ATMA RAM SANATAN DHARMA COLLEGE
UNIVERSITY OF DELHI

Course Title: Latex Typesetting for Beginners

Project Name: Data Privacy Book

Submitted To:

Mr. Anil Kumar Rajak
Faculty of Mathematics

Submitted By:

Name: Sudeep Kumar Singh
Roll No: 22/28021
Course: B.Sc. Computer Science Hons.

Name: Shubham Kushwaha
Roll No: 22/28090
Course: B.Sc. Computer Science Hons.

Name: Ankit Sarawag
Roll No: 22/28006
Course: B.Sc. Computer Science Hons.

FIRST EDITION

2024

DATA PRIVACY ATTACKS, CRYPTOGRAPHY & DATA PROTECTION

PREPARED BY

SUDEEP KUMAR SINGH

SHUBHAM KUSHWAHA

ANKIT SARAWAG

Data Privacy Attacks, Cryptography and Data Protection

Sudeep Kumar Singh, Shubham Kushwaha, Ankit Sarawag

November 19, 2024

Preface

In the contemporary digital age, the security of sensitive data has become a critical concern due to the increasing volume of data exchanged across interconnected systems and the internet. This document explores key concepts surrounding data privacy attacks, cryptographic techniques, and data protection measures. The rise of cyberattacks and the constant evolution of security threats emphasize the need for robust data protection strategies to safeguard personal and organizational information. By focusing on data privacy breaches and cryptographic solutions, this work aims to provide an in-depth understanding of how attacks are executed and what countermeasures can be employed to secure systems. It also covers modern cryptographic techniques such as symmetric and asymmetric algorithms, hash functions, and public-key infrastructure (PKI), which play a vital role in protecting information integrity, confidentiality, and availability. This study is designed to guide students, researchers, and professionals in understanding and mitigating privacy risks and strengthening data security.

Acknowledgements

We would like to express our deepest gratitude to our professors and mentors at Atma Ram Sanatan Dharma College, Delhi University, for their continuous support, insightful feedback, and encouragement throughout the course of this research. Their guidance in both theoretical and practical aspects of cryptography and data protection has been invaluable. Special thanks to our mentor for providing critical feedback that helped shape our approach to understanding data privacy attacks and cryptographic defenses. We are also thankful for the resources and tools provided by ARSD College, which were instrumental in the successful completion of this work. We would like to extend our appreciation to the authors and researchers whose works have been referenced in this document, especially William Stallings' *Information Privacy Engineering and Privacy by Design*, which provided a comprehensive foundation for understanding privacy threats and regulations. Finally, we acknowledge the contributions of our peers, whose discussions and input have enhanced the quality of this study.

About the Authors

Sudeep Kumar Singh

Sudeep Kumar Singh is a third-year student of B.Sc. Computer Science (Hons.) at Atma Ram Sanatan Dharma College, Delhi University.

Shubham Kushwaha

Shubham Kushwaha is also a third-year student pursuing B.Sc. Computer Science (Hons.) at Atma Ram Sanatan Dharma College, Delhi University.

Ankit Sarawag

Ankit Sarawag, is also a third-year student pursuing B.Sc. Computer Science (Hons.) at Atma Ram Sanatan Dharma College, Delhi University.

Credits

- **Sudeep Kumar Singh (28021), Shubham Kushwaha (28090), and Ankit Sarawag (28006)**, students of B.Sc. Computer Science (Hons.) at Atma Ram Sanatan Dharma College, Delhi University, have collaborated on this project.
- The concepts and theories discussed in this work are based on the book: **Stallings, William.** *Information Privacy Engineering and Privacy by Design: Understanding Privacy Threats, Technology, and Regulations Based on Standards and Best Practices.* Addison-Wesley Professional, 2019.

Contents

List of Figures

List of Tables

Chapter 1
Security Attacks

Chapter 1

Security Attacks

1.1 Definitions and Key Concepts

- **Security Attack:** Any action compromising the security of information owned by an organization.
- **Security Mechanism:** A process or device designed to detect, prevent, or recover from a security attack.
- **Security Service:** A service that enhances the security of data processing systems and information transfers, countering security attacks using security mechanisms.

1.2 Threat vs. Attack

- **Threat:** Any circumstance or event with the potential to adversely impact organizational operations, assets, individuals, or the nation via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.
- **Attack:** Any malicious activity attempting to collect, disrupt, deny, degrade, or destroy information system resources or the information itself.

1.3 Types of Security Attacks

1.3.1 Passive Attacks

- **Nature:** Eavesdropping or monitoring transmissions without affecting system resources.
- **Goals:** Obtain information being transmitted.
- **Types:**
  - *Release of Message Contents:* Eavesdropping on communications such as phone conversations, emails, or transferred files.
  - *Traffic Analysis:* Observing the pattern of messages, frequency, and length without examining the contents.
- **Characteristics:** Difficult to detect; prevention usually involves encryption.

1.3.2 Active Attacks

- **Nature:** Involves modification of stored or transmitted data or the creation of false data.
- **Types:**
  - *Masquerade:* One entity pretends to be another, often combined with other active attacks.
  - *Replay:* Passive capture and retransmission of data to produce unauthorized effects.

  - *Data Modification:* Alteration of legitimate messages or reordering/delaying them to produce unauthorized effects.
  - *Denial of Service:* Preventing or inhibiting normal use or management of communication facilities, either targeting specific entities or disrupting entire networks.

- **Characteristics:** Detection and recovery are primary goals since absolute prevention is difficult.

