributes the shares of the encryption key among multiple cloud providers, which gives rise to data availabilty and security concerns as if the adapter is compromised or malicious, it can leak or tamper with the encryption key or the data which means anyone who has access to the shares can reconstruct the secret and decrypt the data. This may pose a risk of unauthorized access or leakage of sensitive data.

Using Attribute-Based Encryption, enabling fine-grained access control based on user-defined attributes [2] presents a key management system Ucloud that, unlike centralized KMS solutions which entail trust in a single authority, distributes trust among users, enhancing resilience against insider threats and single points of failure where the user is in the control of master key which then encrypts the data encryption keys(DEK). [4] Uses Shamir Secret Sharing to guarantee the privacy and security of the data kept in cloud storage systems by distributing the share among multiple CSPs to avoid risk of unauthorized access and single point of failure but the proposed scheme is computationally complex and also is prone to the possibility of insider attacks as well using Lagrange Interpolation which relies on classical number theory concepts, such as polynomial evaluation and modular arithmetic are not quantum-resilient.

[10]Incorporates STRIDE approach to prevent security threats to the data in cloud computing This works well to identify the attack's impact vector before it happens. This method has been applied in the literature before to obtain cloud computing threat capability. [11] points out the flaws of the traditional secret sharing from a security perspective and proposes a distributed mechanism where shares are distributed at different Cloud Service Providers so as to avoid dependence on central master server. [29] employs key access control scheme for a key management system using Shamir secret sharing algorithm and polynomial interpolation for group based access to another group's data where a Credential Generator is responsible for the construction of the secret key and then forwards the key to Client Management Cloud.The scheme lacks security protocols for secure communication and transfer of keys.

[13] presents secure secret-sharing scheme by incorporating Malicious checker analysis where users are verified based on their previous performance to get authorized. Blocks harmful virtual machines (VMs) if they can be found before they are executed, while this is very difficult to do in real-time scenarios. as when it comes to the reconstruction of secret when decrypting data only those shareholders are allowed to take part in

key that are part of qualified set based on their non-malicious performance which makes it hard to compute when malicious shareholders are detected to be equal or greater than the threshold (t). An alternate approach "Variable Elimination" is used in [31] to reconstruct secret instead of langrange interpolation using SSS schemme. Variable Elimination can be more complex than Lagrange interpolation, leading to potential implementation errors. [15] utilizes an authenticated key tansfer protocol using SSS sheme where a Key Generation enter broadast group key info to all the group members instantly requiring only authoruized group member to recover.The protocol includes an authentication mehanism to verify identity. [21] makes use of Diffie-Hellman distribution scheme for key distribution where Public part of the key pair is divided into two parts one to be sent to user and the other to be kept at CKMS. By using long-term keys with the CKMS, any two parties can create a secure communication channel without physically exchanging any type of keying material.The CKMS functions as a key translation center (KTC), making this possible. For the safe preservation of sensitive material, [23] employs a secret split technique whereby portions of the secret are kept on separate disks or places.

2.2 Key Management Systems

[17] Businesses must use encryption in their cloud environment and make sure that their encryption keys are safely kept off-site in order to secure them. It is never a good idea to store keys next to encrypted data. In modern networks, key management schemes generally fall into two major paradigms:

2.2.1 Centralized

As a result, centralized key management solutions have problems like, the KMS acts as a centralized control point hence making it a single point of failure, cross-domain key transfer is hampered since the key management is done centrally making it very slow hence very inefficient since it relies too much on centralization. Automated key management is a communication method that was formulated to address the problems of risks and trust inherent in traditional key management, which relies on centralisation to distribute keys to many members or organisations. The traditional key management methods present problems like, key distribution and security issues and key management complexity adding to the security risks and management problems of the system.

2.2.2 Distributed

A decentralized governance approach disperses key management responsibilities among multiple entities or nodes in the network. [17]However, a dishonest dealer could distribute a fake shadow to a participant, preventing them from ever obtaining the true secret, and a central KDC could still act as a single point of failure. Managing decentralized key management systems (DKMS) tends to be more complex than centralized ones, as handling multiple nodes and ensuring synchronization and consistency across them can be challenging among other chanllenges as mentioned below:

- Latency and Performance: Distributed systems may introduce latency, especially if nodes are geographically dispersed. The time required to reconstruct a key or perform cryptographic operations may increase.
- Coordination and Trust: DKMS often require coordination between multiple nodes or parties. This introduces the need for trust mechanisms to ensure that nodes operate correctly and do not act maliciously.
- Network Reliability: The performance and reliability of a DKMS depend on the underlying network. Network failures or partitions can temporarily prevent the system from functioning properly.

2.3 Data Security Issues to the Cloud Data

[41] The widespread adoption of edge computing, Internet-of-Things(IoT), and fifth-generation (5G) technologies have generated vast amounts of data. IDC projects that by 2025, there will be 175 zeta bytes of data worldwide. The need for data storage is getting crucial. Cloud storage may efficiently combine and make use of the conventional fragmented and isolated data and information as the comprehensive value contained in the data can have a significant impact. When data is outsourced to cloud servers, consumers are relieved of intricate upkeep and control of local storage. But it can cause users to lose ownership of their data and pose serious threat to data security.

[47] According to the ISC2 2024 Cloud Security Report, among other primary barriers to cloud adoption by the organization security and compliance is at the top, (shown in figure 2.1) [17]One of the most important aspects of cloud computing is the secure management of resources (including the data residing there) that are associated with cloud services . There are various challenges and vulnerabilities associated with securing data that is stored, processed, or transmitted within cloud environments. Proper encryption is crucial, if the data is not adequately encrypted at rest or in transit, it is vulnerable to interception and unauthorized access. According to CrowdStrike's Global threat report 2024 cloud environment intrusions increased by 75% from 2022 to 2023.

Another major issue to the data outsourced on the cloud is the risk of data loss. Data stored in the cloud can be lost for multiple reasons, such as failure of hardware, human errors, or activities with malicious intent. Data loss can be disastrous, particularly if

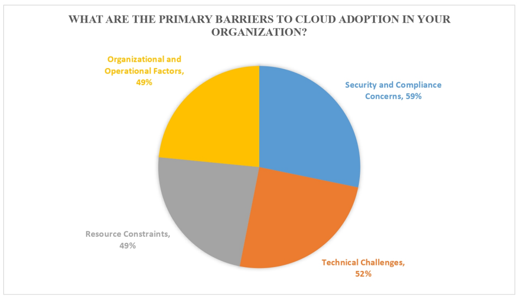


Figure 2.1: ISC2 2024 Report, Barriers to Cloud Adoption

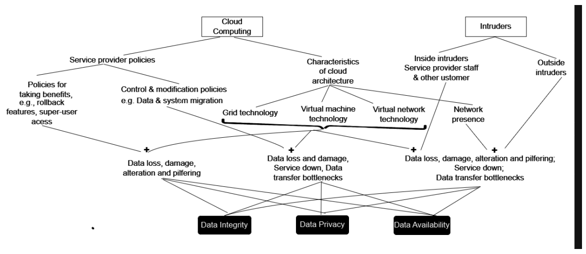

Cloud Computing
Intruders
Service provider policies
Characteristics of cloud architecture
Inside intruders Service provider staff & other ustomer
Outside intruders
Policies for taking benefits, e.g., rollback features, super-user acess
Control & modification policies e.g. Data & system migration
Grid technology
Virtual machine technology
Virtual network technology
Network presence
Data loss, damage, alteration and pilfering
Data loss and damage, Service down, Data transfer bottlenecks
Data loss, damage, alteration and pilfering; Service down; Data transfer bottlenecks
Data Integrity
Data Privacy
Data Availability

Figure 2.2: Issues to Security of Data in the Cloud

backup solutions and disaster recovery plans are not in place, especially for organizations that rely on their data for daily operations. Also, misconfigured or weak access controls can let unauthorized person to have access to sensitive data.

[17] On cloud platforms, there's always a risk of insider attacks, as cloud architectures may not be fully protected against threats from within. [11] In virtual environments, a malicious user could potentially get access into nearby virtual machines which are hosted on the same hardware and then can compromise the integrity and availability of the data. [11]Different factors contribute to risks to data integrity, confidentiality, and availability, including insider attacks as shown in Figure 2.2. [56]According to PWC's Global Digital Trust Insights Survey Report-2024 cloud threats are going to be at the top of the list in coming years as shown in Figure 2.3.

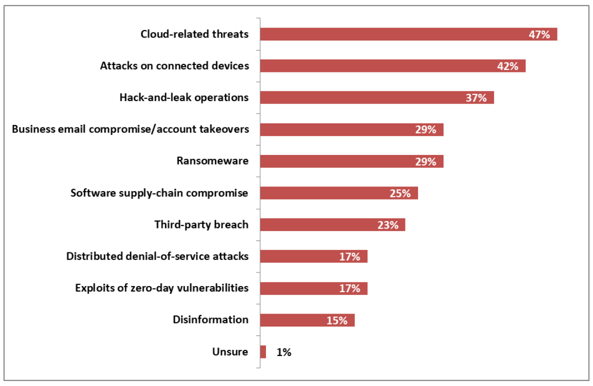


Figure 2.3: Top Cyber Threats Over The Next 12 Months

2.4 Challenges for Cloud-Based Cryptographic Key Management

Traditional key management faces several challenges, including difficulties in key distribution, security threats, management complexity, and trust issues, all of which have worsen the security risks and system's complexity. [43] Resource security is considered as crucial components of cloud computing. Although cloud services are more affordable and provide scalable, elastic, and self-configurable resources, they also necessitate a lot of cryptographic operations from cloud consumers' perspetive. These are required for safe communication with different services as well as for the securiyt of data created/proessed by those services.

The Key Management System needed to facilitate these cryptography operations may be challenging to build due to differences in ownership and management of the infrastructures housing secured assets and KMS. For example, although the data is owned by the cloud customer, the cloud provider manages the storage resources that house the data. For cloud customers looking to guarantee security from these cryptographic activities, this presents a dilemma because frequently, the KMS needed to maintain the cryptographic keys protecting this data also needs to operate on the cloud provider's infrastructure.

Key management in cloud computing presents additional issues, such as ensuring that key stores are protected like any other sensitive data. Key stores need to be safeguarded in storage, during transit, and through backup. Failing to secure key storage can compromise all encrypted data. Access to key stores must be restricted to authorized entities, and policies should be in place to separate roles, ensuring that the entity using a key is not the one storing it. Losing keys means losing the data they protect, which could be disastrous for a business. Therefore, secure storage, backup, and recovery solutions must be implemented to safeguard critical data.

2.5 Current State of Insider Threats and Quantum Attacks in Cryptographic Systems

Malicious insider attacks have long been overlooked, but they are among the most destructive attacks, impacting all layers of cloud infrastructure. Insider threats typically refer to individuals who have legitimate access to an information system and misuse their privileges. [58] The worst-case scenario is when insiders operate within the cloud environment, such as a system administrator with maliious intent working for a cloud provider. This person, by virtue of their role, can use their authorized access to retrieve sensitive data. For example, an authorized administrator who is responsible for system backups (virtual machines, data stores) could take use of this access to obtain private user data. Such indirect access can be very difficult to detect.

An insider attack on the cloud infrastructure could have substantial impact on the cloud provider's business, ranging from data leakage to significant system and data corruption, depending on the insider's motivations. Any kind of cloud service, such as IaaS, PaaS, or SaaS, that is accessible to data centres or cloud Management Systems is susceptible to insider assaults. Verizon's [44] DBIR (Data Breach Investigation Report) reports that while external attackers continue to be the dominant cause of breaches (65%), internal threats have increased significantly to 35% from 20% in 2023 as shown in table 2.1.

Table 2.1. Involvement of Insiders in Attacks

ATTACK VECTORS	**EXTERNAL THREAT ACTORS**	**INTERNAL THREAT ACTORS**
Privilege Misuse	1%	99%
Social Engineering	100%	-
Web Application Attacks	99%	1%
Miscellaneous Errors	-	100%
DOS Attacks	100%	-
Lost and Stolen assets	12%	88%
System Intrusion	100%	-

[46] When malicious insider threats occur, authorised users take use of their access and organisational knowledge, which makes them very challenging to find. By eluding security standards and procedures, these insiders combine their malicious activity with legitimate activities. Their knowledge of security procedures along with the confidence they're given and the emergence of remote work, have made it more difficult to distinguish between benign and actions with malicious intent. In contrast to external threats, which frequently exhibit distinct signs of insider threats requires more sophisticated detection

techniques in order to detect hidden threats. Firewalls and intrusion detection systems often overlook these attacks, since they consider these actions to be acceptable and don't present a risk of being discovered.

Verizon's 2023 report highlights that 99% of threat actors are insiders. The Verizon Data Breach Investigations Report (DBIR) for 2021 reveals the involvement of the insiders in external attacks which is around 22% of security incidents, showing that a significant portion of attacks on cloud or cloud key management systems.

2.6 Quantum-Resilient Cryptography-Why a Necessity Now?

We are aware that the Internet depends on encryption for private emails, security and privacy of the communication, as well as finance-related transactions. If encryption is breached, these crucial could be jeopardized. [54] Today's encryption relies heavily on complex problems in mathematics, such as factoring really big numbers, that are incomprehensible to conventional computers. If quantum computing hadn't been developed, we might still be relying on conventional cryptography for many more years. These gadgets employ the ideas of quantum mechanics to carry out specific calculations significantly more effectively than conventional computers. Regretfully, these intricate computations involve solving the mathematical hard problems that are the basis of majority of the cryptography in broad usage in the modern era.

There are no large and stable quantum computers at present which can break today's cryptography but experts are sure that a Cryptographically-Relevant Quantum Computers (CRQC) will be built soon. However, in a threat model known as the harvest now, decrypt later type of threat model, an attacker can capture encrypted data now and wait for the day he gets access to an advanced quantum computer to decrypt the information. This means that the communication implemented today is already vulnerable from a potential quantum attacker and it is crucial to switch to implementing post-quantum key agreements as soon as possible.

Mathematical difficulties like the discrete logarithm problem and factoring huge integers are the foundation for the security of elliptic curve cryptography (ECC) and other public-key cryptography systems. The RSA and Diffie-Hellman cryptosystems, as well as their elliptic-curve-based variations, are totally broken by Shor's quantum computing method [53].

In $O(n^3)$ time, Peter Shor's quantum method can solve prime factorization for a n-bit integer. It is estimated that the general number field sieve, the quickest classical approach for integer factorization, executes in $O(n1/3)$ time. Using the Pollard-Strassen algorithm, the best rigorously proved upper bound on the classical complexity of factoring is $O(2n/4+o(1))$. If small factors exist, a quantum algorithm employing Grover search can outperform Shor's algorithm in accelerating the factorisation of elliptic curves. It is thought that certain traditional public-key cryptosystems are immune to quantum attacks. Order finding, the foundation of Shor's factoring algorithm, which reduces to the Abelian hidden subgroup problem, can be solved by using Quantum Fourier Transform (QFT) and Other hardness problems, among which few are membership problem for matrix groups over odd order fields and diophantine problems whih is related to quantum circuit creation, also reduce to integer factorization.

[54] The final standards for three post-quantum algorithms—ML-KEM for key agreement, ML-DSA and SLH-DSA for digital signatures—were published by NIST on August 13, 2024. The FFT (fast-Fourier transform) over NTRU-Lattice-Based Digital Signature Algorithm (FN-DSA) will be the main emphasis of the fourth standard, which will be introduced in late 2024 and is based on the FALCON algorithm.

