CRYPTOGRAPHIC HASH FUNCTIONS

AND SECURE COMMUNICATION DESIGN

Members:

1. Sebastián Granizo (00212771)
2. Johana Duchi (00321980)
3. Daniela Jiménez (00322800)

Course: CMP 5006E Fall 2024
Date: November 12th, 2024

Overview

In this assignment, you will explore cryptographic hash functions and design a secure communication system implementing various cryptographic concepts covered in class.

Part 1: Research on Cryptographic Hash Functions

CRYPTOGRAPHIC HASH FUNCTIONS

Abstract

Cryptographic hash functions are essential for maintaining data integrity and security in digital communications. Acting as "digital fingerprints," these functions provide a one-way transformation from arbitrary data to a unique, fixed-length output, ensuring that sensitive information remains secure across applications like password storage, blockchain, and digital signatures. This report investigates the principles underlying cryptographic hash functions, compares widely used algorithms such as MD5, SHA-1, SHA-2, and SHA-3, and examines the vulnerabilities that have led to algorithm deprecation and the adoption of more secure alternatives.

Introduction

In today's digital landscape, ensuring data integrity and confidentiality is essential. Cryptographic hash functions are foundational tools in modern cybersecurity, enabling the verification of data authenticity and preventing unauthorized access. These functions compress variable-length data into a fixed and unique output, known as a hash or "digest," ensuring information security and providing an immutable representation that cannot be reversed to reveal the original data. This process is crucial in high-security applications such as certificate verification, digital signatures, and blockchain technology. Hash functions are widely used to verify the integrity and authenticity of information in various security applications, underscoring their indispensable role in protecting digital data (Anwar, Apriani, & Adianita, 2021).

1. Fundamentals of Cryptographic Hash Functions

Cryptographic hash functions are specialized algorithms that convert input data of any length into a fixed-length output, known as a hash or digest. This transformation is one-way, making it computationally infeasible to reverse-engineer the original data from the

hash, which is crucial for applications requiring unique and irreversible identifiers. These functions play a significant role in password storage, digital signatures, and blockchain technology by ensuring data integrity and authenticity (Cid, 2006).

The security of cryptographic hash functions relies on three key properties. Pre-image resistance prevents attackers from determining the original input from a hash, which is essential for confidentiality in password storage. Second pre-image resistance makes it infeasible to find a different input that produces the same hash, preserving authenticity in digital signatures. Collision resistance ensures that two distinct inputs cannot produce the same hash, a property critical for applications like blockchain, where unique data representation is required (Cantu et al., 2017).

Together, these properties contribute to security by ensuring data remains confidential, authentic, and tamper-resistant. Pre-image resistance protects against unauthorized data access, while second pre-image and collision resistance prevent forgery and ensure data uniqueness. These features make cryptographic hash functions foundational to secure digital communications and data integrity (Maetouq et al., 2018).

2. Common Hash Functions and Their Characteristics

Overview of Widely Used Hash Functions

Cryptographic hash functions play an essential role in ensuring data integrity, authentication, and security across digital systems. Over the past decades, these functions have evolved in response to increasing security demands and vulnerabilities discovered in earlier algorithms. They operate by taking an input (or 'message') and producing a fixed-size string of bytes, typically a 'digest' that is unique to the original data. Hash functions are pivotal in various applications, including digital signatures, password hashing, and blockchain (Debnath et al., 2017). These functions have shaped digital security frameworks, yet with increased computational power and evolving attack strategies, several of these standards have been phased out or are under review.

MD5

It was released in 1991. Initially, MD5 was widely adopted across various protocols due to its efficiency, especially where fast processing was essential. However, in the early 2000s, researchers identified vulnerabilities allowing for collision attacks, where distinct inputs could produce identical hashes. This compromised MD5's reliability in secure environments, leading to its gradual deprecation. Despite its limitations, MD5 is still used in applications such as file checksums for detecting data corruption during downloads (Cilardo et al., 2010).

SHA-1

Developed by the NSA, SHA-1 improved upon MD5 by generating a 160-bit hash, thereby providing a larger security margin. SHA-1 became an integral part of various cryptographic protocols, including SSL/TLS certificates, due to its initial robustness.