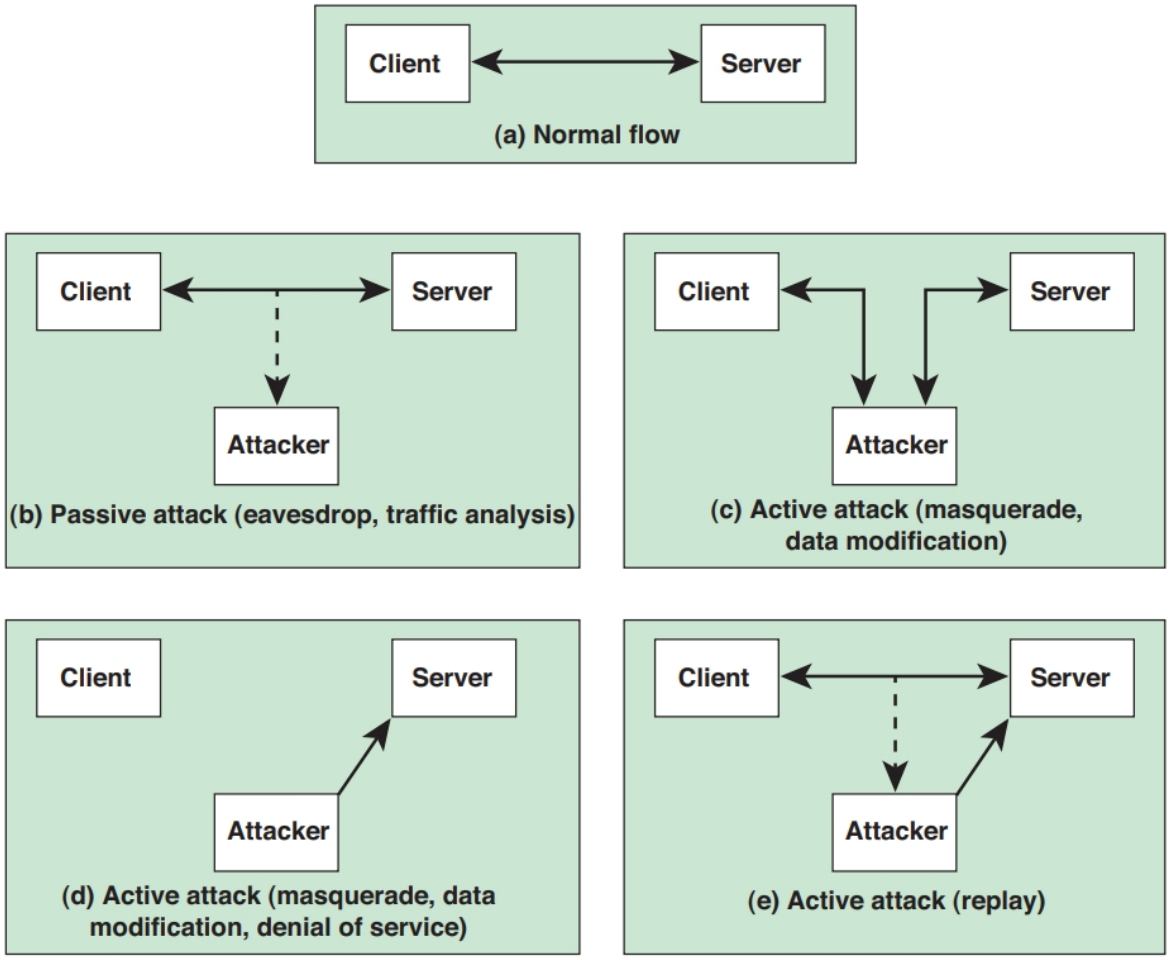


Figure 1.1: Security Attacks

1.4 Illustrations of Security Attacks

- **Passive Attack Example:** Eavesdropping on client-server communication without disturbing the flow.

- **Active Attack Examples:**
  - *Masquerade:* Man-in-the-middle attack, intercepting and pretending to be the client or server.
  - *Data Modification:* Selectively modifying data communicated between client and server.
  - *Replay:* Capturing and reusing client messages.
  - *Denial of Service:* Flooding the server with data or triggering resource-consuming actions.

Chapter 2
Security Services

Chapter 2

Security Services

A security service supports security requirements such as confidentiality, integrity, availability, authenticity, and accountability. These services implement security policies using security mechanisms.

2.1 Key Security Services

Service	Description
Authentication	Determines a person's identity before granting access.
Access Control	Limits or allows access to resources for specific purposes.
Data Confidentiality	Ensures information is only available to intended persons.
Data Integrity	Ensures information is modified only by authorized persons and in appropriate ways.
Nonrepudiation	Prevents a person from denying an action they performed.
Availability	Ensures apps, services, and hardware are ready and perform acceptably when needed.

Table 2.1: Key Security Services

2.2 Authentication

- **Purpose:** Ensures communication is authentic.
  - **Single Message:** Ensures the recipient that the message is from the claimed source.
  - **Ongoing Interaction:** Ensures both entities are authentic and the connection is not interfered with.
- **Types of Authentication:**
  - **Peer Entity Authentication:** Confirms the identity of a peer entity in an association, preventing masquerade or unauthorized replay.
  - **Data Origin Authentication:** Corroborates the source of a data unit, supporting applications like email.

2.3 Access Control

- **Purpose:** Limits and controls access to systems and applications via communications links.
- **Requirement:** Identification or authentication of each entity to tailor access rights.

2.4 Data Confidentiality

- **Purpose:** Protects transmitted data from passive attacks.
- **Protection Levels:**
  - Protection of all user data over a connection.
  - Protection against traffic flow analysis.
- **Requirement:** Prevents attackers from observing source, destination, frequency, length, or other characteristics of communication.

2.5 Data Integrity

- **Purpose:** Ensures messages are received as sent, without modification.
- **Types of Integrity Service:**
  - **Connection-Oriented Integrity:** Addresses message stream modification and denial of service for streams of messages.
  - **Connectionless Integrity:** Provides protection against individual message modification.

2.6 Nonrepudiation

- **Purpose:** Prevents sender or receiver from denying transmitted messages.
  - **Sender Nonrepudiation:** Receiver can prove the sender sent the message.
  - **Receiver Nonrepudiation:** Sender can prove the receiver received the message.

2.7 Availability Service

- **Purpose:** Ensures system or resource is accessible and usable on demand.
- **Requirement:** System provides services as designed whenever users request them.
- **Protection:** Addresses security concerns raised by denial-of-service attacks, depending on access control and other security services.

2.8 Conclusion

These security services are essential for maintaining a secure data processing and communication environment in organizations.

Chapter 3
Security Mechanisms

Chapter 3

Security Mechanisms

3.1 Key Security Mechanisms

Security mechanisms are the methods used to implement security services and ensure the protection of information systems. The key security mechanisms are as follows:

3.1.1 Cryptographic Algorithms

These are techniques used to transform data into a form that is unintelligible to unauthorized parties. Cryptographic algorithms are covered in detail in chapter 4.

3.1.2 Data Integrity

Mechanisms ensuring the integrity of a data unit or stream of data units. These mechanisms verify that data has not been altered during transmission or storage.

3.1.3 Digital Signature

A digital signature is data appended to or a cryptographic transformation of a data unit, allowing the recipient to prove the source and integrity of the data unit. It also protects against forgery by ensuring authenticity.

3.1.4 Authentication Exchange

This mechanism ensures the identity of an entity through the exchange of information, allowing parties to confirm each other's identity.

3.1.5 Traffic Padding

Traffic padding involves the insertion of bits into gaps in a data stream to obscure the size or frequency of the data, thus frustrating traffic analysis attempts. This makes it harder for attackers to discern patterns in the communication.

3.1.6 Routing Control

Routing control enables the selection of secure routes for data and allows routing changes, especially when a breach of security is suspected. It helps in directing the flow of sensitive information through the most secure channels.

3.1.7 Notarization

Notarization involves a trusted third party that verifies certain properties of a data exchange, ensuring that data has not been tampered with and confirming its authenticity.

3.1.8 Access Control

Access control mechanisms enforce access rights to resources, ensuring that only authorized users or systems can access or modify certain information or resources.