Table 2.2. NIST's First Post-Quantum Standards

OLD NAME	**NEW NAME**	**BRANCH**
Kyber	**ML-KEM** (FIPS 203) Module lattice-based key-encapsulation Mechanism	Lattice-Based
Dilithium	**ML-DSA** (FIPS 204) Module lattice-based Digital Signature Standard	Lattice-Based
SPHINCS	**SLH-DSA** (FIPS 205) Stateless Hash-based Digital Signature Standard	Hash-Based
Falcon	**FN-DSA** FFT over NTRU lattices Digital Signature Standard	Lattice-Based

2.7 Conclusion

This chapter reviewed some of the existing techniques for secure management of keys and to provide data security over the cloud environment.It also discussed existing data security issues along with the current state of insiders/internal threats and quantum attacks in cryptographic systems.

Chapter 3

THEORETICAL FOUNDATIONS

This chapter comprehends a detailed explanation of lattice-based cryptography and Shamir Secret Sharing scheme and how the lattice-based version of Shamir Secret Sharing scheme ensures enough security to be resilient against Quantum computers generated attacks.

3.1 Shamir Secret Sharing Scheme

[42] Efficient threshold schemes play a crucial role in managing cryptographic keys. The usual way to protect data is to enrypt it, but we need a different approach to secure the encryption key—increasing the number of encryption levels just makes the issue worse by shifting and complicating it further. A computer, a person's brain, or a safe can all be considered secure locations for the key to be kept in the most secure key management system. This approach, however, is extremely unreliable since the knowledge might become permanently inaccessible due to a single error (such as a computer malfunction, an unexpected death, or sabotage).

Maintaining many copies of the key in different locations is a more obvious approach, but doing so raises the possibility of security breaches (due to hacking, betrayal, or human mistake). With $n = 2k-1$, we implement a (k, n) threshold technique to build a very robust key management system. While adversaries cannot reconstruct the key even if $\lfloor \frac{n}{2} \rfloor = k-1$ of the remaining k pieces are compromised, adversary would not able to retrieve the original key in the event that $\lfloor \frac{n}{2} \rfloor = k-1$ of the n pieces are lost or destroyed.

One of the very first threshold algorithms was Shamir's Secret Sharing Scheme, which was created by Adi Shamir in 1979 [57]. The plan uses polynomial interpolation as its foundation. There are t different points (xi,yi) that uniquely define any polynomial y = f(x) of degree t-1. The Dealer selects a finite field polynomial that is suitable and shares n different locations on the polynomial's graph with **n** users. The Dealer is responsible for key generation, distribution, and reconstruction. The polynomial can then be reconstructed by any group consisting of at least **t** users.

A key feature of Shamir's Secret Sharing is that in order to reconstruct a secret all the shares are not required, instead, the pieces are combined to recover the secret. The threshold value must be smaller than the total number of shares, which prevents decryption failures if a few parties are unavailable. Given that Shamir's scheme provides a efficient solution to the problems of key-sharing it is employed in the protection of the keys for Data that is in an encrypted form and protected with other algorithms or other tools

A simple example is a vault accessible only to a company's board and the passode of vault's is made encrypted using SSS, and a quorum, also known as threshold, of board members is needed to display or release the passcode. Even if one board member is absent, Shamir's Secret Sharing scheme makes it sure that the vault remains secure while still meeting the access requirement.

3.1.1 Algorithm for Shamir's Secret Sharing Scheme

Input: t and n represent threshold and total partiipants, respectively, secret means S
Output: Shares (i, S_i) for $1 \leq i \leq n$, reconstructed secret S

Step 1: Secret Sharing (Generation of Shares) A secret S is generrated by the dealer D, where $S \geq 0$. D selects a prime p such that $p > \max(S, n)$.This prime p must be larger than both the secret S and the total number of participants n. The condition $p > \max(S, n)$ ensures that all operations performed during the secret sharing process are well-defined within the finite field $\mathbb{F}_p$.

D sets $a_0 = S$, where a_0 is the constant term of the polynomial f.This means that the secret S will be the value of the polynomial when evaluated at x=0 (i.e., f(0)= S).

D randomly selects $t-1$ coefficients $a_1, a_2, \ldots, a_{t-1}$, where $0 \leq a_j < p$ for $1 \leq j \leq t-1$. The number of coefficients $t-1$ is determined by the threshold t, which is the minimum number of shares required to reconstruct the secret. In total, there are t coefficients (including a_0) which correspond to the degree of the polynomial.These coefficients define a random polynomial $f(x)$ over the finite field $\mathbb{F}_p$:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} = \sum_{j=0}^{t-1} a_j x^j$$

D computes the shares S_i for each participant P_i by evaluating the polynomial f at $x = i$:

$$S_i = f(i) \mod p, \quad \text{for } 1 \leq i \leq n$$

D securely distributes the share (i, S_i) to each participant P_i.

Step 2: Secret Reconstruction To reconstruct the secret S, at least t participants must combine their shares (x_i, S_i). Each participant P_i provides their share $(x_i, S_i) = (i, S_i)$. Using the t shares, the participants reconstruct the polynomial $f(x)$ by Lagrange interpolation. The polynomial $f(x)$ is calculated as:

$$f(x) = \sum_{i=1}^{t} S_i \cdot L_i(x)$$

Where $L_i(x)$ is the Lagrange basis polynomial associated with the point x_i defined as:

$$L_i(x) = \prod_{\substack{1 \leq j \leq t \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

The secret S represents the constant term of the polynomial, which is $f(0)$:

$$S = f(0) = \sum_{i=1}^{t} S_i \cdot L_i(0)$$

Expanding $L_i(0)$ gives:

$$S = \sum_{i=1}^{t} c_i \cdot S_i, \quad \text{where} \quad c_i = \prod_{\substack{1 \leq j \leq t \\ j \neq i}} \frac{x_j}{x_j - x_i}$$

Once $f(0)$ is computed, the secret S is successfully reconstructed.

3.1.2 Key Attributes of Shamir Secret Sharing Scheme

Some properties of Shamir Secret Sharing scheme are discussed as under:

Secure: The scheme ensures that the secret remains confidential and secure. Even if some participants collude, they cannot uncover the secret unless they have the required number of shares. This is known as information-theoretic security.

Minimal: Each share is exactly the same size as the secret itself. This efficiency tells that the scheme does not need large storage or transmission. resources.

Extensible: The security of the plan is unaffected by the addition or removal of existing shares. Because of its flexibility, the system can change to meet new demands.

Dynamic: This security level can be manipulated by varying with the threshold size of shares required to rebuild the secret. At the same, this means that the scheme can be made more secure if needed.

Convenience: Sharing and rebuilding the secret is a simple process that makes it simple to utilize and implement in practice.

Flexible: The scheme can also be applied to any data and can be used in different application and in different use cases,thus, it can be applied to the majority of practice situations.

3.1.3 Critical Security Features of Shamir Secret Sharing Scheme

[5]If there are less than t individuals in a group trying to reconstruct the shared secret, Shamir's Secret Sharing technique is completely safe. Here, the interpolation is unsuccessful, and the smaller group remains ignorant of the shared secret. Though this is only true if the polynomials are consistently selected arbitrarily and aren't used later to share numerous secrets. The best and most effective method is the Shamir Secret Sharing scheme, where share size is equal to the secret. Shamir's Secret Sharing system is secure because of a number of important factors that work together to withstand different types of attacks and unauthorized efforts to decipher the secret. The following are the primary factors that guarantee the scheme's security:

- Information-Theoretic Security: Perfect secrecy, commonly referred to as information-theoretic security, is attained by Shamir's Secret Sharing. Accordingly, the system offers more than just computational security; it offers absolute security. Without the necessary quantity of shares, an attacker possessing any level of computing power is unable to learn anything about the secret.

- Polynomial Interpolation: The idea is closely connected with the polynomial interpolation in the finite field. The secret is represented as the coefficient of the polynomial based on which the bit and shares are constructed. The security of the SSS scheme is based on the computation of polynomial and its constant term the secret from a subset of the points.

- Randomness and Unpredictability: The generation of shares and polynomial coefficients contain a certain level of randomisation. The randomness of these values is the major determining factor of the scheme's security. The problem encountered with the random numbers is that if an attacker manages to guess these random values then they may have some insight into the secret.

- Threshold Requirement: One of the security standards is the threshold requirement. The secret is further divided into multiple shares and a mandatory (threshold) number of shares are needed to reconstruct the secret. Lacking this threshold, even if an attacker obtains certain shares, he can not restore the secret.

- Shares Independence: The security of the scheme is associated with the fact that shares are generated with equal probability to the number of users. When the shares are created randomly it is hard to gain information on the secret if one has some of the shares, but does not have the shares needed to meet the threshold level.
- Difficulty of Reconstruction: It is indicated that the polynomial interpolation problem, which is defined as reconstructing the original polynomial from a subset of points, is hard.The scheme's security relies on the difficulty of reconstructing the secret using fewer shares than the predetermined threshold.
- Large Finite Fields: This work shows that using large finite fields improves the security of the scheme. The larger the field, the more complex for an attacker to infer the remaining information out of the available shares.
- Adversary Model: The passive adversary model, in which the attacker can only view the shares and not alter or affect their creation, is the basis for the security study of Shamir Secret Sharing. In practical situations, any attempts to alter the shares themselves or the creation of the shares process might cause greater challenges.

  Even though, as a result of implementing these security features, Shamir Secret Sharing is supposedly invulnerable to this type of attack. For instance, it may fail in situations where generation of shares is under threat or where a certain number of malicious shares are produced. Furthermore, the scheme can be threatened by quantum attacks if and when large scale quantum computers are made available.

3.2 Potential Quantum Threats to Classical Shamir Secret Sharing Scheme

Quantum computers have been found to pose a potential threat to the Shamir Secret Sharing scheme's security, though the exact degree of that vulnerability will largely rely on how the system is configured and parameterized.

- Faster Reconstruction: In particular, polynomials that are sorely required for the primary polynomial interpolation operation in SSS may benefit from quantum computing. When using quantum computer it is likely to perform polynomial interpolation over finite fields faster than classical computer and therefore reconstruct the secret.
- Reduced Security: The difficulty to recover the original polynomial from a number of random points is one of the mathematical issues that are presented as proof of

the effectiveness and security of SSS. Quantum computers may actually be able to address these issues more quickly, reducing the scheme's security weakness.

- Attacks on Key Generation: Quantum computers could potentially attack the key generation process in SSS. If the random number generation or key generation process is vulnerable to quantum algorithms, an attacker could predict or manipulate the generated shares, leading to the compromise of the secret.

3.3 Lattice-Based Cryptography

In the course of the technological development, there is also an increased need for the protection of information and data transfer. As attacking power increases and data expands, security has to become simultaneously more powerful and demand less resources. Post all of these, Public Key Cryptography (PKC) is now being implemented in most communication systems. However, more recent developments in mathematics with the advent of Shor's Algorithm, show that public key cryptographic schemes will become ineffective once large scale quantum computers are built for such problems are solvable in poly time. This has led to bring the attention towards post-quantum, or quantum-safe, cryptography (QSC).

In 2016, the international cryptography community , National Institute of Standards and Technology, also known as NIST, initialized the process to transition from classical ryptography to Post Quantum Cryptography(QSC). Initially, lattice-based, code-based, multivariate-based, hash-based, and isogeny-based QSCs were the five primary types that were being considered. The fact that quantum-safe cryptography requires more processing power and bandwidth than existing cryptography is a known drawback. On August 13, 2024, NIST announced the First Post-Quantum Cryptographic Standards, and three of the four standards were lattice-based due to the comparatively high efficiency of lattice-based cryptography. par

Lattice-based cryptography offers several advantages, including strong security guarantees based on the hardness of lattice problems, resistance to quantum computer attacks, and provable security under well-established assumptions. [19] Additionally, it uses linear computations on relatively small integers. Since the secret sharing algorithm must align with the lattice-based public key infrastructure, Shamir's secret sharing scheme should be replaced with a lattice-based secret sharing scheme.

[1]The use of lattice-based cryptography (LBC) offers a viable substitute for the conventional techniques used today. It is thought that lattice issues cannot be solved in a practical amount of time, not even by quantum computers. Ajtai proved in 1995 that the intricacy of these problems in their average situation is at least as challenging as addressing them in their worst scenario. Moreover, in the NIST process, lattice-based

schemes remain the only candidate to offer fundamental primitives such as encryption, key encapsulation, and digital signature techniques.

The analysis of lattice-based cryptography is analyzed from the following aspects as mentioned below

1. **Security**: Lattice-based cryptography provides robust security assurances derived from the intricate nature of lattice problems, including challenges like the Shortest Vector Problem (SVP) and the Learning With Errors (LWE) problem. These inherent security properties offer a robust defense against both classical and quantum attacks, rendering lattice-based schemes highly appealing for post-quantum cryptography purposes.

2. **Efficiency**: Recent improvements in lattice-based cryptography have focused on making cryptographic tools work better. This means trying to make it faster to create keys, make the encrypted messages smaller, and reduce the amount of computing power needed. Structured lattices and smarter ways of doing things with algorithms are really important for making lattice-based cryptography more practical and useful.

3. **Versatility**: Lattice-based cryptography offers versatility and flexibility in designing various cryptographic primitives, including encryption, digital signatures, identity-based encryption (IBE), and homomorphic encryption. These primitives enable secure communication, authentication, and privacy-preserving computations in diverse application domains.

3.3.1 Lattice

[48,49]Modern cryptography includes lattice-based cryptography, whose security depends on solving unsolvable lattice theory challenges. In mathematical terms, a lattice is a collection of vectors. A lattice can be seen of as a vector space created by a collection of vectors that are linearly independent in cryptography. n Linearly independent vectors can be used to define a lattice $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^m$.The lattice generated by these vectors is:

$$\mathcal{L}(B) = \mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$$

Here, $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ are known as the basis vectors of the lattice $\mathcal{L}(B)$. We can also define B as an $m \times n$ matrix, where each column vector is one of the basis vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$. The lattice that this matrix creates is:

$$\mathcal{L}(B) = \mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) = \{B\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$$

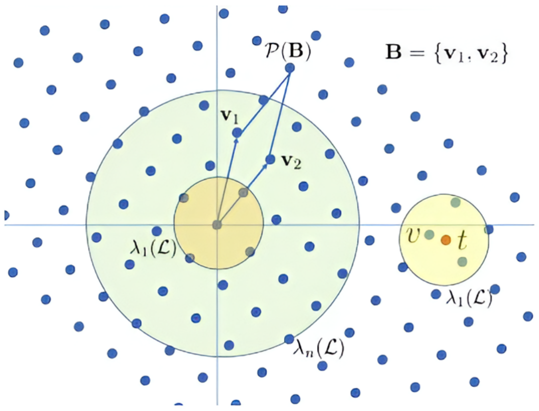


Figure 3.1: A 2-Dimensional Lattice Structure

Here, m and n are integers, where $m \geq n$. The dimension of the lattice is denoted by m, whereas the rank is denoted by n. Full-rank lattices are those in which $m = n$. $\lambda(\mathcal{L}(B))$, which is the length of the shortest non-zero lattice vector, is the shortest distance in the lattice $\mathcal{L}(B)$. It is the smallest distance between any two different lattice points.

3.3.2 Cryptographic Assumptions on Lattices

Lattice-based cryptography uses problems in mathematics with lattice structures as a foundation to build cryptographic techniques including key exchange, encryption, and signatures. latticess often rest on complex problems like the Closest Vector Problem (CVP), the Shortest Vector Problem (SVP), and Learning with Errors (LWE). Lattice-based cryptographic systems provide strong security because these challenges are usually computationally challenging. Lattice-based encryption has a number of benefits, including the ability to provide provable security under well-established assumptions, resistance to attacks utilizing quantum computers, and strong security guarantees based on the hardness of specific lattice issues.

