People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Akli Mohand Oulhaj – Bouira -
Faculty of Science and Applied Sciences
Computer science department

Master thesis

Computer science

Speciality: *Computer Systems Engineering*

Theme

Secure key management mechanism in Big Data

Supervised by

- Mr. BOUDJELABA Hakim

Submitted by

- Mr. Belghit Lounes

2023/2024

Contents

Appreciation

I would like to extend my deepest appreciation to all those who have played a crucial role in my journey.

To my incredible family, whose love and support have been the cornerstone of my success:

To my mother, thank you for your endless patience, wisdom, and encouragement. Your belief in me has been a constant source of strength and inspiration.

To my father, thank you for your unwavering confidence in my abilities and your steadfast support. Your hard work and dedication have been a guiding light throughout my life.

To my wonderful friends, thank you for your companionship, laughter, and encouragement. Your friendship has been a source of immense joy and has helped me through both good times and bad.

To my dedicated teachers, thank you for your tireless efforts and commitment to our education. Your knowledge, passion, and guidance have been instrumental in shaping our futures. Your support has not only helped us achieve our academic goals but also inspired us to strive for excellence in all aspects of our lives.

With heartfelt gratitude and sincere appreciation.

Belghit Lounes

Dedications

I dedicate this project to everyone who has contributed to its completion and success.

To my mentors and supervisors, your guidance and wisdom have been invaluable. Thank you for your patience, your insightful feedback, and your unwavering support throughout this journey.

To my colleagues and peers who collaborated and shared their knowledge, your teamwork has made this project a rewarding experience.

To my family and friends, thank you for your constant encouragement and understanding during the many long hours spent working on this project.

This project would not have been possible without each and every one of you.

Belghit Lounes

Abstract

Our project introduces a sophisticated key management solution tailored for Big Data environments, comprising a Double Key Management Center (KMC) architecture with static Load Balancers and a blockchain-based public record system. This innovative approach addresses the complexities of key management by efficiently distributing cryptographic key operations between the Issuer and Verifier KMCs, guided by predefined routing rules set by the static traffic checker. This ensures optimal resource utilization and scalability without the need for real-time adjustments. Meanwhile, the blockchain-based public record system guarantees transparency, auditability, and security by immutably logging all key management transactions. Through a rigorous evaluation process encompassing simulations, experiments, and real-world testing, we validate the solution's effectiveness across performance metrics, scalability, reliability, security, and auditability, ultimately providing a robust and efficient key management infrastructure tailored to the demands of modern Big Data applications.

Résumé

Notre projet introduit une solution sophistiquée de gestion des clés adaptée aux environnements Big Data, comprenant une architecture KMC (Double Key Management Center) avec contrôle du trafic statique (load balacer) et un système d'enregistrement public basé sur la blockchain. Cette approche innovante aborde les complexités de la gestion des clés en répartissant efficacement les opérations de clés cryptographiques entre l'émetteur et le vérificateur, guidés par des règles de routage prédéfinies définies par le vérificateur de trafic statique. Cela garantit une utilisation optimale des ressources et une évolutivité sans besoin d'ajustements en temps réel. Pendant ce temps, le système d'enregistrement public basé sur la blockchain garantit la transparence, l'auditabilité, et la sécurité en enregistrant immuablement toutes les transactions de gestion des clés. Grâce à un processus d'évaluation rigoureux comprenant des simulations, des expériences et des tests en situation réelle, nous validons l'efficacité de la solution à travers les mesures de performance, l'évolutivité, la fiabilité, la sécurité et l'auditabilité, En fin de compte, fournir une infrastructure de gestion des clés robuste et efficace adaptée aux exigences des applications Big Data modernes.

مُلخص

يقدم مشروعنا حلاً إداريًا رئيسيًا متطورًا مصممًا لبيئات البيانات الضخمة، ويتألف من بنية مركز إدارة المفاتيح المزدوجة مع موازنات مرور ثابتة ونظام سجل عام قائم على تقنية سلسلة الكتل. يعالج هذا النهج المبتكر تعقيدات إدارة المفتاح من خلال التوزيع الفعال لعمليات مفتاح التشفير بين مركز إصدار المفاتيح ومركز تحقق المفاتيح، مسترشدًا بقواعد التوجيه المحددة مسبقًا التي وضعها مدقق المرور الثابت. وهذا يكفل الاستخدام الأمثل للموارد وقابليتها للتوسع دون الحاجة إلى إجراء تعديلات في الوقت الحقيقي. وفي الوقت نفسه، يضمن نظام السجلات العامة القائم على تقنية سلسلة الكتل الشفافية وقابلية التدقيق والأمن من خلال تسجيل جميع معاملات الإدارة الرئيسية بشكل ثابت. من خلال عملية تقييم صارمة تشمل المحاكاة والتجارب والاختبارات الواقعية، فإننا نتحقق من فعالية الحل عبر مقاييس الأداء، وقابلية التوسع، والموثوقية، والأمن، وقابلية التدقيق، مما يوفر في النهاية بنية تحتية إدارة رئيسية قوية وفعالة مصممة خصيصًا لمتطلبات تطبيقات البيانات الضخمة الحديثة.

List of Figures

List of Tables

Abbreviations list

KMS	Key Management System
TLS	Transport Layer Security
SSL	Secure Socket Layer
AES	Advanced Encryption Standard
DES	Data Encryption Standard
3DES	Triple DES
RSA	Rivest–Shamir–Adleman
ECC	Elliptic Curve Cryptography
CRL	Certificate Revocation List
DSA	Digital Signature Algorithm
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
PSK	Pre-Shared Keys
EAP	Extensible Authentication Protocol
APs	Access points
KDC	Key Distribution Centers
TGT	Ticket Granting Ticket
TGS	Ticket Granting Server
SSO	Single Sign-On
PKI	Public Key Infrastructure
CA	Certificate Authority

RA	Registration Authority
OCSP	Online Certificate Status Protocol
EAP	Extensible Authentication Protocol
KDF	Key Derivation Functions
CRLs	Certificate Revocation Lists
OCSP	Online Certificate Status Protocol
ACLs	Access Control Lists
RBAC	Role-based Access Control
TPMs	Trusted Platform Modules
HSMs	Hardware Security Modules
TPMs	Trusted Platform Modules
HKM	Hierarchical Key Management
KMIP	Key Management Interoperability Protocol
CKM	Centralized Key Management
DKMS	Distributed Key Management Systems
HSMs	Hardware Security Modules
RBAC	Role-Based Access Control
BYOK	Bring Your Own Key
AWS	Amazone Web Services
DDoS	Distributed Denial of Service
KMC	Key Management Center
SDN	Software-defined Networking
CDNs	Content Delivery Networks
AS	Authentication Server
TEE	Trusted Execution Environment
MACs	Message Authentication Codes

MU	max users
MUBF	max users before failure
ARTBF	average response time before failure
ARTAF	average response time after failure
95TH	95th percentile time
RECOVER	Does the system recover
NF	no failure
RPS	request per second

General introduction

In the era of big data, the management and security of cryptographic keys have become critical concerns for organizations handling vast amounts of sensitive information. As data volumes grow exponentially, ensuring the confidentiality, integrity, and availability of data through effective key management practices is paramount. Cryptographic keys are essential for securing data at rest and in transit, making their management a cornerstone of any comprehensive data security strategy.

Key management encompasses the processes of generating, distributing, storing, and rotating cryptographic keys, as well as revoking and destroying them when no longer needed. Traditional key management systems often struggle to cope with the sheer scale and complexity of modern data environments. These systems need to handle high transaction volumes, provide robust access controls, and ensure continuous availability and reliability. Additionally, they must integrate seamlessly with various data storage and processing platforms, including cloud services and distributed systems.

Big Data environments, characterized by their volume, velocity, and variety, present unique challenges for key management. The scale of data necessitates efficient and scalable key management solutions that can handle high throughput and low latency requirements. Furthermore, the distributed nature of Big Data platforms requires a key management infrastructure that can operate reliably across multiple nodes and geographic locations. Ensuring data security in such environments demands innovative approaches to key management that can address these challenges effectively.

Security is another crucial aspect of key management in big data. With the increasing sophistication of cyber threats, it is essential to protect cryptographic keys from unauthorized access and misuse. This involves implementing strong authentication and authorization mechanisms, as well as ensuring the integrity and auditability of key management operations. The use of advanced technologies, such as blockchain, can enhance the security and transparency of key management processes by providing immutable and verifiable records of all key-related activities.

Moreover, regulatory compliance is a significant driver for effective key management. Organizations must adhere to various data protection regulations and standards, such as GDPR, HIPAA, and PCI DSS, which mandate strict controls over cryptographic keys and their management. Failure to comply with these regulations can result in severe penalties and damage to an organization's reputation. Therefore, a robust key management solution must also facilitate compliance by providing comprehensive audit trails and supporting regulatory reporting requirements.

The management of cryptographic keys in Big Data environments is a complex yet vital task that underpins the security and integrity of data. As data volumes continue to grow and cyber threats evolve, developing scalable, efficient, and secure key management solutions is essential for safeguarding sensitive information and ensuring compliance with regulatory standards. This project aims to address these challenges by exploring innovative approaches to key management that are tailored to the demands of Big Data environments.

Chapter 1

Key management

1.1 Introduction

In the digital era, safeguarding sensitive information is paramount due to prevalent cyber threats. Key management is crucial, encompassing the generation, distribution, use, storage, and disposal of encryption keys. These keys are vital for secure communication, data protection, and access control across various systems. However, challenges such as ensuring key uniqueness, preventing theft, and managing keys in diverse environments like cloud and IoT persist. Effective key management requires a thorough understanding of principles and the application of suitable technologies and policies. This chapter explores key management components, from generation to storage, guided by industry standards, best practices, and practical examples.

1.2 Fundamentals of Cryptographic Keys

1.2.1 What is encryption?

Encryption is a way of scrambling data so that only authorized parties can understand the information. In technical terms, it is the process of converting human-readable plain text to incomprehensible text, also known as cipher text. In simpler terms, encryption takes readable data and alters it so that it appears random. Encryption requires the use of a cryptographic key: a set of mathematical values that both the sender and the recipient of an encrypted message agree on[19] ,as shown in Figure1.1.

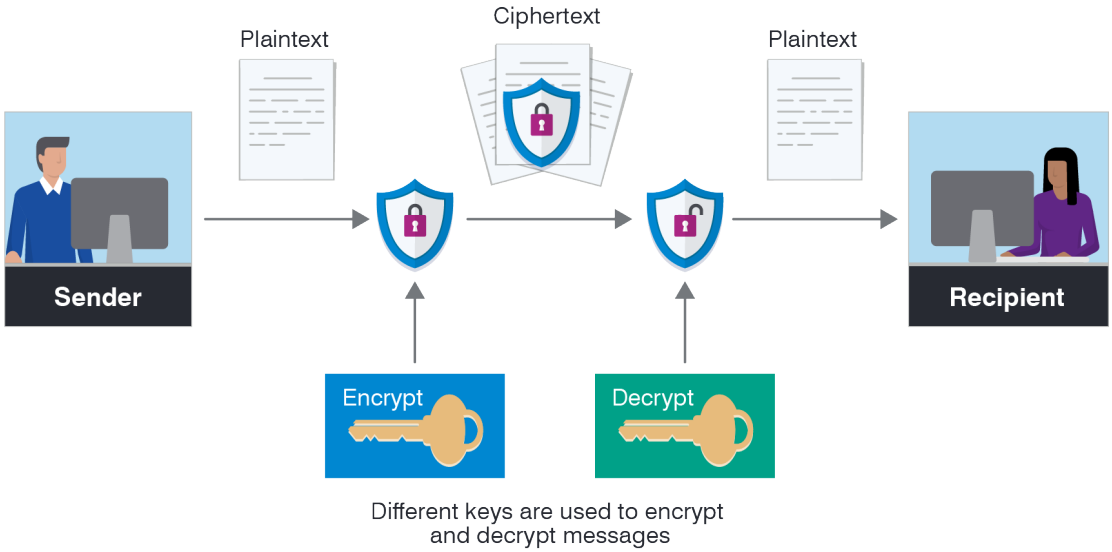


Figure 1.1: Encryption and Decryption method [1]

1.2.2 Cryptographic Keys

Cryptographic keys are an essential component of cryptography, serving as mathematical parameters used in cryptographic algorithms to encrypt, decrypt, authenticate, or digitally sign data. These keys are strings of binary digits, typically represented in a readable format like hexadecimal or base64 [19].

1.2.3 Digital Certificate

A digital certificate is an electronic document used to prove the ownership of a public key. It is issued by a trusted entity known as a Certificate Authority (CA). The certificate contains information about the key, the identity of its owner (individual, organization, or device), and the digital signature of the CA that verifies the certificate's contents. Digital certificates are essential for establishing secure communications over the internet, such as in HTTPS, where they enable encrypted connections and authenticate the identity of websites to prevent fraud and ensure data integrity and confidentiality. They play a crucial role in public key infrastructure, facilitating secure online transactions and communications [20].

1.2.4 Types of Cryptographic Keys

There are two main types of encryption: symmetric and asymmetric. Asymmetric encryption is also referred to as public key encryption.

In symmetric encryption, there exists a singular key, and all parties involved in communication utilize the identical (secret) key for both encryption and decryption. Asymmetric encryption uses two keys, one of which is used for encryption and the other for decryption. The decryption key is kept private, while the encryption key is shared publicly for anyone to use. Asymmetric encryption is a foundational technology for TLS (also known as SSL) [21].

symmetric encryption

Symmetric encryption employs a single key for both encryption and decryption, making it fast and efficient for handling large data sets. The key's symmetric nature means its holder can encrypt and decrypt data. However, securely distributing the key poses a challenge, as compromising it risks unauthorized access to encrypted data. Common symmetric algorithms like AES[22], DES, and 3DES[23], widely used in various applications, ensure data confidentiality. Symmetric encryption, unlike asymmetric methods, offers speed advantages but demands careful key management. Nonetheless, it remains a cornerstone of secure data transmission, balancing efficiency with the need for robust protection against unauthorized access [24].

asymmetric encryption

Asymmetric encryption employs pairs of keys: a public key, shared openly, and a private key, kept secret. Encrypted data with the public key can only be decrypted with the private key, and vice versa, ensuring secure communication. It's vital for SSL/TLS, digital signatures, and key exchange. Popular algorithms include RSA[25], Diffie-Hellman[26], and Elliptic Curve Cryptography [27]. This system enables safe transmission over insecure channels: messages encrypted with the public key can only be decoded by the intended receiver possessing the private key. Asymmetric encryption safeguards sensitive data in various applications, maintaining confidentiality and integrity in digital communications [28].

1.2.5 Key Length and Strength

The key length, or key size, denotes the bits in a cryptographic key for encryption, crucial for safeguarding sensitive data. However, security isn't solely about length, it significantly impacts encryption strength [29]. Brute force attacks pose a serious threat, where adversaries systematically try all possible keys to decrypt data. Consider the Caesar cipher: with only 26 possible keys, it's vulnerable. Modern ciphers employ longer keys, like 128 bits, resulting in an astronomical number of possible keys, making brute force attacks impractical. For instance, a 128-bit key yields $2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$, rendering brute force attacks computationally infeasible even for the most powerful computers [30].

key lengths in the context of symmetric and asymmetric algorithms

1. **Symmetric Cryptography:** Typically, common key lengths such as 128, 192, and 256 bits are utilized. Among these, a 128-bit key is widely regarded as secure for most applications, balancing security and computational efficiency. However, in cases demanding heightened security, such as military or government systems, key lengths of 256 bits or even higher are employed to ensure maximum protection against cryptographic attacks [31].

2. **Asymmetric Cryptography:** Asymmetric algorithms utilize pairs of keys: a public key for encryption and a corresponding private key for decryption. The security of these algorithms relies on mathematical properties such as factorization or discrete logarithms. Common key lengths vary depending on the algorithm used. For RSA, key lengths of 2048 bits are recommended for keys used until 2030, while 3000 bits are suggested for keys used beyond 2022. For Elliptic Curve Cryptography (ECC), roughly half the key length provides effective security [31].

1.2.6 Key Revocation and Renewal

Key revocation: Key revocation is crucial for invalidating compromised, expired, or unnecessary public key certificates or symmetric encryption keys, preventing unauthorized access and maintaining data security. Compromised keys pose serious risks, enabling attackers to intercept communications, impersonate users, or access sensitive data. Expired

keys can disrupt communication or data access, while unnecessary keys clutter the system, increasing the risk of misuse. Various methods ensure key revocation: Certificate Revocation Lists (CRLs) are commonly used for public key certificates, listing revoked certificates for verification. Online Certificate Status Protocol (OCSP) allows direct queries to Certificate Authorities (CAs) about certificate status. Symmetric encryption keys are managed through Key Management Systems (KMS), enabling deletion or marking as invalid. These practices ensure only authorized and current keys are utilized, enhancing overall security [32].

Key renewal: This is an extension of the validity period of an expiring certificate. It safeguards against interruptions in secure communications and services. The renewal process generates a new key and distributes it to replace the revoked one. Key renewal is essential for ensuring the security and integrity of Public Key Infrastructure (PKI) and encryption systems [33].

- For public key certificates, the common method is to use a certificate renewal process. Certificate holders can request a new certificate from the CA before the old one expires.
- Renewal prevents disruptions in secure communications and services by ensuring that certificates remain valid.
- The process involves generating a new certificate with an extended validity period.

1.3 Key Generation

1.3.1 Symmetric Key Generation

Symmetric key encryption relies on a singular secret key that is utilized for both encryption and decryption.

Symmetric keys are typically generated using a cryptographicly secure random number generator. The length of the key is critical to security. Common key lengths comprise of 128 bits (16 bytes) or 256 bits (32 bytes) Longer keys provide greater security, but they may have an impact on performance. Using OpenSSL, we can generate a random symmetric key using the following command:

OpenSSL rand 128 > symkeyfile.key

This creates a 128-bit key and stores it in the symkeyfile.key file. On Unix-like systems, we can also use **/dev/random** to generate random bytes and create a symmetric key.

The process of key exchange becomes imperative once the symmetric key has been generated. Methods include secure channels, pre-shared keys, and key derivation from a passphrase.

1.3.2 Asymmetric Key Pair Generation

Asymmetric key encryption involves a pair of keys: a public key for encryption and a private key for decryption. Here is how the key pairs are generated in some algorithms :

1. RSA (Rivest–Shamir–Adleman) [34]:
   (a) Choose Two Prime Numbers:
      - Select two large prime numbers, denoted as p and q.
      - These primes serve as the foundation for the RSA key pair.
   (b) Calculate the Modulus (n):
      - Compute the product of p and q: $(n = p \cdot q)$.
      - The modulus (n) is used in both the public and private keys.
   (c) Calculate Euler's Totient Function ($\phi(n)$):
      - Calculate $(\phi(n) = (p-1) \cdot (q-1))$.
      - Euler's totient function represents the count of positive integers less than n that are coprime (have no common factors) with n.
   (d) Choose the Public Exponent (e):
      - Select a small positive integer e such that:
        - $(1 < e < \phi(n))$.
        - e is coprime with $\phi(n)$ (i.e., $(\gcd(e, \phi(n)) = 1)$).
      - The public key consists of (e, n).
   (e) Calculate the Private Exponent (d):
      - Find an integer d such that:

- $(d \cdot e \equiv 1 \pmod{\phi(n)})$.
- In other words, d is the modular multiplicative inverse of e modulo $\phi(n)$.

- The private key consists of (d, n).

(f) Key Pair Creation:

- The public key is (e, n), where e is the chosen exponent and n is the modulus.
- The private key is (d, n), where d is the calculated exponent and n remains the same.

2. ECC (Elliptic Curve Cryptography) [27]:

(a) Components of ECC:

- Elliptic Curves: These curves are defined by an equation in the form of a Diophantine equation.
- Finite Fields: ECC operates over finite fields (modulo arithmetic).
- Base Point (G): A fixed point on the curve used for key generation.
- Private Key (d): A random integer.
- Public Key (Q): Calculated as $(Q = d \cdot G)$.

(b) Key Generation:

i. Generate a private key (d) randomly.

ii. Compute the public key (Q) using $(Q = d \cdot G)$.

(c) Encryption and Decryption: **To encrypt a message:**

i. Represent the message as a point on the curve (usually using a hash function).

ii. Multiply the point by the recipient's public key.

To decrypt:

i. Multiply the received point by the recipient's private key.

3. DSA (Digital Signature Algorithm) [35]:

(a) Select Prime Numbers:

- Choose a large prime number, denoted as p (typically with 1024 or 2048 bits).
- Select another prime number, q, such that q divides $p - 1$.

(b) Generate Private Key:

- Pick a random number, k, where $1 \leq k \leq q - 1$.
- Compute r as: $r = (g^k mod\ p) mod\ q$ where g is a generator (a fixed value).

(c) Compute Public Key:

- Calculate y (the public key) as: $y = g^x mod\ p$ where x is the private key.

(d) Key Pair:

- The private key consists of x.
- The public key consists of p, q, g, and y.

1.4 Key Distribution Mechanisms

1.4.1 Symmetric Key Distribution

Symmetric key cryptography involves the use of a single key for both the encryption and decryption of data. Here's a rundown of common methods for symmetric key distribution:

Pre-shared Keys

Keys are distributed manually or through a secure channel before communication begins. This method is common in scenarios where the number of communicating parties is limited and known in advance [36].

1. Characteristics of Pre-Shared Keys:
   - The secret or key format can vary. It might be a password, a passphrase, or a hexadecimal string.
   - All cryptographic systems employ pre-shared keys to safeguard communication flow.
   - These keys are crucial for ensuring confidentiality in crypto systems.
2. Security Considerations:

- While sufficiently long and randomly chosen pre-shared keys can resist practical brute-force attacks, they can still be compromised if one end is breached.
- Choosing strong keys is essential. Avoid patterns and use random key choices to make brute-force attacks as difficult as possible.
- Avoid using non-cryptographically secure pseudo-random number generators for key generation.

Key Distribution Centers (KDC)

A KDC is a centralized service that manages and distributes cryptographic keys for secure communication between entities (such as users or services) within a network. It ensures that users and services can securely authenticate themselves and establish secure sessions without directly exchanging sensitive information like passwords, as shown in Figure 1.2 [37].