Chapter 4
Cryptographic Algorithms

Chapter 4

Cryptographic Algorithms

4.1 Definitions

Cryptography: The discipline that involves the principles, means, and methods for transforming data to hide its content, prevent unauthorized use, or undetected modification. It provides information security, including confidentiality, data integrity, non-repudiation, and authenticity.

Cryptographic Algorithm: A well-defined computational procedure related to cryptography that takes variable inputs, often including a cryptographic key, and produces an output.

4.2 Categories of Cryptographic Algorithms

Category	Description
Keyless	Algorithms that do not use any keys during cryptographic transformations.
Single-Key	Algorithms where the transformation result is a function of the input data and a single secret key.
Two-Key	Algorithms that use two related keys (private key and public key) during different stages of the calculation.

Table 4.1: Categories of Cryptographic Algorithms

4.3 Keyless Algorithms

4.3.1 Cryptographic Hash Function:

Converts a variable amount of text into a small, fixed-length value called a hash value, hash code, or digest.
It has properties that make it useful in other cryptographic algorithms, such as message authentication codes or digital signatures.

4.3.2 Pseudorandom Number Generator:

Produces a deterministic sequence of numbers or bits that appear to be random, sufficient for some cryptographic purposes.

4.4 Single-Key Algorithms (Symmetric Encryption)

4.4.1 Symmetric Encryption Algorithms:

Use a secret key shared between two parties to encrypt and decrypt data.
This ensures that communication between the parties is protected from outsiders.

4.4.2 Message Authentication Code (MAC):

A data element generated by a cryptographic transformation involving a secret key and typically a cryptographic hash function of the message.
It is used to verify the integrity of the message by those possessing the secret key.

4.5 Two-Key Algorithms (Asymmetric Encryption)

4.5.1 Asymmetric Encryption Algorithms:

Use two related keys: a private key (known only to the user) and a public key (available to others). It can work in two ways:

- Encrypting data with the private key and decrypting with the public key.
- Encrypting data with the public key and decrypting with the private key.

4.6 Applications

4.6.1 Digital Signature Algorithm:

A value computed with a cryptographic algorithm associated with a data object, used to verify the data's origin and integrity. The signer uses their private key to generate the signature, and anyone with the public key can verify it.

4.6.2 Key Exchange:

Securely distributing a symmetric key to two or more parties.

4.6.3 User Authentication:

Authenticating a user attempting to access an application or service and verifying that the service is genuine.

These cryptographic algorithms are fundamental in ensuring the secure storage, transmission, and interaction of data.

Chapter 5
Symmetric Encryption

Chapter 5

Symmetric Encryption

5.1 Definition

Symmetric encryption, also known as secret-key encryption, is a cryptographic scheme where the same key is used for both encryption and decryption.

5.2 Ingredients of Symmetric Encryption

- **Plaintext:** The original message or data block that is input into the encryption algorithm.
- **Encryption Algorithm:** The algorithm that performs various substitutions and transformations on the plaintext using the secret key.
- **Secret Key:** An input to the encryption algorithm; the exact substitutions and transformations depend on this key.
- **Ciphertext:** The scrambled message produced as output. It depends on both the plaintext and the secret key. Different keys produce different ciphertexts for the same plaintext.
- **Decryption Algorithm:** The inverse of the encryption algorithm; it uses the ciphertext and the secret key to reproduce the original plaintext.

5.3 Requirements for Secure Symmetric Encryption

- **Strong Encryption Algorithm:**
  - The algorithm should be such that an opponent, even with access to the algorithm and multiple ciphertexts, cannot decipher the ciphertext or determine the key.
  - Ideally, the opponent should not be able to decrypt the ciphertext or discover the key even if they possess multiple ciphertext-plaintext pairs.
- **Secure Key Distribution:**
  - The sender and receiver must obtain copies of the secret key in a secure manner and must keep the key secure.
  - If an opponent discovers the key, they can read all communications encrypted with that key.

5.4 Key Generation and Distribution

A key generation algorithm typically generates a random number and derives a secret key from it.

For two parties to communicate, key distribution can occur through various methods:

- One party generates the key and securely transfers it to the other party.
- A secure key exchange protocol enables both parties to jointly generate a key known only to them.
- A third party generates the key and securely transfers it to the two communicating parties.

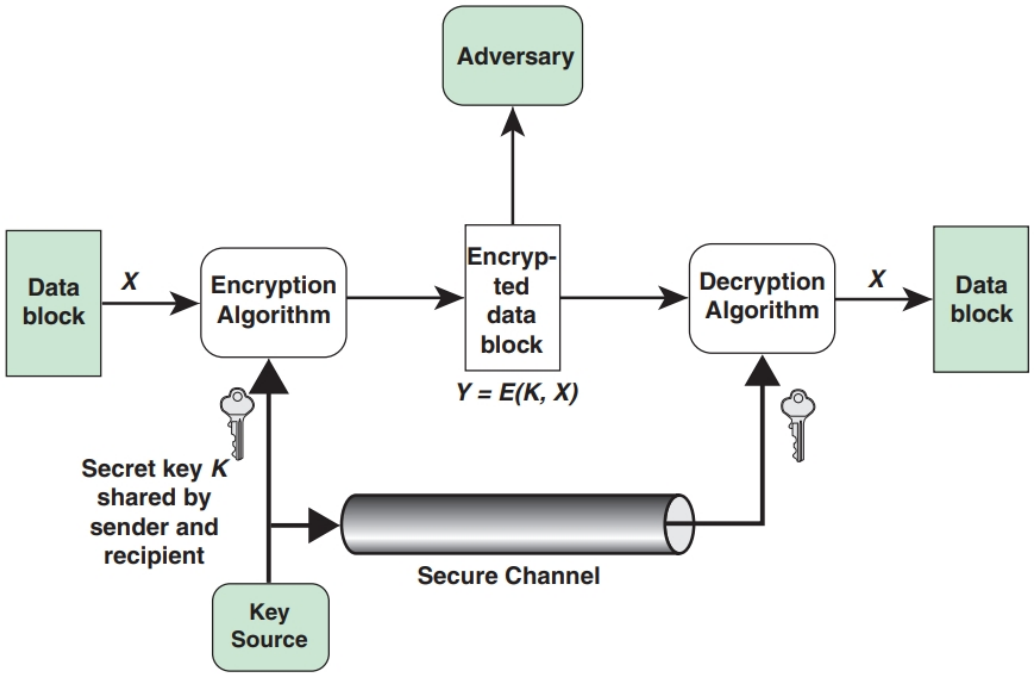


Figure 5.1: Model of Symmetric Cryptosystem

5.5 Potential Adversary Attacks

- **Cryptanalysis:**
  - Relies on the nature of the algorithm and knowledge of the plaintext or sample plaintext/ciphertext pairs.
  - It attempts to deduce the key or specific plaintext.
  - If successful, all past and future messages encrypted with that key are compromised.
- **Brute-Force Attack:**
  - Involves trying every possible key on a piece of ciphertext until an intelligible translation into plaintext is found.
  - On average, half of all possible keys must be tried to achieve success.
  - A secure symmetric encryption scheme requires an algorithm resistant to cryptanalysis and a key of sufficient length to defeat brute-force attacks.