1. **Small Integer Solution Definition 2 (SIS).** Take a matrix $A \in \mathbb{Z}_q^{n\times m}$ with a norm satisfying $\|\mathbf{z}\| \leq \beta$ formed by m uniformly distributed random vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$.

2. **Learning with Errors Definition 3 (LWE).** A random matrix $A \in \mathbb{Z}_q^{m\times n}$ and $(A, A\mathbf{s} + \mathbf{x})$ are involved in the LWE problem for a prime number q, a positive integer n, and a Gaussian distribution χ^m on $\mathbb{Z}_q^m$. Here, $\mathbf{x} \in \chi^m$. The LWE problem presents a substantial probability challenge that is to locate $\mathbf{s} \in \mathbb{Z}_q$.

3. **Shortest Vector Problem Definition 4 (SVP).** For a lattice $\mathcal{L}(B) \subseteq \mathbb{R}^n$ of dimension d, with a basis $B \in \mathbb{Z}^{n\times m}$, the SVP problem requires finding a non-zero

vector $\mathbf{x}$ in $\mathcal{L}(B)$ such that $\|\mathbf{x}\| \leq \lambda(\mathcal{L}(B))$. **Definition 5 (SVP$_\gamma$).** Given a lattice $\mathcal{L}(B) \subseteq \mathbb{R}^n$ of dimension d, with a basis $B \in \mathbb{Z}^{n\times m}$, the SVP$_\gamma$ problem involves finding a non-zero vector $\mathbf{x}$ in $\mathcal{L}(B)$ such that $\|\mathbf{x}\| \leq \gamma \cdot \lambda(\mathcal{L}(B))$, where $\gamma \geq 1$ is an approximation factor. **Definition 6 (GapSVP$_\gamma$).** In this problem, for a lattice $\mathcal{L}(B) \subseteq \mathbb{R}^n$ of dimension d, with a basis $B \in \mathbb{Z}^{n\times m}$, the GapSVP$_\gamma$ problem requires judging whether $r \cdot \gamma < \lambda(\mathcal{L}(B))$ or $r \geq \lambda(\mathcal{L}(B))$, where r is a rational number.

4. **Closest Vector Problem Definition 7 (CVP).** For a lattice $\mathcal{L}(B) \subseteq \mathbb{R}^n$ of dimension d, with a basis $B \in \mathbb{Z}^{n\times m}$, the CVP problem requires finding a non-zero vector $\mathbf{v}$ such that, for any non-zero vector $\mathbf{u} \in \mathcal{L}(B)$, the inequality $\|\mathbf{v} - \mathbf{t}\| \leq \|\mathbf{u} - \mathbf{t}\|$ holds, where $\mathbf{t}$ is a target vector.

   Definition 8 (CVP$_\gamma$). Similar to CVP, but with an approximation factor $\gamma \geq 1$, the CVP$_\gamma$ problem involves finding a non-zero vector $\mathbf{v}$ in $\mathcal{L}(B)$ such that $\|\mathbf{v} - \mathbf{t}\| \leq \gamma \cdot \|\mathbf{u} - \mathbf{t}\|$ for any non-zero vector $\mathbf{u}$. **Definition 9 (GapCVP$_\gamma$).** Given a lattice $\mathcal{L}(B) \subseteq \mathbb{R}^n$ and a basis B, the GapCVP$_\gamma$ problem involves deciding whether $r\gamma \leq \|\mathbf{u} - \mathbf{t}\|$ or $r \geq \|\mathbf{u} - \mathbf{t}\|$, with r being a rational number and $\mathbf{t}$ a target vector.

3.4 Lattice-Based Shamir Secret sharing scheme

Using a lattice-based version of Shamir Secret Sharing can potentially provide better security compared to the traditional Shamir Secret Sharing scheme that works using polynomial interpolation. In a lattice-based variant of Shamir Secret Sharing, instead of using polynomial interpolation, the operations are performed within the lattice framework. The shares and reconstruction process involve operations on lattice points, leveraging lattice problems like the Shortest Vector Problem(SVP),and also the other problems known as Ring Learning With Errors (RLWE) or Learning With Errors (LWE) problem.

3.4.1 How Lattice-Based Shamir Secret Sharing Scheme Strengthens Security of the Scheme

The Lattice-Based Shamir Secret Sharing Scheme significantly strengthens the security of traditional Shamir Secret Sharing (SSS) schemes by integrating lattice-based cryptographic techniques, which offer several advantages, particularly in the context of quantum computing threats. Here's a comprehensive overview of how this integration enhances the security of the scheme:

1. Post-Quantum Security:

- Lattice-Based Advantage: On the other hand, lattice-based systems rely on the difficulty of lattice problems, such Learning With Errors (LWE) and the Shortest Vector Problem (SVP). It is thought that these issues are immune to quantum algorithms, guaranteeing that the security of the shared secret will endure even in the presence of quantum computers. Lattice-based SSS provides a future-proof solution against changing cryptographic threats because of this post-quantum security.

2. Threshold Flexibility: Another advantage of using lattice-based methods is thatit is possible to change the thresholds, and no other communication is needed with the dealer or shareholders. By this flexibility, security within the scheme can be changed as a result of changing security demands or threat levels in the environment and this makes security even better. Since the number of shares that is required to reconstruct the secret can be changed to meet the current security requirements, the protocol remains rather immune to the attacks.

3. Robustness Against Collusion:

   - Common Risk: If shareholders control a significant enough number of shares in a typical SSS, there is a chance that they will collaborate to reconstruct the secret.
   - Lattice-Based Strength: The method based on lattice, on the other hand, offers better protection from collusion into such schemes. The security of the lattice problems makes it much more difficult for a part of shareholders to reconstruct the secret in case of reaching the required threshold only. This way guarantees that even if some of the participants are involved in conspiracy they cannot garner back the secret without meeting the minimum number of shares apart from compromising the secret.

4. Efficient Verification:In lattice based schemes, for example, various techniques for reconstruction and distribution include efficient checks on the shares to detect authenticity.They help prevent manipulation so that the secret can only be rebuilt using the correct shares. This improves security and gives the secret exchange procedure an extra degree of security.

5. Enhanced Privacy: According to such a lattice, the privacy of individual shares is greatly enhanced through transforming mathematical representations of shares into other equivalent forms. The mathematical form of lattices allows it that the potential opponent has no chance to obtain any information about the secret from single shares if he succeeded in getting one or more of the shares.This enhanced privacy is crucial in maintaining the confidentiality of the secret in a distributed environment.

6. Resilient Key Generation: The lattice-based approach also fortifies the key generation process. Quantum computers could potentially attack the key generation process in traditional SSS, but lattice-based schemes are designed to withstand such attacks, ensuring that the shares and the secret remain secure.

7. Long-Term Security Assurance: As the cryptographic landscape continues to evolve, lattice-based schemes offer a future-proof approach to securing secrets. By preparing for the potential capabilities of quantum computers, the lattice-based Shamir Secret Sharing Scheme provides long-term security assurances, making it a robust choice for both current and future cryptographic needs.

3.5 Key Management Functions

[43]The key management life cycle/phases are the operations that govern the creation, use, storage and destruction of keys. The primary key management functions operations are discussed as follows:

- Generate Key: A vital component of security is the creation of high-quality keys. Cryptographic modules that have been authorised to create keys for a given algorithm should be used to produce the keys for that algorithm.
- Generate Domain Parameters: These establish the fundamental parameters and guidelines in which cryptographic operations will take place. Before the keys or shares are formed, lattice-based algorithms need the development of domain parameters, such as lattice dimension and basis vectors, which are generated using authorised cryptographic modules. The domain parameters are then used to construct the keys.
- Bind Key and Metadata: An associated Access Control List (ACL) that identifies which users, entities, or processes are authorised to access or use the key or share, as well as usage constraints (like authentication, encryption, or key establishment) and the key's owner's identification, are bound to the key.
- Key Activation: During key generation, this step often activates a key.
- Key Deactivation: This is usually done after a key has expired or been replaced by another, making it unusable for cryptographic protection.
- Key Backup: In the case where a key is accidently deleted or hets unavailable, it can still be restored from where the backup is stored.

- Key Recovery: When a key is lost but still required by authorised persons, this feature is used in addition to key backup. Both symmetric and private keys are often covered by key backup and recovery.
- Metadata Modification: When a key's associated metadata needs to be updated, this function is called. Examples include updating the validity period of a public key certificate or adding or removing a person or device from the ACL of a key or share.
- Rekeying: This operation replaces an existing key with a new one or its updated version Typically, the current key is used for authentication and authorization for the replacement of key .
- Key Suspension: This feature, which is comparable to reversible revocation, temporarily halts the use of a key. It can be used when the owner of a key wants to temporarily halt using it (for example, while on a prolonged absence) or when the key's status is unknown. This can also be accomplished for secret keys by deactivating the key. Suspension of public keys and their matching private keys is often accomplished by sending out a suspension notification for the public key.
- Key Restoration:As soon as the secure status of a locked key is determined, this function allows the user to unlock it. Key activation with secret keys is another way to accomplish it. This is mostly handled by a revocation notification for the public key and any associated companion private identifiers; otherwise, the revoked public key item is simply removed using the unique identifier that certifies the key's validity.
- Key Revocation: This feature notifies the parties that are dependent on the public key to stop utilising it. The owner discontinuing to utilise the private linked with it and the treachery of a companion companion private key are two possible causes of this.
- Key Archival: After a key has been deactivated, expired, or compromised, this function allows it to be saved in the long-term memory.
- Key Destruction: If a key is not needed any longer, this function is used to destroy it.

A symmetric key or public/private key pair being used in proposed Key Management System can undergo different states as shown in figure 3.2.

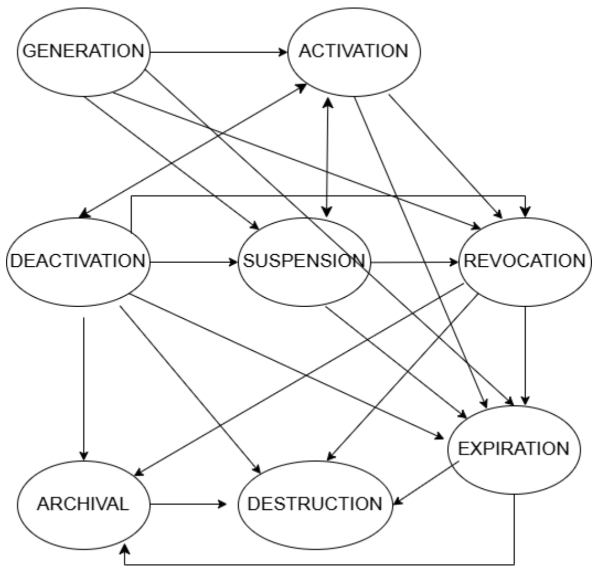


Figure 3.2: State Diagram of Keys Life-Cycle

3.6 Key Management - Generic Security Requirements

[28]Managing keys is crucial in a cloud environment to ensure data flow synchronization in networks. Encryption provides security but requires significant computational power. Storing encryption keys presents challenges because of clouds' dynamic nature. to the dynamic nature of the cloud. Keeping keys with the consumer, rather than in the cloud, avoids excessive computations for decryption during data retrieval, ensuring security and efficiency. Here are some security requirements regarding key management, discussed as under:

1. Appropriately authenticated and authorizations parties are allowed to perform the key management functions for a given key.

2. Concerning the major management commands together with the related information all protected from spoofing, i. e. the methods involved in source authentication are done before a command is to be issued.

3. Since any alterations to information, even those made by an unauthorized person, are referred to as unlawful modifications, all crucial management commands and pertinent data are protected against sneak attacks, guaranteeing the integrity.

4. The private and secret keys are kept hidden to prevent unauthorised use.

5. Integrity protection is offered in distributed environments to protect all keys and metadata from unauthorised and undetected modifications.

3.7 Conclusion

The chapter discussed the core principles of Shamir Secret Sharing and Lattice-based Cryptography, highlighting their strengths in secure information distribution and quantum-resistant security. By merging these two frameworks, Lattice-based Shamir Secret Sharing offers a robust and future-proof approach to cryptography, ensuring both the efficient sharing of secrets and protection against quantum attacks.

Chapter 4

PROPOSED KEY MANAGEMENT SYSTEM

This chapter, we will discusses the architecture and working of the proposed scheme using an improved version of Lattice Based Shamir Secret Sharing Scheme as an efficient solution for existing security related concerns in key Management System (KMS).

4.1 Architecture and Components of the KMS

As the pattern of proposed KMS is distributed, therefore, some of the components lie at the cloud-side and while some are functioning at the on-premise based KMS. The figure 4.1 shows the architectural components of Cloud-Based KMS and Premise-Based KMS and interactions amongst them.

4.1.1 Architectural Components

1. On-premise KMS Components:

    - Authorization System: Responsible for handling authentication tasks for users, ensuring secure access control. Could implement multifactor authentication (MFA) and monitor session activity.
    - Encryption/Decryption Application: Uses keys provided by the KMS for encrypting and decrypting user files. Likely integrated with the enterprise user interface for seamless operations.
    - Dealer: Performs the lattice-based Shamir secret sharing tasks. Handles the distribution of shares to the key provider on both on-premise and cloud KMS.

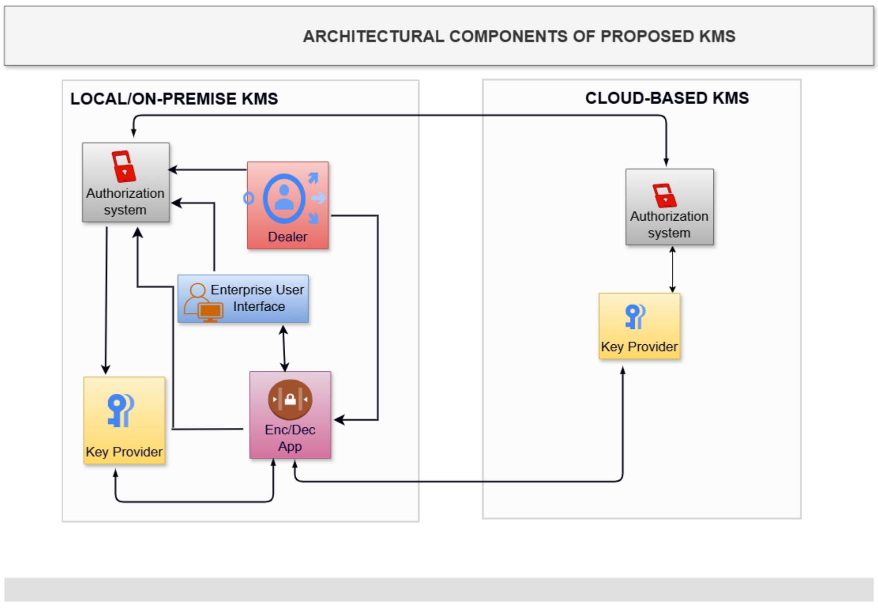


Figure 4.1: Architectural Components of Proposed KMS

- Key Provider: Stores and manages key shares locally. Works with the dealer to distribute and reconstruct secret shares for user operations.
- Enterprise User Interface: A user-facing component where users interact to encrypt/decrypt files. Can be designed to allow users to manage access control, key versions, etc.

2. Cloud-based KMS Components:

- Key Provider: Stores keys or key shares that are mirrored or distributed from the on-premise system and Act as a backup or redundancy in case of on-premise system failure.
- Authorization System: Does mutual authentication with the on-premise KMS using mTLS and also ensures only authorized communication occurs between cloud and local systems.