Despite that, as computational resources advanced, SHA-1's susceptibility to collision attacks emerged, ultimately leading to its deprecation. In 2017, Google and CWI Amsterdam publicly demonstrated a successful collision, which highlighted SHA-1's vulnerabilities and accelerated its phase-out across industries.

SHA-2 Family

The SHA-2 family, comprising variants such as SHA-224, SHA-256, SHA-384, and SHA-512, represents a more resilient approach to hash function design. SHA-256 is widely used in applications demanding high security, including blockchain and SSL/TLS certificates. SHA-2's design allows for hash lengths up to 512 bits, significantly enhancing its resistance to brute-force and collision attacks (Sun et al., 2007). This robustness has led to SHA-2's adoption as a standard across numerous high-security sectors, including finance and government applications.

Differences Between MD5, SHA-1, and SHA-2

MD5, SHA-1, and SHA-2 exhibit key differences in bit length and structural resilience, factors that directly impact their security against collision attacks. MD5 produces a shorter, 128-bit hash, making it more susceptible to attacks compared to SHA-1's 160-bit output. However, SHA-2, with its range of bit lengths (up to 512), provides a superior level of security suitable for applications where data integrity and confidentiality are paramount. SHA-256, for example, makes it more resistant to brute-force attacks than MD5 and SHA-1, because of its larger hash length.

SHA-3: A New Standard in Hash Functions

Standardized by NIST in 2015, SHA-3 introduces a distinct sponge construction method, differentiating it from SHA-2's structure. Rather than replacing SHA-2, SHA-3 was developed as a complement, providing an alternative in cases where enhanced security is required. The unique construction of SHA-3 allows it to resist certain attacks that could potentially affect SHA-2, offering an additional layer of security for applications such as governmental and enterprise-level encryption.

3. Applications of Cryptographic Hash Functions

Digital Signatures

In digital signatures, cryptographic hash functions are fundamental for ensuring data authenticity and integrity. A digital signature works by hashing the data, followed by encrypting the hash with the sender's private key. Upon receipt, the recipient decrypts the hash with the sender's public key and compares it to a hash of the received data. If these

hashes match, the data's authenticity and integrity are confirmed, as it proves that the document originates from the claimed source and has not been altered (Kpieleh, 2022). Digital signatures are widely used in sectors that require high data integrity and source verification, as they integrate well with digital authentication protocols, enhancing the security of electronic transactions.

Password Storage

Cryptographic hash functions are also pivotal in secure password storage. Systems use hash functions to store only the hashed version of passwords, making it unnecessary to store the actual password. When a user logs in, the system hashes the entered password and compares it with the stored hash, which minimizes the risk of password exposure if the database is compromised. Additionally, techniques like 'salting,' which adds random data to each password before hashing, are commonly employed to enhance security against rainbow table attacks. Salting ensures that identical passwords produce different hashes, increasing security and making decryption attempts more difficult. Major authentication systems, including those utilized by large platforms like Google and Facebook, implement salted hashes for password storage to protect against data breaches (Upadhyay et al., 2022).

Blockchain Technology

Cryptographic hash functions are integral to blockchain technology, where they secure the integrity of transaction data across decentralized systems. In a blockchain, each block contains a hash of the previous block, linking them in an unbreakable chain that deters tampering. Any change in a block's data alters its hash, thus breaking the link with subsequent blocks, which signals potential tampering. Bitcoin, for example, utilizes SHA-256 to secure transactions within its blockchain, ensuring that any attempt to modify past transactions would require recalculating all subsequent hashes a computationally prohibitive task. This structure ensures data immutability and transparency, essential for the decentralized and trustless nature of blockchain systems (Martínez et al., 2020). The reliance on cryptographic hash functions for data security in blockchain underscores their critical role in this field.

HMAC (Hash-based Message Authentication Code)

Hash-based Message Authentication Codes (HMACs) combine a cryptographic hash function with a secret key to provide both data integrity and authenticity. HMAC is extensively utilized in network protocols like SSL and IPsec to ensure data integrity during transmission. By using a shared secret key, HMAC verifies that the data has not been altered during transit, providing a secure means of authentication. For example, HMAC is critical in online banking, where message authentication is essential to prevent fraudulent activities during online transactions. Its security is derived from the underlying hash function, often SHA-2, which makes HMAC a highly effective method for secure communication (Khali et al., 2005). The robustness of HMAC in securing messages underlines its relevance in financial and network security applications.