Chapter 6
Asymmetric Encryption

Chapter 6

Asymmetric Encryption

6.1 Definition

Public-key cryptography, also known as asymmetric cryptography, uses two separate keys for encryption and decryption. This contrasts with symmetric encryption, which uses only one key.

6.2 Ingredients of Asymmetric Encryption

- **Plaintext:** The readable message or data block input into the encryption algorithm.
- **Encryption Algorithm:** Performs various transformations on the plaintext using a key.
- **Public Key and Private Key:** A pair of keys selected such that if one key is used for encryption, the other is used for decryption. The transformations depend on the key used.
- **Ciphertext:** The scrambled block produced as output, depending on the plaintext and the key used.
- **Decryption Algorithm:** The inverse of the encryption algorithm, it accepts the ciphertext and the matching key to produce the original plaintext.

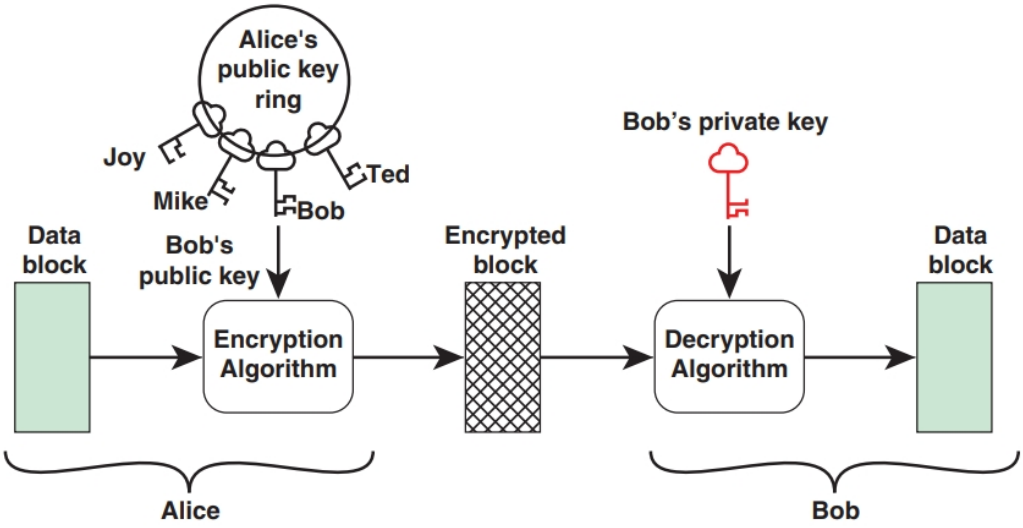
6.3 Process of Asymmetric Encryption

- **Key Generation:** Each user generates a pair of keys (public and private) for encryption and decryption.
- **Public Key Sharing:** One of the keys (public key) is placed in a public register or accessible file, while the companion key (private key) is kept secret.
- **Encryption:** If Alice wants to send a confidential message to Bob, she encrypts the message using Bob's public key.
- **Decryption:** Bob decrypts the message using his private key. No one else can decrypt the message because only Bob knows his private key.

6.4 Comparison with Symmetric Encryption

6.5 Alternative Uses

- **Authentication:**
  Alice can use her private key to encrypt a message. When Bob decrypts it with Alice's public key, he can be certain the message came from Alice.



(a) Public-key encryption/decryption (Alice encrypts block for Bob only)

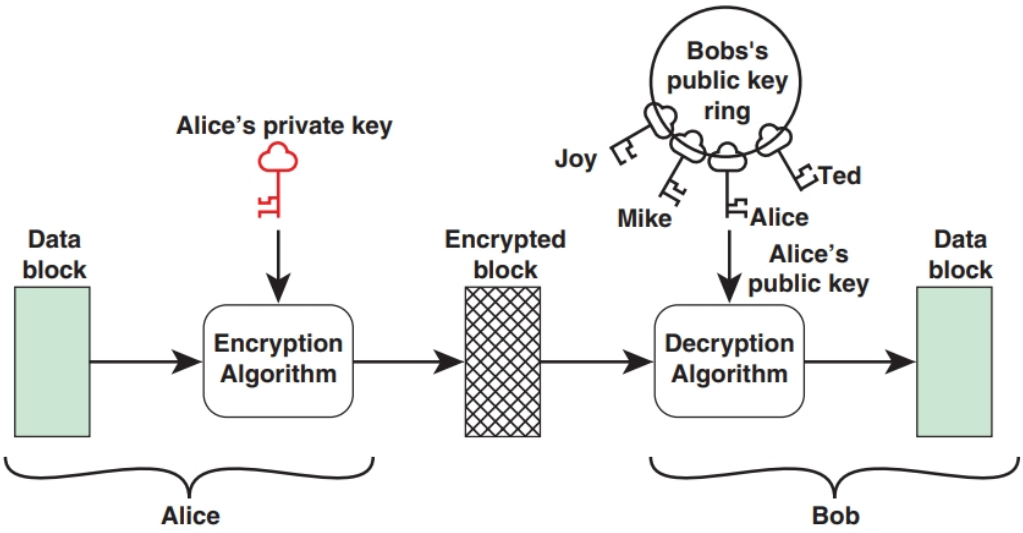


Figure 6.1: Model of Asymmetric Cryptosystem

Symmetric Encryption	**Asymmetric Encryption**
Uses the same algorithm with the same secret key for encryption and decryption.	Uses one algorithm for encryption and a related algorithm for decryption, with a pair of keys (public and private).
Sender and receiver must share the algorithm and the secret key.	Sender and receiver must each have a unique public/private key pair.
Key must be kept secret.	Private key must be kept secret.
Algorithm plus samples of ciphertext must be insufficient to determine the key.	Algorithm plus public key plus samples of ciphertext must be insufficient to determine the private key.

Table 6.1: Symmetric Vs Asymmetric Encryption

6.6 Security Considerations

The security of public-key encryption relies on the strength of the algorithm and the length of the private key.
Public-key cryptographic algorithms are generally slower than symmetric algorithms and are often limited to small blocks of data, such as a secret key or a hash value.

Chapter 7
Cryptographic Hash Functions

Chapter 7

Cryptographic Hash Functions

7.1 Definition

A hash function takes an input of arbitrary length and maps it to a fixed-length data block. Multiple input blocks may produce the same output, known as the hash value or hash digest.

7.2 Requirements for Cryptographic Hash Function

Requirement	**Description**
Variable input size	Can be applied to a block of data of any size.
Fixed output size	Produces a fixed-length output.
Efficiency	Relatively easy to compute for any given input, making hardware and software implementations practical.
Preimage resistant	Computationally infeasible to find an input that maps to a given hash value.
Second preimage resistant	Computationally infeasible to find a different input with the same hash value.
Collision resistant	Computationally infeasible to find any two different inputs that map to the same hash value.
Pseudorandomness	Output appears to be a random sequence of bits.