4.1.2 Overview of the KMS

The proposed key management system is based on lattice-based Shamir Secret Sharing scheme(based on Shortest vector problem hardness problem). The Key Management System (KMS) is distributed and has two sub KMSs , one is the local/on-premise based and other one is cloud based. The premise-based has 5 components namely authorization system who performs authentication related task, encryption/decryption application that performs the encryption/decryption of the files using the keys provided to it, dealer is the entity that will perform all the secret sharing activities, key provider has the keys and the shares of the keys and enterprise user interface through which user interact with the encryption-decryption application to encrypt or decrypt the files . On the other hand, the components at the cloud side has the key provider where the rest of the keys are stored and the authorization system which helps in the mutual authentication between the cloud KMS and premised-based KMS.

First of all at the start of each session the cloud KMS and Enterprise KMS mutually authenticate each other using mTLS and then a secure channel is established between the two. When a user wants to send an encrypted file to any other user 2, he first login using the Enterprise User Interface(EUI) to access the enryption/decryption application, he is then navigated to authorization system to get access to key provider in order to select the keys he is authorized to use, after he selects the key, the key is sent to the encryption application after mutual authentication of key provider and encryption application. If the user is not registered already then user is first registered and then process of enryption/decryption starts as explained in Section: High-Level Algorithm of Proposed KMS.

The proposed Key Management System is designed to ensure confidentiality, integrity, and proper authentication throughout the file encryption and decryption stages. Below is a generalized overview of how the process works:

1. User Authentication for Key Access: The user gets a chance to login into the system an identify himself through a secured method. After user authentication the user communicates with the authorization system to obtain the required cryptographic keys. If the user is not registered, he proceeds through a secure registration to connect the account to a smart key.

2. File Encryption Using Session and Master Key: To encrypt the file there is a session key and to maintaining the master key, the system uses improved method like Lattice-Based Shamir Secret Sharing(LBSSS). The master key then generated and divided into shares and stored securely. They then utilizes this master key later to further encrypt the session key enhancing the level of security.

3. Secure Exchange of Public Key: The system receivers the receivers's public key and encrypt the file with that so it may be readable only by the recipient. This public key encryption ensure secure transfer of the encrypted file.

4. Secure Communication using Mutual Authenticaton: In the proposed KMS, each of the system components are authenticated mutually, allowing only administrative entities to engage in the encryption, key management and files transfer. There is a pub/pri key pair for each component which is stored at key Provider.

4.2 Authentication Between Different Modules of the Framework

1. Authentication between User and key provide
   Adapted Public Key Authentication in proposed Framework works as explained below:

   - Key Pair Generation and Storage:
     The user's public/private key pair is generated by the key provider. Both the public key and the private key are stored securely by the key provider.
   - Request for Authentication:
     When the user wants to authenticate, they send an authentication request to the Authorization System. This request can be initiated, for example, when the user logs in or requests access to certain services.
   - Challenge-Response Mechanism:
     The Authorization System generates a challenge (a random piece of data) and forwa it to the key provider. The key provider, holding the user's private key, signs the challenge using the private key.
   - Verification Process:
     The signed challenge is sent back to the Authorization System. The Authorization System uses the stored public key (which it retrieves from the key provider or its own secure store) to verify the signature. If the signature is valid, the Authorization System authenticates the user.
   - Access granted:
     Upon successful authentication, the Authorization System authorizes the user to access the key provider. The Authorization System then securely passes an authentication token or a temporary access credential to the key provider, indicating that the user has been authenticated. The key provider can then trust that the user has been properly authenticated by the Authorization System and grant access to the requested cryptographic services.

In the same manner, mutual-authentication is done between En/Dec app and Key Provider, and between Dealer and the Enc/Dec app.

4.2.1 Why mutual authentication is needed

- To authorize the dealers to generate and share the shares of the secret and logging of events starts after the dealer's authentication.

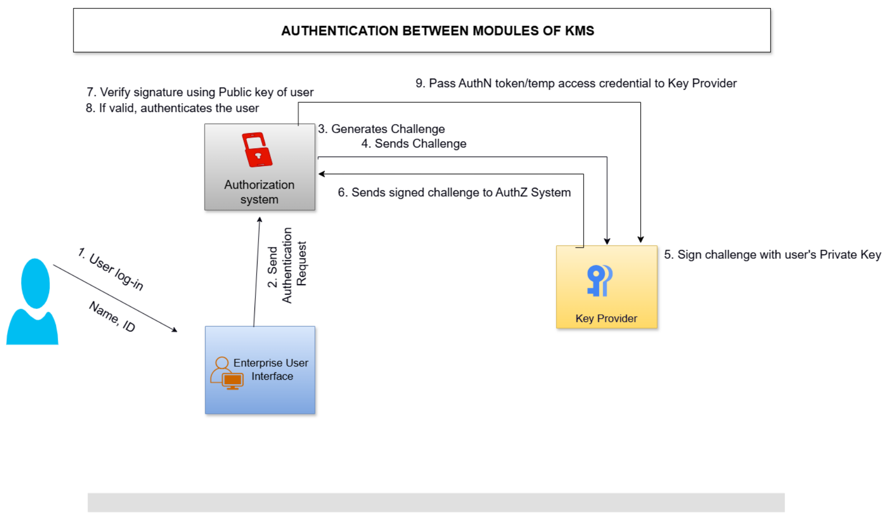

AUTHENTICATION BETWEEN MODULES OF KMS
7. Verify signature using Public key of user
8. If valid, authenticates the user
9. Pass AuthN token/temp access credential to Key Provider
3. Generates Challenge
4. Sends Challenge
Authorization system
6. Sends signed challenge to AuthZ System
5. Sign challenge with user's Private Key
Key Provider
1. User log-in
Name, ID
2. Send Authentication Request
Enterprise User Interface

Figure 4.2: Authentication b/w User and Key Provider

- To keep record of the secrets being generated and for whom and also with whom the shares are shared.
- To prevent any rogue dealer from impersonating a legitimate dealer and perform reconstruction of the secret by retrieving shares.

4.3 High-level Algorithm of Proposed KMS

1. User Registration:
    - User Request: A new user (e.g., User 3) initiates the registration process by submitting a request through the Enterprise User Interface (EUI). The user provides required information such as name, ID, etc.
    - Authorization Check: The request is passed to the Authorization System. The system validates the user's details to ensure that the user is authorized to register (is an employee of the enterprise).
    - Smart Key Generation: Once the authorization check is successful, the Key Provider generates a set of keys for the user such as Public/Private Key Pairs for authentication and encryption/decryption, Team Keys, Contingency Keys for collaborative and backup purposes, Key Versions for future updates of the keys and the ACLs specifying the permissions.
    - Access Control Lists (ACLs): The Key Provider creates an Access Control List (ACL) for the new user. This ACL defines Which keys the user can access, the user can use the keys (e.g., for encryption only, or both encryption and

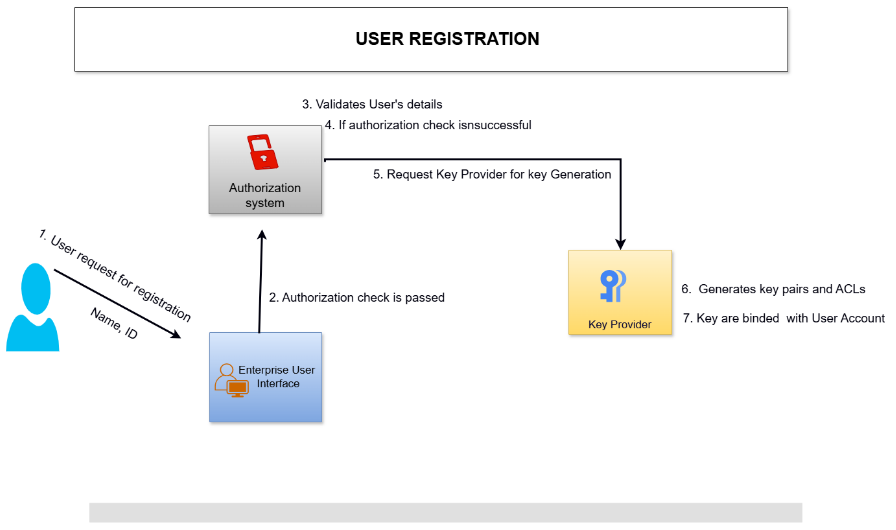

USER REGISTRATION
3. Validates User's details
4. If authorization check isnsuccessful
Authorization system
5. Request Key Provider for key Generation
1. User request for registration
Name, ID
2. Authorization check is passed
Key Provider
6. Generates key pairs and ACLs
7. Key are binded with User Account
Enterprise User Interface

Figure 4.3: User Registration

decryption) and list of users who can access the same keys (for collaborative or team-based tasks).

- User Authentication: The new user is required to authenticate with the Authorization System to bind the smart key with their account. The system may use multi-factor authentication (MFA) for added security during this step.
- Registration Complete: After completing the registration, the user can now access the Encryption/Decryption Application and perform secure file transfers. The user's smart key will be used to encrypt/decrypt files, and all operations will be logged for auditing purposes.

2. Encryption Algorithm: After the user gets authenticated and selects a key to be used for encryption of the file as explained earlier the encryption/decryption app receives the key selected by the user from the key provider and the unencrypted file to be encrypted along with the ID of the receiver from the user through EUI, the process of encrypting the file starts. Firstly, Enc/Dec app creates Session key and encrypt file with Session key and place it in .zip file. To create Master key using the LBSSS (Lattice Based Shamir Secret Sharing)mechanism, Dealer authenticates itself to get access to the encryption-decryption application and local-key provider and initializes LBSSS process(as explained in Section 4.4). The shares of Master key are stored at local Key provider (with master key ID). Dealer reconstruct Master key and sends it to encryption-decryption application application which then uses Master key to encrypt the session key, places it into .zip file. Encryption-decryption application must prove it has access to the cloud

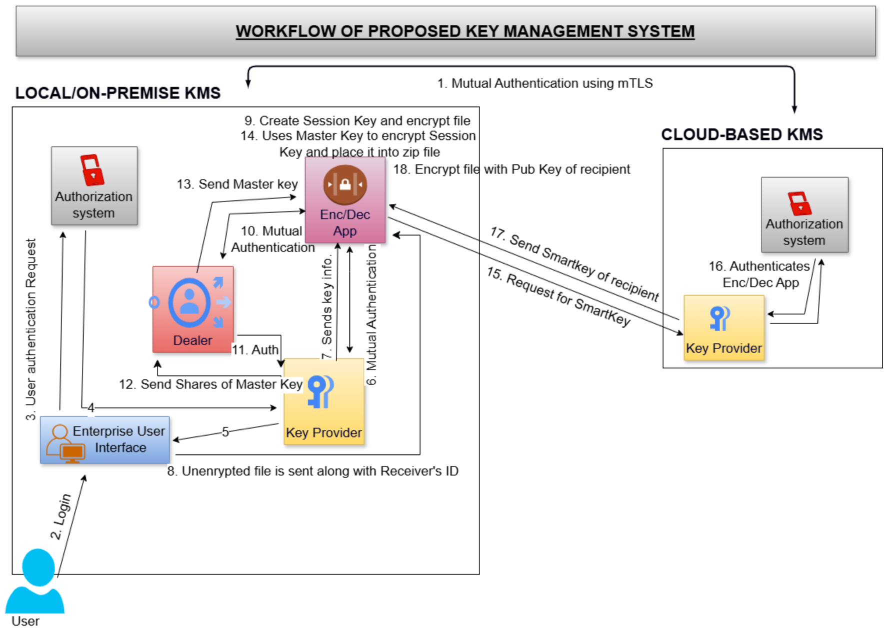

WORKFLOW OF PROPOSED KEY MANAGEMENT SYSTEM
LOCAL/ON-PREMISE KMS
CLOUD-BASED KMS
1. Mutual Authentication using mTLS
Authorization system
Enc/Dec App
Dealer
Key Provider
Enterprise User Interface
User
2. Login
3. User authentication Request
4
5
6. Mutual Authentication
7. Sends key info.
8. Unenrypted file is sent along with Receiver's ID
9. Create Session Key and encrypt file
10. Mutual Authentication
11. Auth
12. Send Shares of Master Key
13. Send Master key
14. Uses Master Key to encrypt Session Key and place it into zip file
15. Request for SmartKey
16. Authenticates Enc/Dec App
17. Send Smartkey of recipient
18. Encrypt file with Pub Key of recipient

Figure 4.4: Encryption Process of Proposed KMS

based key provider in order to get the public key of the recepient depending on the ID mentioned by the sender and use it to encrypts the.zip file. Based on the above steps, figure 4.4 illustrate the process of encrypting in proposed KMS.

3. Decryption Algorithm: The recipient (User2) request Encryption-decryption application to decrypt file using Enterprise User Interface. User2 get authenticated and granted access to Key Pprovider to select the key to be used for decryption. Encryption-decryption application app request Key provider for the receiver's Smart key. Key provider provides all the keys ID to user and user selects the one to be used for decryption of file which is kept in .zip file along with the encrypted file. Key provider sends key info. to the Encryption-Decryption application which then requests for the smartkey against the ID to the Cloud-based key provider.

   Encryption-Decryption application gets authenticated to get the services of Key provider which sends the smartkey back to Encryption-Decryption application. Encryption-decryption application uses private Key (K_pri) of receiver and decrypts .zip file. Encryption-decryption application app authenticates itself to Dealer to reconstruct Master key by providing ID Of the Master key. Dealer forwards Master key to Encryption-decryption application app after reconstruction so Encryption-decryption application may use Master key to decrypt Session key. Encryption-decryption application use encrypted session key to decrypt file and The decrypted file is sent to User2. Encryption-decryption application terminates session with User2. Figure 4.5 shows the decryption process of proposed KMS.