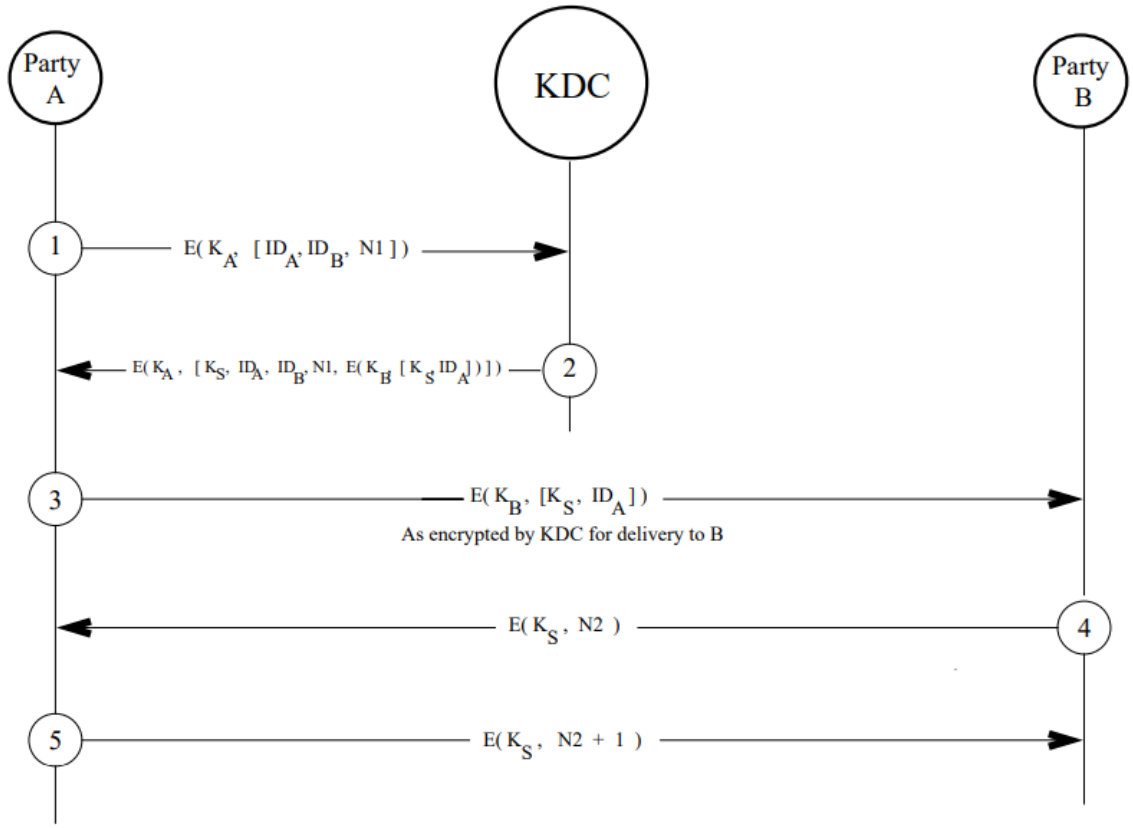


Figure 1.2: A pictorial depiction of the Needham-Schroder protocol[2]

1. Components of a KDC:

   - Authentication Server (AS): The AS is the initial point of contact for a user or service seeking authentication. It verifies the user's identity and issues a ticket-granting ticket (TGT).
   - Ticket Granting Server (TGS): The TGS is responsible for granting service tickets to users or services. It verifies the TGT and issues service tickets for specific services.
   - Database: The KDC maintains a database containing user and service information, including their secret keys.

2. How Kerberos Works with a KDC:

   (a) A user logs in and requests authentication from the AS.

   (b) The AS verifies the user's credentials and issues a TGT.

   (c) The user presents the TGT to the TGS when requesting access to a specific service.

   (d) The TGS validates the TGT and issues a service ticket.

   (e) The user presents the service ticket to the desired service, which grants access.

3. Benefits of KDC and Kerberos:

   - Single Sign-On (SSO): Users authenticate once and can access multiple services without re-entering credentials.
   - Strong Authentication: Kerberos uses symmetric encryption and avoids transmitting plain text passwords.
   - Mutual Authentication: Both the client and server authenticate each other.

Public Key Infrastructure (PKI)

Public key infrastructure encompasses everything used to establish and manage public key encryption. It involves software, hardware, policies, and procedures for the creation, distribution, management, storage, and revoking of digital certificates [38].

1. Digital Certificates:

- A digital certificate cryptographically links a public key with the device or user who owns it.
- It helps authenticate users and devices and ensures secure digital communications.
- A certificate authority (CA) is a trustworthy source that issues digital certificates.
- Think of them as digital passports that verify the sender's identity.

2. Components of PKI:
   - Certificate Authority (CA):
     - The CA issues, stores, and signs digital certificates.
     - It signs the certificate with its private key and publishes the public key for verification.
   - Registration Authority (RA):
     - The RA verifies the identity of users or devices requesting digital certificates.
     - It can be a third party or the CA itself.
   - Certificate Database:
     - Stores digital certificates and metadata (e.g., validity period).
   - Central Directory:
     - Securely indexes and stores cryptographic keys.
   - Certificate Management System:
     - Manages certificate delivery and access.
3. Use Cases:
   - Web Security:
     - PKI secures and authenticates traffic between web browsers and servers.
     - It ensures privacy and verifies sender identity.
   - Internal Communications:

- PKI can secure messages within an organization.
- Ensures confidentiality and integrity.

Diffie-Hellman Key Exchange

The Diffie–Hellman key exchange is a mathematical method that allows two parties to securely establish a shared secret key over an insecure channel without directly transmitting the key themselves [26].

- **Origin and Pioneers:** The Diffie–Hellman key exchange was conceived by Ralph Merkle and named after Whitfield Diffie and Martin Hellman. Published in 1976, it was one of the earliest practical examples of public-key exchange within cryptography.Prior to this, secure communication required exchanging keys through physical means, like paper lists transported by trusted couriers.

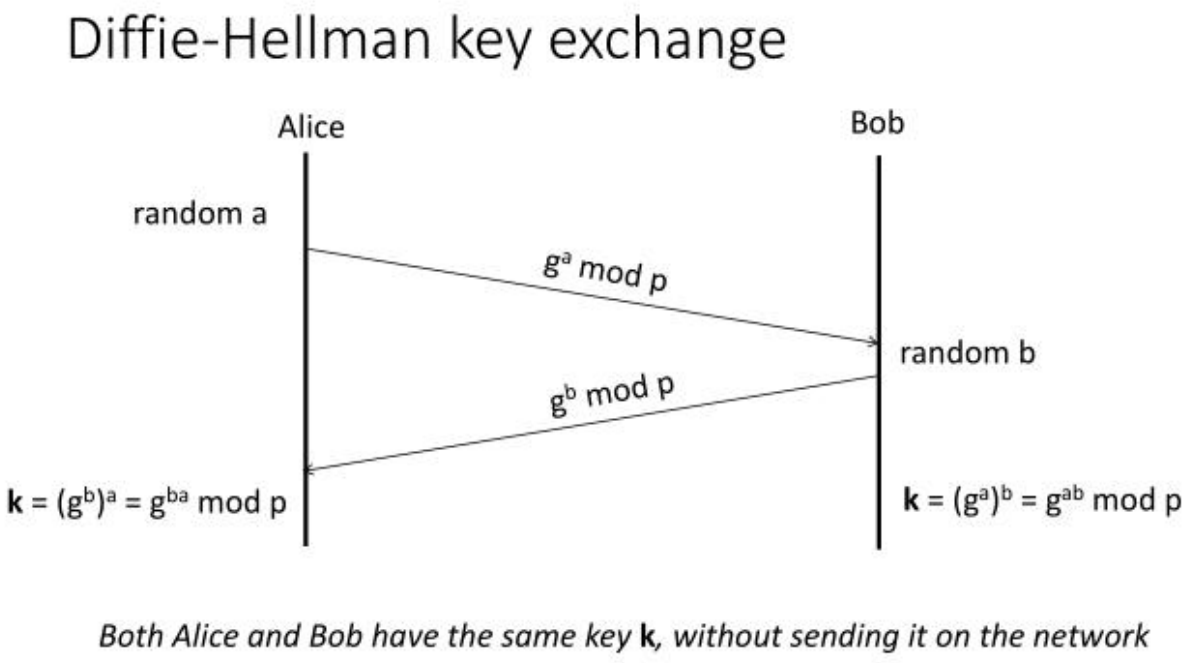


Figure 1.3: Diffie-Hellman Key Exchange [3]

- **How It Works:** Here is how Diffie-Hellman operates, as shown in Figure 1.3:

  1. **Setup:**

- Both parties agree on publicly available parameters: a large prime number p and a base g.

2. **Private Keys:**
   - Each party chooses a secret private key. Let's call these private keys a and b for the two parties.
3. **Public Keys:**
   - Using the agreed-upon parameters and their private keys, each party calculates their public key:
     * Party A computes $A = g^a \ mod \ p$.
     * Party B computes $B = g^b \ mod \ p$.
4. **Exchange Public Keys:**
   - Both parties exchange their calculated public keys A and B over the public channel.
5. **Shared Secret:**
   - Finally, both parties independently compute the shared secret key using their own private key and the other party's public key:
     * Party A computes $S = B^a \ mod \ p$.
     * Party B computes $S = A^b \ mod \ p$.

- **Applications:** Diffie–Hellman is used to secure various Internet services. It provides the basis for authenticated protocols and offers forward secrecy in Transport Layer Security (TLS) ephemeral modes. Security Considerations: Research suggests that some parameters used in DH Internet applications may not be strong enough to prevent compromise by well-funded attackers. However, it remains a fundamental building block for secure communication.

Physical Delivery

In some cases, especially for highly sensitive information, symmetric keys may be physically delivered using secure couriers or hardware tokens to ensure confidentiality.

Key Derivation

Key derivation functions play a crucial role in encryption protocols, such as TLS, by securely deriving session keys from the principle key [39].

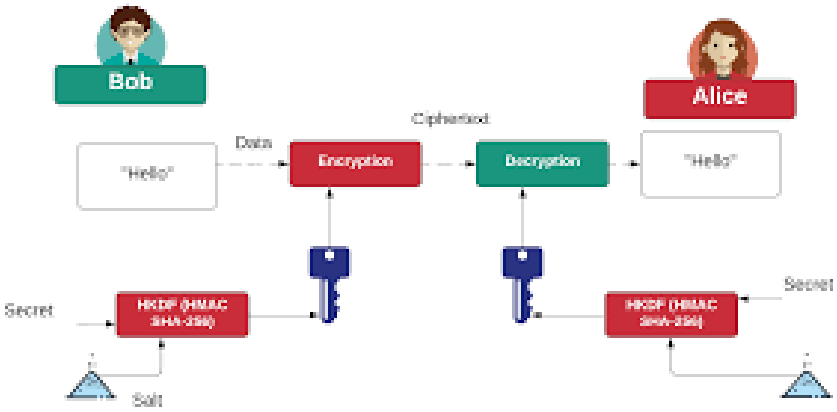


Figure 1.4: HKDF key derivation [4]

how it work:

1. **Master Key Generation:** Initially, a secure master key is generated through a key establishment protocol. This master key is shared between the communicating parties, often established through asymmetric encryption and authenticated key exchange mechanisms.

2. **Parameters for KDF:** Both parties agree on parameters for the key derivation function. These parameters might include the master key, additional data (such as nonces or identifiers), and possibly some other context-specific information.

3. **Key Derivation:** Using the agreed-upon parameters, the key derivation function computes session keys. This function takes the master key and other inputs and produces session keys that are unique to the current session. The function is designed to be computationally expensive to resist brute-force attacks and ensure that the derived keys are indistinguishable from random.

4. **Session Key Usage:** The derived session keys are then used for encrypting and decrypting the data exchanged during the session. Since these keys are unique to the session and derived from the master key, even if one session key is compromised, it doesn't affect the security of other sessions.

5. **Periodic Renegotiation:** Depending on the protocol, session keys might be renegotiated periodically to enhance security. This involves re-deriving new session keys using the same master key but possibly different parameters or additional data.

1.4.2 Asymmetric Key Distribution

Asymmetric key cryptography, also known as public-key cryptography, involves the use of two keys, one public and one private. These keys are mathematically related, but they are different. The public key is disclosed plainly, whereas the private key is kept confidential [40].

1. **Key Generation:** The first step is to generate a key pair: a public key and a private key. This is usually done by the intended recipient of encrypted messages.

2. **Public Key Distribution:** The public key is then distributed to anyone who needs to send encrypted messages to the recipient or verify the sender's digital signatures. Public keys can be freely distributed without compromising security.

3. **Private Key Protection:** The private key is kept securely by the recipient. It should never be shared with anyone else.

4. **Encryption:** When someone wants to send an encrypted message to the recipient, they use the recipient's public key to encrypt the message. Only the recipient, who holds the corresponding private key, can decrypt the message.

5. **Digital Signatures:** If the recipient wants to sign a message to prove its authenticity, they use their private key to create a digital signature. Anyone with access to the public key can verify that the signature was created by the holder of the corresponding private key.

6. **Certificate Authorities (Optional):** In many cases, a trusted third party known as a Certificate Authority (CA) is involved in the distribution of public keys. CAs issue digital certificates, which bind public keys to the identities of their owners, providing additional assurance of authenticity.

7. **Key Revocation:** If a private key is compromised or no longer needed, it should be revoked. This typically involves updating certificate revocation lists (CRLs) or using mechanisms such as the Online Certificate Status Protocol (OCSP) to inform users that the key is no longer trusted.

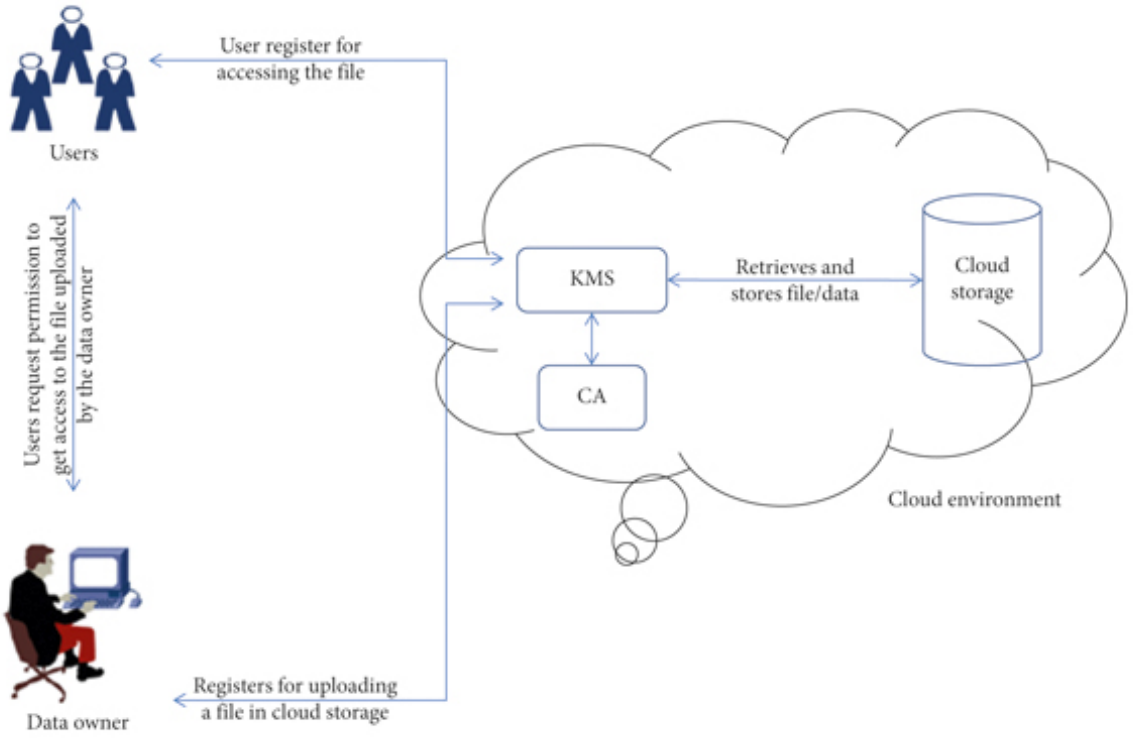


Figure 1.5: Asymmetric Key Cryptography Distribution [5]

1.5 Key Storage and Protection

Storage and protection are key aspects of encryption and security. Keys are used to encrypt and decrypt data, authenticate users, and ensure communication integrity. Proper storage and protection of keys is of great importance for preventing unauthorized access and maintaining information security.
Here are some key principles and practices for key storage and protection:

1.5.1 Best Practices for Key Storage

When it comes to key storage, especially in the context of cybersecurity, adhering to best practices is crucial for maintaining the security and integrity of sensitive information. Here are some guidelines to follow when storing keys [41]:

- **Encryption**: Keys themselves should be encrypted when stored to prevent unauthorized access. Encryption adds an extra layer of security, ensuring that even if an attacker gains access to the storage medium, they cannot use the keys without decryption.

- **Access Control**: Limit access to keys based on the principle of least privilege. Only authorized users or applications should have access to keys, and access should be tightly controlled using Access Control Lists (ACLs), Role-based Access Control (RBAC), or similar mechanisms.

- **Key Rotation**: Regularly rotate keys to mitigate the impact of key compromise or cryptographic vulnerabilities. Key rotation involves replacing old keys with new ones at predefined intervals while ensuring seamless operation of systems and applications.

- **Secure Transmission**: When distributing keys, use secure channels such as encrypted connections (e.g., TLS) or secure protocols (e.g., HTTPS) to prevent interception or tampering during transmission.

- **Auditing and Monitoring**: Implement logging and monitoring mechanisms to track key usage, detect suspicious activities, and facilitate forensic analysis in case of security incidents. Regularly review logs and audit trails to ensure compliance with security policies and regulations.

- **Key Destruction**: Properly manage the life cycle of keys, including secure deletion or destruction when no longer needed. This prevents unauthorized access to sensitive information, especially in the case of decommissioned systems or services.

- **Secure Development Practices**: Follow secure coding practices to avoid exposing keys inadvertently in source code or configuration files. Use secure key storage APIs provided by programming frameworks and libraries.

- **Regular Security Assessments**: Conduct regular security assessments, including penetration testing and vulnerability scanning, to identify and address potential weaknesses in key storage and protection mechanisms.

1.5.2 Key Management Systems

Implementing a strong key management system is critical. KMS provides a central platform for generating, storing and distributing encryption keys safely. It often includes features such as main rotation, access controls and audit capabilities. Key management

systems play an important role in managing encryption keys within the encryption system. Here are some important points about KMS [42]:

1. Definition KMS refers to the management of cryptographic keys, including their generation, exchange, storage, use, destruction, and replacement. It encompasses cryptographic protocol design, key servers, user procedures, and relevant protocols.

2. Purpose KMS ensures the secure handling of keys, which are essential for encryption, decryption, and authentication. Proper key management is vital for maintaining data confidentiality, integrity, and authenticity.

3. Components

   - **Key Generation**: Creating strong cryptographic keys.
   - **Key Exchange**: Securely sharing keys between parties.
   - **Key Storage**: Safely storing keys to prevent unauthorized access.
   - **Key Usage**: Properly using keys for encryption, decryption, and other cryptographic operations.
   - **Key Rotation**: Regularly replacing keys to enhance security.
   - **Key Destruction (Crypto-Shredding)**: Ensuring keys are securely deleted when no longer needed.

1.5.3 Secure hardware storage

Keys should be stored in secure environments, such as hardware security modules (HSMs), trusted platform modules (TPMs), or secure enclaves. These physical or virtual devices provide tamper-resistant protection for keys and cryptographic operations.

Hardware Security Modules

Hardware Security Modules are specialized hardware devices designed to manage digital keys, perform cryptographic operations, and provide secure storage for sensitive data such as encryption keys, and digital certificates. They are used in various industries where security is paramount, including finance, healthcare, government, and cloud services [43].

Here are some key features and functions of HSMs:

- **Key Management:** HSMs securely generate, store, and manage cryptographic keys used for encryption, decryption, digital signatures, and other cryptographic operations.
- **Secure Key Storage:** HSMs provide a secure environment for storing cryptographic keys, protecting them from unauthorized access and tampering. Keys stored within an HSM are typically stored in encrypted form and are only accessible through the HSM's cryptographic operations.
- **Hardware-based Security:** HSMs utilize specialized hardware components and security mechanisms to protect against various forms of attacks, including physical tampering, side-channel attacks, and software-based attacks.
- **Cryptographic Operations:** HSMs can perform a wide range of cryptographic operations, including encryption, decryption, digital signing, key generation, and key wrapping/unwrapping. These operations are performed securely within the HSM, ensuring the confidentiality and integrity of sensitive data.
- **Compliance and Auditing:** HSMs often support compliance requirements such as FIPS 140-2, PCI DSS, and GDPR by providing features such as audit logging, tamper-evident seals, and role-based access control.
- **Integration with Applications:** HSMs can be integrated with various applications and systems using standard cryptographic APIs such as PKCS#11, Microsoft CNG, and Java JCE, allowing applications to offload cryptographic operations to the HSM for improved security.
- **High Availability and Scalability:** HSMs are designed for high availability and scalability, with features such as redundant components, fail over mechanisms, and support for clustering and load balancing.

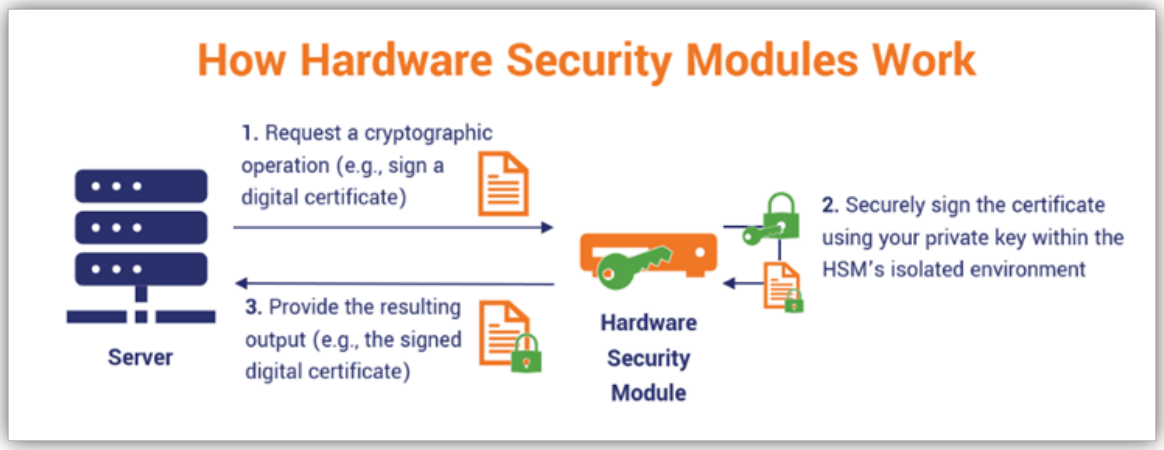


Figure 1.6: How a hardware security module works [6]

Trusted platform modules

Trusted Platform Modules (TPMs) are specialized hardware components used to enhance the security of computing devices. They are typically integrated into the motherboard of a device, such as a computer or a smartphone. TPMs provide a secure environment for storing cryptographic keys[44].
Here are some key features and functionalities of TPMs:

- **Secure Storage:** TPMs provide a secure area, often called a "secure enclave," where cryptographic keys and other sensitive data can be stored. This storage is isolated from the main system memory and inaccessible to unauthorized software or users.
- **Hardware-based Security:** Unlike software-based security solutions, TPMs operate at the hardware level, making them more resistant to attacks that target software vulnerabilities. This hardware-based approach enhances the overall security of the system.
- **Key Management:** TPMs can generate, store, and manage cryptographic keys used for various security purposes, such as encryption, authentication, and digital signatures. These keys are protected within the TPM and can be used without being exposed to the rest of the system.
- **Secure Boot:** TPMs can verify the integrity of the system firmware, boot loader, and operating system during the boot process. This helps prevent unauthorized

modifications to the boot process, such as the installation of malware or unauthorized software.

- **Remote Attestation:** TPMs can generate a cryptographic attestation of the system's configuration and integrity, which can be used to prove the trustworthiness of the system to external entities. Remote attestation enables secure communication and collaboration in distributed computing environments.

- **Secure Execution Environment:** Some TPMs support the execution of secure code within a trusted execution environment (TEE). This allows sensitive operations to be performed in a secure and isolated environment, protecting them from attacks and unauthorized access.