Conclusion

Cryptographic hash functions have proven to be vital pillars in securing the integrity and authenticity of digital information. Beyond their technical attributes, these functions embody the constant interplay between security needs and innovation in the digital age. As cybersecurity challenges continue to evolve, the development of stronger hash algorithms, such as SHA-2 and SHA-3, reflects a proactive approach to counter vulnerabilities that emerge alongside technological advancement. This adaptability underscores the resilience of cryptographic principles, ensuring that data remains protected even as threats grow more sophisticated. Looking ahead, the role of cryptographic hash functions will likely expand, not only in traditional areas like password protection and digital signatures but also within emerging fields such as decentralized finance, quantum-resistant cryptography, and advanced IoT security. The continued refinement of hash functions exemplifies the commitment to building trust in digital interactions, underscoring their significance as the silent yet steadfast guardians of data in an increasingly interconnected world.

References

Ali Maetouq, Salwani Mohd Daud, Noor Azurati Ahmad, Nurazean Maarop, Nilam Nur Amir Sjarif and Hafiza Abas, "Comparison of Hash Function Algorithms Against Attacks: A Review" International Journal of Advanced Computer Science and Applications(ijacsa), 9(8), 2018. http://dx.doi.org/10.14569/IJACSA.2018.090813

Anwar, M. R., Apriani, D., & Adianita, I. R. (2021). Hash Algorithm in Verification of Certificate Data Integrity and Security. *Aptisi Transactions on Technopreneurship*, 3(2), 1-10. https://doi.org/10.34306/att.v3i2.212

Cid, C. (2006). Recent developments in cryptographic hash functions: Security implications and future directions. *Information Security Technical Report, 11*(2), 100-107. https://doi.org/10.1016/j.istr.2006.03.007

Cilardo, A., Esposito, L., Veniero, A., Mazzeo, A., Beltran, V., & Ayguadé, E. (2010). A CellBE-based HPC application for the analysis of vulnerabilities in cryptographic hash functions. *2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC)*, 450-457. https://doi.org/10.1109/HPCC.2010.113

Debnath, S., Chattopadhyay, A., & Dutta, S. (2017). Brief review on journey of secured hash algorithms. *2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix)*, 1-5. https://doi.org/10.1109/OPTRONIX.2017.8349971

Khali, H., Mehdi, R., & Araar, A. (2005). A system-level architecture for hash message authentication code. *2005 12th IEEE International Conference on Electronics, Circuits and Systems*, 1-4. https://doi.org/10.1109/ICECS.2005.4633372

Kpieleh, F. (2022). Cryptographic Hash Functions for Digital Stamping. *Journal of Digital Innovations & Contemporary Research in Science, Engineering & Technology*, 10(4), 65-72. https://fcc08321-8158-469b-b54d-f591e0bd3df4.filesusr.com/ugd/185b0a_0be6a284409b4b1db623c7cc1220df06.pdf

Stallings, William. *Cryptography and Network Security: Principles and Practice*. 7th Global Edition, Pearson Education, 2017.

Sun, W., Guo, H., He, H., & Dai, Z. (2007). Design and optimized implementation of the SHA-2(256, 384, 512) hash algorithms. *2007 7th International Conference on ASIC*, 858-861. https://doi.org/10.1109/ICASIC.2007.4415766

Upadhyay, D., Gaikwad, N., Zaman, M., & Sampalli, S. (2022). Investigating the avalanche effect of various cryptographically secure hash functions and hash-based applications. *IEEE Access*, *10*, 112472-112486. https://doi.org/10.1109/ACCESS.2022.3215778

Part 2: Secure Communication System Design

Design Requirements

- ***Confidentiality:*** Only Alice and Bob can read the communication.
- ***Integrity:*** Messages cannot be altered during transmission.
- ***Authenticity:*** Alice and Bob can verify each other's identities.
- ***Non-repudiation:*** Neither party can deny the authenticity of the message.