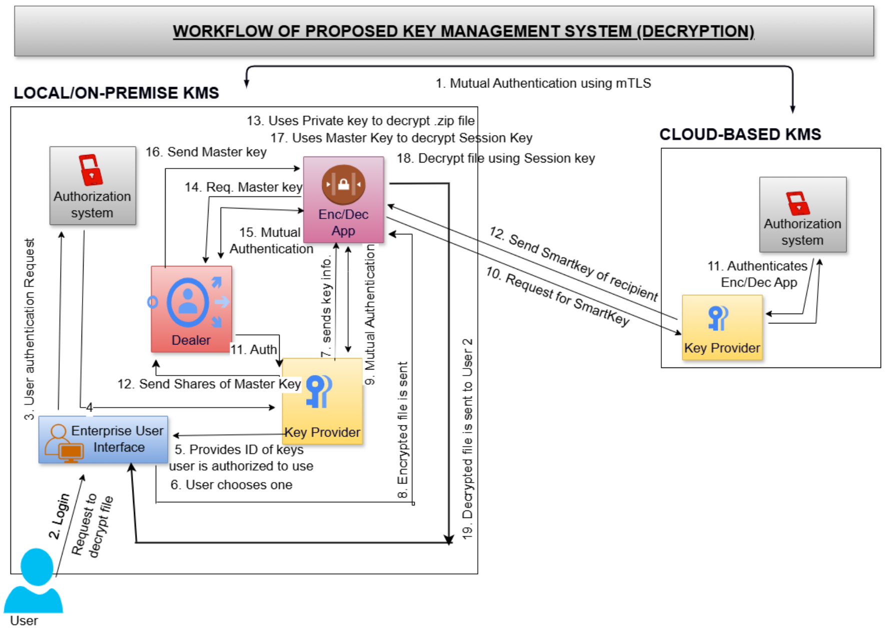

WORKFLOW OF PROPOSED KEY MANAGEMENT SYSTEM (DECRYPTION)
LOCAL/ON-PREMISE KMS
1. Mutual Authentication using mTLS
CLOUD-BASED KMS
13. Uses Private key to decrypt .zip file
17. Uses Master Key to decrypt Session Key
16. Send Master key
18. Decrypt file using Session key
Authorization system
14. Req. Master key
Enc/Dec App
15. Mutual Authentication
12. Send Smartkey of recipient
10. Request for SmartKey
Authorization system
11. Authenticates Enc/Dec App
Key Provider
3. User authentication Request
Dealer
11. Auth
7. sends key info.
9. Mutual Authentication
12. Send Shares of Master Key
Key Provider
4
Enterprise User Interface
5. Provides ID of keys user is authorized to use
6. User chooses one
8. Encrypted file is sent
19. Decrypted file is sent to User 2
2. Login
Request to decrypt file
User

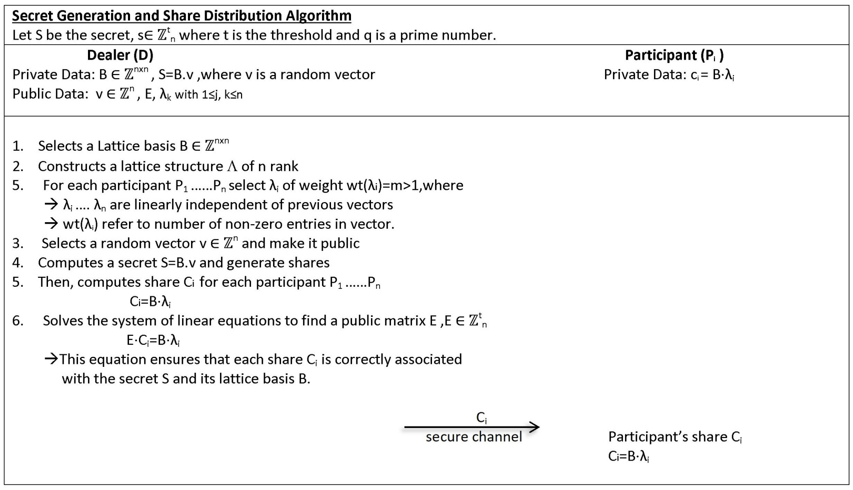
Figure 4.5: Decryption Process of Proposed KMS

4.4 Implementation Details of the Lattice-Based Shamir Secret Sharing Scheme

In the proposed KMS, an improved version of lattice based Shamir secret sharing is used for constructing Master Key, where shares are not stored as received but there is stored a computed share which is called Pseudo-secret which is sent back to the dealer/combiner at the time of reconstruction, the reason is to improve security of the shares at the time of rest and also Participants should not divulge the initial shares while trying to retrieve the secret or secrets. In the worst scenario, lattice-based cryptosystems can be proven to be secure based on the hardness of the lattice issues.

- Threshold Requirement: The correct reconstruction of B requires at least t linearly independent λ_i vectors. This ensures that the threshold t is respected, meaning that fewer than t participants cannot reconstruct the secret.
- Security: The security of this process is based on the hardness of reconstructing the lattice basis B without sufficient information (i.e., fewer than t shares).

The use of lattice problems, such as the Shortest Vector Problem (SVP), ensures that the scheme is resistant to unauthorized reconstruction. use of Permutation matrix The permutation matrix P transforms the matrix E into a new matrix =Ei=EP.



Secret Generation and Share Distribution Algorithm

Let S be the secret, s$\in \mathbb{Z}^t_n$ where t is the threshold and q is a prime number.

Dealer (D)

Private Data: B $\in \mathbb{Z}^{nxn}$, S=B.v ,where v is a random vector

Public Data: v $\in \mathbb{Z}^n$, E, λ_k with 1≤j, k≤n

Participant (P_i)

Private Data: c_i= B·λ_i

```
1.  Selects a Lattice basis B ∈ $\mathbb{Z}^{nxn}$
2.  Constructs a lattice structure $\Lambda$ of n rank
5.  For each participant $P_1$ ......$P_n$ select $\lambda_i$ of weight wt($\lambda_i$)=m>1,where
      → $\lambda_i$ .... $\lambda_n$ are linearly independent of previous vectors
      → wt($\lambda_i$) refer to number of non-zero entries in vector.
3.  Selects a random vector v ∈ $\mathbb{Z}^n$ and make it public
4.  Computes a secret S=B.v and generate shares
5.  Then, computes share $C_i$ for each participant $P_1$ ......$P_n$
          $C_i$=B·$\lambda_i$
6.  Solves the system of linear equations to find a public matrix E ,E ∈ $\mathbb{Z}^t_n$
          E·$C_i$=B·$\lambda_i$
      →This equation ensures that each share $C_i$ is correctly associated
        with the secret S and its lattice basis B.
```

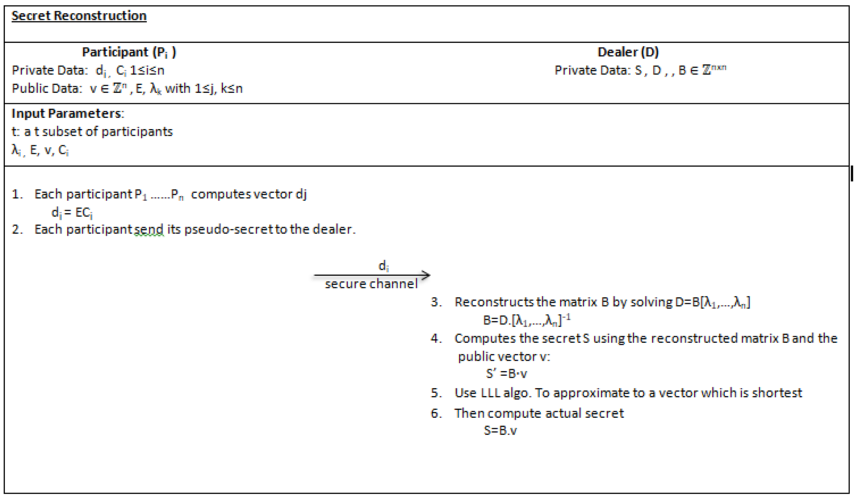
$\xrightarrow[\text{secure channel}]{C_i}$ Participant's share C_i

C_i=B·λ_i

Figure 4.6: Secret Generation and Distribution Algorithm

This operation essentially reorders the rows or columns (depending on whether P is applied to the left or right of E) of the matrix E. This reordering adds an extra layer of complexity for anyone trying to infer the structure or properties of E just from observing the public matrices Ei. Since P is a permutation matrix, the rows or columns are merely shuffled, not altered in value. Still, this shuffling can prevent certain attacks by making it harder to correlate Ei directly with the underlying secret.

By introducing a permutation matrix P, the scheme becomes more resistant to attacks that exploit predictable structures in the matrix E. For instance, without the permutation, if E has certain patterns, an attacker might use these to deduce the secret more easily. The permutation obscures such patterns. Instead of generating a new matrix Ei for each share or each participant, the same matrix E can be used across different parts of the scheme by simply applying different permutations via P. This reduces the memory footprint because you only need to store one base matrix E and the permutation matrix P, rather than multiple distinct matrices Ei. The permutation matrix effectively allows the same base matrix E to serve multiple roles or be associated with different participants or secrets, thereby reducing the overall amount of public information that needs to be stored and managed.


Secret Reconstruction

Participant (P_i)
Private Data: d_i, C_i $1 \le i \le n$
Public Data: $v \in \mathbb{Z}^n$, E, λ_k with $1 \le j, k \le n$

Dealer (D)
Private Data: S, D, , $B \in \mathbb{Z}^{n \times n}$

Input Parameters:
t: a t subset of participants
λ_i, E, v, C_i

1. Each participant P_1 ......P_n computes vector dj
   $d_j = EC_j$
2. Each participant send its pseudo-secret to the dealer.

$\xrightarrow[\text{secure channel}]{d_i}$

3. Reconstructs the matrix B by solving $D=B[\lambda_1,\ldots,\lambda_n]$
   $B=D.[\lambda_1,\ldots,\lambda_n]^{-1}$
4. Computes the secret S using the reconstructed matrix B and the public vector v:
   $S' = B \cdot v$
5. Use LLL algo. To approximate to a vector which is shortest
6. Then compute actual secret
   $S=B.v$

Figure 4.7: Secret Reconstruction Algorithm

Figure 4.6 and 4.7 explains how the secret is generated, distributed and reconstruction of the Lattice-Based Shamir Secret Sharing Scheme with its improved version which involves pseudo-secret at the time of reconstructing the secret.

The modulo operation in both the algorithms ensures that the vector is correctly aligned with the original lattice basis. As we are using the Shortest Vector Problem (SVP), after the dealer reconstructs the secret S', it uses a lattice reduction algorithm known as (LLL) algorithm or BKZ (Block Korkin-Zolotarev), in order to approximate the shortest vector in the lattice in order to retrieve the actual secret S. The shortest vector v_{short} found from the lattice reduction process should be very close to the original secret S.

The reconstructed secret S' is obtained by taking the shortest vector and adjusting it if necessary to account for any small errors introduced by the noise vectors v_i. The relation is:

$$S \equiv v_{\text{short}} \mod B$$

4.4.1 Mutual TLS (mTLS) Certificate

At the start of each session, the KMSs at the local/On-Premise and at the Cloud-Based are mutually authenticated to make the transfer of the key material secure using the mTLS protocol. With the help of digital certificates, the client and server authenticate one another using the mTLS handshake, which expands on the basic TLS (Transport Layer Security) protocol. For the proposed system mTLS ensures secure transfer of key material between the local KMS and the Cloud-Based KMS.

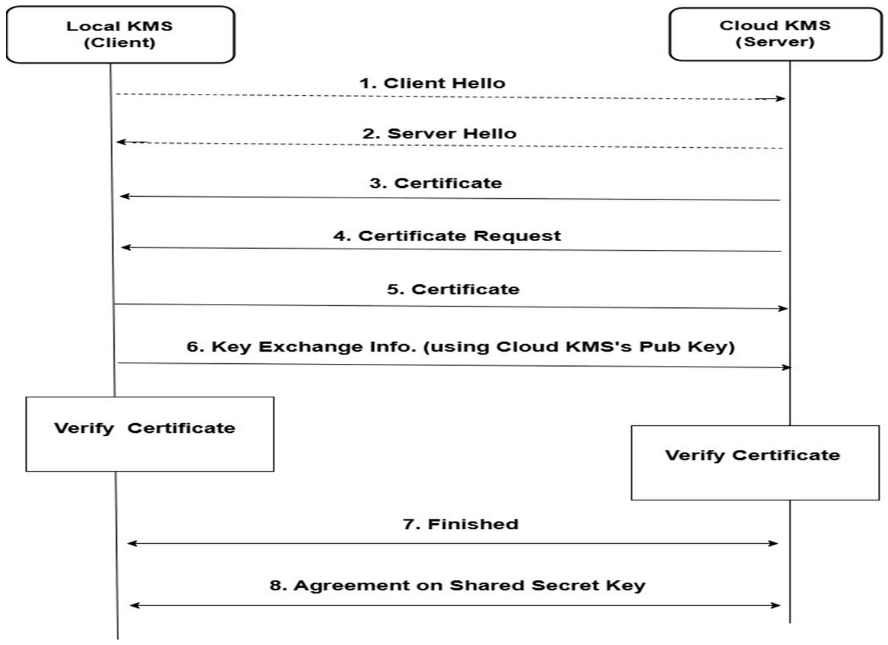


Figure 4.8: mTLS Handshake

Here are some details for the mTLS Handshake:

- **Local-KMS Hello:** For the handshake to initiate, the Local-KMS first sends a "Local-KMS Hello" message containing information about the supported TLS version(s), cipher suites (sets of encryption algorithms), a nonce value (a random number used to generate session keys), and, if it's desired, the Local-KMS's public key.
- **Cloud-KMS Hello:** The Cloud-KMS then sends a "Cloud-KMS Hello" message that contains the selected TLS version, the cipher suite, a nonce that aids in the generation of session keys, and any extensions that provide further configuration data.
- **Cloud-KMS Certificate:** To authenticate itself, the Local-KMS uses the digital certificate that the server transmits along with public key. A reliable Certificate Authority (CA) has signed the certificate.
- **Cloud-KMS Certificate Request:** To ensure mutual authentication, the Cloud-KMS makes a request for the Local-KMS's certificate. This is a key difference between mTLS and standard TLS.
- **Cloud-KMS Hello Done:** The Cloud-KMS indicates it has finished its portion of the handshake and is waiting for the Local-KMS's response by sending a "Cloud-KMS Hello Done" message.

- **Local-KMS Certificate:** For authentication, the client transmits the Cloud-KMS a digital copy of its certificate.
- **Local-KMS Key Exchange:** Making use of Cloud-KMS's public key, a "pre-master secret" is created by the Local-KMS and is sent to the Cloud-KMS encrypted. To generate session keys, both sides utilise the nonces and pre-master secret.
- **Certificate Verify:** Possession of the private key associated with the certificate is demonstrated by the Local-KMS signing and sending to the Cloud-KMS a hash of all previous handshake messages.
- **Local-KMS Finished:** With the session key encrypted, the client sends a "Finished" message that is a hash of all prior handshake messages.
- **Cloud-KMS Finished:** When the server replies with a "Finished" response of its own, the handshake is complete. Along with being encrypted with the session key, this message also contains a hash of previous communications.
- **After the Handshake:** After the handshake, a secure, encrypted communication channel has been formed and both the client and server have verified one another. To maintain secrecy and integrity, all upcoming communications will be encrypted using session keys that are generated from the pre-master secret.

4.4.2 Lenstra–Lenstra–Lovász (LLL) Algorithm

An orthogonal, virtually structured lattice basis can be obtained by applying the polynomial-time LLL method to a given lattice basis. One can tackle issues such as these using a reduced basis:

- The Shortest Vector Problem (SVP) is the task of determining the shortest possible non-zero lattice vector
- A lattice vector's closest match to a given vector is found using the Closest Vector Problem (CVP).

The LLL algorithm doesn't always find the exact shortest vector but provides a good approximation of the shortest vector. This makes it practical for many cryptographic applications where exact solutions are not necessary but good approximations are sufficient.

How Does the Lattice-Based Shamir Secret Sharing Scheme Benefit from LLL? The initial secret is encoded as a lattice vector in our lattice-based Shamir secret sharing system, and shares are generated by appending noise to this vector. The objective of the reconstruction phase is to solve the Shortest Vector Problem (SVP) on the lattice made up of the shares in order to retrieve the original secret. Here's where the LLL algorithm comes into play:

1.

2. Lattice Reduction for Reconstruction: The LLL algorithm can reduce the lattice basis formed by the shares. This reduced basis makes it easier to find an approximate solution to the SVP. Lattice reduction techniques like LLL provide a good approximation to the shortest vector. Since exact solutions to SVP are computationally expensive, LLL offers a balance between efficiency and accuracy.

3. Noise Tolerance: The LLL algorithm helps reduce the impact of the noise added to the shares during the splitting process. By reducing the basis, it removes redundant and noisy components, bringing the reconstructed vector closer to the original secret vector. This reduction minimizes the errors that arise from noise during the reconstruction phase.

4. Efficient Approximation: Exact solutions to SVP are NP-hard, and attempting to solve them directly for large lattices is computationally infeasible. The LLL algorithm provides an approximation to the shortest vector in polynomial time, making it practical for cryptographic applications.

In our case, using LLL during the reconstruction phase helps to efficiently approximate the secret vector from noisy shares. Steps involved using LLL for Reconstruction:

- Construct the Lattice:

  During the reconstruction phase, the shares form the lattice. We construct a lattice using the given shares (vectors), which will serve as the basis for the LLL algorithm.