Table 7.1: Requirements for Cryptographic Hash Function

7.3 Uses of Cryptographic Hash Functions

7.3.1 Message Authentication:

Ensures data integrity and verifies the authenticity of the message source.
Example process:

1. Generate a hash value for the source message.
2. Encrypt the hash value using a secret key shared by a cooperating partner.
3. Transmit the message plus the encrypted hash value to the destination.
4. The recipient decrypts the hash value, generates a new hash value from the incoming message, and compares the two hash values.

7.3.2 Digital Signatures:

Verifies the origin and integrity of a message. The sender's private key is used to encrypt the hash value, and the recipient can use the sender's public key to decrypt it and verify the message.

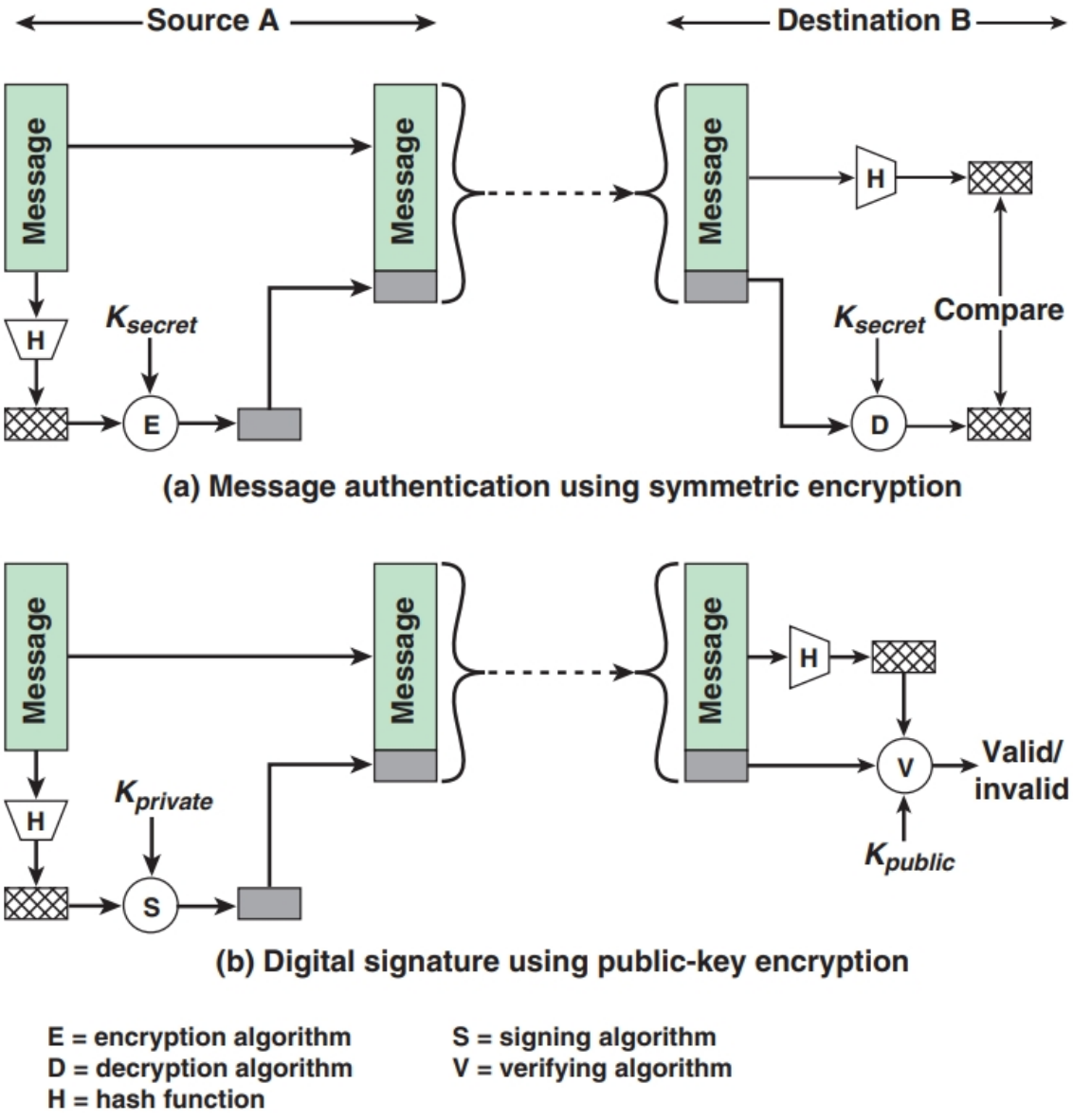


Figure 7.1: Uses for a Secure Hash Function

7.4 Digital Signatures

7.4.1 Definition:

A cryptographic transformation of data ensuring origin authentication, data integrity, and non-repudiation (NIST FIPS 186-4).

7.4.2 Process:

1. **Hash Generation:** Bob generates a hash value for his message using a secure hash function.
2. **Signature Creation:** Bob uses his private key and the hash value to create a digital signature.
3. **Sending:** The message is sent with the digital signature attached.
4. **Verification:**
   (a) The receiver generates a hash value of the received message.
   (b) The receiver uses Bob's public key, the generated hash, and the received signature to verify the message.
   (c) If valid, it confirms the message was indeed sent by Bob and hasn't been altered.

7.4.3 Applications:

- Signing email messages for sender authentication.
- Signing software programs to authenticate their source and counter software tampering.
- Verifying authorship or origin of digital data.
- Ensuring the integrity of digital data against tampering.
- Authenticating online entities.

Hash Function Algorithm

The SHA-256 algorithm is a cryptographic hash function that produces a fixed-size 256-bit hash value from input data. It is used widely for data integrity verification and digital signatures. Below is the description of the SHA-256 algorithm and its mathematical representation.

Mathematical Equation Representation of SHA-256

The SHA-256 function involves several mathematical operations, including bitwise operations, modular additions, and logical functions. The core operations are defined as:

$$T_1 = h + \Sigma_1(e) + Ch(e, f, g) + K_j + W_j$$

$$T_2 = \Sigma_0(a) + Maj(a, b, c)$$

Where: - $\Sigma_1(x)$ and $\Sigma_0(x)$ are the functions that represent the bitwise rotations and shifts. - $Ch(x, y, z)$ is the "choose" function, which selects between values based on certain conditions. - $Maj(x, y, z)$ is the "majority" function, which returns the majority bit from the three input bits.