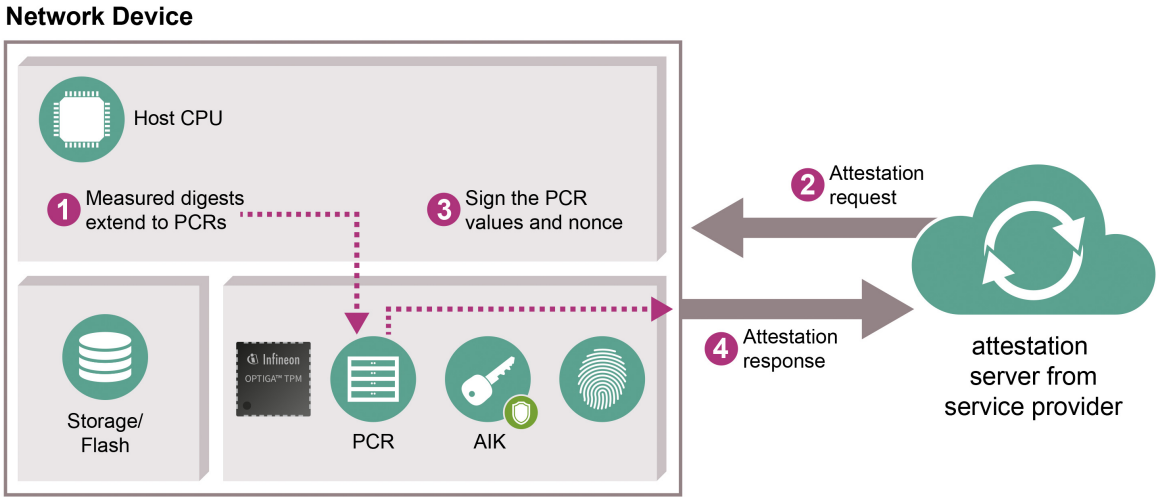


Figure 1.7: trusted platform modules [7]

1.6 Conclusion

Cryptographic key management is vital for modern information system security. This chapter explored its importance, principles, and practices. We covered encryption keys, distinguishing between symmetric and asymmetric types, and their roles in data confidentiality, integrity, and authenticity. The entire key life cycle (from generation to disposal) was discussed, along with best practices. Key generation techniques, distribution mechanisms, and storage strategies were examined to highlight key management's multifaceted

nature. Emphasizing strong practices to mitigate security risks and ensure compliance, we acknowledged evolving challenges. The next chapter will delve into key management in Big Data , focusing on advanced techniques, emerging trends, and practical strategies.

Chapter 2

Key management in Big Data

2.1 Introduction

In today's era of rapid data growth and digital transformation, effective cryptographic key management is essential for ensuring the integrity, confidentiality, and accessibility of sensitive information in Big Data ecosystems. As organizations harness vast datasets to drive innovation, robust key management practices become increasingly crucial.

This chapter examines the importance, challenges, and evolving approaches to key management in Big Data . Amidst ubiquitous data breaches and stringent compliance requirements, understanding key management principles is vital for navigating modern data landscapes. We will start by exploring Big Data 's fundamentals and its impact, followed by the necessity for effective key management. Additionally, we will address the challenges of securing cryptographic keys and evaluate emerging trends and technologies that enhance key management in dynamic Big Data environments.

2.2 Fundamentals of Big Data

2.2.1 What is Big Data ?

The term "Big Data " refers to a vast and diverse array of data that cannot be effectively managed by conventional data storage and processing systems. The term encompasses a considerable quantity of information derived from diverse sources, including but not limited to business procedures, machines, social media platforms, networks, and human

interactions. The significance of Big Data lies in its potential to furnish valuable insights and facilitate decision-making across diverse domains [45].

2.2.2 Characteristics of Big Data

The key Characteristics of Big Data can be summarized as follows [46]:

- **Volume:**
  - **Definition:** Big Data is characterized by its enormous size. It involves massive volumes of data generated daily.
  - **Example:** Facebook alone generates approximately a billion messages, records 4.5 billion "Like" button clicks, and uploads over 350 million new posts each day.
  - **Handling:** Big Data technologies can efficiently handle these large amounts of data.
- **Variety:**
  - **Definition:** Big Data can be structured, unstructured, or semi-structured. It comes from diverse sources, including databases, PDFs, emails, audios, social media posts, photos, videos, and more.
  - **Examples:**
    - **Structured Data:** Tabular data with well-defined columns stored in relational databases.
    - **Semi-structured Data:** Formats like JSON, XML, CSV, and email.
    - **Unstructured Data:** Includes log files, audio files, and image files.
  - **Importance:** Handling this variety of data is crucial for extracting meaningful insights.
- **Veracity:**
  - **Definition:** Veracity refers to the reliability and trustworthiness of data. It involves filtering and managing data effectively.

  - **Example:** Ensuring that data is accurate and can be used confidently for analysis.
  - **Business Impact:** Veracity is essential for business development and decision-making.

- **Value:**
  - **Definition:** Value represents the essential characteristic of Big Data . It's not just about processing or storing data; it's about storing, processing, and analyzing valuable and reliable information.
  - **Significance:** Extracting actionable insights from data adds value to organizations.

- **Velocity:**
  - **Definition:** Velocity plays a crucial role. It refers to the speed at which data is generated in real-time.
  - **Examples:**
    * Incoming data from application logs, business processes, networks, social media sites, sensors, and mobile devices.
    * Activity bursts and rapid changes.
  - **Purpose:** Big Data velocity ensures timely access to critical information.

2.3 Importance of Key Management in Big Data

Key management is paramount when it comes to Big Data , due to the vast volume, velocity, and variety of data being generated and processed. In Big Data environments, sensitive information is often stored, processed, and transmitted across various systems. This makes it vulnerable to security threats. This is why key management is crucial [47] [48]:

- **Data Integrity and Confidentiality:**
  - Key management plays a pivotal role in ensuring the integrity and confidentiality of data.

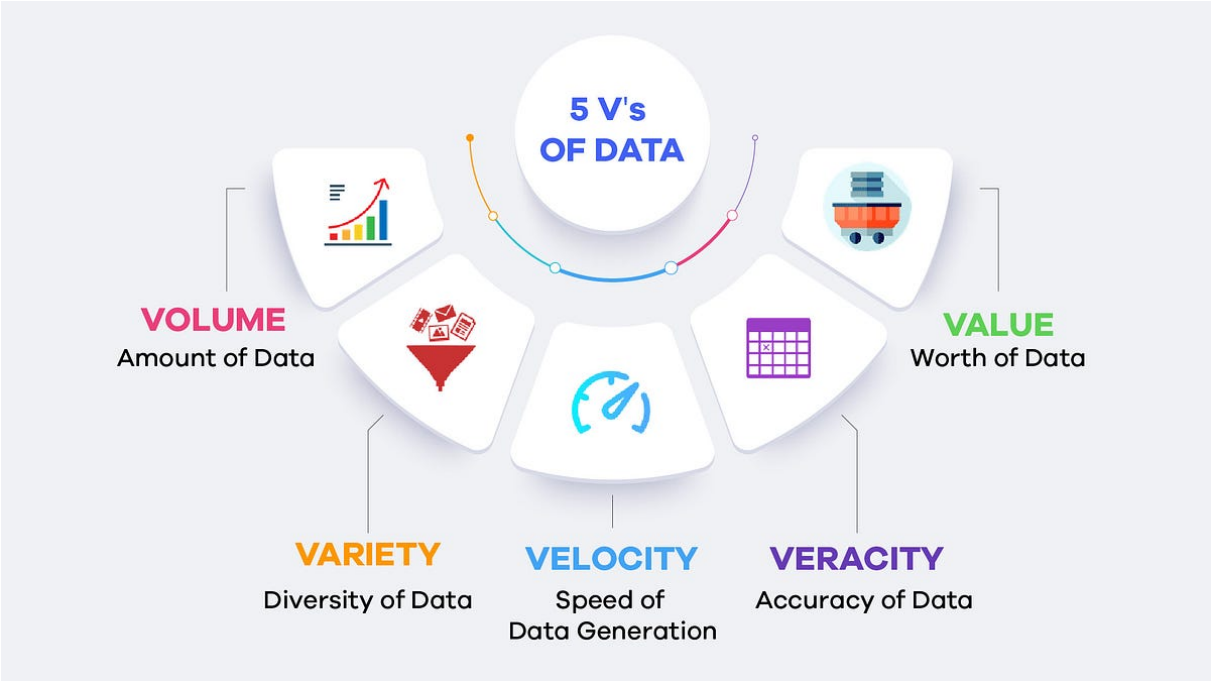


Figure 2.1: Characteristics of Big Data [8]

  - In Big Data environments, vast amounts of sensitive information are processed and stored. Properly managed encryption keys help protect this data from unauthorized access and tampering.
  - Without robust key management, data integrity can be compromised, leading to incorrect insights and decisions.

- **Regulatory Compliance:**
  - Big Data often involves handling personal information, financial records, and other sensitive data.
  - Compliance with data protection laws (such as GDPR, HIPAA, or CCPA) is essential. Proper key management ensures that data encryption adheres to regulatory requirements.
  - Failure to comply can result in hefty fines and reputational damage.

- **Secure Data Sharing and Collaboration:**
  - Organizations collaborate and share data across departments, partners, and cloud services.

  - Effective key management enables secure data sharing by allowing authorized parties to access encrypted data.
  - Without proper key management, sharing encrypted data becomes cumbersome and risky.

- **Protection Against Insider Threats:**
  - Insiders, including employees, contractors, or vendors, can pose security risks.
  - Adequate key management ensures that only authorized personnel can access sensitive data.
  - Without it, insiders might misuse or leak encryption keys, leading to data breaches.

- **Scalability and Performance:**
  - Big Data systems handle massive volumes of information.
  - Efficient key management practices allow for scalability without compromising performance.
  - Well-managed keys facilitate seamless encryption and decryption operations.

2.3.1 Risks Associated with Inadequate Key Management

There are several risks associated with poor key management practices [49]:

- **Data Breaches:**
  - Weak key management can lead to data breaches.
  - If encryption keys are compromised, attackers can decrypt sensitive data, exposing it to unauthorized parties.
  - Breaches can result in financial losses, legal consequences, and damage to reputation.

- **Unauthorized Access:**
  - Inadequate key management allows unauthorized users to gain access to encrypted data.

  - Without proper controls, anyone with access to the keys can decrypt and view sensitive information.
  - This jeopardizes data privacy and security.

- **Data Loss:**
  - Lost or mismanaged keys can render encrypted data permanently inaccessible.
  - If keys are lost, data recovery becomes impossible.
  - Organizations may lose critical information, affecting business continuity.

- **Compliance Violations:**
  - Poor key management practices violate data protection regulations.
  - Failure to protect encryption keys can result in non-compliance penalties.
  - Organizations risk legal actions and financial penalties.

- **Operational Disruptions:**
  - Inefficient key management processes can disrupt operations.
  - Slow decryption due to poorly managed keys impacts system performance.
  - Delays in accessing data hinder decision-making and productivity.

2.4 Challenges in Key Management for Big Data

Key management in Big Data environments poses several unique challenges due to the scale, complexity, and distributed nature of the data. Here are some of the key challenges [50] [51] [52]:

1. **Scalability:** Big Data often involves massive datasets spread across numerous servers and potentially even cloud platforms. Traditional key management systems designed for smaller, on-premise setups struggle to handle this sprawl.
   - **Challenge:** Keeping track of a vast number of keys spread across different locations securely and ensuring all have proper access control.
   - **Impact:** Increased risk of keys being lost or compromised due to the complexity of managing them.

2. **Diverse Data Sources:** Big Data incorporates information from a wide variety of sources, each with potentially unique security requirements. This heterogeneity makes a one-size-fits-all key management approach impractical.

    - **Challenge:** Developing and implementing different key management strategies for various data types (structured, unstructured, etc.) while maintaining consistency.
    - **Impact:** There is an increased risk of vulnerabilities if specific data types aren't secured with appropriate key management practices.

3. **Real-time Processing:** Big Data often demands real-time analysis, making traditional key retrieval processes that might involve manual intervention a bottleneck.

    - **Challenge:** Balancing the need for secure key access with the speed required for real-time analytics.
    - **Impact:** Potential delays in data processing or compromising security by using less secure key access methods for speed.

    - **Key Rotation:** With a larger attack surface, Big Data environments necessitate even more frequent key rotation to minimize the damage if a key is compromised. Managing this rotation across a vast number of keys securely adds another layer of complexity.
    - **User Access Control:** Granular access control becomes even more critical in Big Data to ensure only authorized users have access to specific datasets and the keys that unlock them.

2.5 Centralized Key Management Techniques in Big Data

Centralized key management refers to the practice of managing encryption keys, which are used to secure sensitive information within large-scale data environments, from a single centralized location. Instead of distributing management responsibilities across various systems or departments, all aspects of key management, including generation, storage,

access control, rotation, and auditing, are handled centrally. This approach ensures that consistent security measures are applied across the organization's Big Data infrastructure, thereby simplifying management and enhancing data protection Figure 2.2 [53] [54].

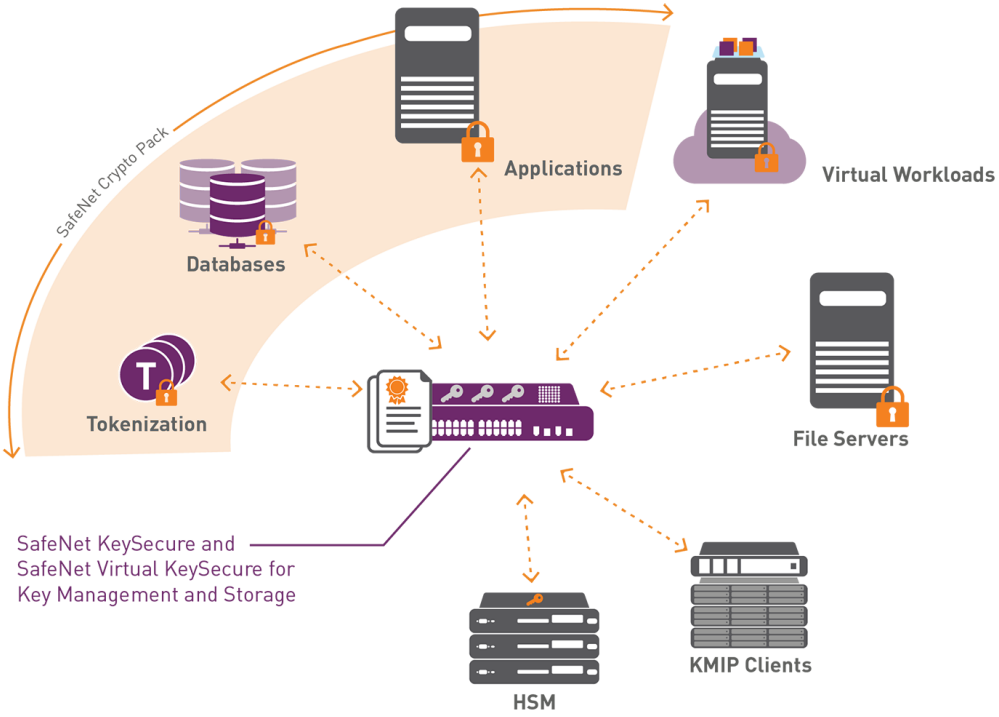


Figure 2.2: centralized key management [9]

2.5.1 Cloud Key Management Services

Cloud key management services are cloud-based services that help manage cryptographic keys used to encrypt data. These services typically offer secure key storage, key generation, key rotation, and key usage auditing capabilities. The Cloud Key Management System allows users to encrypt their data stored in the cloud and control access to the encryption keys, providing an additional layer of security for sensitive information, as shown in Figure 2.3 [55] [56].

how it works:

Cloud Key Management Services work by providing a centralized platform for generating, storing, managing, and controlling cryptographic keys used for encryption and decryption in cloud environments. Here's a simplified explanation of how it works [57]:

- **Key Generation:** KMS generates cryptographic keys using strong random number generators. These keys can be used for various cryptographic operations, such as encryption, decryption, digital signing, and verification.
- **Secure Storage:** The generated keys are securely stored within the KMS infrastructure. This storage is typically designed with robust security measures to protect the keys from unauthorized access, theft, or tampering. Hardware Security Modules (HSMs) are often used to provide additional layers of protection for the keys.
- **Key Management:** KMS provides APIs or interfaces for managing the keys throughout their life cycle. This includes tasks such as creating new keys, rotating keys periodically to enhance security, revoking or disabling keys when necessary (e.g., in case of compromise), and securely deleting keys that are no longer needed.
- **Access Control:** KMS enforces access controls to ensure that only authorized users or applications can access the keys. Access policies can be defined to specify which users or services have permission to perform specific cryptographic operations using the keys.
- **Encryption and Decryption:** When an application or service needs to encrypt data, it sends a request to the KMS, specifying the appropriate key and the data to be encrypted. The KMS then retrieves the key, performs the encryption operation,

and returns the encrypted data to the requester. Similarly, when decryption is required, the requester sends the encrypted data and the corresponding key to the KMS, which performs the decryption operation and returns the plain text data.

- **Auditing and Logging:** KMS logs all key management activities, including key creation, usage, rotation, and deletion. This auditing capability allows administrators to track and monitor key usage, detect any suspicious activities, and ensure compliance with security policies and regulations.
- **Integration with Cloud Services:** KMS services are often integrated with other cloud services, allowing seamless encryption and decryption of data stored in databases, object storage, or transmitted over networks within the cloud environment. This integration simplifies the implementation of data security measures for cloud-based applications and services.

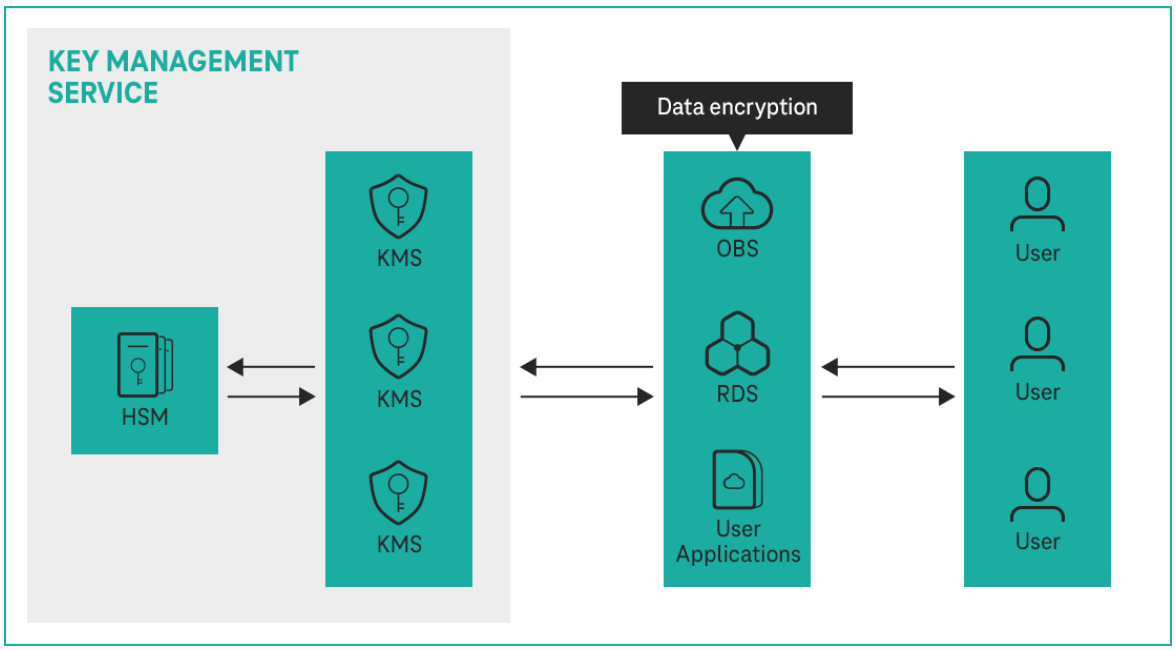


Figure 2.3: Cloud Key Management Service [10]

2.5.2 Hierarchical Key Management

Hierarchical Key Management (HKM) organizes cryptographic keys in a hierarchy, starting with a main "root" key from which other keys are derived. These derived keys

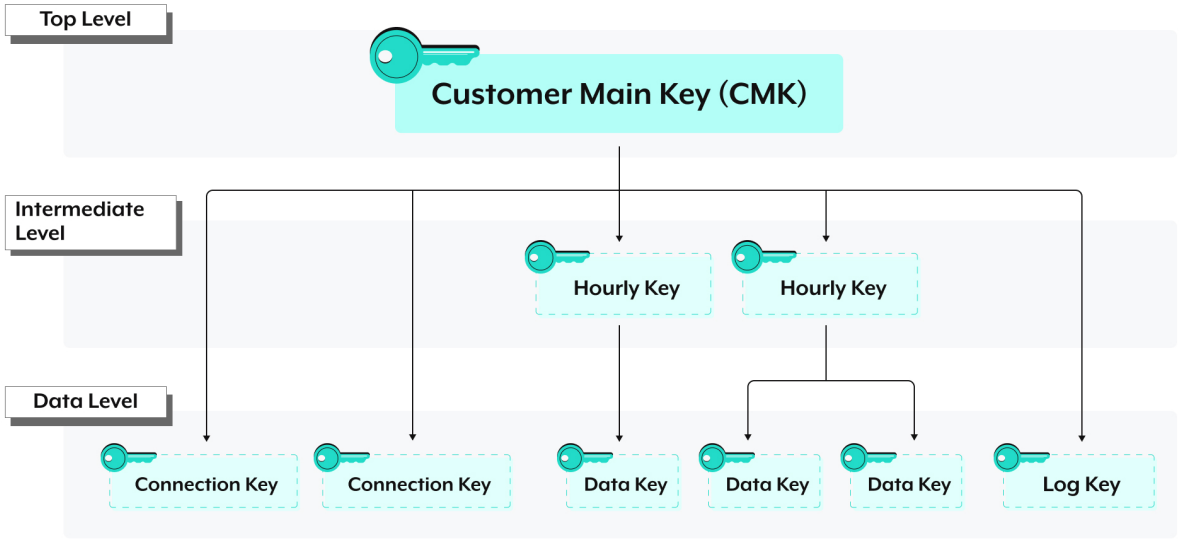


Figure 2.4: Hierarchical Key Management [11]

are used for specific tasks like encrypting data or signing messages, enabling efficient and secure key management with different levels of access control. This limits the impact of a compromised key. In Big Data , HKM secures and controls access to sensitive data by organizing encryption keys hierarchically. This method ensures security, scalability, and granular access control in large-scale data systems dealing with massive data volumes [58] [59][60] as shown in Figure 2.4.

How it works:

Here's a simplified explanation of how HKM works [58] [61]:

- **Root Key Generation:** The process starts with generating a highly secure root key. This key serves as the top-level key in the hierarchy and is typically stored in a secure location, such as a hardware security module (HSM).
- **Derivation of Intermediate Keys:** From the root key, intermediate keys are derived. These intermediate keys can represent different levels of the hierarchy or be associated with specific data domains or services within the Big Data environment.
- **Assignment of Leaf Keys:** At the lowest level of the hierarchy, leaf keys are assigned. These keys are used for specific cryptographic operations, such as encrypting

or decrypting data, and are typically associated with individual pieces of data or specific tasks within the Big Data system.

- **Encryption and Decryption:** When data needs to be encrypted or decrypted within the Big Data system, the appropriate leaf key is used for the cryptographic operation. The leaf keys may be dynamically generated as needed or pre-assigned based on predefined policies.

- **Access Control and Auditing:** Access to keys at different levels of the hierarchy is controlled through access control mechanisms. This ensures that only authorized users or processes can access sensitive cryptographic material. Additionally, auditing mechanisms may be employed to track key usage and detect any unauthorized access attempts.

- **Key Rotation and Management:** Periodic key rotation and management practices are implemented to enhance security. This involves replacing old keys with new ones, updating key metadata, and ensuring that encryption algorithms and key lengths remain up-to-date with best practices.

2.5.3 Key Management Interoperability Protocol (KMIP)

KMIP serves as a universal language for cryptographic key management, facilitating seamless communication between different components within a cryptographic infrastructure. It operates at a higher level of abstraction, enabling interoperability between diverse key management systems, cryptographic devices, and applications [62].