- Apply LLL Reduction:

  The LLL algorithm is applied to reduce the lattice basis. This step transforms the shares into a more structured form, helping us recover the approximate shortest vector, which corresponds to the original secret. Reconstruct the Secret:

After applying LLL, the secret is recovered by solving the SVP on the reduced lattice. The reduced lattice allows us to efficiently approximate the original secret vector.

4.5 How Security Is Enhanced Against Insider And Quantum Threats

The proposed Key Management System (KMS) using Lattice-Based Shamir Secret Sharing Scheme (LBSSS) strengthens security against both insider and quantum threats through several critical design features:

- Decentralization to Counter Insider Threats: The proposed KMS decentralizes key storage and generation across both on-premise and cloud components, reducing the reliance on any single entity. Insider threats, such as malicious administrators with elevated privileges, are mitigated since no single insider can access enough information to reconstruct the master key. The keys are split into multiple shares distributed across independent systems, and only a predefined threshold number of shares can reconstruct the key. This division ensures that even if an insider accesses some shares, they cannot compromise the entire system without controlling the majority.
- Quantum-Resistant Cryptography: LBSSS leverages lattice-based cryptographic techniques that are resistant to quantum attacks. Lattice problems, such as the Shortest Vector Problem (SVP), are considered hard even for quantum computers. This ensures that cryptographic secrets remain secure against adversaries using quantum algorithms, such as Shor's algorithm, which can break classical cryptographic schemes. By employing post-quantum cryptography, LBSSS future-proofs the KMS against the inevitable rise of quantum computing.
- Threshold-Based Security: In the proposed scheme, LBSSS divides the master key into a number of shares and a threshold value is set in order to reconstruct original secret, this means that even though a quantum adversary or insider gains access to fewer than the threshold number of shares, they cannot reconstruct the key. The difficulty of the lattice problems further enhances the security to reconstruct key ensures it remains resilient against attacks that exploit quantum computing power.
- Mutual Authentication and Secure Communication: The KMS employs mutual Transport Layer Security (mTLS) to ensure secure communication between the cloud-based and on-premise components. This prevents unauthorized entities from intercepting or tampering with key transfer processes, further enhancing the security of the system against insider and external threats. Additionally, the authorization system ensures that only authenticated users and processes can access the key provider and encryption/decryption applications.

- Noise-Resilient Key Reconstruction: The LBSSS enhances resilience by using lattice reduction algorithms, such as Lenstra-Lenstra-Lovász (LLL), during the key reconstruction process. These algorithms provide robustness against noise and errors that may arise from insider manipulation or external quantum-based attempts to reconstruct the key. Even in the presence of noisy or compromised shares, the LBSSS can accurately reconstruct the original secret.
- Access Control: Associating an access control policy with each key to permit, deny or restricts the access to the keys resolves the security issues as no other than the authorized person can have access to the key material.
- Shortest Vector Problem(SVP): Using SVP with lattices makes the algorithm computationally difficult to solve, especially in high dimensions, which makes it a strong foundation for cryptographic schemes. The hardness of SVP is thought to be immune to even quantum computing.

4.6 Conclusion

The chapter provided a detailed explanation of the working and implementation of the proposed Lattice-Based Shamir secret sharing scheme-based Key Management System. The chapter also discussed the authentication mechanism used to authenticate the different modules/components of the proposed Key Management System.

Chapter 5

ANALYSIS OF LATTICE-BASED SHAMIR SECRET SHARING SCHEME (LBSSS) AND PROPOSED KMS

Using lattice-based Shamir Secret Sharing not only enhances security, also efficiency is improved as there's no need to make the key size larger to achieve better security.

We may now summarise the results in the following context if we wish to look into the secrecy, integrity, and availability:

- Secrecy: To find out the secret, an eavesdropper must get hold of at least t shareholders and take their shares.
- Integrity: To destroy or change the secret, an eavesdropper must corrupt at least n-t + 1.
- Availability: If the adversary knows threshold value (t) is known, then the probability of secret availability will grow as n increases. Secrecy and integrity will be improved with the increases in t and n (number of shareholders).

The suggested technique uses r+n memory to store the size of public values for each secret size.

The memory used by proposed scheme regarding the size of public values per secret size is r+n and size of each share per secret is equal to r/(t log q), which approximately equals to 0.5, if $rmaxtlogt, n \approx tlogt$.

5.1 Description Of Experiments And Evaluation Metrics

In the context of complexity, secret recovery consists of two steps:

- The side of the participants: Each member calculates his pseudo-secret share from his share in this stage. Since the shares are binary vectors, we only need to do a simple column addition in matrix (E_i) in order to get the pseudo-secret shares. For every participant in this process, the complexity is $\mathbf{O}(t_r)$, which is less than the modular exponentiation complexity found in other techniques. As such, this technique is appropriate for applications that are not too complex.
- The combiner's side: The complexity of the step is $\mathbf{O}(t^3)$, which includes two operations, one is matrix inversion and other one is matrix multiplication. The complexity of inverting a general t x t matrix is O(t^3) using standard algorithms such as Gaussian elimination or LU decomposition. This complexity arises because matrix inversion involves multiple steps, including row reductions and back substitutions, which require t^2 operations for each of the t rows in the matrix. Thus, the overall complexity of inverting a matrix is proportional to t^3.

5.2 Comparative Analysis of Experimental Findings

5.2.1 Classical Shamir Secret Sharing Scheme

1. Complexity Analysis:
   - Secret Splitting (Polynomial Evaluation): The time complexity for polynomial evaluation is $O(k)$, where each share represents a value generated by evaluating the degree $k-1$ polynomial at some point x.
   - Reconstruction (Lagrange Interpolation): The time complexity of Lagrange interpolation is $O(k^2)$, where k represents the total number of shares which are used for reconstruction. For each share, it needs to run a nested loop over all other shares and compute the Lagrange basis polynomial. This process involves computing modular inverses and performing multiplications.
2. Performance Analysis: Secret sharing scheme has been implemented on a 1.00GHz Intel m3-7Y30, running Windows 10 pro 64 Bit (the code can be found in the Appendix). The performance results of the implementation reported for n=100 and t=15, p = 7919, where P is prime number larger than the secret and larger than all intermediate values, of Shamir Secret Sharing scheme, is shown in Table 5.1.

Table 5.1. Performance Analysis of Shamir Secret Sharing Scheme

Algorithm	**Memory Consumption**	**Computational time**
Secret, share generation	0.011719 MB	0.220823 seconds
Secret Reconstruction	0.031250 MB	0.204482 seconds

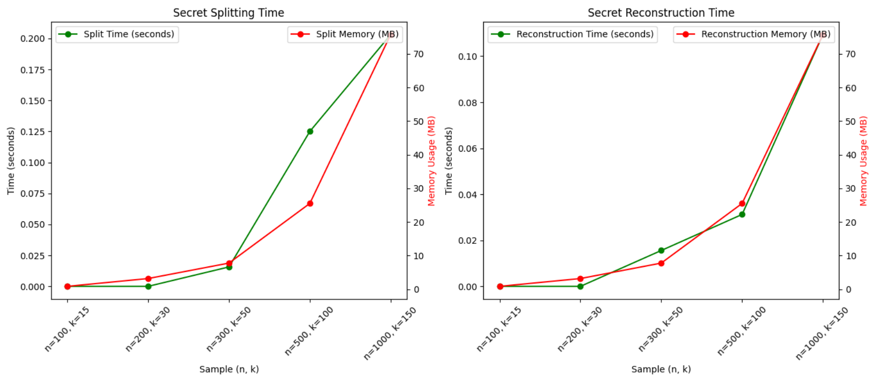


Figure 5.1: Memory and Time Consumption Analysis of SSS

5.2.2 Lattice-Based Shamir Secret Sharing Scheme

1. Complexity Analysis:

   - Secret Creation and Distribution Phase:

     The computational complexity in the distribution phase is centered around matrix A_i, which is pivotal for generating and distributing shares. The computational complexities include:

     - $O(t^2n)$: Represents the complexity of generating shares for n participants, where t is the threshold for secret reconstruction.

     - $O(tn(r-n))$: Accounts for the matrix multiplications involved in computing shares using lattice encoding.

     - $O(n^3)$: Comes from matrix inversion, which is the most computationally intensive operation in this phase, making it the dominant cost.

     - $O(tn^2)$: Represents the matrix multiplications during the distribution process to ensure shares are properly encoded within the lattice framework.

     Summing these complexities, the overall computational complexity for share distribution is $O(n^3)$ per secret, primarily due to matrix operations, particularly matrix inversion. While this phase is computationally expensive, the lattice structure offers strong security against quantum threats. For the proposed KMS, the use of pseudo-secrets introduces an additional computation step, as the original secret must first be obfuscated before generating the shares. However, this step does not impose significant overhead, as pseudo-secret generation can be performed efficiently using standard cryptographic primitives.

- Share Distribution: Distributing shares to participants requires securely transmitting them over a network. This process introduces minimal computational overhead since it mainly involves communication protocols. The complexity for share generation and distribution remains $O(n * k)$, making it scalable for a moderate number of participants.
- Secret Reconstruction Phase:

  In the reconstruction phase, participants compute the pseudo-secret shares to reconstruct the original secret. The binary nature of the shares simplifies operations in this phase, making computations more efficient:

  - Binary representations allow participants to perform simple bitwise operations, which reduces the overall computational cost. - Unlike the distribution phase, where matrix operations dominate, the recovery phase involves simpler computations, making it less computationally intensive. The system ensssures that even though the distribution phase is complex, the recovery process remains efficient for participants.

2. Performance Analysis:

   - Memory Usage:

     In the first phase of the Lattice-Based Shamir secret sharing scheme, memory usage is primarily dominated by the matrices A_i, which are crucial for the operations of the Lattice-Based Shamir Secret Sharing Scheme. These matrices are used with the purpose to create shares for multiple participants and occupy a significant portion of available memory resources. Each participant receives a share, s_j, which also contributes to memory usage.

     The size of each share s_j is optimized to $r \cdot t \log q$, where:

     - q is defined as $\frac{t^2}{2}$, - r is $t \log t$.

     These optimizations help reduce memory overhead, which is vital when managing many shares and participants. By limiting the share sizes, the system ensures scalability, even in resource-constrained environments.

     In the second phase, after the shares are distributed, the matrices A_i are updated to A_i'. At this stage, only additional vectors s_j'' are transmitted to participants. This step is designed to minimize the amount of data sent over secure channels, reducing bandwidth consumption and improving system efficiency—especially in distributed cloud environments where communication costs are key.

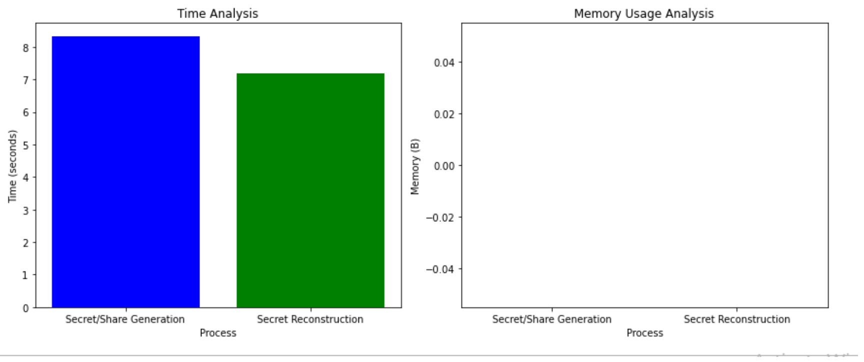


Figure 5.2: Memory and Time Consumption Analysis of LBSSS

For n=3 and t=3, p = 11 where P is prime number larger than the secret and larger than all intermediate values, analysis of computational speed and memory consumption of Lattice-Based Shamir Secret Sharing scheme (the code implementation can be found in Appendix), is shown in Table 5.2

Table 5.2. Performance Analysis of Lattice-Based Shamir Secret Sharing Scheme

Algorithm Phase	**Memory Consumption**	**Computational time**
Secret Generation and Distribution	0.007188 B	8.318776 seconds
Secret Reconstruction	0.003906 B	7.201234 seconds

The figure below shows the memory and time consumed during the secret generation, share distribution and secret reonstrution phases of the LBSSS. Memory graph shows the memory consumption of the scheme in Bytes, as the process consumed ignorable memory whih is less than a Byte, therefore, graph does not show any value.

5.3 Security Analysis of Proposed KMS Based on LBSSS

The combination of the proposed key management system (KMS) and LBSSS ensures a robust, future-proof, and tamper-resistant security model.

1. Quantum Resistance: LBSSS mitigates quantum threats, which traditional cryptosystems are vulnerable to.
2. Confidentiality and Integrity: Both systems ensure that sensitive data, keys, and secrets remain protected, valid, and only accessible to authorized participants.

3. Collusion and Insider Threat Mitigation: The use of thresholds and distributed shares in LBSSS, along with the KMS's role-based access, prevents collusion and insider attacks.

4. Availability and Fault Tolerance: The distributed nature of both the KMS and LBSSS ensures the availability of keys and secrets during system failures or unavailability of participants.

   This integrated security framework offers significant resilience against both traditional and emerging threats, ensuring long-term security and reliability for the key management system as shown below in table 5.3 and 5.4.

Attack Type	**Threat**	**Defense**	**How LBSSS Helps**
Quantum Attacks (Shor's Algorithm)	Quantum computers can efficiently break traditional cryptosystems such as RSA and ECC using Shor's algorithm.	Lattice-based cryptography is resistant to quantum attacks because the underlying problems (e.g., SVP and LWE) are believed to be quantum-resistant.	Hard lattice problems make it infeasible to solve encoded and reconstructed shares, even with quantum computing power.
Brute Force Attacks	The attacker tries all possible combinations of shares or secrets to recover the original secret.	With a large modulus q and properly generated matrices, the search space is exponentially large, making brute force infeasible.	High-dimensional lattices and modular arithmetic make brute-forcing computationally infeasible.
Eavesdropping (Man-in-the-Middle Attack)	An attacker intercepts communication between parties to collect shares and reconstruct the secret.	Modular arithmetic with a large prime q makes it difficult to derive the secret. Communication can also be encrypted or authenticated.	Intercepted shares are useless unless the attacker obtains at least t shares.

Table 5.3. LBSSS - Common Attacks and Defense Mechanisms (Part 1)

Attack Type	**Threat**	**Defense**	**How LBSSS Helps**
Collusion Attacks	Some participants collaborate to reconstruct the secret without meeting the threshold.	The threshold t ensures that fewer than t participants cannot successfully reconstruct the secret, even if they collude.	Lattice encoding ensures partial information from collusion does not reveal the complete secret.
Insider Attacks (Malicious Participants)	A dishonest participant tries to manipulate their share or provide incorrect data to disrupt reconstruction.	Modular arithmetic and lattice-based encoding ensure that tampered or incorrect shares fail the reconstruction process.	Matrix inversion and consistency checks ensure only valid shares are accepted during reconstruction.
Key Compromise Attack (Single Point of Failure)	In traditional systems, the compromise of a single key or share can expose the secret.	LBSSS distributes the secret across multiple shares, and the threshold t ensures that fewer than t shares reveal no useful information.	The distributed nature of LBSSS eliminates the risk of a single point of failure.
Replay Attacks	An attacker intercepts a share during distribution and tries to reuse it to access the secret later.	Time-sensitive or session-specific secrets can be used, and shares can be recomputed with different lattice bases per session.	Randomized share generation ensures uniqueness, preventing reuse across sessions.
Fault Injection Attacks	The attacker manipulates input or computations (e.g., matrices or shares) to produce incorrect results.	Redundancy and verification during reconstruction (e.g., matrix inversion checks) ensure tampered data fails to pass.	If any share or matrix is tampered with, the reconstructed matrix and secret will not match, ensuring fault detection.