Introduction

Using the bases of the **NIST Cybersecurity Framework**, in order to secure the messaging app between Bob and Alice, the NIST Cybersecurity Framework was applied to manage risks and strengthen defenses against cyber threats. Using the five core functions—**Identify**, Protect, Detect, Respond, and Recover—each step was carefully addressed to minimize vulnerabilities. In the Identify phase, potential threats to the application and communication channels were assessed. This included recognizing valuable data, like the message content between Bob and Alice, and identifying which system components were at risk. Then, the **Protect** function involved encrypting the messages, ensuring that both data at rest and in transit were secure, using methods like end-to-end encryption (E2EE). Multi-factor authentication (MFA) was also implemented to prevent unauthorized access to user accounts. Firewalls and intrusion detection systems (IDS) were deployed to monitor traffic for any suspicious activity, adding a proactive defense layer.

Moving to **Detect**, the system was set up to continuously monitor for any potential breaches or anomalies. Threat intelligence and analysis tools were integrated to spot unusual patterns in real-time, enabling quick detection of any intrusions. For the **Respond** function, an incident response plan was established to mitigate the effects of any detected threats, outlining actions to take if a hacker attempted to access the app. This plan ensured minimal downtime and limited damage to both data and user trust. Lastly, the **Recover** phase included regular backups and secure restore mechanisms to bring the system back online if a compromise occurred, ensuring that Bob and Alice could safely resume communication. Together, these NIST framework elements provided a comprehensive security approach that safeguarded the messaging app against hackers and potential threat actors.

Additionally, we took into account other cybersecurity frameworks which helped design a better system. According to **ISO/IEC 27001** standards, several advanced cryptographic measures were implemented to ensure non-repudiation, authenticity, and message integrity. Digital Signatures were introduced to verify the identity of both users and ensure that messages truly originated from the sender, providing non-repudiation. Using their private keys, Alice and Bob digitally sign each message, which allows the recipient to verify the sender's authenticity with the public key issued by a Certificate Authority (CA). This approach prevents impersonation and verifies that messages are genuinely from Alice or Bob. Message Integrity Checks were also applied by hashing each message and signing the hash before transmission. Upon receipt, the recipient verifies the hash against the sender's digital signature, ensuring that the message remains unaltered during transit.

Additional mechanisms were set up to prevent common attacks. For Replay Attack Prevention, each message includes a timestamp and a nonce to prevent the reuse of messages in different contexts or sessions. The timestamp ensures that old messages cannot be resented as new ones, while the nonce secures each session by making every exchange unique. Optionally, Perfect Forward Secrecy (PFS) was implemented through Elliptic Curve Diffie-Hellman (ECDH) for session key exchange, which ensures that session keys from past communications cannot be derived even if private keys are later compromised. The final System Architecture enables Bob and Alice to communicate securely over an encrypted channel where each message is digitally signed and verified. Key components include Alice and Bob as the main users, the encrypted messaging channel, and the CA responsible for issuing and verifying public keys. Through these layered security measures, the ISO/IEC 27001 framework enhances the app's resilience against threats, preserving message integrity, authenticity, and confidentiality.

System Components

Key Management

Key Generation Algorithm: Use Elliptic Curve Cryptography (ECC) or RSA for asymmetric key generation and AES for symmetric encryption.

Generation Method: Public-private key pairs are generated locally on both devices. Symmetric session keys are generated for each session using a secure random generator.

Key Exchange

Public Key Exchange: Public keys are exchanged via a trusted Certificate Authority (CA). Alice and Bob each obtain each other's public keys from the CA.

Session Key Exchange: Use Diffie-Hellman or Elliptic Curve Diffie-Hellman (ECDH) to securely negotiate symmetric session keys.

Key Storage

Private Keys: Stored securely on each party's device in a hardware security module (HSM) or encrypted in secure storage.

Session Keys: Temporarily stored in memory for the session duration and securely deleted after use.

Key Rotation

Frequency: Session keys are generated for each new communication session. Public-private key pairs should be rotated periodically or as required by policy.