At its foundation, KMIP defines a set of standardized operations for managing cryptographic keys and related objects. These operations encompass key life cycle management tasks such as creation, distribution, activation, deactivation, and destruction of keys, as well as operations for managing key attributes, permissions, and usage policies [63].

One of the key features of KMIP is its support for a wide range of cryptographic algorithms and key types, allowing organizations to manage keys used in symmetric encryption, asymmetric encryption, digital signatures, and other cryptographic operations. This

flexibility ensures compatibility with various cryptographic standards and requirements, enabling organizations to adapt to evolving security needs and regulatory frameworks [64].

How it works:

The Key Management Interoperability Protocol works by defining a standardized set of messages and operations for managing cryptographic keys and related objects. Here's a simplified overview of how it typically works [63]:

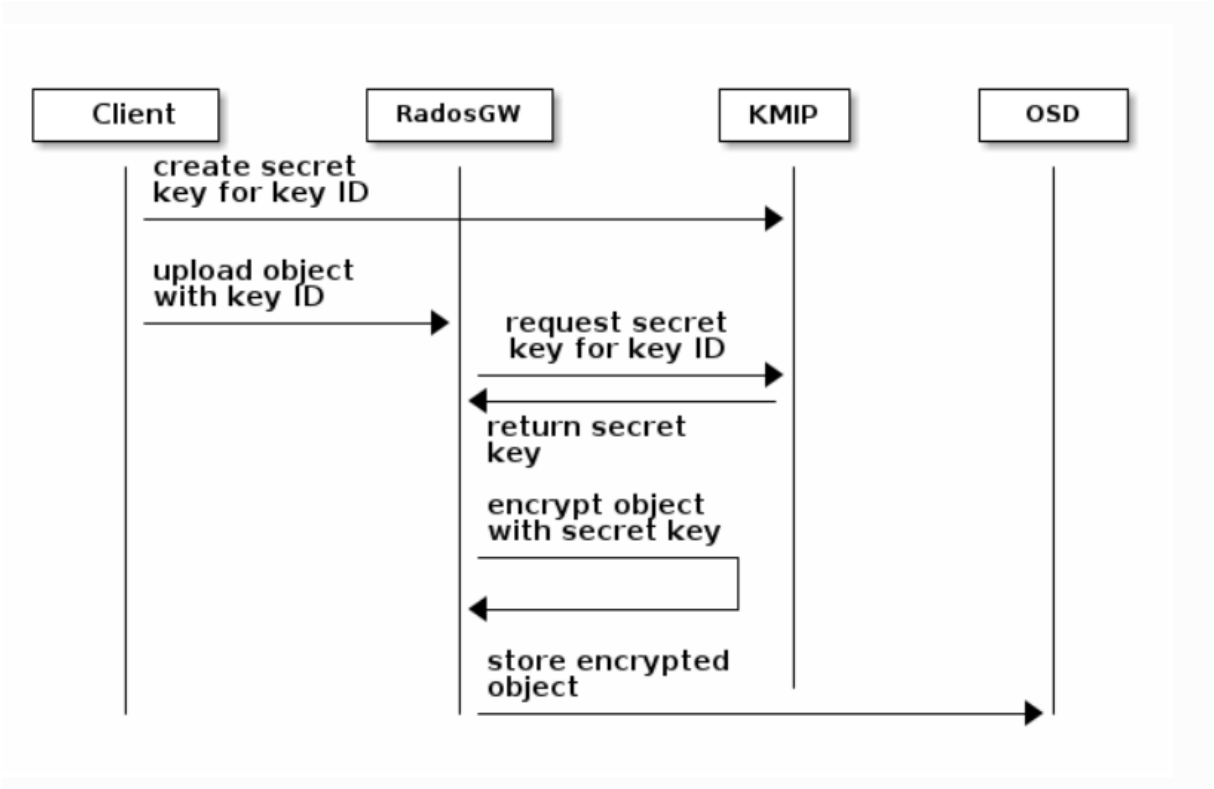


Figure 2.5: Key Management Interoperability Protocol [12]

1. **Initialization**: When a KMIP client (such as an application or device) wants to communicate with a KMIP server (a key management system), it initiates a connection to the server.

2. **Authentication**: The client authenticates itself to the server using credentials such as a username and password or through other authentication mechanisms supported by KMIP, such as client certificates or authentication tokens.

3. **Session Establishment**: Once authenticated, the client and server establish a session for communication. During this phase, they negotiate parameters such as encryption algorithms and key lengths to ensure secure communication.

4. **Key Operations**: The client can then send requests to the server to perform various key management operations, such as:

   - Creating new cryptographic keys
   - Retrieving existing keys
   - Deleting keys
   - Updating key attributes (e.g., changing access permissions)
   - Performing cryptographic operations using keys stored on the server (e.g., encryption, decryption, signing)

5. **Response Handling**: The server processes the client's requests and sends back appropriate responses, indicating the success or failure of the operations. If successful, the response may also include requested data, such as keys or key metadata.

6. **Session Termination**: When the client no longer needs to communicate with the server, it can terminate the session, freeing up resources on both sides.

2.5.4 Centralized Blockchain-based Key Management

Centralized Blockchain-based Key Management involves a central authority managing cryptographic keys in a blockchain network. Unlike traditional systems where participants manage their own keys, this centralized approach delegates the creation, distribution, and revocation of keys to a single entity. This system ensures streamlined key management,

potentially enhancing security and efficiency by reducing the burden on individual participants and centralizing control within the network. [65].

How it works:

In a Centralized Blockchain-based Key Management system, the process typically involves the following steps [65] [66]:

1. Key Generation:
    - The central authority employs cryptographic algorithms to generate pairs of public and private keys for each participant.
    - Public keys are shared openly and are used for verifying signatures and encrypting messages.
    - Private keys are kept secret and are used for signing transactions and decrypting messages.
2. Key Distribution:
    - Once keys are generated, they are securely distributed to participants. This may involve encryption and secure channels to prevent interception.
    - Participants may be authenticated through various means to ensure that keys are delivered to the intended recipients.
3. Key Revocation:
    - If a participant's key is compromised, lost, or if the participant is no longer authorized to access the network, the central authority revokes the corresponding key.
    - Revocation prevents unauthorized access and fraudulent activities by invalidating compromised or outdated keys.
4. Key Recovery:
    - In the event that a participant loses their private key or is unable to access it, the central authority may provide a key recovery mechanism.

- This could involve identity verification procedures to ensure that only legitimate participants regain access to their keys.

5. Key Rotation:

   - Periodically, the central authority may enforce key rotation policies to enhance security.
   - Key rotation involves generating new key pairs for participants and replacing existing keys.
   - This mitigates the risk of long-term key exposure and potential compromise.

6. Centralized Monitoring and Control:

   - The central authority maintains centralized oversight of key management processes.
   - It monitors key usage, detects anomalies or suspicious activities, and enforces security policies.
   - This centralized control enables swift response to security incidents and ensures compliance with key management protocols.

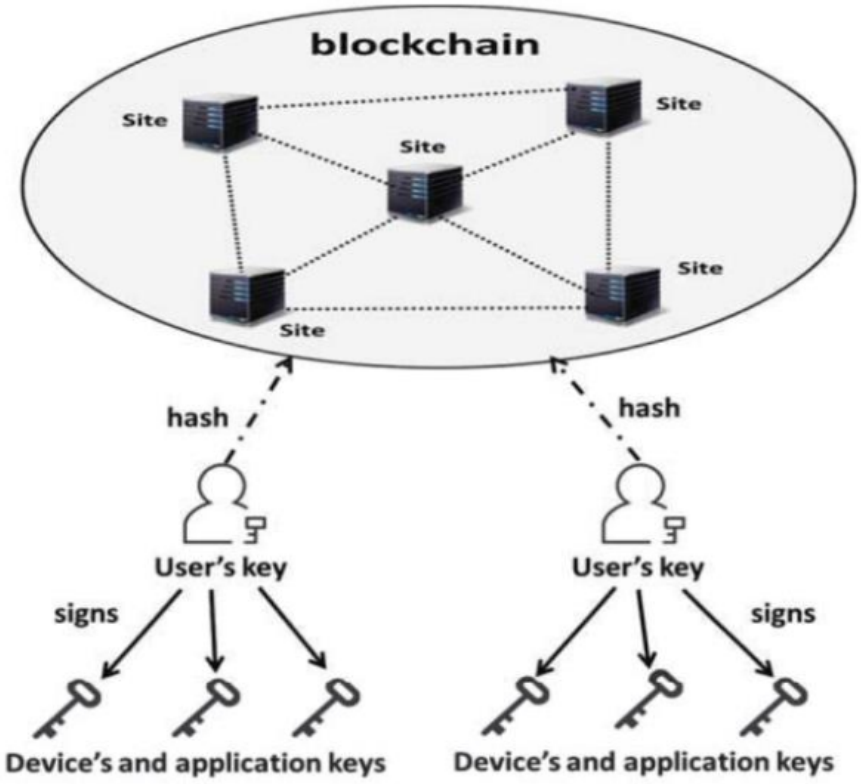


Figure 2.6: Centralized Blockchain-based Key Management [13]

2.5.5 Advantages of centralized key management

centralized key management (CKM) offers numerous advantages [67]:

- **Simplified Key Management:** Managing cryptographic keys in a centralized manner simplifies key generation, distribution, rotation, and retirement processes. It eliminates the need for disparate key management solutions across multiple systems and platforms, reducing complexity and administrative overhead.

- **Improved Compliance:** Many industries are subject to regulatory requirements mandating the protection of sensitive data through encryption and access controls. CKM facilitates compliance with regulatory standards (such as GDPR, HIPAA, and PCI DSS) by providing a centralized approach to key management, audit logging, and access control enforcement.

- **Efficient Resource Utilization:** Centralized key management allows organizations to optimize resource utilization by consolidating key management functions within a single system or service. This minimizes the need for redundant key man-

agement infrastructure and streamlines key-related operations, resulting in cost savings and operational efficiencies.

- **Scalability and Flexibility:** CKM solutions are designed to scale with the growing needs of Big Data environments. They can accommodate large volumes of cryptographic keys and support integration with various Big Data platforms, frameworks, and cloud services, ensuring scalability and flexibility in key management operations.

- **Resilience and Disaster Recovery:** Centralized key management enhances resilience and disaster recovery capabilities by providing redundancy and backup mechanisms for cryptographic keys. Organizations can implement robust key replication, backup, and recovery strategies to ensure continuous access to keys and data in the event of system failures or disasters.

- **Ease of Integration:** CKM solutions are often designed to seamlessly integrate with existing Big Data infrastructure, applications, and workflows. They provide APIs, SDKs, and plugins that facilitate integration with popular Big Data platforms (e.g., Hadoop, Spark, Kafka) and cloud services, enabling organizations to deploy centralized key management without disrupting existing workflows or architectures.

2.5.6 Inconveniences of centralized key management

While centralized key management offers numerous advantages, it also comes with some potential inconveniences and challenges [67]:

- **Single Point of Failure:** Centralizing cryptographic keys introduces a single point of failure. If the centralized key management system experiences downtime or becomes inaccessible, it can disrupt data access and processing across the entire Big Data environment, leading to potential service interruptions and operational issues.

- **Security Risks:** A centralized key management system becomes an attractive target for attackers. If compromised, it could result in unauthorized access to sensitive cryptographic keys, leading to data breaches and confidentiality breaches. Therefore, robust security measures, such as encryption, access controls, and monitoring, must be implemented to mitigate these risks.

- **Performance Bottlenecks:** Depending on the scale and architecture of the centralized key management system, it may introduce performance bottlenecks, particularly during key retrieval and distribution operations. As the number of keys and the volume of data increases, the centralized key management system may struggle to keep up with the demands, impacting the overall performance of data processing workflows.

- **Complexity and Scalability Challenges:** Implementing and managing a centralized key management system can introduce complexity, especially in large-scale and distributed Big Data environments. As the volume of data and the number of users/systems accessing the keys grow, managing key life cycle, access controls, and compliance requirements can become increasingly complex and challenging to scale.

- **Vendor Lock-In:** Organizations that opt for proprietary, centralized key management solutions may face vendor lock-in, limiting their ability to switch to alternative solutions or migrate to different platforms in the future. Vendor lock-in can restrict flexibility, increase dependency on specific vendors, and potentially lead to higher costs in the long run.

2.6 Decentralized Key Management Techniques in Big Data

Decentralized key management refers to managing cryptographic keys in a distributed manner, avoiding reliance on a centralized authority. Traditional systems, like centralized databases or cloud services, often have a single entity managing these keys, making them potential targets for attacks or compromise [68].

In decentralized key management systems, control over cryptographic keys is distributed among multiple participants or nodes in a network. This enhances security, resilience against attacks, and privacy. Blockchain technology, often associated with decentralized key management, uses cryptographic keys to control access to digital assets and execute transactions. This ensures that no single entity controls the entire network, reducing the risk of a single point of failure [69].

2.6.1 Distributed Key Management Systems (DKMS)

A Distributed Key Management System (DKMS) is a cryptographic framework that handles the generation, distribution, storage, and maintenance of keys across interconnected nodes or devices. Unlike centralized systems, DKMS spreads key management across multiple entities, reducing single-point failure risks. It uses encryption algorithms, authentication protocols, and access control to ensure secure key handling, protecting data confidentiality, integrity, and availability. DKMS is essential in distributed environments like cloud computing, blockchain, IoT ecosystems, and multi-party communication systems, ensuring robust security for sensitive information [70].

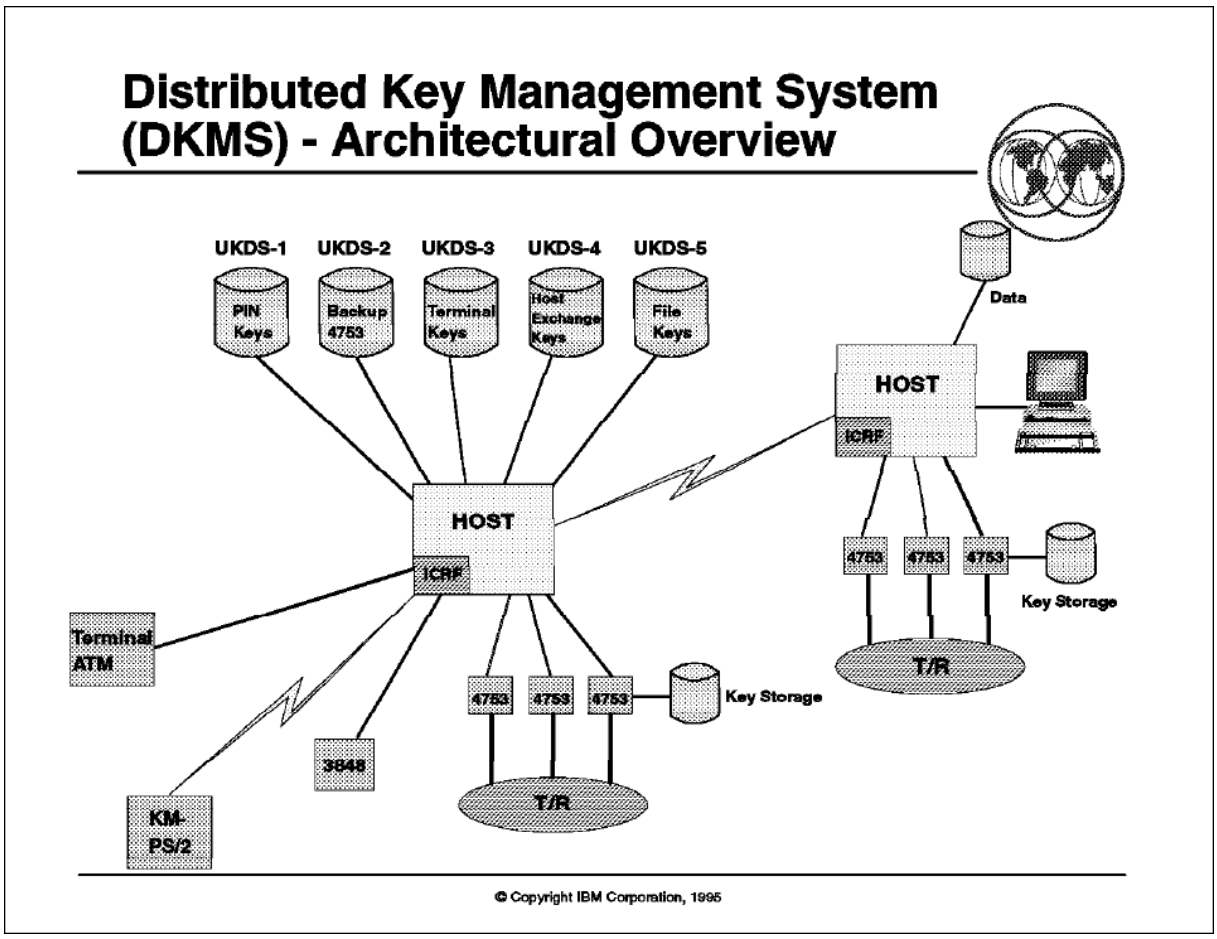


Figure 2.7: Distributed Key Management Systems [14]

How it works

Let's delve into more details about how a Distributed Key Management System operates [70]:

1. **Key Generation**:
    - DKMS employs secure random number generators and cryptographic algorithms to generate keys with sufficient entropy and strength.
    - Depending on the application requirements, DKMS may generate different types of keys, such as symmetric keys for encryption and decryption or asymmetric key pairs for digital signatures and key exchange.
2. **Key Distribution**:
    - DKMS utilizes secure communication channels and protocols to distribute keys to authorized entities.
    - Key distribution mechanisms may include key agreement protocols like Diffie-Hellman, where two parties can agree on a shared secret key without explicitly transmitting it, or public key infrastructure (PKI) for distributing public keys and certificates.
3. **Key Storage**:
    - Keys are stored in secure repositories or cryptographic hardware modules to prevent unauthorized access.
    - Hardware Security Modules (HSMs) are often used to store and manage keys securely, providing tamper-resistant hardware and cryptographic operations.
    - Access controls are enforced to restrict key access to authorized users or applications, typically through role-based access control (RBAC) or cryptographic access policies.
4. **Key Rotation and Update**:
    - DKMS regularly rotates keys to limit exposure to potential compromise or cryptographic attacks.
    - Key rotation intervals and algorithms are carefully chosen based on security requirements and best practices.
    - Automated processes and protocols ensure seamless key rotation without disrupting services, including mechanisms for distributing updated keys to relevant parties.

5. **Key Recovery**:
    - DKMS implements key recovery mechanisms to recover lost or corrupted keys to maintain operational continuity.
    - Key escrow services may be employed to securely store copies of keys, accessible only under specific conditions or by authorized personnel.
    - Multi-factor authentication and secure procedures are used to verify the identity of individuals requesting key recovery.
6. **Key Revocation**:
    - DKMS provides mechanisms for revoking compromised or unauthorized keys to prevent their misuse.
    - Revocation processes are initiated based on security incidents, key compromise events, or policy violations.
    - Real-time revocation mechanisms, such as Certificate Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP), enable immediate invalidation of revoked keys.
7. **Auditing and Monitoring**:
    - DKMS includes comprehensive auditing and monitoring capabilities to track key management activities and security events.
    - Audit logs capture key-related operations, access attempts, and security incidents for forensic analysis and compliance auditing.
    - Monitoring tools provide real-time visibility into key usage patterns, anomalies, and potential security threats, enabling proactive response and mitigation.

2.6.2 Bring Your Own Key

Bring Your Own Key (BYOK) is a cloud security practice where customers manage their own encryption keys for encrypting data stored in the cloud. This gives customers more control and ownership over their data security. By managing their encryption keys, organizations can enforce stricter access controls and encryption policies, ensuring only

authorized users can decrypt sensitive data. BYOK also helps meet compliance requirements by demonstrating control over encryption keys. It is commonly used with cloud services like storage and databases, but requires careful planning to maintain key integrity and prevent unauthorized access or loss [71].

How it works:

Bring Your Own Key operates along these fundamental steps [72]:

- **Key Generation:** The customer generates encryption keys using their own key management system (KMS) or cryptographic hardware module. These keys are typically generated using strong cryptographic algorithms to ensure security.
- **Key Import:** The customer securely transfers the generated encryption keys to the cloud service provider's environment. This transfer usually occurs through a secure channel, such as a secure API or encrypted file transfer.
- **Key Registration:** The cloud service provider registers the imported encryption keys within their system. This involves storing metadata about the keys, such as key identifiers and associated policies, in a secure manner.
- **Data Encryption:** When data is uploaded to the cloud service, it is encrypted using the encryption keys provided by the customer. This ensures that the data is protected with customer-controlled encryption keys rather than keys managed solely by the cloud provider.
- **Access Control:** The cloud service provider enforces access controls to ensure that only authorized users or applications can access the encrypted data. Access policies are typically managed by the customer and enforced by the cloud provider's access control mechanisms.
- **Key Usage:** Whenever data needs to be decrypted, the encryption keys provided by the customer are used to perform the decryption operation. This ensures that the customer maintains control over the data encryption process and can revoke access to the data by revoking or rotating the encryption keys as needed.

2.6.3 Decentralized Blockchain-based Key Management

Decentralized blockchain-based key management refers to a method of securely managing cryptographic keys using blockchain technology in a decentralized manner.

Blockchain technology provides an ideal platform for decentralized key management due to its inherent properties of transparency, immutability, and decentralization. In a blockchain-based key management system, cryptographic keys are stored on the blockchain in a tamper-proof and transparent manner. Access to these keys is controlled by smart contracts, which enforce predefined rules and conditions for key management.

One common application of decentralized blockchain-based key management is in cryptocurrency wallets. In this scenario, users have control over their private keys, which are used to access and transfer their digital assets. By storing these keys on a blockchain in a decentralized manner, users can mitigate the risk of losing access to their funds due to centralized failures or security breaches .

Another application is in secure communication systems, where users can exchange encrypted messages using decentralized key management protocols. By leveraging blockchain technology, these systems can ensure the integrity and confidentiality of communication channels without relying on centralized authorities [69] [73].

How it works:

The steps for implementing decentralized blockchain-based key management are as follows[74] [73]:

- Key Generation: Users generate a pair of cryptographic keys - a public key and a private key. The private key is kept secret and is used to sign transactions or decrypt messages, while the public key is shared publicly and is used to verify signatures or encrypt messages.
- Blockchain Registration: Users register their public keys on a blockchain network. This registration process typically involves creating a transaction that includes the public key and broadcasting it to the network.