Table 5.4. LBSSS - Common Attacks and Defense Mechanisms (Part 2)

5.4 Threat Model of Proposed Key Management System

The threat model for proposed KMS uses lattice-based cryptography, mutual authentication, and threshold-based key sharing to identify and mitigate potential threats. Secure key management across cloud and on-premise components is maintained, and strong resistance against insider threats, external attacks, and quantum threats is enforced.

1. **Assets in the System**
    - Master Keys and Session Keys: Critical for encryption and decryption of sensitive data.
    - Key Shares: Parts of the secret key distributed across cloud and on-premise systems.
    - Lattice-based Parameters: Includes public matrix E, lambda vectors, and lattice basis B.
    - Access Control Policies: Define which users or components have permission to access keys and perform operations.
2. **Threat Actors**
    - Malicious Insiders: Employees with access to key shares who may attempt unauthorized reconstruction.
    - External Attackers: Hackers or adversaries targeting cloud components to access shares or keys.
    - Quantum Adversaries: Attackers using quantum computing to break classical cryptographic protections.
    - Man-in-the-Middle (MitM) Attackers: Adversaries attempting to intercept communication between cloud and on-premise systems.
    - Rogue Administrators or Trusted Third-Party Misuse: System administrators misusing elevated privileges to reconstruct secrets.
3. **Attack Vectors**
    - Unauthorized Key Reconstruction: Insiders gaining access to a subset of shares in an attempt to reconstruct the full key.
    - Compromised Communication Channels: Intercepting data transferred between cloud and on-premise systems.

- Compromise of Cloud Storage: Attackers gaining unauthorized access to key shares stored in the cloud.
- Quantum Threats: Exploiting quantum algorithms (e.g., Shor's algorithm) to break classical encryption used for securing key material.
- Side-Channel Attacks: Extracting information through timing analysis, cache access, or power consumption during encryption and decryption operations.

4. **Vulnerabilities in the System**

   - Single Point of Failure at Key Providers: If cloud or on-premise key providers are compromised, partial key information may be leaked.
   - Matrix Inversion Operations: Matrix inversion can be computationally expensive, leaving the system vulnerable to timing-based side-channel attacks.
   - Complexity of Access Control Lists (ACLs): Incorrectly configured ACLs may lead to unauthorized users gaining access to key material.
   - Noisy or Tampered Shares: If attackers modify shares, key reconstruction may fail, resulting in denial of service.

5. **Mitigation Strategies**

   - Decentralization of Key Storage: The use of threshold-based reconstruction ensures that no single entity (cloud or on-premise) has full control over the key.
   - Lattice-Based Cryptography: The use of lattice-based encryption (SVP and LLL algorithms) ensures quantum resistance and noise tolerance in key reconstruction.
   - Secure Communication via mTLS: Mutual TLS (mTLS) ensures encrypted communication between cloud and on-premise systems, preventing MitM attacks.
   - Access Control Policies: ACLs are used to strictly define access permissions, reducing the risk of unauthorized access to key material.
   - Audit Logging and Monitoring: Log every access attempt to key material to detect potential insider threats or rogue administrators.
   - Noise-Tolerant Key Reconstruction: The use of the LLL algorithm ensures that the system can still function even if some shares are noisy or tampered with.

     The STRIDE framework provides a structured way to identify, classify, and mitigate potential security threats to proposed KMS, as shown in Table 5.5.

Threat	**Type (STRIDE)**	**Description**	**Mitigation**
Unauthorized key reconstruction	Elevation of Privilege (E)	Insider attempts to gather enough shares to reconstruct the key.	Threshold reconstruction (t-out-of-n).
Compromised cloud storage	Information Disclosure (I)	Cloud-based key shares are accessed by attackers.	Decentralized storage and ACLs.
Man-in-the-Middle attack	Spoofing (S)	Attacker intercepts communication between cloud and on-premise systems.	Use mTLS for secure channels.
Quantum computing attack	Tampering (T)	Quantum algorithms attempt to break classical encryption.	Lattice-based cryptography (post-quantum).
Rogue administrator access	Elevation of Privilege (E)	Admin misuses access to retrieve and reconstruct keys.	ACLs and audit logs.
Noisy or tampered shares	Denial of Service (D)	Shares are modified, causing reconstruction to fail.	LLL algorithm to tolerate noise.

Table 5.5. STRIDE Threat Categorization for Proposed KMS

5.5 Conclusion

The Lattice-Based Shamir Secret Sharing Scheme (LBSSS) strengthens security while improving efficiency, without the need for larger key sizes. Its lattice-based structure provides strong post-quantum resistance, ensuring secrecy, integrity, and availability, with enhanced protection as the number of participants increases. Although LBSSS involves higher computational complexity due to matrix operations, it remains highly efficient for modern cryptographic requirements. Experimental results confirm its scalability and suitability for lightweight, secure applications, making it a strong candidate for future cryptographic systems.

Chapter 6

DISCUSSION

Certain aspects has to be examined for potential future enhancements to improvise the security of the proposed Key Management System (KMS) based on the Lattice-Based Shamir Secret Sharing Scheme (LBSSS). Issues including user authentication, data leakage measures, access control, and the location of key material is unavoidable to have a strong system. This chapter explains specific changes that could be made to augment security and scalability of the framework so that it can meet new cryptographic challenges in the future.

6.1 Addressing Potential Future Improvements

A few major security concerns in the Key Management System are thought to be user authentication, access control, data loss, key material availability, and key management. . Some of the improvements to bolster the security of these concerns are suggested as below:

- Verifiability in the Lattice-Based Shamir Secret Sharing Scheme A key improvement for the framework is adding verifiability to the lattice-based Shamir secret sharing scheme. This would allow participants to verify the correctness of their shares, enhancing security by preventing potential malicious behavior and ensuring trust in distributed environments.

- Expanding Beyond Enterprise Applications Currently focused on enterprise-based applications, the framework can be expanded to broader contexts, such as cloud environments and multi-organizational collaborations. This would increase its flexibility and scalability, making it applicable to a wider range of modern cryptographic needs.

- Evaluating the Framework Against Common KMS Attacks Another important future step is to evaluate the framework against common attacks on Key Management Systems (KMSs). By rigorously testing the framework's resilience to these attacks, any potential vulnerabilities can be identified and addressed, further strengthening the overall security of the system.
- Integrating Post-Quantum Standards By incorporating the post-quantum cryptographic standards for key encryption and digital signatures the security of the proposed Key Management System will offer a high level of efficiency and protection in the future for authentication, key management, and key exchange processes.

6.2 Considerations For Real-World Deployment and Scalability

When deploying the LBSSS-based KMS in real-world applications, several important factors must be considered:

1. Performance Overhead: The scheme introduces some overhead, especially during the reconstruction phase, where lattice reduction algorithms like LLL or BKZ are used. Ensuring the system can efficiently handle this overhead is crucial, particularly in high-traffic environments or those requiring frequent encryption and decryption operations.
2. Scalability: LBSSS is well-suited for scalable systems. With increase in number of participants or shares, the system can manage larger workloads. However, larger enterprises will need to allocate sufficient resources, such as memory and computational power, to ensure smooth operation.
3. Security in Distributed Systems: The proposed KMS distributes key management between on-premise and cloud-based systems, offering redundancy and enhanced security. Ensuring secure communication between these systems is essential. The use of mTLS for mutual authentication helps address this, but continuous monitoring is needed to prevent insider threats and misconfigurations.
4. Adoption of Post-Quantum Standards: As post-quantum cryptography advances, integrating LBSSS into existing infrastructures will require adherence to emerging standards. Ensuring that the system can adapt to future updates in cryptographic protocols is key to maintaining long-term viability.
5. Cost and Resource Management: In practice, deploying LBSSS will require investment in hardware, software, and regular maintenance. Organizations need

to plan for both the initial setup and ongoing resource management, ensuring there is enough computational power to handle the complexity of cryptographic operations. A cost-benefit analysis will be essential to balance the enhanced security against the additional resources required for large-scale implementation.

6.3 Conclusion

This chapter identifies the major issues that could affect security in the proposed system and also indicates possible future changes for increasing the system reliability for implementation. These enhancements are designed to enhance the security of Lattice-Based Shamir Secret Sharing Scheme (LBSSS) and to expand the framework's capabilities to go beyond enterprise systems, to consider the verifiability of the scheme, and to lay the ground work for post-quantum standards.

Chapter 7

SUMMARY OF RESEARCH WORK

The objective of the thesis is to develop a quantum-safe KMS for cloud-based computing with the use of a combined cryptographic environment. The service of cloud computing paradigms has incorporated various security issues, particularly when data is managed through the multi-tenant environment in public cloud services. There are extra security threats from the internal and the external environment these include the insider threat and the quantum threat that threatens to overshadow the traditional cryptographic models. This work aims to address these problems using a proposed Future-oriented Distributed Key Management System (FDKMS), and implementing the Lattice-Based Shamir Secret Sharing Scheme (LBSSS) as part of the post-quantum cryptography.

The research is driven by the existing drawbacks of conventional centralised systems, where key generation and storage are frequently handled by the single controlling centre/entity, making it a possible target for hackers. The cryptography key management hierarchical structures can be threatened by internal attackers who have high-level accesses and threaten the encrypted information by corrupting the central key managing entity, also centralized management systems are also prone to external quantum threats that could exploit the classical cryptographic algorithms. This thesis introduces the utilization of LBSSS for decentralized key management within cloud or on-premise KMS components as presented in Figure 4.1. The proposed Key Management System incorporate lattice-based cryptography which is post-quantum secure to encrypt and protect the keys from both classical and quantum threats thereby minimizing on the issue of single point failure concession.

One key innovation introduced is the integration of Lattice-Based cryptography into classical Shamir Secret Sharing Scheme, that combines the robustness of lattice-based cryptographic techniques with the threshold-based security model of Shamir's Secret Sharing Scheme (SSS). The system splits cryptographic keys into many shares, which are distributed among different entities/users and to reonstruct the key a threshold number of shares is required, which ensures that unauthorized access is prevented even if some shares are compromised. Lattice-based cryptography further enhances this scheme by making it resistant to algorithms of Post Quantum Cryptography (PQC), such as Shor's algorithm, which is capable to, otherwise break traditional SSS. By combining these approaches, the system ensures that keys remain secure even in a future where quantum computing becomes a reality.

The thesis outlines the architecture of the proposed DKMS as consisting of the authorization system, the system for encryption/decryption application, dealer for key sharing, key provider, and the integrated enterprise user interface. It is mostly integrated in a hybrid form with some elements at the Local/ On-Premise side and others in cloud environment as it is both secure and elastic. The mTLS (Mutual Transport Layer Security) protocol is used to authenticate the on-premise and cloud components when exchanging information and encrypt when the system performs key transfer to make it less vulnerable to attacks. Furthermore, the LLL reduction developed for the Lenstra-Lenstra-Lovász (LLL) algorithm improves the key-sharing process to be immune to inside attackers and quantum cryptanalysis. In terms of security analysis, the thesis evaluates the performance of the Lattice-Based Shamir Secret Sharing Scheme in comparison to the classical Shamir Secret Sharing Scheme. The system's performance is assessed based on key generation, distribution, and reconstruction times, as well as its ability to withstand quantum adversaries. The results demonstrate that the proposed system enhances security while maintaining efficient performance, making it suitable for deployment in real-world cloud environments.

Lastly, the thesis discusses potential future improvements to the system. One focus is the scalability of the DKMS, as it must handle large numbers of users and keys in cloud environments. The study also explores challenges of real-world deployment, including integration with cloud infrastructure and compliance with security regulations. The thesis concludes that the proposed Quantum-Safe Distributed Key Management System offers a robust solution to the security challenges of cloud computing, providing long-term protection against insider threats and quantum computing. Future research directions include optimizing the system for large-scale cloud deployments and exploring additional post-quantum cryptographic techniques to further enhance key management security.

Chapter 8

CONCLUSION

This section of the thesis presents the key insights and highlights the novel architecture of the proposed Quantum-Safe Key Management System (KMS), which uses the extremely secure Lattice-Based Shamir Secret Sharing Scheme (LBSSS) in particular. The study focused on security flaws in cloud-based key management, particularly those related to insider threats and the increasing danger of quantum attacks. By eliminating the chance for one point of failure ,decentralising key management across on-premise and cloud components improves overall security.

By splitting up cryptographic keys among multiple parties i.e. On-Premise KMS and Cloud-Based KMS, requiring a threshold of shares for key reconstruction and lattice-based cryptography combined with Shamir Secret Sharing enhances robustness against quantum adversaries. Improved security against quantum adversaries and efficiency in large-scale cloud deployments were validated by experimental results. The LLL algorithm fully supports the system in ensuring secure communication and key distribution via the mTLS protocol.

In order to solve present and future cryptographic issues, the research presents a next-generation, quantum-safe DKMS that provides a scalable, secure solution for cloud environments. This study establishes a solid basis for future research on distributed key management systems and quantum-safe cryptography.

Bibliography

[1] McCarthy, S. (2020). Practical Lattice-Based Cryptography over Structured Lattices, Queen's University Belfast (United Kingdom).

[2] Kao, Y.-W., et al. (2013). "uCloud: a user-centric key management scheme for cloud data protection." IET Information Security 7(2): 144-154.

[3] Varalakshmi, P., et al. (2014). A framework for secure cryptographic key management systems. 2014 Sixth International Conference on Advanced Computing (ICoAC), IEEE.

[4] Lin, C.-T. (2015). A Secret-Sharing-Based Method for Cloud Storage System, NSYSU.

[5] Pundkar, S. N. and N. Shekokar(2016). Cloud computing security in multi-clouds using Shamir's secret sharing scheme. 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), IEEE.

[6] Naik R, M. and S. Sathyanarayana (2017). "Key management infrastructure in cloud computing environment-a survey." ACCENTS Transactions on Information Security 2: 52-61.

[7] Schiefer, G., et al. (2017). Security in a distributed key management approach. 2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS), IEEE.

[8] Huang, X. and R. Chen (2018). A survey of key management service in cloud. 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), IEEE.

[9] Bentajer, A.,et al. (2021). Secure Cloud Key Management based on Robust Secret Sharing. CS IT Conference Proceedings, CS IT Conference Proceedings.

[10] Abdulsalam, Y. S. and M. Hedabou (2022). "Security and Privacy in Cloud Computing: Technical Review." Future Internet 14(1): 11.

[11] Attasena, V., et al. (2017). "Secret sharing for cloud data security: a survey." The VLDB Journal 26(5): 657-681.

[12] Celiktas, B., et al. (2021). "A higher-level security scheme for key access on cloud computing." IEEE Access 9: 107347-107359.

[13] Chhabra, S. and A. Kumar Singh (2020). "Security enhancement in cloud environment using secure secret key sharing." Journal of Communications Software and Systems 16(4): 296-307.

[14] Evertsson, D. (2013). Cloud computing from a privacy perspective. UMNAD. Independent thesis Basic level (degree of Bachelor).

[15] Harn, L. and C. Lin (2010). "Authenticated group key transfer protocol based on secret sharing." IEEE transactions on computers 59(6): 842-846.

[16] Jeya, J. J., et al. (2023). An Efficient Algorithm for Secure Key Management in Cloud Environment. 2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), IEEE.

[17] Mishra, A. (2014). Data security in cloud computing based on advanced secret sharing key management scheme.

[18] Naik R, M. and S. Sathyanarayana (2017). "Key management infrastructure in cloud computing environment-a survey." ACCENTS Transactions on Information Security 2: 52-61.

[19] Pilaram, H., et al. (2021). "An efficient lattice-based threshold signature scheme using multi-stage secret sharing." IET Information Security 15(1): 98-106.