Finally, the 256-bit output hash value is obtained by concatenating the hash values after processing all blocks:

$$H = h_0 \parallel h_1 \parallel h_2 \parallel h_3 \parallel h_4 \parallel h_5 \parallel h_6 \parallel h_7$$

Algorithm 1 SHA-256 Hash Function

Input: A message M of any length
Output: A fixed-length hash value H (256 bits)
Initialize hash values:
$h_0 = 0x6a09e667f3bcc908$
$h_1 = 0xbb67ae85a4a1f7e9$
$h_2 = 0x3c6ef372fe94f82b$
$h_3 = 0xa54ff53a5f1d36f1$
$h_4 = 0x510e527fade682d1$
$h_5 = 0x9b05688c2b3e6c1f$
$h_6 = 0x1f83d9abfb41bd6b$
$h_7 = 0x5be0cd19137e2179$
Padding: Add padding to the message M to make its length congruent to 448 mod 512.
Append the length of M (in bits) as a 64-bit integer.
Processing blocks: Process the padded message in 512-bit blocks.
for each block M_i in the padded message **do**
    Break M_i into 16 32-bit words $W_0, W_1, ..., W_{15}$
    for j = 16 to 63 **do**
        $W_j = \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}$
    end for
    Initialize working variables:
    $a = h_0$, $b = h_1$, $c = h_2$, $d = h_3$, $e = h_4$, $f = h_5$, $g = h_6$, $h = h_7$
    for j = 0 to 63 **do**
        Compute $T_1 = h + \Sigma_1(e) + Ch(e, f, g) + K_j + W_j$
        $T_2 = \Sigma_0(a) + Maj(a, b, c)$
        Update working variables:
        $h = g$, $g = f$, $f = e$, $e = d + T_1$, $d = c$, $c = b$, $b = a$, $a = T_1 + T_2$
    end for
    Update hash values:
    $h_0 = h_0 + a$, $h_1 = h_1 + b$, $h_2 = h_2 + c$, $h_3 = h_3 + d$,
    $h_4 = h_4 + e$, $h_5 = h_5 + f$, $h_6 = h_6 + g$, $h_7 = h_7 + h$
end for
Output: The final hash value is $H = h_0 \parallel h_1 \parallel h_2 \parallel h_3 \parallel h_4 \parallel h_5 \parallel h_6 \parallel h_7$.

Output of the Hash Function

The output of a hash function like SHA-256 is a 256-bit (32-byte) fixed-length value, irrespective of the input size. This means that even if the input data is extremely large, the output will always be a fixed 256-bit hash.

For example, applying SHA-256 to the string "Hello, World!" produces the following hash output:

SHA-256("Hello, World!") = dffd6021bb2bd7a3643b4d5c891d94b6f97c2bc0c5e9b8482f5a03b4777e0e8f

Chapter 8
Practical Considerations

Chapter 8

Practical Considerations

8.1 Cryptographic Algorithms and Key Lengths

Security Over Time:

- Processor speeds and scrutiny increase over time, which can lead old algorithms to become insecure.
- Key lengths and hash values that were secure before may now be too weak due to increased computational power and analysis techniques.

8.1.1 Guidance Sources:

- **FIPS[2] 140-2A:** Approved Security Functions for FIPS PUB 140-2.
- **SP[3] 800-131A:** Transitioning the Use of Cryptographic Algorithms and Key Lengths.
- **ENISA[4]:** Algorithms, Key Size, and Protocol Report [ECRY18].

8.1.2 Recommendations:

1. **Symmetric Encryption:**
   - **Advanced Encryption Standard (AES):** Key lengths of 128, 192, or 256 bits.
2. **Hash Functions:**
   - **SHA[5]-2 or SHA-3:** Hash lengths range from 224 to 512 bits.
3. **Digital Signatures:**
   - **Digital Signature Algorithm (DSA)[7]:** 2048 bits.
   - **RSA[6] Algorithm:** 2048 bits.
   - **Elliptic-Curve Digital Signature Algorithm (ECDSA)[8]:** 224 bits.

8.1.3 Implementation Considerations:

- **Standards Selection:**
  - Rely on standardized algorithms (e.g., AES, SHA, DSS).
  - Use standards developed by **NIST[1]** and other trusted organizations.
- **Implementation Methods:**
  - Choose between hardware, software, and firmware based on security, cost, simplicity, efficiency, and ease of implementation.
- **Key Management:**
  - Administer and manage cryptographic keys (generation, protection, storage, etc.).

  - For more details, refer to *Effective Cybersecurity: Best Practices and Standards [STAL19].*

- **Cryptographic Module Security:**
  - Secure design, implementation, and use of cryptographic modules.
  - Use **NIST[1]** Cryptographic Module Validation Program (CMVP) for validation.

8.2 Lightweight Cryptographic Algorithms

8.2.1 Focus:

- Develop secure algorithms minimizing execution time, memory usage, and power consumption.
- Suitable for embedded systems (e.g., IoT devices).

8.2.2 NIST[1] Project:

- **NIST[1]** is working on developing a portfolio of lightweight algorithms.
- Initial focus is on symmetric encryption and secure hash functions.

8.3 Post-Quantum Cryptographic Algorithms

Concerns:

- Quantum computers may break current asymmetric cryptographic algorithms, leading to vulnerabilities.

8.3.1 NIST[1] Effort:

- **NIST[1]** is working on standardizing algorithms that can replace or complement existing asymmetric cryptographic schemes.
- They are exploring mathematical approaches for new asymmetric cryptographic algorithms resistant to quantum computing threats.

NIST[1]: National Institute of Standards and Technology, **FIPS[2]:** Federal Information Processing Standards, **SP[3]:** Special Publication, **ENISA[4]:** European Union Agency for Cybersecurity, **SHA[5]:** Secure Hash Algorithm, **RSA[6]:** Rivest-Shamir-Adleman, **DSA[7]:** Digital Signature Algorithm, **ECDSA[8]:** Elliptic Curve Digital Signature Algorithm

Chapter 9
Public-Key Infrastructure (PKI)

Chapter 9

Public-Key Infrastructure (PKI)

9.1 Definition:

- PKI supports the distribution and identification of public encryption keys.
- It enables secure data exchange over networks and verifies the identity of the other party.
- PKI binds public keys to entities and manages public key bindings.

9.2 Public-Key Certificates:

- A public-key certificate is a set of data that uniquely identifies an entity, containing the entity's public key and other data.
- It is digitally signed by a trusted party, known as the Certification Authority (CA), binding the public key to the entity.
- The certificate solves the problem of public-key distribution by ensuring authenticity.

9.3 Certificate Generation and Use:

- **Certificate Components:**
  - Contains unique identifying information for the entity.
  - Public key of the entity.
  - Information about the CA.
  - Certificate details (e.g., expiration date).
- **Signing:**
  - Information is hashed, and a digital signature is generated using the CA's private key.
  - The certificate is signed and can be broadcast or attached to documents.
- **Verification:**
  - Users verify the certificate's validity using the CA's public key.
  - Ensures the public key in the certificate is authentic.