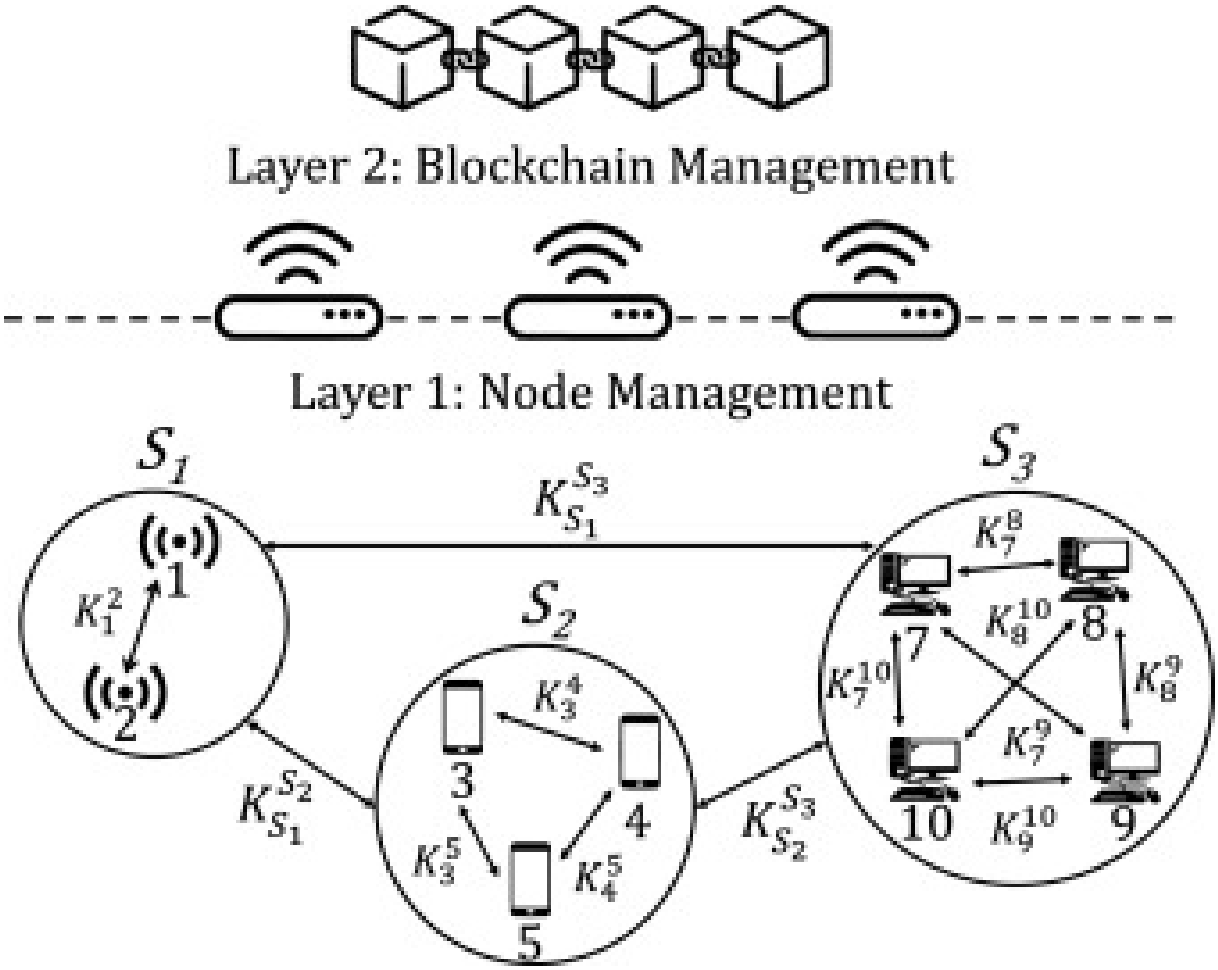


Figure 2.8: Decentralized Blockchain-based Key Management [15]

- Smart Contract Deployment: Smart contracts are deployed on the blockchain to manage key-related operations. These smart contracts define rules and conditions for key management, such as who can access the keys and under what circumstances.

- Access Control: The smart contracts enforce access control policies based on predefined rules. For example, a smart contract might specify that only the owner of a private key can initiate transactions using that key.

- Key Usage: When a user wants to use their private key to sign a transaction or decrypt a message, they interact with the smart contract on the blockchain. The smart contract verifies that the user is authorized to use the key based on the predefined rules.

- Transaction Verification: Transactions initiated by users are broadcasted to the blockchain network. These transactions include cryptographic signatures generated

using the private key. Other nodes on the network verify the signatures using the corresponding public keys stored on the blockchain.

- Consensus: The blockchain network reaches consensus on the validity of transactions through mechanisms such as proof-of-work or proof-of-stake. Valid transactions are added to the blockchain in a sequential and immutable manner.

- Key Revocation: In case a user wants to revoke access to their keys (for example, in the event of a compromised private key), they can interact with the smart contract to revoke the associated public key. Once revoked, the key cannot be used for further transactions.

- Key Recovery: Some systems may incorporate key recovery mechanisms to help users regain access to their keys in case of loss or compromise. These mechanisms typically involve using additional cryptographic techniques, such as multi-signature schemes or secret sharing, to securely recover the keys.

2.6.4 Advantages of Decentralized key management

Decentralized key management offers many advantages over centralized approaches [75]:

1. **Enhanced Security:** With decentralized key management, there is no single point of failure. Even if one node is compromised, the entire system's security is not compromised because the keys are distributed across multiple nodes. This makes it more difficult for attackers to gain unauthorized access to sensitive data.

2. **Improved Privacy:** Decentralized key management can enhance privacy by limiting the number of entities that have access to encryption keys. This reduces the risk of unauthorized access to sensitive data and helps protect user privacy.

3. **Resilience and Fault Tolerance:** Decentralized systems are more resilient to failures because they distribute key management tasks across multiple nodes. If one node fails or goes offline, the system can continue to operate using the keys stored on other nodes.

4. **Scalability:** Decentralized key management can scale more easily than centralized approaches. As the volume of data increases or the number of users grows, additional

nodes can be added to the network to handle the increased load without sacrificing performance or security.

5. **Reduced Dependency on Trust:** In a decentralized key management system, trust is distributed across multiple nodes rather than relying on a single trusted authority. This reduces the risk of abuse or misuse of authority by any single entity and increases the overall trustworthiness of the system.

6. **Compliance and Regulatory Compliance:** Decentralized key management can help organizations comply with regulatory requirements related to data security and privacy. By distributing key management responsibilities, organizations can demonstrate a higher level of control and accountability over their data encryption practices.

2.6.5 Inconveniences of Decentralized key management

While decentralized key management offers many advantages, it also presents some challenges and inconveniences [76] [77]:

- **Complexity:** Decentralized key management systems tend to be more complex than centralized ones. Coordinating key distribution, synchronization, and access control across multiple nodes requires sophisticated protocols and mechanisms, which can increase implementation and maintenance complexity.

- **Key Recovery:** In a decentralized system, if a node storing encryption keys becomes inaccessible or compromised, key recovery can be challenging. Ensuring that keys can be recovered without compromising security requires careful planning and robust mechanisms for key backup and recovery.

- **Coordination Overhead:** Managing encryption keys across multiple nodes requires coordination and communication among those nodes. This coordination overhead can introduce latency and performance bottlenecks, particularly in large-scale distributed systems with a high volume of data transactions.

- **Potential for Misconfiguration:** Decentralized key management systems may be more susceptible to misconfiguration errors, as the responsibility for key man-

agement is distributed across multiple entities. A misconfigured node could lead to security vulnerabilities or data breaches if not properly identified and addressed.

- **Regulatory Compliance:** Decentralized key management systems may pose challenges in terms of regulatory compliance, particularly in highly regulated industries such as finance or healthcare. Ensuring compliance with data protection regulations while distributing key management responsibilities across multiple entities requires careful planning and adherence to relevant compliance requirements.

2.7 Conclusion

In conclusion, managing cryptographic keys in Big Data is a complex challenge requiring careful consideration and innovative solutions. This chapter explored the interplay between vast volumes, varied sources, and the dynamic nature of Big Data , highlighting the need for robust key management to ensure confidentiality, integrity, and availability. We discussed the inadequacies of traditional key management techniques in Big Data environments and emphasized the importance of emerging trends and technologies for scalability, agility, and efficiency. The next chapter will present our proposed approach, focusing on a framework designed to address these multifaceted challenges with scalability, agility, and efficiency in mind.

Chapter 3

Proposed Approach

3.1 Introduction

In the preceding chapters, we explored key management's importance in ensuring data integrity, confidentiality, and availability within modern computing environments. We examined established methods and best practices for key management, highlighting its role in protecting sensitive information from unauthorized access and malicious attacks.

Building on this, we investigated key management within the Big Data domain, addressing the challenges posed by the scale, speed, and diversity of data. We assessed conventional approaches and identified areas for improvement.

Now, we aim to develop more robust, efficient, and scalable key management solutions for Big Data . This chapter presents our vision, combining established principles, emerging technologies, and new methodologies.

3.2 Proposed Idea for Key Management

We offer a unique approach based on the notion of a Double Key Management Center (KMC). This novel technique aims to improve the security, transparency, and scalability of key management processes by utilizing blockchain technology (Public Record (Blockchain) system).

At the center of our proposed design are two interconnected KMCs that play unique but complementary functions in the key management life cycle. These KMCs' major job is to produce, store, distribute, and revoke cryptographic keys for securing data assets in the Big Data ecosystem.

The first KMC, termed the "Issuer KMC," is responsible for the initial provisioning and distribution of cryptographic keys to authorized entities within the system. It serves as the centralized authority for key generation and management, ensuring that keys are securely disseminated to legitimate users and revoked promptly upon request or expiration.

The second KMC, known as the "Verifier KMC", operates in tandem with the issuer KMC to validate the authenticity and integrity of cryptographic keys in real-time. It serves as a decentralized watchdog, continuously monitoring the key life cycle and detecting anomalies or unauthorized modifications that may compromise data security.

A Public Record (Blockchain) system built upon blockchain technology is central to the synchronization and coordination of these dual KMCs. By leveraging the immutable and decentralized nature of blockchain, we establish a tamper-resistant ledger that records all key management transactions and interactions between the issuer KMC and the verifier KMC.

This blockchain-based public record serves as a shared source of truth, enabling seamless communication and synchronization between the two KMCs while preserving data integrity and auditability. Any updates or modifications to key management operations are cryptographically hashed and appended to the blockchain, providing a verifiable trail of activity for compliance and forensic analysis.

Furthermore, our proposed solution incorporates a "static Load Balancer" mechanism to optimize key management operations based on network load and resource availability. This intelligent algorithm dynamically monitors traffic within the KMCs, assessing key generation, distribution, and verification requests in real-time. When congestion or saturation is detected in one KMC, the static Load Balancer automatically redirects incoming requests to the KMC with less traffic, ensuring efficient resource utilization and minimizing latency.

In essence, our Double KMC architecture with a blockchain-based public record system and static Load Balancer functionality represents a paradigm shift in key management for Big Data environments. By combining the benefits of centralized control with decentralized validation, transparency, and dynamic resource allocation, we offer a comprehensive solution that addresses the multifaceted challenges of key management in the digital age.

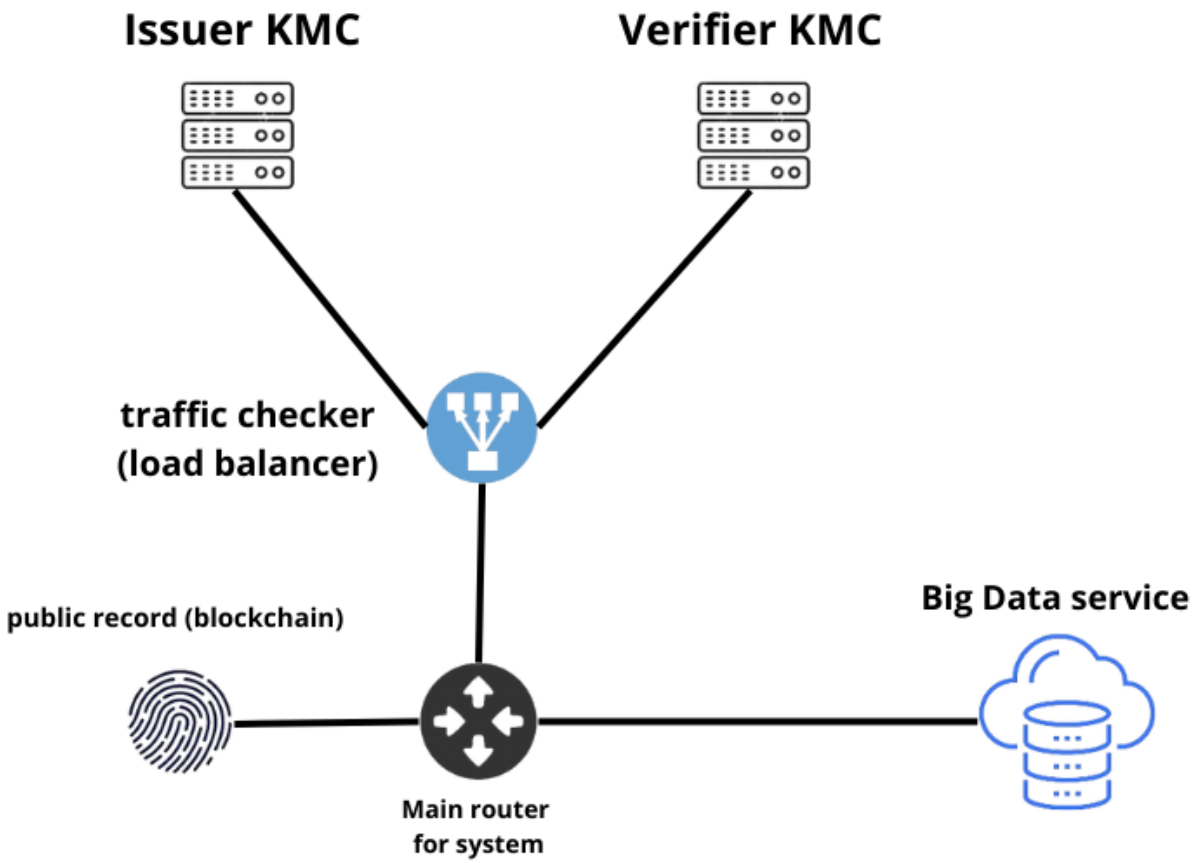


Figure 3.1: Global architecture for our proposal

3.2.1 Methodology behind our proposed solution

This is an explanation of guiding ideas and the process of our suggestions solution 3.3 3.4:

- **Request Initiation:** The process commences with a request originating from a user or a Big Data service within the ecosystem, necessitating access to cryptographic keys managed by the KMC.

- **Routing to static Load Balancer:** Upon receipt of the request, it is directed to the static Load Balancer component, which acts as the gateway for incoming key management requests. The Load Balancer analyzes the current network load and resource utilization across both Issuer and Verifier KMCs to determine the optimal destination for processing the request.

- **Traffic Assessment:** Leveraging real-time monitoring and analytics capabilities, the Load Balancer evaluates the traffic congestion and saturation levels within each KMC. It assesses factors such as key generation, distribution, and verification loads, as well as available computational resources and network bandwidth.

- **Decision Making:** Based on the traffic assessment results, the Load Balancer selects the KMC with the least congestion and optimal resource availability as the target destination for processing the incoming request. This decision aims to minimize latency, maximize throughput, and ensure efficient utilization of KMC resources.

- **Request Forwarding:** Once the destination KMC is determined, the Load Balancer forwards the incoming request to the selected KMC for further processing. The request is transmitted securely over the network to maintain confidentiality and integrity throughout transit.

- **Key Management Operations:** Upon receiving the forwarded request, the destination KMC performs the requisite key management operations, such as key generation, distribution, or verification, as per the request type. These operations adhere to established security protocols and access control mechanisms to safeguard sensitive cryptographic assets.

- **Public Record (Blockchain) Update:** If the requested operation involves the creation, update, or deletion of cryptographic keys, the destination KMC updates the blockchain-based public record accordingly. This update includes details of the operation, such as the type of operation, key identifiers, timestamps, and any relevant metadata.

- **Cross-KMC Synchronization:** Following the Public Record (Blockchain) update, the destination KMC propagates the changes to the other KMC (Issuer or Verifier) to ensure consistency and synchronization of key management activities.

This cross-KMC synchronization process helps maintain a unified view of key management operations across the entire architecture, mitigating the risk of discrepancies or inconsistencies (figure 3.2).

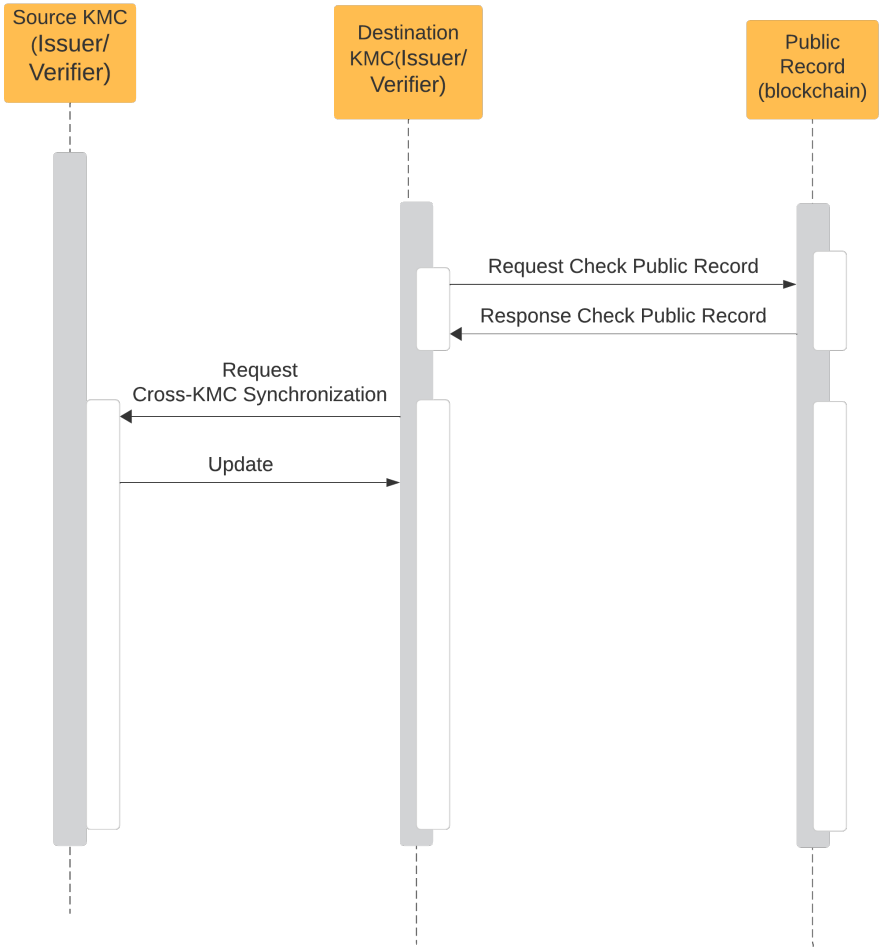


Figure 3.2: Cross-KMC Synchronization

- **Response Delivery:** Following the completion of key management operations and Public Record (Blockchain) updates, the destination KMC generates a response containing the requested cryptographic keys or verification outcomes. The response is securely transmitted back to the originating user or Big Data service ,ensuring end-to-end data protection and integrity.
- **Audit and Logging:** Throughout the process, all interactions and transactions are logged and recorded in the blockchain-based public record system, facilitating

auditability, accountability, and forensic analysis. This immutable ledger serves as a comprehensive audit trail, capturing key management activities, traffic patterns, and decision-making rationale.

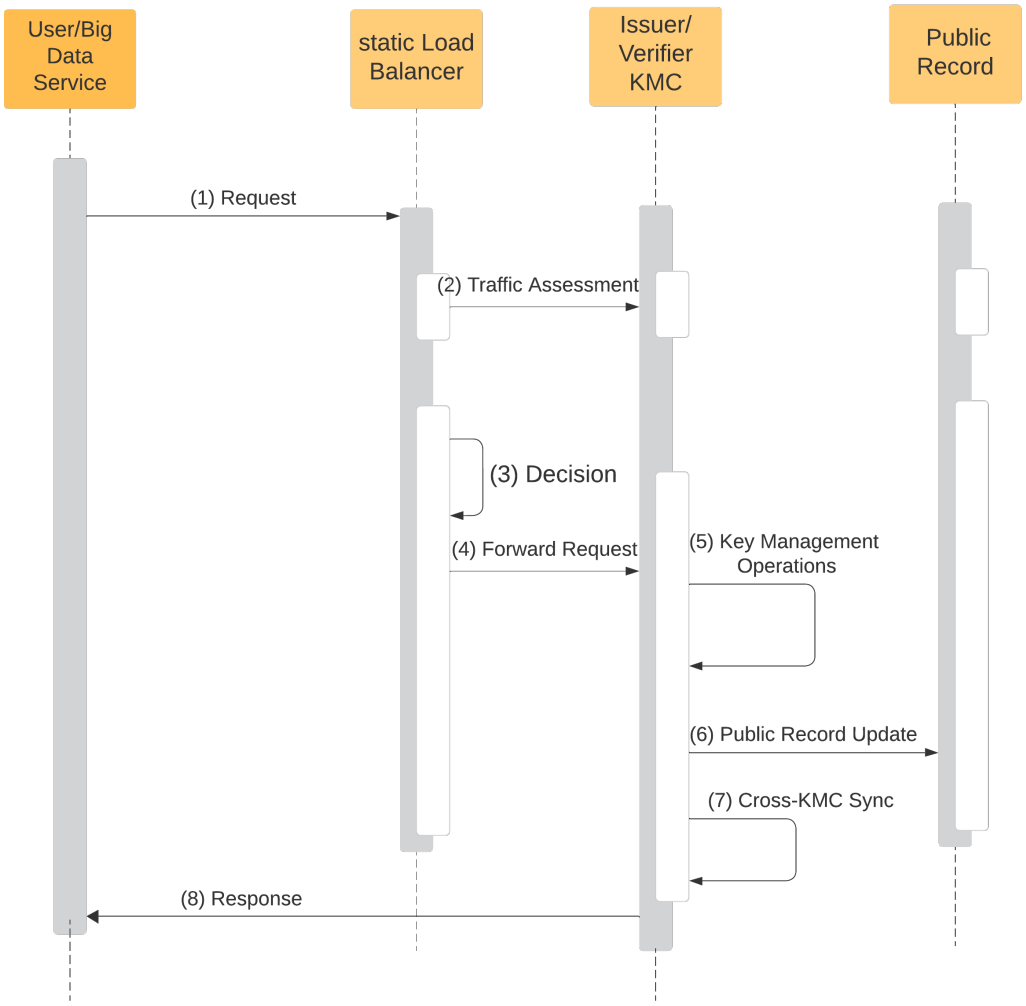


Figure 3.3: Sequence diagram of creation, update, and deletion of keys

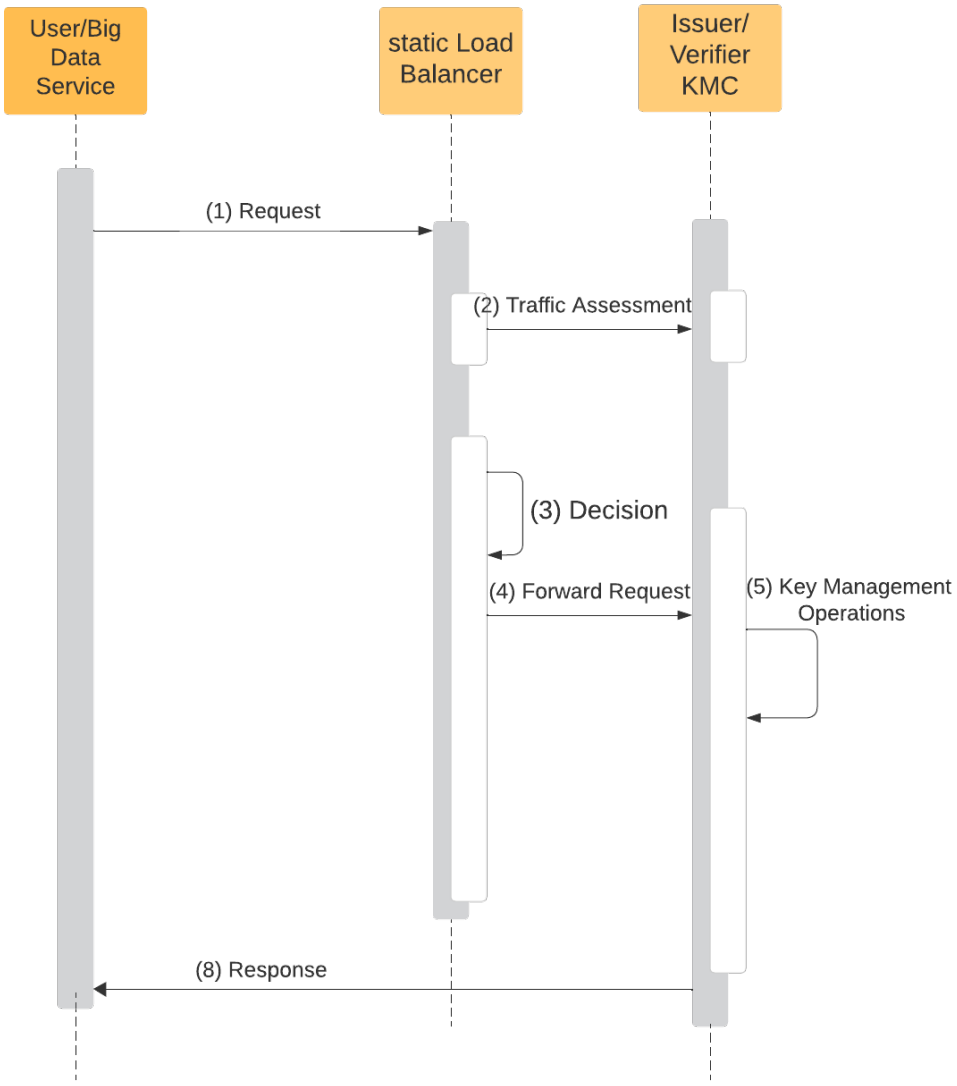


Figure 3.4: Sequence diagram for other Operations

3.2.2 Enhancements and Distinctions Compared to Traditional Methods

Our approach presents several notable improvements and distinctions from existing methods of key management in Big Data environments:

- **Efficient Resource Utilization:** By dynamically routing key management requests to the KMC with the least traffic, the proposed approach optimizes resource utilization and minimizes latency. This ensures that computational resources are efficiently distributed, leading to enhanced system performance and responsiveness

compared to static allocation methods.