Message Flow Design

Encryption Process (Alice Sending to Bob)

Session Setup: Alice initiates communication and establishes a session key with Bob using ECDH.

Message Signing: Alice hashes the message and signs it with her private key to ensure integrity and non-repudiation.

Encryption: Alice encrypts the signed message with the session key (AES-256).

Transmission: Alice sends the encrypted message and her digital signature to Bob.

Decryption Process (Bob Receiving from Alice)

Session Key Verification: Bob verifies the session key.

Message Decryption: Bob decrypts the message with the session key.

Signature Verification: Bob verifies Alice's signature using her public key, confirming integrity and authenticity.

Message Processing: If verification is successful, the message is processed; otherwise, it is discarded.

Error Handling

Decryption Failure: If decryption fails, an error message is sent back to the sender, logging the issue.

Signature Verification Failure: If the signature verification fails, the message is discarded, and Alice is notified.

Session Management

Session Timeout: Sessions automatically terminate after a fixed duration of inactivity.

Session Renewal: Upon session expiration, a new session key is negotiated.

Security Features

Digital Signatures

Purpose: To ensure non-repudiation and authenticity.

Method: Alice and Bob use their private keys to send messages.

Message Integrity Checks

Method: Hash each message and sign it before transmission.

Verification: Receiver verifies the hash against the signature to ensure message integrity.

Replay Attack Prevention

Timestamps: Each message includes a timestamp to prevent replay attacks.

Nonce: Use nonces (unique random values) for each session to ensure messages cannot be reused.

Perfect Forward Secrecy (Optional)

Key Agreement: Use ECDH for session key exchange to ensure past session keys cannot be derived, even if private keys are compromised in the future.

Deliverables

System Architecture

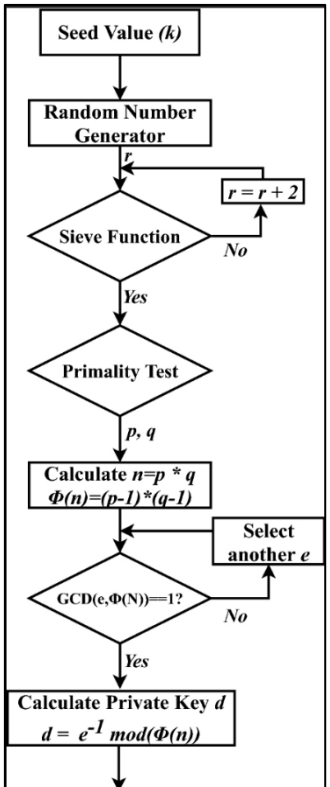
Overview: Alice and Bob communicate over an encrypted channel with each message digitally signed.