[20] Li, W., et al. (2019). "Design of secure authenticated key management protocol for cloud computing environments." IEEE Transactions on Dependable and Secure Computing 18(3): 1276-1290.

[21] Pradeep, K. V., et al. (2019). "An Efficient Framework for Sharing a File in a Secure Manner Using Asymmetric Key Distribution Management in Cloud Environment." Journal of Computer Networks and Communications 2019: 9852472.

[22] Huang, J.-Y., et al. (2011). Efficient identity-based key management for configurable hierarchical cloud computing environment. 2011 IEEE 17th International Conference on Parallel and Distributed Systems, IEEE.

[23] Fakhar, F. and M. A. Shibli (2013). Management of symmetric cryptographic keys in cloud based environment. 2013 15th International Conference on Advanced Communications Technology (ICACT), IEEE.

[24] Song, N. and Y. Chen (2014). "Novel hyper-combined public key based cloud storage key management scheme." China Communications 11(14): 185-194.

[25] Chhabra, S. and A. Kumar Singh (2020). "Security enhancement in cloud environment using secure secret key sharing." Journal of Communications Software and Systems 16(4): 296-307.

[26] Nikhade, M. and T. Hiwarkar (2022). "A Study on Key Management Infrastructure in Cloud Computing Environmental Survey for Secured Communication." International Journal of Innovations in Engineering and Science 7: 12-19.

[27] Nikhade, G. and T. Hiwarkar (2022). "A Study on Techniques for Access Control and Key Management in the Cloud for Secured Communication." International Journal of Innovations in Engineering and Science 7: 1-3.

[28] Karanam, P. and V. Varadarajan (2015). "Survey on the Key Management for Securing the Cloud." Procedia Computer Science 50.

[29] Dr, R. P., et al. (2023). "A Higher-Level Security Scheme For Key Access Management On Cloud Computing." International Journal Of Scientific Research In Engineering and Management.

[30] Fatima, S. and S. Ahmad (2020). "Secure and effective key management using secret sharing schemes in cloud computing." International Journal of e-Collaboration (IJeC) 16(1): 1-15.

[31] Gupta, V. and S. Bedekar "Alternative to Shamir's secret sharing scheme Lagrange interpolation over finite field." Int. J. Tech. Res. Sci.

[32] Han, J., et al. (2021). A decentralized document management system using blockchain and secret sharing. Proceedings of the 36th Annual ACM Symposium on Applied Computing.

[33] Jäger, H., et al. (2013). "A novel set of measures against insider attacks–sealed cloud." Open Identity Summit 2013.

[34] Kannan, N. R. and N. Salian (2016). "Cloud Computing Security in Multi-Clouds using Shamir's Secret Sharing Scheme." International Journal of Computer Applications 975: 8887.

[35] Ravi, P., et al. (2021). "Lattice-based key-sharing schemes: A survey." ACM Computing Surveys (CSUR) 54(1): 1-39.

[36] Schiefer, G., et al. (2017). Security in a distributed key management approach. 2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS), IEEE.

[37] Yoo, S., et al. (2013). User-Centric Key Management Scheme for Personal Cloud Storage. 2013 International Conference on Information Science and Applications (ICISA)..

[38] Gupta, V. and S. Bedekar "Alternative to Shamir's secret sharing scheme Lagrange interpolation over finite field." Int. J. Tech. Res. Sci.

[39] Han, J., et al. (2021). A decentralized document management system using blockchain and secret sharing. Proceedings of the 36th Annual ACM Symposium on Applied Computing.

[40] Pilaram, H. and T. Eghlidos (2015). "An efficient lattice based multi-stage secret sharing scheme." IEEE Transactions on Dependable and Secure Computing 14(1): 2-8.

[41] Jiang, J., et al. (2023). "QPause: Quantum-Resistant Password-Protected Data Outsourcing for Cloud Storage." IEEE Transactions on Services Computing.

[42] Shamir, A. (1979). "How to share a secret." Communications of the ACM 22(11): 612-613.

[43] Chandramouli, R., et al. (2013). "Cryptographic key management issues and challenges in cloud services." Secure Cloud Computing: 1-30.

[44] https://www.verizon.com/business/resources/T3dc/reports/2024-dbir-data-breach-investigations-report.

[45] https://go1.gurucul.com/2023-Insider-Threat-Report.

[46] https://www.cybersecurity-insiders.com/2024-insider-threat-report-trends-challenges-and-solutions.

[47] https://www.isc2.org/-/media/Project/ISC2/Main/Media/Marketing-Assets/CCSP/2023-Cloud-Security-Report

[48] Chen, J., Deng, H., Su, H., Yuan, M., Ren, Y. (2024). Lattice-Based Threshold Secret Sharing Scheme and Its Applications: A Survey. Electronics, 13(2), 287.

[49] Khalid, A.; McCarthy, S.; O'Neill, M.; Liu, W. Lattice-based cryptography for IoT in a quantum world: Are we ready? In Proceedings of the 2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI), Otranto, Italy,13–14 June 2019; pp. 194–199.

[50] El Bansarkhani, R., Meziani, M., 2012. An Efficient Lattice-Based Secret Sharing Construction, in: Lecture Notes in Computer Science. Lecture Notes in Computer Science, pp. 160–168.. https://doi.org/10.1007/978-3-642-30955-7-14

[51] https://www.datauniverseevent.com/en-us/blog/general/AI and the Global Datasphere, how much information will humanity have by 2025.

[52] https://www.readkong.com/page/the-digitization-of-the-world-from-edge-to-core-8666239

[53] https://quantumalgorithmzoo.org/

[54] https://blog.cloudflare.com/nists-first-post-quantum-standards

[55] https://blog.cloudflare.com/pq-2024

[56] https://PWC-Global-Digital-Trust-Insights-Report-2024.pdf

[57] Čuřík, P., Ploszek, R., Zajac, P. (2022). Practical use of secret sharing for enhancing privacy in clouds. Electronics, 11(17), 2758.

[58] Kandias, M., Virvilis, N., Gritzalis, D. (2013). The insider threat in cloud computing. Critical Information Infrastructure Security: 6th International Workshop, CRITIS 2011, Lucerne, Switzerland, September 8-9, 2011.

Appendix A

APPENDIX: CODE IMPLEMENTATION OF LATTICE-BASED SHAMIR SECRET SHARING SCHEME(LBSSS)

This section is a write-up for the Python and SageMath implementation of the Lattice-Based Shamir Secret Sharing Scheme (LBSSS).

A.1 Overview

The LBSSS secret sharing scheme shares a single secret among several participants, dividing the secret between multiple shares. The participants must combine these so they can fully reconstruct it. A quorum/threshold of a pre-set number of shares is required, and only those shares can recreate the secret. This scheme follows these steps:

1. Generate a Lattice Basis and Participant Vectors.
2. Shares are distributed according to the generated lattice basis.
3. Find the Modular Inverse:** Reassemble the secret from the combined shares.
4. Performance Analysis:** Conduct memory profiling and time measurement.

The implementation contains (1) Secret Generation and Share Distribution, and (2) Secret Reconstruction algorithm.

A.2 Explanation of Code Components

- Lattice Basis Generation: The B matrix is created as a n x n matrix, with randomness added by computing each element using a modulus operation.
- Lambda Vectors: A distinct lambda vector produced in a manner akin to the lattice basis is given to every participant.
- Modular Inverse: To make secret reconstruction easier, the modular_ inverse() function calculates the lambda matrix's inverse.
- Generating Secrets and Shares: The participants' secret and shares are calculated using the generate_ secret_ and_ shares() function.
- Secret Reconstruction: The secret is reconstructed using the lambda matrix inverse by the reconstruct_ secret_ exact() method.

A.3 Pseudo-Code

Algorithm: LBSSS Implementation Input: Lattice basis B, secret vector v, prime modulus q, lambda vectors Output: Secret S and reconstructed secret S'

```
1. Generate lattice basis B (n x n matrix).
2. Generate lambda vectors for participants.
3. Compute secret: S = (B * v) mod q
4. Compute shares: $C_i$ = (B * $lambda_i$) mod q for all i.
5. To reconstruct the secret:
    • Compute d vectors: $d_i$= (E * $C_i$) mod q.
    • Compute the modular inverse of lambda matrix.
    • Reconstruct lattice basis B' = ($lambda - matrix^{-1} * D$) mod q.
    • Compute reconstructed secret: S' = (B'* v) mod q.
6. Compare S and S' to verify the reconstruction.
```

A.4 Annotated Code Implementation of LBSSS

Below is the code implementation of the LBSSS scheme. The code is written in Python using the SageMath library for matrix operations.

```python
from sage.all import Matrix, vector, ZZ, gcd, inverse_mod

# Fixed values based on the matching configuration
B = Matrix(ZZ, [[4, 10, 4], [10, 4, 9], [8, 3, 7]])
lambdas = [vector(ZZ, [9, 9, 3]), vector(ZZ, [2, 3, 2]), vector(ZZ, [1, 6, 9])]
v = vector(ZZ, [3, 4, 4])
E = Matrix(ZZ, [[9, 10, 10], [5, 10, 11], [5, 3, 6]])
q = 11  # Modulus

# Precomputed shares
shares = [(B * lambdas[i]) % q for i in range(3)]
print(f"Lattice_Basis_(B):\n{B}")
print(f"Lambda_Vectors:_{lambdas}")
print(f"Vector_(v):_{v}")
print(f"Public_Matrix_(E):\n{E}")
print(f"Shares_(Ci):_{shares}")

# Original secret
S = (B * v) % q
print(f"Original_Secret(S=B*v_mod{q}):{S}")

# Function to compute modular inverse
def modular_inverse(matrix, q):
    print(f"Computing_modular_inverse_of_matrix:\n{matrix}")
    matrix_sage = Matrix(ZZ, matrix)
    det = matrix_sage.det() % q
    print(f"Determinant_(mod{q}):{det}")

    if gcd(det, q) != 1:
        raise ValueError(f"Matrix_not_invertible_mod{q}(det={det})")

    det_inv = inverse_mod(det, q)
    print(f"Inverse_of_determinant_(mod{q}):{det_inv}")

    adjugate = matrix_sage.adjugate() % q
    print(f"Adjugate_of_matrix_(mod{q}):\n{adjugate}")

    modular_inv = (det_inv * adjugate) % q
    print(f"Modular_Inverse_(mod{q}):\n{modular_inv}")

    return modular_inv

# Secret reconstruction
```

```
def reconstruct_secret_exact(shares, E, v, q):
    print("Starting_reconstruction...")

    # Step 1: Compute d_vectors = E * Ci mod q
    d_vectors = [(E * shares[i]) % q for i in range(3)]
    print(f"d_vectors(E*Ci_mod{q}):{d_vectors}")

    D = Matrix(ZZ, d_vectors) % q
    print(f"D_Matrix(mod{q}):\n{D}")

    # Step 2: Create Lambda matrix from lambda vectors
    lambda_matrix = Matrix(ZZ, lambdas) % q
    print(f"Lambda Matrix(mod_{q}):\n{lambda_matrix}")

    # Step 3: Compute the inverse of the lambda matrix
    B_inverse = modular_inverse(lambda_matrix, q)

    # Step 4: Compute reconstructed B = B_inverse * D mod q
    B_reconstructed = (B_inverse * D) % q
    print(f"Reconstructed_Lattice_Basis(B')(mod{q}):\n{B_reconstructed}")

    # Step 5: Compute the reconstructed secret S' = B' * v mod q
    S_reconstructed = (B_reconstructed * v) % q
    print(f"Reconstructed_Secret(S'=B'*v_mod{q}):{S_reconstructed}")

    return S_reconstructed

# Reconstruct the secret
reconstructed_secret = reconstruct_secret_exact(shares, E, v, q)

# Print final comparison
print(f"\nFinal_Reconstructed_Secret:{reconstructed_secret}")
print(f"Original_Secret:{S}")
```

A.5 Code Snippet for Performance Analysis

```
import time  # For time measurement
from memory_profiler import memory_usage  # For memory profiling
import matplotlib.pyplot as plt  # For plotting graphs

# Performance analysis for secret/share generation
def profile_generation(generate_secret_and_shares):
    start_time = time.time()  # Start time measurement
    mem_gen = memory_usage((generate_secret_and_shares,), interval=0.01)  # Profile memory usag
    shares, S = generate_secret_and_shares()  # Execute the function
```

```python
    time_gen = time.time() - start_time  # End time measurement

    print(f"Secret and Share Generation:")
    print(f"Secret: {S}")
    print(f"Time: {time_gen:.6f} seconds | Memory: {max(mem_gen) - min(mem_gen):.2f} MiB")

    return shares, S, time_gen, max(mem_gen) - min(mem_gen)

# Performance analysis for secret reconstruction
def profile_reconstruction(reconstruct_secret_exact, shares):
    start_time = time.time()  # Start time measurement
    mem_recon = memory_usage((reconstruct_secret_exact, (shares,)), interval=0.01)  # Profile m
    reconstructed_secret = reconstruct_secret_exact(shares)  # Execute the function
    time_recon = time.time() - start_time  # End time measurement

    print(f"\nSecret Reconstruction:")
    print(f"Reconstructed Secret: {reconstructed_secret}")
    print(f"Time: {time_recon:.6f} seconds | Memory: {max(mem_recon) - min(mem_recon):.2f} MiB"

    return time_recon, max(mem_recon) - min(mem_recon)

# Plotting the results
def plot_analysis(time_gen, mem_gen, time_recon, mem_recon):
    labels = ['Secret/Share Generation', 'Secret Reconstruction']
    times = [time_gen, time_recon]
    memories = [mem_gen, mem_recon]

    # Plotting time analysis
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.bar(labels, times, color=['blue', 'green'])
    plt.xlabel('Process')
    plt.ylabel('Time (seconds)')
    plt.title('Time Analysis')

    # Plotting memory analysis
    plt.subplot(1, 2, 2)
    plt.bar(labels, memories, color=['red', 'purple'])
    plt.xlabel('Process')
    plt.ylabel('Memory (MiB)')
    plt.title('Memory Usage Analysis')

    # Show the plots
    plt.tight_layout()
    plt.show()

# Example usage (replace these with actual function calls from your main code)
def generate_secret_and_shares():
    time.sleep(0.01)  # Simulate a delay
    shares = [(i, i + 1, i + 2) for i in range(3)]  # Example shares
    S = (2, 5, 9)  # Example secret
    return shares, S

def reconstruct_secret_exact(shares):
    time.sleep(0.01)  # Simulate a delay
    return (2, 5, 9)  # Example reconstructed secret

# Run the analysis
shares, S, time_gen, mem_gen = profile_generation(generate_secret_and_shares)
time_recon, mem_recon = profile_reconstruction(reconstruct_secret_exact, shares)
plot_analysis(time_gen, mem_gen, time_recon, mem_recon)
```

A.5.1 Libraries Used in Implementation

The performance of the scheme is analyzed by making use of python and SageMath libraries/dependencies such as:

- **memory_ profiler** to monitor and profile memory usage of functions.
- **matplotlib** plots graphs to visualize the results for time and memory consumption.
- **tracemalloc** for advanced memory tracking by tracking memory allocations.
- **sage.all** provides access to the core functionalities of SageMath like matrices, vectors, and modular arithmetic.

A.5.2 Subcomponents Used from sage.all Library

- **Matrix** creates and manipulates matrices over the integer ring (Z).
- **Vector** creates and manipulates vectors over the integer ring (Z).
- **Z** represents the ring of integers in SageMath, ensuring all matrix and vector operations occur over the correct field.
- **gcd** computes the gcd to determine if a matrix is invertible under modular arithmetic.
- **Inverse_ mod** Computes the modular inverse of a determinant modulo q. Essential for modular arithmetic in matrix inversion.