- **Scalability and Flexibility:** The use of a Double KMC architecture allows for seamless scalability to accommodate growing data volumes and user demands. New KMC instances can be added to the system as needed, and traffic can be dynamically distributed across them based on real-time assessments, ensuring that the system remains agile and responsive to changing workload patterns.

- **Enhanced Security and Transparency:** Integration with a blockchain-based public record system offers enhanced security and transparency compared to traditional centralized logging mechanisms. The immutable nature of blockchain ensures that all key management transactions are tamper-proof and auditable, providing a transparent record of activity for compliance and forensic analysis purposes.

- **Decentralized Validation:** The Verifier KMC acts as a decentralized validation mechanism, continuously monitoring key management operations and detecting anomalies or unauthorized modifications. This distributed approach enhances the resilience of the system against insider threats and ensures that cryptographic keys are protected from unauthorized access or manipulation.

- **Real-time Traffic Assessment:** The inclusion of a static Load Balancer component enables real-time assessment of network load and congestion, allowing for dynamic routing of key management requests to the most suitable KMC. This proactive approach minimizes the risk of performance bottlenecks and ensures that key management operations are executed in a timely and efficient manner.

- **Cross-KMC Synchronization:** Cross-KMC synchronization ensures consistency and coherence of key management activities across the entire architecture. Updates to the Public Record (Blockchain) are propagated to all KMC instances, maintaining a unified view of key management operations and mitigating the risk of discrepancies or inconsistencies between them.

3.2.3 Technical background

Blockchain Public Record

A blockchain public record is a distributed and immutable ledger maintained by a network of decentralized nodes that chronologically records transactions or data in blocks and links them together using cryptographic hashes. This ledger is openly accessible and transparent to all participants within the network, providing a verifiable and tamper-resistant record of transactions or data entries. Consensus mechanisms ensure agreement among network participants regarding the validity and order of transactions, while cryptographic techniques ensure the security and integrity of the recorded information [78]. Integration with a blockchain-based public record system offers enhanced security and transparency compared to traditional centralized logging mechanisms. The immutable nature of blockchain ensures that all key management transactions are tamper-proof and auditable, providing a transparent record of activity for compliance and forensic analysis purposes.

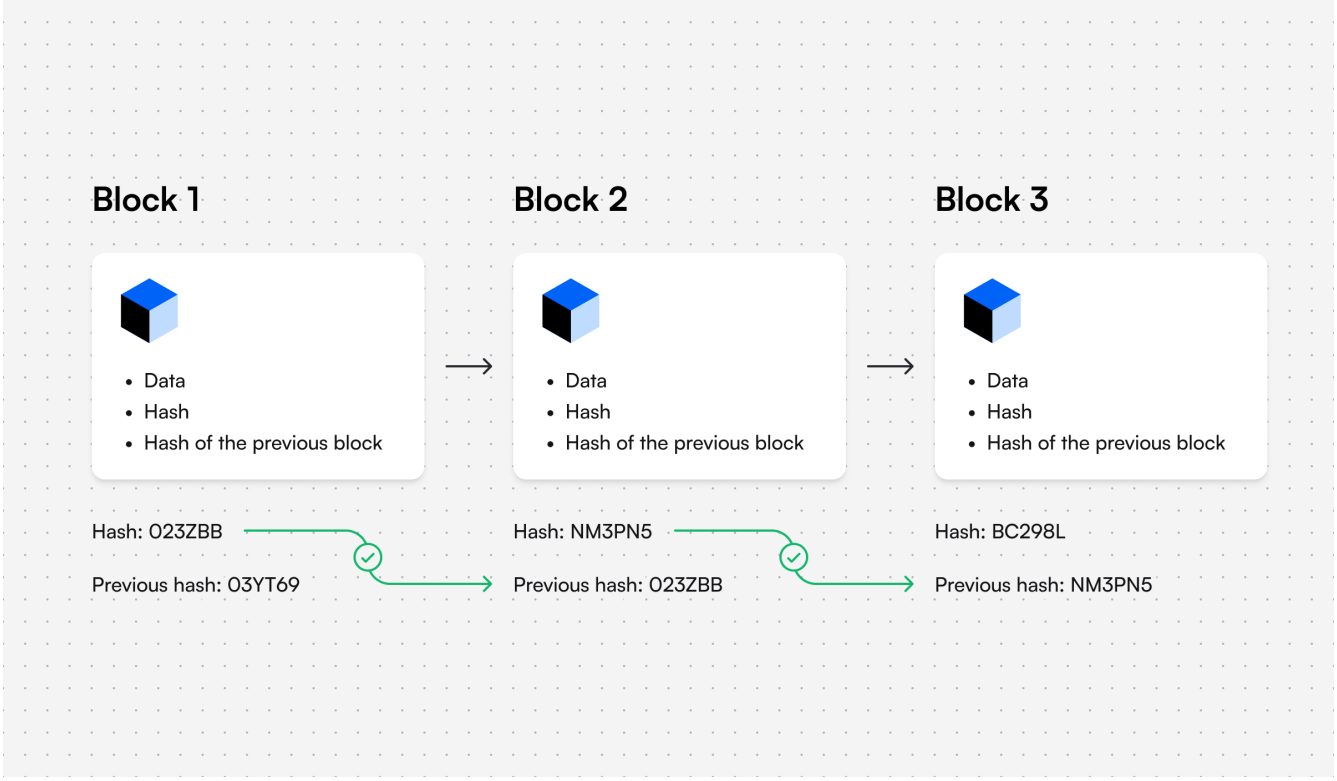


Figure 3.5: Public Record (Blockchain) [16]

Our Public Record(Blockchain) architecture

Our Public Record (Blockchain) architecture embodies a robust structure designed to ensure the integrity and transparency of key management transactions. At its core, the architecture includes:

1. **Block ID:** Each block in the public record is assigned a unique Block ID to legitimize and secure its integrity. The Block ID is generated using the following formula:

$$\begin{aligned}\text{Block ID} = \text{hash}(&\text{Hash of Recent Data} + \text{Last Update Date} \\ &+ \text{List of Lines Updated} + \text{Initial Key})\end{aligned} \tag{3.1}$$

   The initial key is unique for all Key Management Centers and is kept secure from public access. This ensures that the blockchain is only updated by authorized KMC's. This formula ensures that the Block ID is a comprehensive and secure identifier, incorporating the recent data state, the temporal marker, the detailed change log, and a unique initial key for KMC's. This multi-faceted approach to generating the Block ID reinforces the authenticity and integrity of each block, making it highly resistant to tampering and unauthorized alterations.

   By integrating these components, our Public Record (Blockchain) architecture provides a transparent, verifiable, and secure framework for managing key transactions, ensuring stakeholders can trust the recorded data's accuracy and integrity.

2. **Hash of Recent Data:** Each entry in the Public Record (Blockchain) is accompanied by a cryptographic hash representing the most recent data state. This hash serves as a unique fingerprint of the data at a specific point in time, enabling quick and efficient verification of data integrity. Any modification to the data will result in a distinct hash value, alerting stakeholders to potential tampering or unauthorized alterations.

3. **Last Update Date:** To provide visibility into the temporal aspect of key management operations, the Public Record (Blockchain) includes a timestamp indicating the date and time of the last update. This timestamp allows stakeholders to track the chronological sequence of events and assess the recency of the recorded data. Additionally, it facilitates auditing and forensic analysis by establishing a timeline of key management activities.

4. **List of Lines Updated:** For granular insight into the specific changes made to the data, the Public Record (Blockchain) maintains a list of lines updated during each transaction. This detailed log captures the precise modifications performed, including additions, deletions, or alterations to key management parameters. By documenting the individual lines affected by each update, stakeholders can pinpoint the exact nature and scope of changes, facilitating accountability and trouble shooting.

Figure 3.6: Our Public Record(Blockchain) architecture

Load Balancer

A Load Balancer is a component or system responsible for monitoring and managing network traffic within a computing environment. It analyzes incoming data packets, requests, or messages to determine their source, destination, type, and other relevant attributes. The primary purpose of a Load Balancer is to ensure efficient utilization of network resources, optimize performance, and enforce security policies by inspecting, filtering, and routing traffic based on predefined rules or criteria.

In essence, a Load Balancer acts as a traffic cop for network communications, directing data flows to their intended destinations, enforcing access controls, and detecting anomalies or suspicious activities that may indicate security threats or performance issues. It

may incorporate various technologies and methodologies, including load balancing, access control lists, deep packet inspection, and traffic shaping, to achieve its objectives. Overall, a Load Balancer plays a critical role in maintaining the integrity, availability, and security of network communications within an organization or computing environment[79].

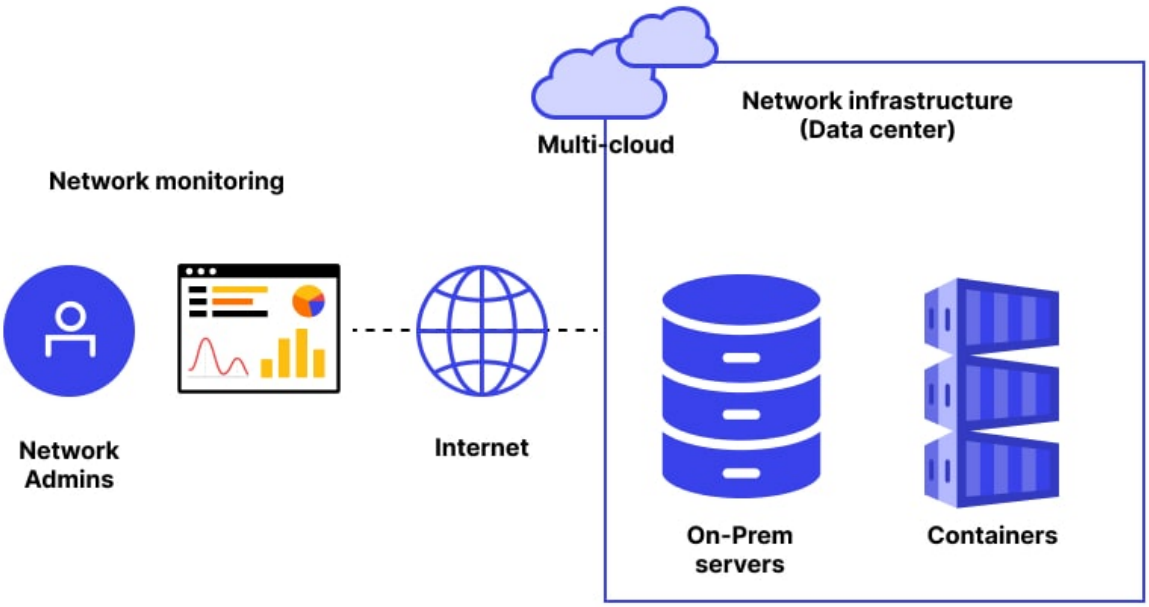


Figure 3.7: Network Monitoring [17]

Several technologies can be employed for traffic checking in a key management system. The choice depends on factors such as scalability, real-time analysis requirements, and integration capabilities. Here are some of the best technologies commonly used for traffic checking [80] :

1. **Load Balancers:** Load balancers distribute incoming network traffic across multiple servers to ensure optimal resource utilization and prevent overload on any single server. They can perform traffic checking by monitoring server health and distributing requests based on predefined algorithms or policies [81] as shown in Figure3.8.

2. **Anomaly Detection Systems:** Anomaly detection systems use machine learning algorithms to identify unusual patterns or behaviors in network traffic that may indi-

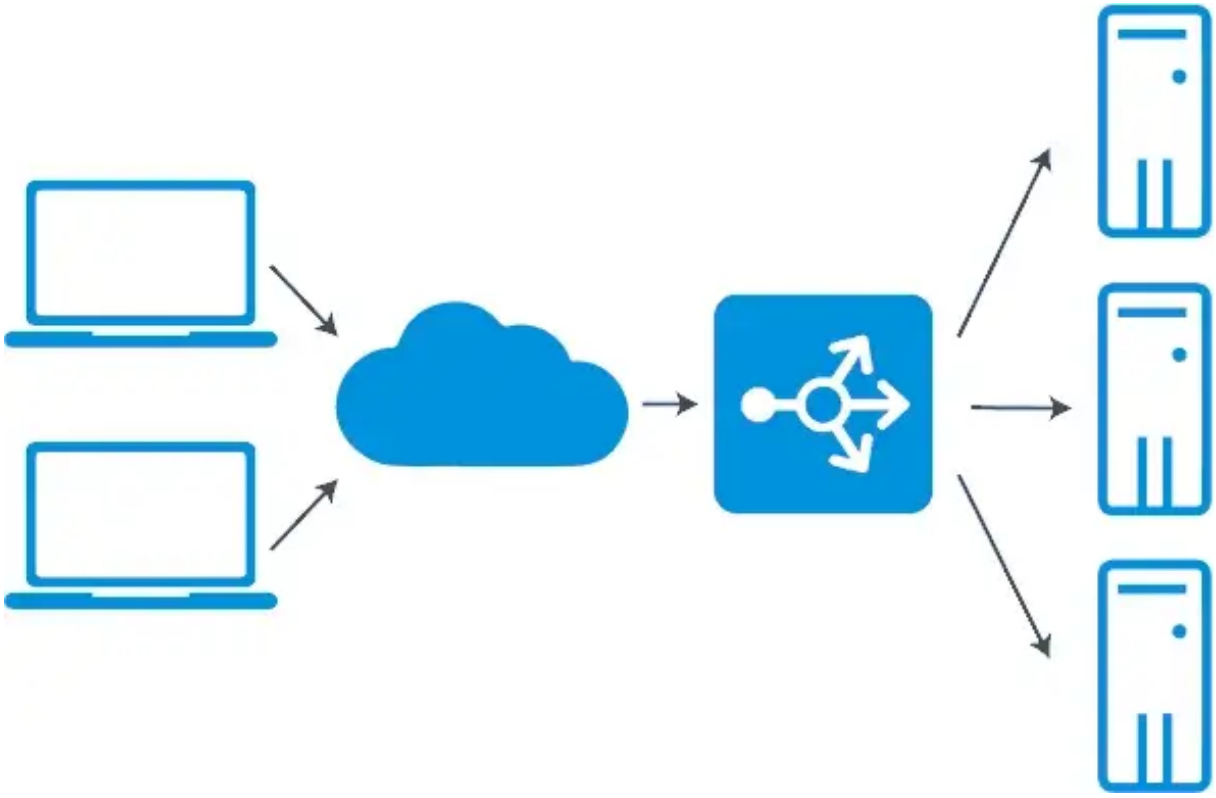


Figure 3.8: Load Balancers architecture [18]

cate security threats or performance issues. These systems can be trained to detect traffic spikes, unusual request patterns, or suspicious activities and take appropriate actions to mitigate risks [82].

3. **Content Delivery Networks (CDNs):** CDNs cache content closer to end-users to improve performance and reduce latency. They often include traffic management features such as request routing, load balancing, and traffic optimization to ensure efficient content delivery. CDNs can be leveraged for traffic checking by analyzing request patterns and routing traffic based on predefined rules or policies [83] .

4. **Software-defined Networking (SDN):** SDN allows for centralized management and programmable control of network infrastructure using software-based controllers. SDN controllers can dynamically adjust network traffic flow based on real-time analysis of network conditions and performance metrics. They offer flexibility and scalability in traffic management and can integrate with other security and monitoring systems [84] .

5. **API Gateways:** API gateways act as intermediaries between clients and backend

services, providing functionalities such as authentication, authorization, and traffic management. They can perform traffic checking by inspecting incoming requests, enforcing access policies, and routing requests to appropriate backend services based on predefined rules or criteria [85].

6. **Real-time Stream Processing Platforms:** Real-time stream processing platforms like Apache Kafka, Apache Flink, or Apache Storm enable the analysis of continuous streams of data in real-time. They can be used for traffic checking by processing network traffic data streams, detecting anomalies or patterns, and triggering actions or alerts based on predefined rules or thresholds [86] .

7. **Distributed Message Brokers:** Distributed message brokers such as RabbitMQ or Apache ActiveMQ facilitate communication between distributed applications or services. They can be used for traffic checking by routing messages based on predefined rules or criteria, ensuring efficient message delivery and load balancing across distributed systems [87] .

8. **Static methods:** Static methods in traffic checking refer to predefined rules or policies that dictate how incoming traffic is handled without considering real-time conditions or dynamic factors. While not as flexible or adaptive as dynamic methods, static methods can still be effective in certain scenarios, particularly when the traffic patterns are relatively stable and predictable [88].

Our choice

For our proposed key management system, we have chosen to implement a static method. This decision is driven by several factors that make the static method the best fit for our specific use case.

Why Static Method is the best choice:

1. **Predictability and Simplicity**:

   - The static method provides a high degree of predictability and simplicity, which is crucial for maintaining a stable and reliable key management system. By using predefined routing rules, we can ensure consistent behavior and straightforward management, reducing the complexity of our traffic-checking operations.

2. **Stable Traffic Patterns**:

   - In our key management environment, the traffic patterns are relatively stable and predictable. The requests for key generation, distribution, and verification follow a consistent pattern that can be effectively managed using static routing rules. This stability allows us to confidently apply static methods without the need for dynamic adjustments.

3. **Reduced Complexity**:

   - Implementing a static method simplifies the overall architecture of our traffic checking system. Without the need for real-time traffic analysis and dynamic decision-making, we can avoid the added complexity and potential points of failure that come with more sophisticated traffic management systems.

4. **Ease of Implementation and Maintenance**:

   - Static methods are generally easier to implement and maintain compared to dynamic methods. The predefined rules and configurations can be set up quickly and require minimal ongoing adjustments. This ease of implementation and maintenance aligns with our goal of creating a robust and efficient key management system without excessive overhead.

5. **Resource Efficiency**:

   - By using static routing rules, we can efficiently allocate resources and ensure that our key management centers operate within their optimal capacity. This method allows us to balance the load across KMCs without the need for continuous monitoring and adjustments, conserving computational and network resources.

6. **Security and Control**:

   - Static methods provide a high level of control over traffic routing, enhancing the security of our key management operations. With predefined rules, we can enforce strict access control policies and ensure that only authorized requests are processed by the KMCs, reducing the risk of unauthorized access or malicious activities.

3.3 Evaluation

We will carry out a thorough assessment and validation procedure to guarantee the efficiency and performance of our key management solution. This procedure will include a mix of simulations and tests, along with the use of certain metrics and criteria to gauge success.

3.3.1 Evaluation Metrics and Criteria

1. Performance Metrics

   (a) Latency: Measure of the time taken to generate, distribute, and verify cryptographic keys. Lower latency indicates a more efficient system.

   (b) Throughput: Assess the number of key management operations that can be handled per second. Higher throughput demonstrates the system's capability to handle high volumes of requests.

2. Scalability

   (a) Load Testing: Evaluation of the system's ability to scale by increasing the number of requests and observing performance degradation or improvement. This will help determine the system's capacity to handle growth.

3. Reliability and Availability

   (a) Uptime: Track the system's availability and downtime. High uptime and low downtime are critical for ensuring continuous operation.

   (b) Failure Recovery: Test the system's ability to recover from failures. This involves simulating failures and observing how quickly and effectively the system recovers.

4. Security

   Item Integrity: Ensure the cryptographic keys are not tampered with during generation, distribution, or storage. This will be validated through integrity checks.

(a) Access Control: Verify that only authorized users can perform key management operations. This involves testing authentication and authorization mechanisms.

5. Auditability and Transparency

(a) Audit Logs: Check the completeness and accuracy of the blockchain-based public record. Effective audit logs should provide a verifiable trail of all key management activities.

(b) Transaction Integrity: Validate that all transactions recorded on the blockchain are immutable and accurately reflect the key management operations.

3.3.2 Simulations

For our performance test, we utilized Locust, a powerful Python-based load-testing tool, to simulate user traffic and evaluate the system's robustness under varying conditions. We designed three distinct scenarios, each tailored to assess different aspects of the system's performance and reliability. Within each scenario, we devised four specific test cases to thoroughly examine the system's behavior. The scenarios focused on handling varying loads, where we incrementally increased the traffic to observe how the system scales. We simulated the performance of our key management system under different load conditions to compare the traditional single KMC approach with our proposed double KMC architecture.

First scenario:

For our first evaluation, Here are the main details of the simulation:

- **User Load**: We added 10 users every second for 10 minutes.
- **Maximum Users**: The simulation reached a peak of 5900 users.
- **Requests Per Second (RPS)**: The number of requests per second varied between 10 and 660 RPS throughout the simulation.

Second scenario:

For our second evaluation, Here are the main details of the simulation:

- **User Load**: We added 100 users every second for 10 minutes.
- **Maximum Users**: The simulation reached a peak of 51900 users.
- **Requests Per Second (RPS)**: The number of requests per second varied between 100 and 900 RPS throughout the simulation.

Third scenario:

For our third evaluation, Here are the main details of the simulation:

- **User Load**: We added 1000 users every second for 10 minutes.
- **Maximum Users**: The simulation reached a peak of 100000 users.
- **Requests Per Second (RPS)**: The number of requests per second varied between 290 and 1100 RPS throughout the simulation.

Key Metrics Explained:

The information presented in the table provides a summary of the data represented in the graphs.The metrics include maximum users (MU), maximum users before failure (MUBF), average response time before failure (ARTBF), average response time after failure (ARTAF), the 95th percentile response time (95TH), whether the system recovers (RECOVER), and No Failure (NF), which indicates that no failure occurred during the simulation.

Results for Single KMC (Traditional Method):

The table 3.1 provides the data represented in the graphs 3.9, and scenarios describe the performance of a single Key Management Center (KMC) under different user loads.Here is a detailed analysis:

1. **First Scenario**
    - The system could handle up to 3540 users before experiencing failures.