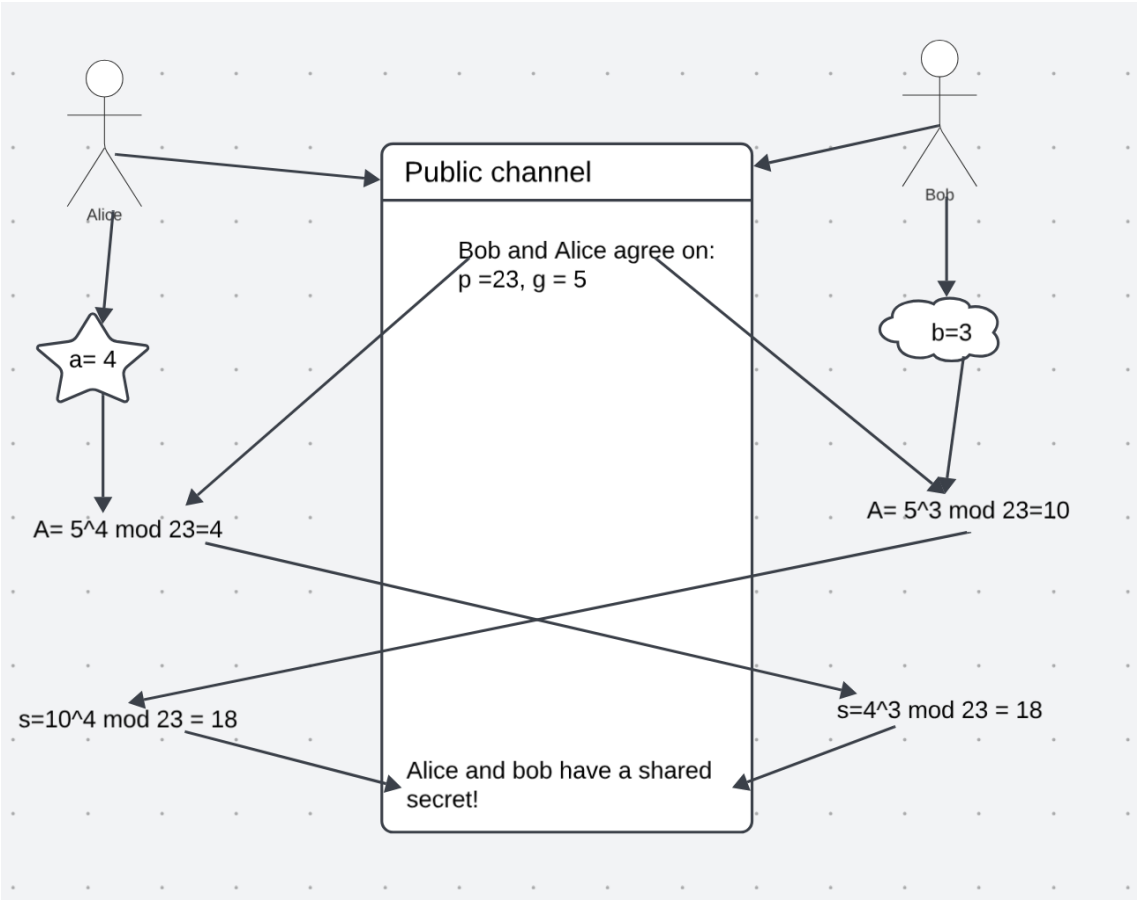
Components:

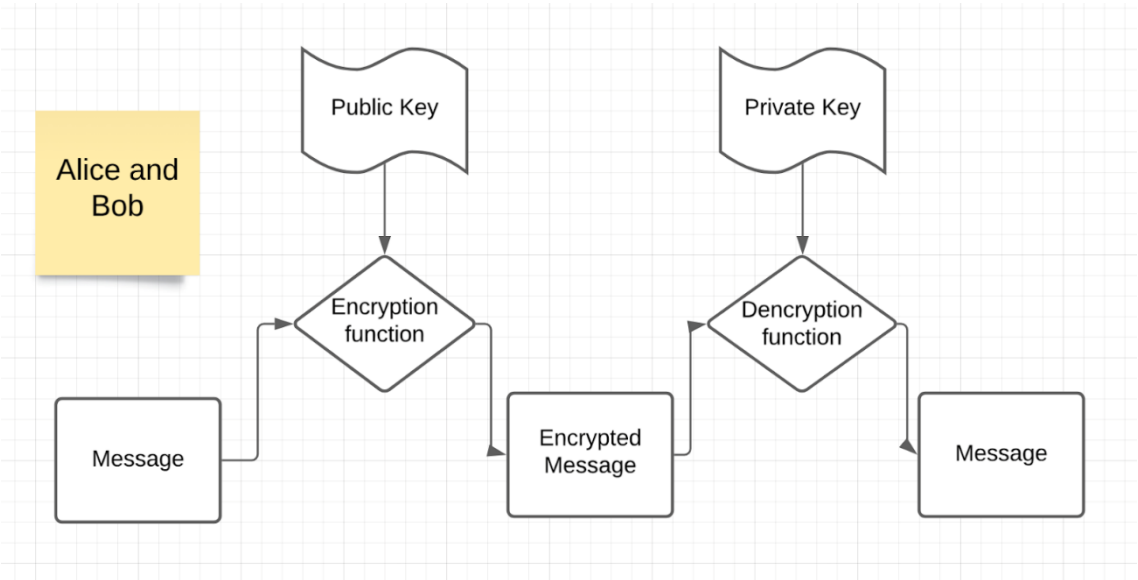
Alice and Bob: Users who exchange messages.