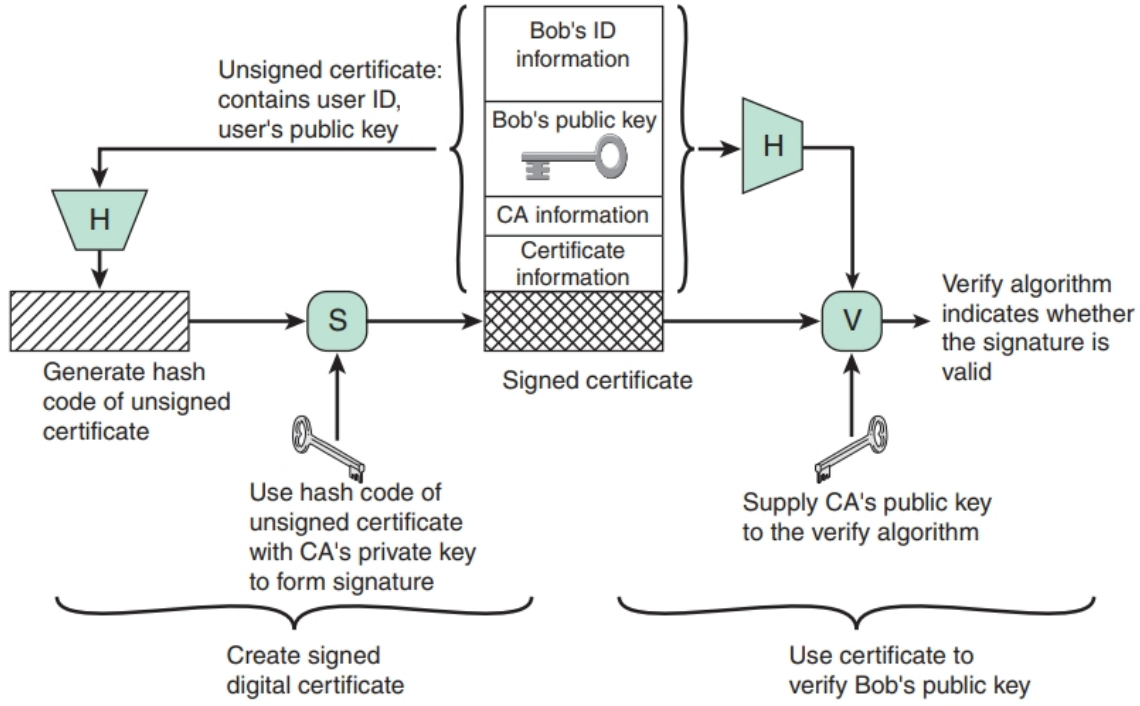


FIGURE 1.8 Public-Key Certificate Use

Figure 9.1: Public-Key Certificate Use

9.4 PKI Architecture:

9.4.1 Requirements:

- Any participant can read and verify a certificate's owner and authenticity.
- Only the CA can create and update certificates.
- Participants can verify the current validity of certificates.

9.4.2 Components:

- **End Entity:** Users, devices, or processes identified in certificates.
- **Certification Authority (CA):** Creates and assigns certificates, issues certificate revocation lists (CRLs).
- **Registration Authority (RA):** Optional, offloads administrative functions from the CA, verifies end entity identity.
- **Repository:** Stores and retrieves PKI-related information (certificates and CRLs).
- **Relying Party:** Users or agents relying on certificate data for decision-making.

9.5 Certificate Use Example:

- Alice needs Bob's public key.
- Alice obtains the CA's public key securely.

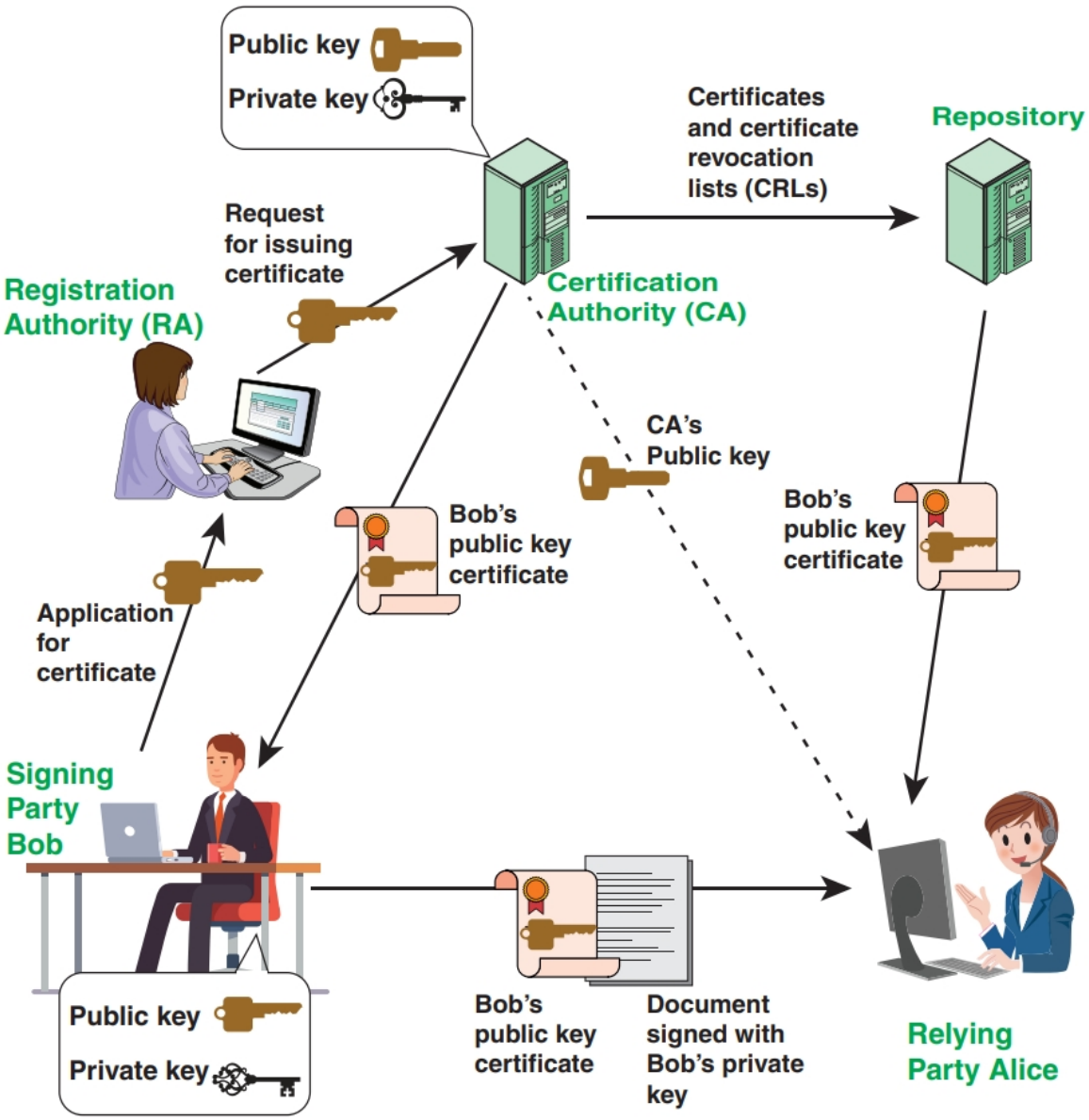


Figure 9.2: PKI Scenario

- Alice checks the repository for Bob's certificate and verifies its validity.
- Alice encrypts data to Bob using Bob's public key.
- Bob can send a signed document to Alice, who verifies the signature with Bob's public key.