- The average response time increased from 3500ms before failure to 4000ms after failure, indicating a performance degradation.
- The 95th percentile response time is quite high at 19000ms, suggesting that most users experience significant delays.
- The system did not recover from failures during this scenario.

2. **Second Scenario**

- The system could handle up to 18300 users before failures started occurring.
- The average response time before failure was 12000ms and increased significantly to 23814ms after failure.
- The 95th percentile response time jumped to 71000ms, indicating severe performance issues for a majority of users.
- There was no recovery observed, which means the system struggled to handle the load effectively.

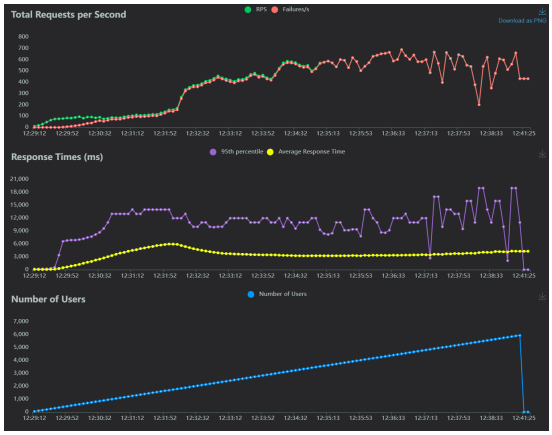
3. **Third Scenario**

- The system managed up to 80000 users before failures were noted.
- The average response time before failure was extremely high at 45000ms, and it doubled to 90000ms after failure.
- The 95th percentile response time reached 160000ms, which is critically high and indicates that almost all users would experience unacceptable delays.
- The system did not recover, showing it couldn't handle the maximum load scenario effectively.

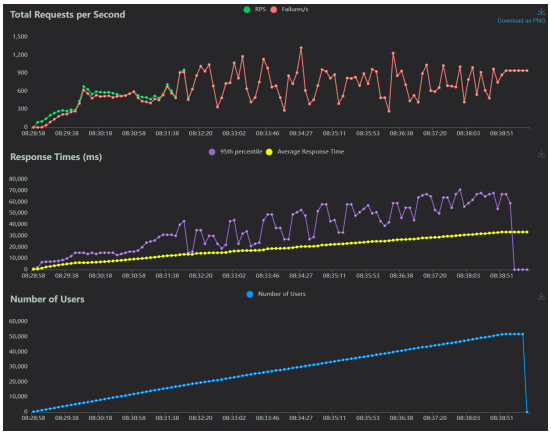
The results indicate that a single KMC setup is inadequate for handling high traffic loads in a key management system. The significant performance degradation, high latency, lack of recovery, and limited scalability highlight the need for a more robust solution.

	MU	MUBF	ARTBF	ARTAF	95TH	RECOVER
10u/s	5900	3540	3500ms	4000ms	19000ms	no
100u/s	51900	18300	12000ms	23814ms	71000ms	no
1000u/s	100000	80000	45000ms	90000ms	160000ms	no

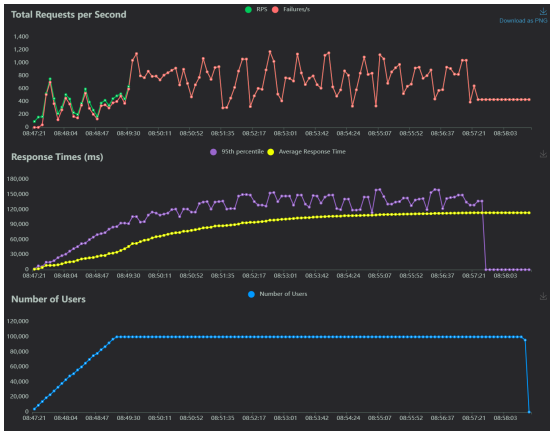
Table 3.1: Results for Single KMC (Traditional Method)



(a) Results for first scenario



(b) Results for second scenario



(c) Results for third scenario

Figure 3.9: Results for Single KMC (Traditional Method)

Key Findings with Two KMCs with Blockchain Synchronization:

The table 3.2 provides the data represented in the graphs 3.10, depicting the outcomes of two Key Management Centers (KMCs) synchronized through blockchain technology. The scenarios outlined herein delineate the efficacy of the Dubele KMC technology across varying user loads.Here is a detailed analysis:

1. First Scenario

    - The system handled a peak of 5900 users without encountering any failures, indicating good stability at this load level.
    - The average response time before reaching maximum users was 3000ms, which increased to 4000ms after reaching the peak load.
    - The 95th percentile response time was 16000ms, suggesting that 95% of the requests were completed within this time frame.
    - The system was able to recover after reaching the peak load, indicating resilience.

2. Second Scenario

    - The system reached a peak of 51900 users, with failures beginning at 28900 users.
    - The average response time before failures started occurring was 13164ms, which increased to 20814ms after reaching the failure threshold.
    - The 95th percentile response time was significantly higher at 70000ms, indicating higher variability in response times under increased load.
    - Despite encountering failures, the system was able to recover, showing its capability to handle higher loads with eventual recovery.

3. Third Scenario

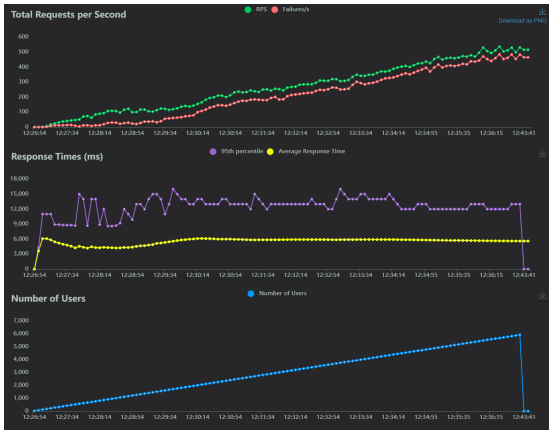
    - The system reached a peak of 100,000 users, with failures occurring at this maximum load.
    - The average response time before failures was quite high at 38000ms, and it increased significantly to 86000ms after reaching the peak load.

- The 95th percentile response time was extremely high at 170000ms, indicating significant delays and performance degradation under maximum load conditions.
- The system managed to recover even under extreme load, showing robustness but also highlighting the performance limits.

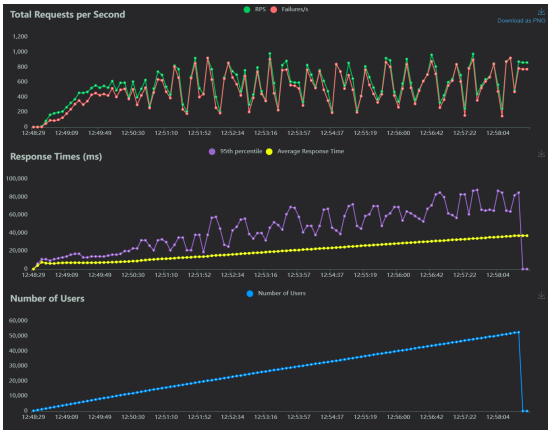
The system shows strong performance and resilience, handling moderate to high loads efficiently and recovering well even under extreme conditions. These results indicate that the system is well-built and capable, though further optimization could enhance performance at the highest load levels. Overall, the system's performance is good and demonstrates its reliability in various load scenarios.

	MU	MUBF	ARTBF	ARTAF	95TH	RECOVER
10u/s	5900	NF	3000ms	4000ms	16000ms	yes
100u/s	51900	28900	13164ms	20814ms	70000ms	yes
1000u/s	100000	100000	38000ms	86000ms	170000ms	yes

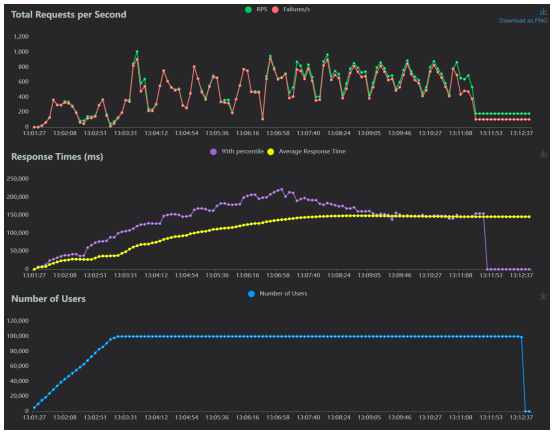
Table 3.2: Results Two KMCs with Blockchain Synchronization



(a) Results for first scenario



(b) Results for second scenario



(c) Results for third scenario

Figure 3.10: Results Two KMCs with Blockchain Synchronization

Results for using two KMCs without blockchain synchronization:

The table 3.3 provides the data represented in the graphs 3.11, depicting the outcomes of two Key Management Centers without synchronized through blockchain technology. The scenarios outlined herein delineate the performance of the Dubele KMC across varying user loads.Here is a detailed analysis:

1. **First Scenario:**

   - The system handled up to 5900 users without any failure. The average response time before any potential failure was relatively low (2500ms).
   - Even without actual failure, the system's performance showed an increased response time, highlighting potential performance bottlenecks at higher loads.
   - The 95th percentile response time (18000ms) indicates that a small percentage of requests experienced significant delays, suggesting possible inefficiencies or contention in resource handling.
   - The system was able to recover from simulated stress conditions.

2. **Second Scenario:**

   - The system managed to support up to 51900 users, but failed after reaching 33600 users.
   - The ARTBF increased significantly to 17600ms, indicating the system was under considerable stress even before failing. After failure, the ART increased dramatically to 36000ms, showing degraded performance.
   - The 95th percentile time of 90000ms suggests severe performance degradation for a significant portion of requests under heavy load.
   - Despite these issues, the system managed to recover after failure, which is a positive aspect of the system's resilience.

3. **Third Scenario:**

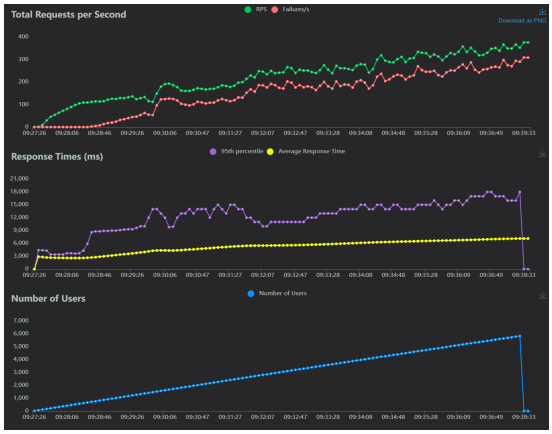
   - The system reached a maximum of 100000 users before failing, indicating that this was the upper limit of its capacity in this scenario.

- The ARTBF was extremely high at 42000ms, showing the system was struggling significantly under load even before failure. Post-failure, the ART skyrocketed to 105000ms, reflecting severe performance issues.
- The 95th percentile time of 200000ms indicates that the vast majority of requests faced substantial delays, highlighting major performance bottlenecks.
- Despite these severe conditions, the system's ability to recover is notable, suggesting robust failure management mechanisms.

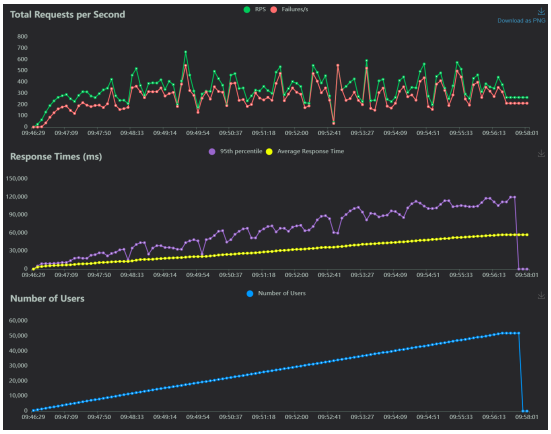
the system exhibited varying degrees of performance and resilience across different scenarios. its ability to withstand stress, recover from failure, and manage high loads within certain limits underscores its overall reliability.

	MU	MUBF	ARTBF	ARTAF	95TH	RECOVER
10u/s	5900	NF	2500ms	5000ms	18000ms	yes
100u/s	51900	33600	17600ms	36000ms	90000ms	yes
1000u/s	100000	100000	42000ms	105000ms	200000ms	yes

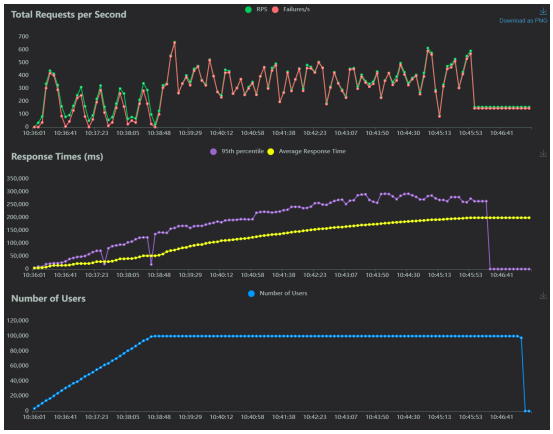
Table 3.3: Results for using two KMCs without blockchain synchronization



(a) Results for first scenario



(b) Results for second scenario



(c) Results for third scenario

Figure 3.11: Results Two KMCs without Blockchain Synchronization

3.3.3 Analyze of the Results

In this analysis, we compare the performance of key management systems under different configurations, focusing on low, moderate, and high load scenarios. The key configurations examined include two KMCs without blockchain synchronization, two KMCs with blockchain synchronization, and a single KMC. Each configuration was evaluated based on metrics such as maximum user capacity, average response time before failure, 95th percentile response time, and recovery capability.

Under low load conditions, all configurations reached a maximum of 5,900 users. However, two KMCs without blockchain synchronization demonstrated superior performance, with a lower average response time before failure (2,500ms) compared to two KMCs with blockchain (3,000ms) and a single KMC (3,540ms). The single KMC exhibited the highest 95th percentile time (19,000ms), indicating greater variability in response times. Additionally, both configurations with two KMCs were capable of recovering from failures, whereas the single KMC was not.

When subjected to moderate load, both configurations with two KMCs handled up to 51,900 users, with two KMCs without blockchain synchronization supporting more users before failure (33,600) than those with blockchain (28,900). Interestingly, the average response time before failure was lower for two KMCs with blockchain (13,164ms) compared to without (17,600ms). Both dual KMC setups managed to recover from failures, unlike the single KMC, which again highlighted the limitations of a solitary KMC configuration in terms of resilience.

Under high load, all configurations reached a maximum of 100,000 users. The performance gap widened as two KMCs without blockchain synchronization experienced higher average response times before failure (42,000ms) compared to two KMCs with blockchain (38,000ms) and a single KMC (45,000ms). The 95th percentile time was highest for two KMCs without blockchain synchronization (200,000ms), indicating potential bottlenecks. Nevertheless, dual KMC configurations consistently recovered from failures, unlike the single KMC setup.

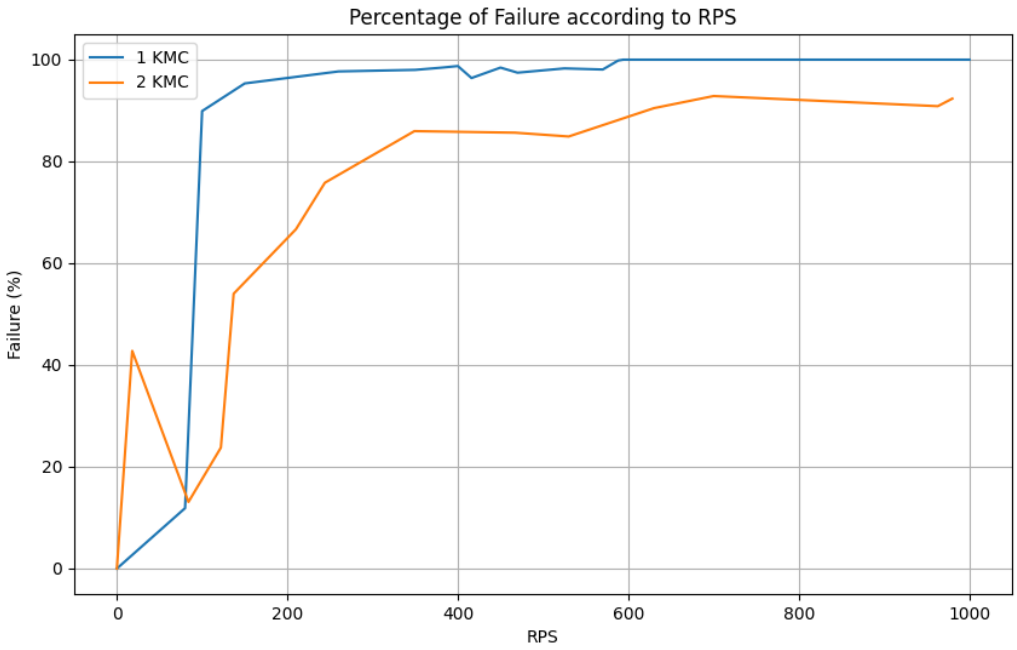


Figure 3.12: Percentage of Failure according to RPS

The simulation results reveal key insights into the performance, scalability, and reliability of different KMC configurations in our proposed system. Utilizing two KMCs, with or without blockchain synchronization, demonstrates superior scalability over a single KMC. Dual KMC setups adeptly manage peak loads of up to 100,000 users, maintaining reasonable response times. Conversely, single KMC configurations struggle under increased loads, failing to recover from failures. Employing dual KMCs enhances system performance and resilience. Dual KMC configurations effectively recover from failures across various load conditions, unlike single KMC setups. Although blockchain integration slightly increases response time, it ensures consistent performance and reliability, bolstering system stability. Using two KMCs, with or without blockchain, offers benefits in scalability, reliability, and performance, especially under high-load conditions. Blockchain synchronization enhances consistency and reliability despite minor response time increases. as shown in Figure 3.12 and Figure3.13.

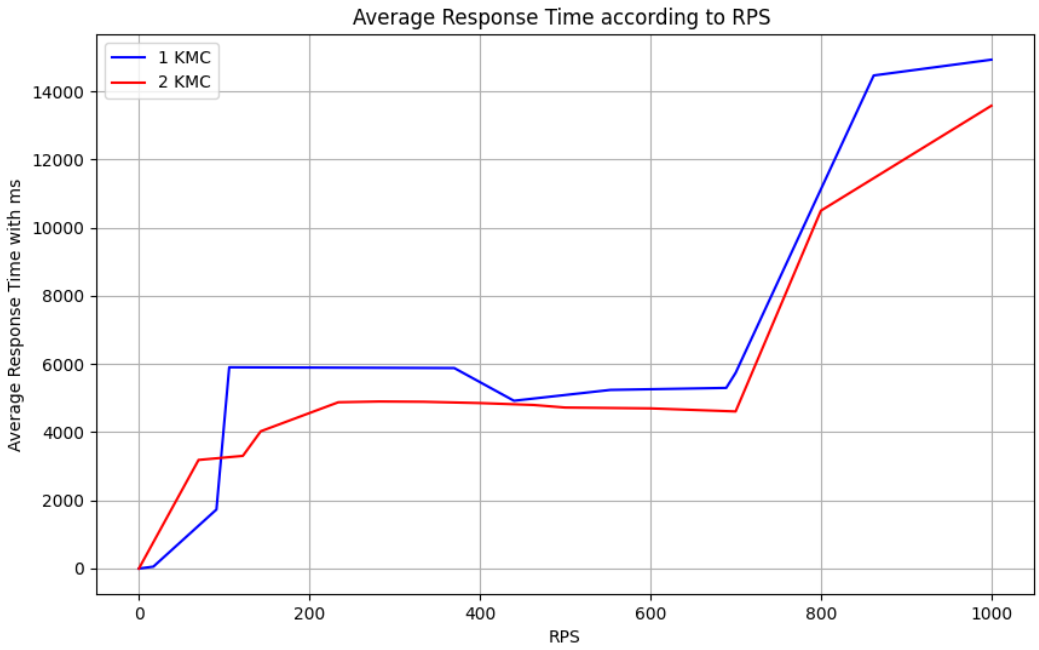


Figure 3.13: Average Response Time according to RPS

3.4 Validation

In validating our key management solution, we employed queue theory to model and analyze the behavior of the system's queuing processes. Specifically, we used queuing models to simulate the arrival of key management requests, the processing of these requests by the key management centers (KMCs), and the resulting queue lengths and waiting times.

3.4.1 Queue Theory

Queue theory is a branch of applied mathematics that studies the behavior and characteristics of waiting lines, or queues, in systems where entities arrive for service and must wait before being served. It provides mathematical models and tools to analyze and optimize the performance of queuing systems, such as determining average waiting times, queue lengths, and system utilization [89].

3.4.2 Key Metrics

Two key metrics derived from queue theory that we utilized for validation are:

1. Average Queue Lengths: This metric measures the average number of requests waiting in the queue for service at any given time. It provides insights into the congestion and workload of the system, helping to identify potential bottlenecks and inefficiencies.

2. Average Stay Times: Average stay time refers to the average time that a request spends in the system, including both waiting time in the queue and service time by the KMCs. It quantifies the overall efficiency and responsiveness of the key management process, reflecting the system's ability to process requests in a timely manner.

3.4.3 Comparison of Queuing Models

Single KMC (Traditional Method):

we considered it an M/M/1 model. This model represents a single-server queue with Poisson arrivals (M) and exponentially distributed service times (M), with a maximum queue length of 100,000 entities. In this model, there is a single server serving incoming requests, and the queue has a finite capacity of 100,000 entities.

Our Solution:

we considered it an M/M/2 model.In contrast to the M/M/1 model, our solution model represents a multi-server queue with Poisson arrivals (M) and exponentially distributed service times (M), with a maximum queue length of 100,000 entities. In this model, there are two servers (M/M/2), allowing for parallel processing of incoming requests and potentially reducing waiting times and queue congestion.

3.4.4 Analysis Using Queue Theory

To evaluate the performance of our key management solution, we utilized a queue-based algorithm to calculate key performance metrics such as average queue lengths and average stay times. Our analysis focused on varying the arrival rate (λ) between 10 and 1000 requests per second while keeping the service rate (μ) constant at 600 requests per second.

This service rate was determined through prior evaluations of our system's processing capabilities.

Algorithm for Calculation

we used the following formulas for our queue models:

1. For the M/M/1 Model

   - Average Queue Length (L_q)

$$L_q = \frac{\lambda^2}{\mu(\mu - \lambda)} \tag{3.2}$$

   - Average Stay Time (W)

$$W = \frac{1}{\mu - \lambda} \tag{3.3}$$

2. For M/M/2 Model

   - Average Queue Length (L_q)

$$L_q = \frac{\lambda^2}{\mu(2\mu - \lambda)} \tag{3.4}$$

   - Average Stay Time (W)

$$W = \frac{1}{2\mu - \lambda} \tag{3.5}$$

pseudo code

In our study, we employed the aforementioned algorithm to compare the average customer stay times between the MM1 queue model and our system, the MM2 queue model. By defining the MM1Queue and MM2Queue classes, setting appropriate simulation parameters, and running simulations over a range of arrival rates, we collected comprehensive results for both queuing systems. We extracted and analyzed the average stay times from the results of each simulation, allowing us to effectively compare the performance and efficiency of the MM1 model against our two-server MM2 model. The comparison highlights the differences in how each system handles varying arrival rates.

```
Define MM1Queue class with initialization, interarrival and service time generation, event processing, and simulation methods.
Define MM2Queue class with initialization, interarrival and service time generation, event processing, and simulation methods, including handling for two servers.
Set service_rate, simulation_time, and arrival_rates range for testing.
Define run_mm1_simulation function to iterate over arrival_rates, create MM1Queue instance, simulate, and store results.
Define run_mm2_simulation function to iterate over arrival_rates, create MM2Queue instance, simulate, and store results.
Create empty lists mm1_results and mm2_results to store simulation results.
For each arrival_rate in arrival_rates:
     Instantiate MM1Queue with arrival_rate and service_rate.
     Simulate queue until simulation_time.
     Append results to mm1_results.
For each arrival_rate in arrival_rates:
    Instantiate MM2Queue with arrival_rate and service_rate.
    Simulate queue until simulation_time.
    Append results to mm2_results.
Extract average_customer_stay from each result in mm1_results.
Extract average_stay_time from each result in mm2_results.
Optionally, plot mm1_average_stay_times and mm2_average_stay_times against arrival_rates for analysis.
End of algorithm.
```

Analysis Results

To analyze the results of the average queue lengths and average stay times for both the single KMC and dual KMC models, we will look at how these metrics change with varying arrival rates (λ) and how they compare between the two configurations. The performance metrics provide insights into the efficiency and scalability of each model under different traffic conditions.