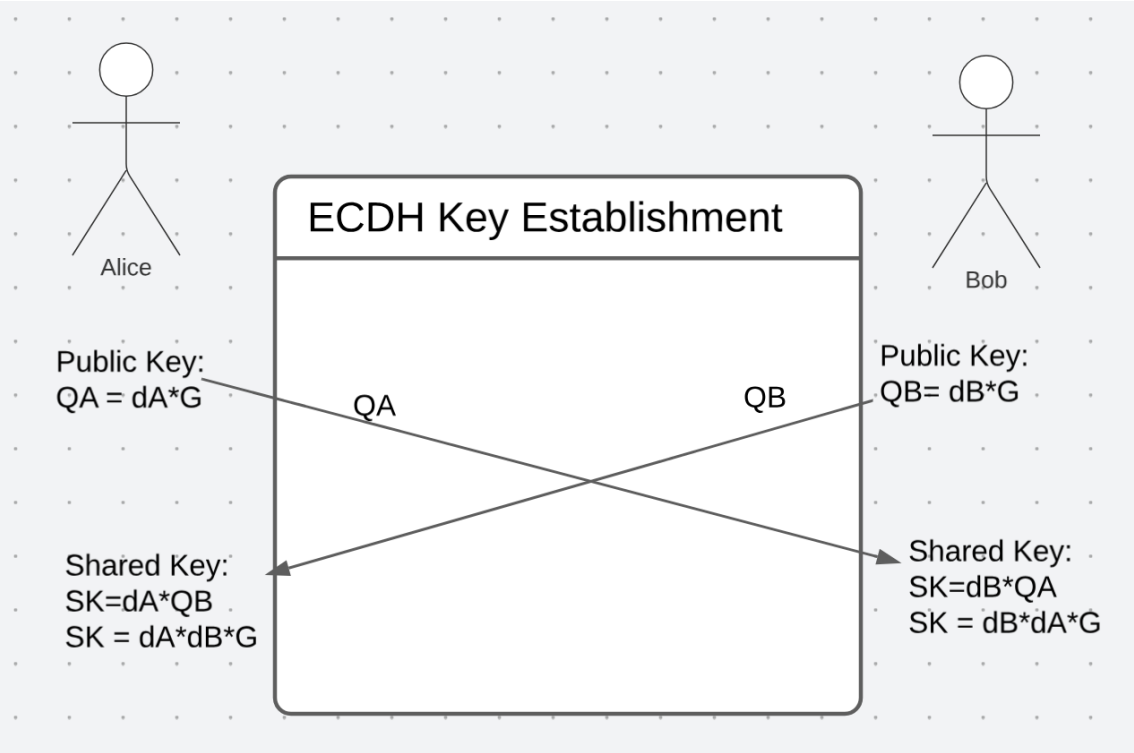
Certificate Authority (CA): Verifies identities and issues public keys.

Flow Diagrams




Alice
Bob
Public channel
Bob and Alice agree on:
p =23, g = 5
a= 4
b=3
A= 5^4 mod 23=4
A= 5^3 mod 23=10
s=10^4 mod 23 = 18
s=4^3 mod 23 = 18
Alice and bob have a shared secret!


Alice and Bob
Public Key
Private Key
Encryption function
Dencryption function
Message
Encrypted Message
Message



Security Analysis

Confidentiality: Achieved through AES-256 encryption and secure session keys.

Integrity: Message integrity is preserved via digital signatures and hashing.

Authenticity: Ensured through digital signatures verified with public keys.

Non-repudiation: Alice and Bob cannot deny sending signed messages.

Implementation Considerations

Cryptographic Libraries: Use libraries with proven security

Secure Storage: Store private keys and session keys securely.

Session Management: Implement secure session lifecycle policies.

Audit Logging: Log key exchange, message transmission, and error handling events for troubleshooting.

Potential Vulnerabilities and Mitigations

Vulnerability	**Mitigation**
Man-in-the-Middle (MitM)	Use a trusted CA for key validation.
Replay Attacks	Timestamps and nonces for each session.

Key Compromise	Implement periodic key rotation and perfect forward secrecy.
Message Tampering	Verify integrity via hashing and digital signatures.
Session Hijacking	Use short session lifetimes and re-authentication.