9.6 Hierarchical CA Organization:

- Multiple CAs can be organized hierarchically, with a root CA signing subordinate CAs.
- Root certificates are embedded in browsers and other software for built-in trust.

Appendix

Appendix A: SHA-256 Hash Function Algorithm

The SHA-256 algorithm is a cryptographic hash function that produces a fixed-size 256-bit hash value from input data. It is used widely for data integrity verification and digital signatures. Below is the description of the SHA-256 algorithm and its mathematical representation.

Algorithm 2 SHA-256 Hash Function

Input: A message M of any length
Output: A fixed-length hash value H (256 bits)
Initialize hash values:
$h_0 = 0x6a09e667f3bcc908$
$h_1 = 0xbb67ae85a4a1f7e9$
$h_2 = 0x3c6ef372fe94f82b$
$h_3 = 0xa54ff53a5f1d36f1$
$h_4 = 0x510e527fade682d1$
$h_5 = 0x9b05688c2b3e6c1f$
$h_6 = 0x1f83d9abfb41bd6b$
$h_7 = 0x5be0cd19137e2179$
Padding: Add padding to the message M to make its length congruent to 448 mod 512.
Append the length of M (in bits) as a 64-bit integer.
Processing blocks: Process the padded message in 512-bit blocks.
for each block M_i in the padded message **do**
    Break M_i into 16 32-bit words $W_0, W_1, ..., W_{15}$
    for j = 16 to 63 **do**
        $W_j = \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}$
    end for
    Initialize working variables:
    $a = h_0$, $b = h_1$, $c = h_2$, $d = h_3$, $e = h_4$, $f = h_5$, $g = h_6$, $h = h_7$
    for j = 0 to 63 **do**
        Compute $T_1 = h + \Sigma_1(e) + Ch(e, f, g) + K_j + W_j$
        $T_2 = \Sigma_0(a) + Maj(a, b, c)$
        Update working variables:
        $h = g$, $g = f$, $f = e$, $e = d + T_1$, $d = c$, $c = b$, $b = a$, $a = T_1 + T_2$
    end for
    Update hash values:
    $h_0 = h_0 + a$, $h_1 = h_1 + b$, $h_2 = h_2 + c$, $h_3 = h_3 + d$,
    $h_4 = h_4 + e$, $h_5 = h_5 + f$, $h_6 = h_6 + g$, $h_7 = h_7 + h$
end for
Output: The final hash value is $H = h_0 \parallel h_1 \parallel h_2 \parallel h_3 \parallel h_4 \parallel h_5 \parallel h_6 \parallel h_7$.

Appendix B: Mathematical Equation Representation of SHA-256

The SHA-256 function involves several mathematical operations, including bitwise operations, modular additions, and logical functions. The core operations are defined as:

$$T_1 = h + \Sigma_1(e) + Ch(e, f, g) + K_j + W_j$$

$$T_2 = \Sigma_0(a) + Maj(a, b, c)$$

Where: - $\Sigma_1(x)$ and $\Sigma_0(x)$ are the functions that represent the bitwise rotations and shifts. - $Ch(x, y, z)$ is the "choose" function, which selects between values based on certain conditions. - $Maj(x, y, z)$ is the "majority" function, which returns the majority bit from the three input bits.

Finally, the 256-bit output hash value is obtained by concatenating the hash values after processing all blocks:

$$H = h_0 \parallel h_1 \parallel h_2 \parallel h_3 \parallel h_4 \parallel h_5 \parallel h_6 \parallel h_7$$

Appendix C: Output of the Hash Function

The output of a hash function like SHA-256 is a 256-bit (32-byte) fixed-length value, irrespective of the input size. This means that even if the input data is extremely large, the output will always be a fixed 256-bit hash.

For example, applying SHA-256 to the string "Hello, World!" produces the following hash output:

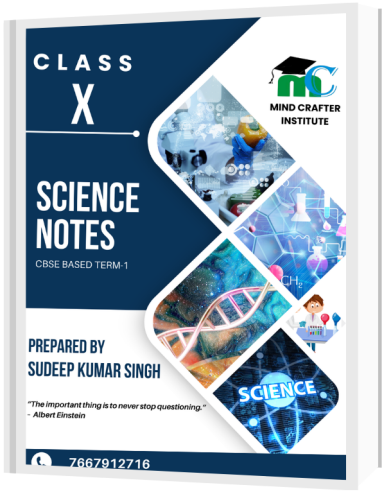
SHA-256("Hello, World!") = dffd6021bb2bd7a3643b4d5c891d94b6f97c2bc0c5e9b8482f5a03b4777e0e8f

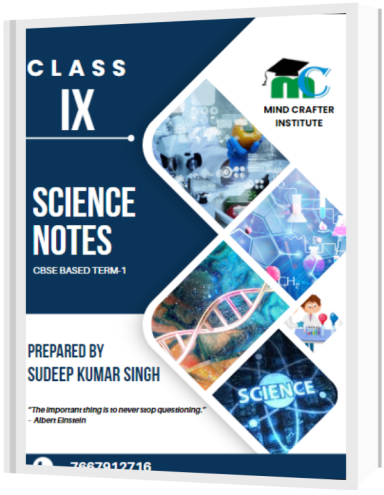
Index

ABOUT THE BOOK

This book, Data Privacy Attacks, Cryptography, and Data Protection, offers a concise overview of modern security challenges. It covers data privacy attacks, their impacts, and counter measures. Key cryptographic algorithms, including symmetric and asymmetric methods, are explored for effective data protection. The book equips readers with both theoretical knowledge and practical solutions for securing digital environments.

OTHER PUBLICATIONS



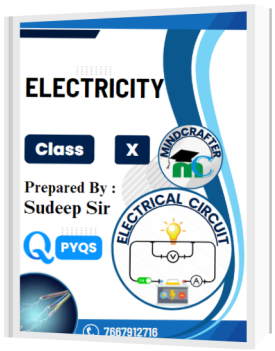


AVAILABLE IN BOTH MEDIUM: HINDI & ENGLISH

PYQS SERIES (CHAPTER WISE) - 10

CHEMISTRY

BIOLOGY



PHYSICS

& MORE

SCAN TO GIVE FEEDBACK