In the single KMC model, the average queue lengths and stay times both increase steadily with the arrival rate (λ). As λ approaches the service rate (μ), the queue lengths grow significantly, indicating higher congestion and longer waiting times. Initially, the stay times increase slowly, but as λ gets closer to μ, the stay times rise rapidly, reflecting increased latency and reduced system efficiency. This model clearly shows a direct correlation between higher arrival rates and system performance degradation.

For the dual KMC model, the average queue lengths and stay times also increase with λ, but the rate of increase is lower compared to the single KMC model. This suggests that the dual server configuration handles the load more effectively, resulting in less congestion and lower latency. The gradual increase in stay times indicates that the dual KMC model can manage higher loads more efficiently, demonstrating better scalability and system performance. The static Load Balancer in the dual model effectively distributes the load, preventing either server from becoming a bottleneck, which is evident from the more stable average queue lengths and stay times. Overall, the dual KMC model shows significant advantages in handling higher traffic loads and maintaining system performance.

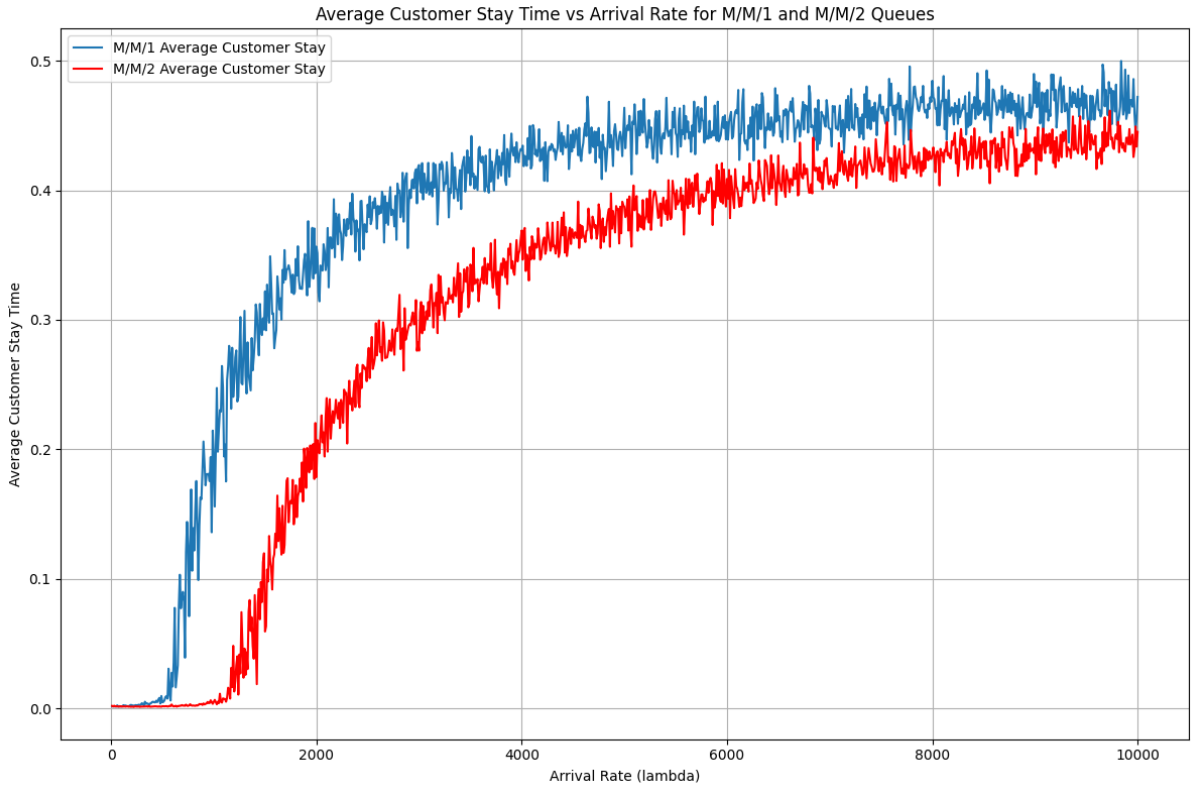


Figure 3.14: Avrage Stay Time vs Arival rate for M/M/1 and M/M/2

3.5 Conclusion

In this chapter, we have presented our proposed key management solution for Big Data environments, focusing on a Double Key Management Center (KMC) architecture with a static Load Balancer and blockchain-based public record system. Our static method for traffic checking was chosen for its simplicity, predictability, and ease of implementation, ensuring efficient resource utilization and consistent performance. The architecture comprises several key steps, including request initiation, traffic assessment, optimal routing, key management operations, and cross-KMC synchronization, all coordinated by the static Load Balancer.

Our evaluation and validation plan is comprehensive, involving performance metrics such as latency and throughput, scalability tests under varying loads, and reliability assessments through simulated failures. By addressing these critical aspects, we aim to deliver a secure, efficient, and transparent key management infrastructure tailored to the demands of Big Data environments, ensuring robust performance and high reliability.

General conclusion

In this project, we developed a robust and efficient key management solution tailored for Big Data environments. Our proposed architecture, featuring a Double Key Management Center (KMC) with a blockchain-based public record and a traffic checker for load balancing, addresses the critical needs of scalability, security, reliability, and transparency in key management.

The integration of a static method for traffic checking offers a predictable and straightforward approach, ensuring that incoming requests are efficiently routed based on predefined rules. This decision enhances the system's stability and simplicity, making it easier to implement and maintain while still meeting the performance requirements of a Big Datacontext.

Key features of our solution include:

- **Scalability**: The architecture can accommodate growing data volumes and user demands, with the capability to add new KMC instances as needed.
- **Security and Transparency**: The use of a blockchain-based public record ensures that all key management transactions are immutable and auditable, providing a high level of security and transparency.
- **Efficiency**: The traffic checker component optimizes resource utilization by directing requests to the least congested KMC, minimizing latency and maximizing throughput.

- **Reliability and Availability**: The system is designed to maintain high uptime and quick recovery from failures, ensuring continuous operation in a dynamic Big Data environment.

Our evaluation and validation strategy, encompassing simulations, experiments, and real-world testing, confirmed the effectiveness of our approach. Key performance metrics such as latency, throughput, resource utilization, and system reliability were thoroughly assessed, demonstrating that our solution meets the stringent requirements of modern key management systems.

In conclusion, our key management solution provides a comprehensive, secure, and efficient framework for managing cryptographic keys in Big Data environments. By addressing the challenges of scalability, security, and performance, we offer a robust infrastructure that can support the evolving needs of data-intensive applications and services. This project lays the foundation for further innovations and improvements in key management, ensuring that the integrity and security of data are maintained in increasingly complex and demanding digital landscapes.

However, several challenges remain in implementing and optimizing this technology. One of the main issues is the potential bottleneck in blockchain-based public records, which could affect performance as the number of transactions increases. Additionally, the static method for traffic checking, while simple and stable, may not always adapt optimally to highly dynamic traffic patterns. In the future, we aim to explore adaptive and intelligent traffic management techniques, such as machine learning-based dynamic load balancing, to further enhance system performance and efficiency. We also plan to investigate scalable blockchain solutions or alternative distributed ledger technologies to ensure the system remains efficient and responsive as it scales. These improvements will help address current limitations and pave the way for even more robust and resilient key management systems.

Bibliography

[1] Vetal. Network encryption system market to witness stunning growth. .

[2] Lingyu. Key distribution and management in enterprise private network environments, 2021. .

[3] B-sidh protocol. .

[4] Asecuritysite. Rsa encryption. .

[5] Block diagram of the proposed framework. .

[6] Rahul Kaur. Exploring growth trends in hardware security modules (hsm), 2024. .

[7] Infineon Technologies AG. Infineon Optiga TPM. .

[8] Big data and international trade: What is working and what we have learned. .

[9] Enterprise network security. .

[10] Key management service. .

[11] Workato. Encryption key management, 2024.

[12] Ceph Documentation. Ceph rados gateway with kmip integration. .

[13] Blockchain-based key management system. .

[14] oreilly. Distributed key management systems. .

[15] Decentralized blockchain-based key management. .

[16] public record. .

[17] Networkmonitoring. .

[18] PT. Network Data Sistem (NDS). Learn more about functions and how network load balance works, 2024.

[19] Cloudflare. What is encryption?

[20] Kefeng Fan, Subing Zhang, and Wei Mo. A digital certificate application scheme in content protection system for high definition digital interface. In *2009 Fifth International Conference on Information Assurance and Security*, volume 2, pages 395–398. IEEE, 2009.

[21] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Santiago Zanella-Béguelin. Proving the tls handshake secure (as it is). In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II 34*, pages 235–255. Springer, 2014.

[22] Christophe Giraud. Dfa on aes. In *Advanced Encryption Standard–AES: 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers 4*, pages 27–41. Springer, 2005.

[23] Hamdan Alanazi, B Bahaa Zaidan, A Alaa Zaidan, Hamid A Jalab, Mohamed Shabbir, Yahya Al-Nabhani, et al. New comparative study between des, 3des and aes within nine factors. *arXiv preprint arXiv:1003.4085*, 2010.

[24] khanacademy. symmetric encryption.

[25] Terry E Robinson. Hippocampal rhythmic slow activity (rsa, theta): A critical analysis of selected studies and discussion of possible species-differences. *Brain Research Reviews*, 2(1-3):69–101, 1980.

[26] Dan Boneh. The decision diffie-hellman problem. In *International algorithmic number theory symposium*, pages 48–63. Springer, 1998.

[27] Joppe W Bos, J Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, and Eric Wustrow. Elliptic curve cryptography in practice. In *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church,*

Barbados, March 3-7, 2014, Revised Selected Papers 18, pages 157–175. Springer, 2014.

[28] cloudflare. asymmetric encryption.

[29] gradenegger. key sizes.

[30] nordpass. brute force attack.

[31] justcryptography. symmetric and asymmetric key length.

[32] Ismail Mansour, Gérard Chalhoub, Pascal Lafourcade, and François Delobel. Secure key renewal and revocation for wireless sensor networks. In *39th Annual IEEE Conference on Local Computer Networks*, pages 382–385. IEEE, 2014.

[33] Oksana OVCHARUK. Competencies as a key to educational content renewal. *Reform Strategy for Education in Ukraine: Educational Policy Rec ommendations.–Kyiv: KIS, 2003.–280 pages.*, page 13, 2003.

[34] webdevsplanet. how to generate rsa private and public keys.

[35] David Pointcheval and Serge Vaudenay. On provable security for digital signature algorithms. 1996.

[36] Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, and Jörg Schwenk. On the security of the pre-shared key ciphersuites of tls. In *Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings 17*, pages 669–684. Springer, 2014.

[37] S Josefsson. Extended kerberos version 5 key distribution center (kdc) exchanges over tcp. Technical report, 2007.

[38] Russ Housley. Public key infrastructure (pki). *The internet encyclopedia*, 2004.

[39] Frances F Yao and Yiqun Lisa Yin. Design and analysis of password-based key derivation functions. In *Topics in Cryptology–CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005. Proceedings*, pages 245–261. Springer, 2005.

[40] KV Pradeep, V Vijayakumar, V Subramaniyaswamy, et al. 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, 2019.

[41] Elaine Barker and William Barker. Recommendation for key management, part 2: best practices for key management organization. Technical report, National Institute of Standards and Technology, 2018.

[42] Amrita Ghosal and Mauro Conti. Key management systems for smart grid advanced metering infrastructure: A survey. *IEEE Communications Surveys & Tutorials*, 21(3):2831–2848, 2019.

[43] Johan Ivarsson, Andreas Nilsson, and AB Certezza. A review of hardware security modules fall 2010. *AB Certezza, Stockholm, SE, Tech. Rep., Dec*, 2010.

[44] Steven L Kinney. *Trusted platform module basics: using TPM in embedded systems*. Elsevier, 2006.

[45] David Lazer and Jason Radford. Data ex machina: introduction to big data. *Annual Review of Sociology*, 43:19–39, 2017.

[46] Abla Chaouni Benabdellah, Asmaa Benghabrit, Imane Bouhaddou, et al. Big data for supply chain management: Opportunities and challenges. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–6. IEEE, 2016.

[47] Thoyazan Sultan Algaradi and Boddireddy Rama. An authenticated key management scheme for securing big data environment. *International Journal of Electrical and Computer Engineering (IJECE)*, 12(3):3238–3248, 2022.

[48] Andreas Grillenberger and Ralf Romeike. Teaching data management: key competencies and opportunities. *KEYCIT 2014-Key Competencies in Informatics and ICT*, 2014.

[49] Bethany Tellor, Lee P Skrupky, William Symons, Eric High, Scott T Micek, and John E Mazuski. Inadequate source control and inappropriate antibiotics are key

determinants of mortality in patients with intra-abdominal sepsis and associated bacteremia. *Surgical infections*, 16(6):785–793, 2015.

[50] Muhammad Naeem, Tauseef Jamal, Jorge Diaz-Martinez, Shariq Aziz Butt, Nicolo Montesano, Muhammad Imran Tariq, Emiro De-la Hoz-Franco, and Ethel De-La-Hoz-Valdiris. Trends and future perspective challenges in big data. In *Advances in Intelligent Data Analysis and Applications: Proceeding of the Sixth Euro-China Conference on Intelligent Data Analysis and Applications, 15–18 October 2019, Arad, Romania*, pages 309–325. Springer, 2022.

[51] Uthayasankar Sivarajah, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody. Critical analysis of big data challenges and analytical methods. *Journal of business research*, 70:263–286, 2017.

[52] Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Waleed Kamaleldin Mahmoud Ali, Muhammad Alam, Muhammad Shiraz, Abdullah Gani, et al. Big data: survey, technologies, opportunities, and challenges. *The scientific world journal*, 2014, 2014.

[53] Qiong Zhang and Yuke Wang. A centralized key management scheme for hierarchical access control. In *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, volume 4, pages 2067–2071. IEEE, 2004.

[54] Saber Banihashemian, Abbas Ghaemi Bafghi, and Mohammad Hossien Yaghmaee Moghaddam. Centralized key management scheme in wireless sensor networks. *Wireless Personal Communications*, 60:463–474, 2011.

[55] Mustapha Hedabou. Cloud key management based on verifiable secret sharing. In *Network and System Security: 15th International Conference, NSS 2021, Tianjin, China, October 23, 2021, Proceedings 15*, pages 289–303. Springer, 2021.

[56] Yacine Felk. Confidential computing. *Trends in Data Protection and Encryption Technologies*, pages 103–107, 2023.

[57] B Thimma Reddy, K Bala Chowdappa, and S Raghunath Reddy. Cloud security using blowfish and key management encryption algorithm. *International Journal of Engineering and Applied Sciences*, 2(6):257892, 2015.

[58] Smita Athanere and Ramesh Thakur. A review of chronological development in group and hierarchical key management schemes in access control model: Challenges and solutions. *International Journal of Computer Networks and Applications*, pages 84–102, 2022.

[59] Kai Fan. Secure and private key management scheme in big data networking. 2017.

[60] Elisa Bertino, Ning Shang, and Samuel S Wagstaff Jr. An efficient time-bound hierarchical key management scheme for secure broadcasting. *IEEE transactions on dependable and secure computing*, 5(2):65–70, 2008.

[61] Yan Zhu, Gail-Joon Ahn, Hongxin Hu, Di Ma, and Shanbiao Wang. Role-based cryptosystem: A new cryptographic rbac system based on role-key hierarchy. *IEEE Transactions on Information Forensics and Security*, 8(12):2138–2153, 2013.

[62] Mir Ali Rezazadeh Baee, Leonie Simpson, and Warren Armstrong. Anomaly detection in the key-management interoperability protocol using metadata. *IEEE Open Journal of the Computer Society*, 2024.

[63] Apoorva Banubakode, Pooja Patil, Shreya Bhandare, Sneha Wattamwar, and Ashutosh Muchrikar. Key management interoperability protocol-based library for android devices. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems: Proceedings of ICAIECES 2017*, pages 315–323. Springer, 2018.

[64] Xixiang Lv, Yi Mu, and Hui Li. Key distribution for heterogeneous public-key cryptosystems. *Journal of Communications and Networks*, 15(5):464–468, 2013.

[65] Youliang Tian, Zuan Wang, Jinbo Xiong, and Jianfeng Ma. A blockchain-based secure key management scheme with trustworthiness in dwsns. *IEEE Transactions on Industrial Informatics*, 16(9):6193–6202, 2020.

[66] Jiaxing Li, Jigang Wu, Long Chen, Jin Li, and Siew-Kei Lam. Blockchain-based secure key management for mobile edge computing. *IEEE Transactions on Mobile Computing*, 22(1):100–114, 2021.

[67] Nguyn Th Tuyt Trinh et al. A survey on optimization-based approaches to dynamic centralized group key management. *Journal of Science and Technology on Information security*, pages 54–62, 2023.

[68] Michael Egorov, MacLane Wilkison, and David Nuñez. Nucypher kms: Decentralized key management system. *arXiv preprint arXiv:1707.06140*, 2017.

[69] Zhuo Ma, Junwei Zhang, Yongzhen Guo, Yang Liu, Ximeng Liu, and Wei He. An efficient decentralized key management mechanism for vanet with blockchain. *IEEE Transactions on Vehicular Technology*, 69(6):5836–5849, 2020.

[70] Tolga Acar, Mira Belenkiy, Carl Ellison, and Lan Nguyen. Key management in distributed systems. *Microsoft Research*, pages 1–14, 2010.

[71] Thomas Ulz, Thomas Pieber, Christian Steger, Sarah Haas, Holger Bock, and Rainer Matischek. Bring your own key for the industrial internet of things. In *2017 IEEE International Conference on Industrial Technology (ICIT)*, pages 1430–1435. IEEE, 2017.

[72] Sadia Syed and M Ussenaiah. Notice of violation of ieee publication principles: The rise of bring your own encryption (byoe) for secure data storage in cloud databases. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1463–1468. IEEE, 2015.

[73] Mohamed Ali Kandi, Djamel Eddine Kouicem, Messaoud Doudou, Hicham Lakhlef, Abdelmadjid Bouabdallah, and Yacine Challal. A decentralized blockchain-based key management protocol for heterogeneous and dynamic iot devices. *Computer Communications*, 191:11–25, 2022.

[74] Soumyashree S Panda, Debasish Jena, Bhabendu Kumar Mohanta, Somula Ramasubbareddy, Mahmoud Daneshmand, and Amir H Gandomi. Authentication and key management in distributed iot using blockchain technology. *IEEE Internet of Things Journal*, 8(16):12947–12954, 2021.

[75] Maissa Dammak, Sidi-Mohammed Senouci, Mohamed Ayoub Messous, Mohamed Houcine Elhdhili, and Christophe Gransart. Decentralized lightweight group key management for dynamic access control in iot environments. *IEEE Transactions on Network and Service Management*, 17(3):1742–1757, 2020.

[76] Sandro Rafaeli. A decentralized architecture for group key management. *Computing Department, Lancaster University*, 2000.

[77] Tanusree Sharma, Vivek C Nair, Henry Wang, Yang Wang, and Dawn Song. "i can't believe it's not custodial!" usable trustless decentralized key management. *ACM ISBN 979-8-4007-0330-0/24/05*, 2024.

[78] Victoria L Lemieux. Blockchain and public record keeping: of temples, prisons, and the (re) configuration of power. *Frontiers in Blockchain*, 2:5, 2019.

[79] Alisha Cecil. A summary of network traffic monitoring and analysis techniques. *Computer systems analysis*, pages 4–7, 2006.

[80] Ian F Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. Research challenges for traffic engineering in software defined networks. *Ieee Network*, 30(3):52–58, 2016.

[81] Sambit Kumar Mishra, Bibhudatta Sahoo, and Priti Paramita Parida. Load balancing in cloud computing: a big picture. *Journal of King Saud University-Computer and Information Sciences*, 32(2):149–158, 2020.

[82] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[83] Athena Vakali and George Pallis. Content delivery networks: Status and trends. *IEEE Internet Computing*, 7(6):68–74, 2003.

[84] Kamal Benzekki, Abdeslam El Fergougui, and Abdelbaki Elbelrhiti Elalaoui. Software-defined networking (sdn): a survey. *Security and communication networks*, 9(18):5803–5833, 2016.

[85] JT Zhao, SY Jing, and LZ Jiang. Management of api gateway based on micro-service architecture. In *Journal of Physics: Conference Series*, volume 1087, page 032032. IOP Publishing, 2018.

[86] Wolfram Wingerath, Felix Gessert, Steffen Friedrich, and Norbert Ritter. Real-time stream processing for big data. *it-Information Technology*, 58(4):186–194, 2016.

[87] Vineet John and Xia Liu. A survey of distributed message broker queues. *arXiv preprint arXiv:1704.00411*, 2017.

[88] Radu Stoenescu, Matei Popovici, Lorina Negreanu, and Costin Raiciu. Symnet: Static checking for stateful networks. In *Proceedings of the 2013 workshop on Hot topics in middleboxes and network function virtualization*, pages 31–36, 2013.

[89] Ivo Adan and Jacques Resing. Queueing theory. *Eindhoven University of Technology*, 180, 2002.

[90] cyberpedia. What is key length?

[91] Anne VDM Kayem, Selim G Akl, and Patrick Martin. On replacing cryptographic keys in hierarchical key management systems. *Journal of Computer Security*, 16(3):289–309, 2008.

[92] Albert Treytl and Thilo Sauter. Hierarchical key management for smart grids. In *2015 IEEE International Symposium on Systems Engineering (ISSE)*, pages 496–500. IEEE, 2015.

[93] Mathias Björkqvist, Christian Cachin, Robert Haas, Xiao-Yu Hu, Anil Kurmus, René Pawlitzek, and Marko Vukolić. Design and implementation of a key-lifecycle management system. In *International Conference on Financial Cryptography and Data Security*, pages 160–174. Springer, 2010.

[94] Gregory Linklater, Christian Smith, Alan Herbert, and Barry Irwin. Toward distributed key management for offline authentication. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 10–19, 2018.

[95] Laurent Eschenauer and Virgil D Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41–47, 2002.

[96] Pınar Nuhoğlu Kibar, Abdullah Yasin Gündüz, and Buket Akkoyunlu. Implementing bring your own device (byod) model in flipped learning: Advantages and challenges. *Technology, Knowledge and Learning*, 25(3):465–478, 2020.

[97] Abdulrezzak Zekiye and Öznur Özkasap. Blockchain-based federated learning for decentralized energy management systems. In *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)*, pages 186–193. IEEE, 2023.

[98] Conghui Zhang, Yi Li, Wenwen Sun, and Shaopeng Guan. Blockchain based big data security protection scheme. In *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pages 574–578. IEEE, 2020.

[99] amazon. bring your own key in aws kms. .