Public-key Cryptography
Theory and Practice

Abhijit Das

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

Chapter 1: Overview

What is Cryptography?

What is Cryptography?

- **Cryptography** is the study of techniques for preventing access to sensitive data by parties who are not authorized to access the data.

What is Cryptography?

- **Cryptography** is the study of techniques for preventing access to sensitive data by parties who are not authorized to access the data.
- **Cryptanalysis** is the study of techniques for breaking cryptographic systems.

What is Cryptography?

- **Cryptography** is the study of techniques for preventing access to sensitive data by parties who are not authorized to access the data.
- **Cryptanalysis** is the study of techniques for breaking cryptographic systems.
- **Cryptology = Cryptography + Cryptanalysis**

What is Cryptography?

- **Cryptography** is the study of techniques for preventing access to sensitive data by parties who are not authorized to access the data.
- **Cryptanalysis** is the study of techniques for breaking cryptographic systems.
- **Cryptology = Cryptography + Cryptanalysis**
- Cryptanalysis is useful for strengthening cryptographic primitives.

What is Cryptography?

- **Cryptography** is the study of techniques for preventing access to sensitive data by parties who are not authorized to access the data.
- **Cryptanalysis** is the study of techniques for breaking cryptographic systems.
- **Cryptology = Cryptography + Cryptanalysis**
- Cryptanalysis is useful for strengthening cryptographic primitives.
- Maintaining security and privacy is an ancient and primitive need.

What is Cryptography?

- **Cryptography** is the study of techniques for preventing access to sensitive data by parties who are not authorized to access the data.
- **Cryptanalysis** is the study of techniques for breaking cryptographic systems.
- **Cryptology = Cryptography + Cryptanalysis**
- Cryptanalysis is useful for strengthening cryptographic primitives.
- Maintaining security and privacy is an ancient and primitive need.
- Particularly relevant for military and diplomatic applications.

What is Cryptography?

- **Cryptography** is the study of techniques for preventing access to sensitive data by parties who are not authorized to access the data.
- **Cryptanalysis** is the study of techniques for breaking cryptographic systems.
- **Cryptology = Cryptography + Cryptanalysis**
- Cryptanalysis is useful for strengthening cryptographic primitives.
- Maintaining security and privacy is an ancient and primitive need.
- Particularly relevant for military and diplomatic applications.
- Wide deployment of the Internet makes everybody a user of cryptographic tools.

Message Encryption

Message Encryption

- Required for secure transmission of messages over a public channel.

Message Encryption

- Required for secure transmission of messages over a public channel.
- Alice wants to send a **plaintext** message M to Bob.

Message Encryption

- Required for secure transmission of messages over a public channel.
- Alice wants to send a **plaintext** message M to Bob.
- Alice **encrypts** M to generate the **ciphertext** message $C = f_e(M, K_e)$.

Message Encryption

- Required for secure transmission of messages over a public channel.
- Alice wants to send a **plaintext** message M to Bob.
- Alice **encrypts** M to generate the **ciphertext** message $C = f_e(M, K_e)$.
- K_e is the **encryption key**.

Message Encryption

- Required for secure transmission of messages over a public channel.
- Alice wants to send a **plaintext** message M to Bob.
- Alice **encrypts** M to generate the **ciphertext** message $C = f_e(M, K_e)$.
- K_e is the **encryption key**.
- C is sent to Bob over the public channel.

Message Encryption

- Required for secure transmission of messages over a public channel.
- Alice wants to send a **plaintext** message M to Bob.
- Alice **encrypts** M to generate the **ciphertext** message $C = f_e(M, K_e)$.
- K_e is the **encryption key**.
- C is sent to Bob over the public channel.
- Bob **decrypts** C to recover the plaintext message $M = f_d(C, K_d)$.

Message Encryption

- Required for secure transmission of messages over a public channel.
- Alice wants to send a **plaintext** message M to Bob.
- Alice **encrypts** M to generate the **ciphertext** message $C = f_e(M, K_e)$.
- K_e is the **encryption key**.
- C is sent to Bob over the public channel.
- Bob **decrypts** C to recover the plaintext message $M = f_d(C, K_d)$.
- K_d is the **decryption key**.

Message Encryption

- Required for secure transmission of messages over a public channel.
- Alice wants to send a **plaintext** message M to Bob.
- Alice **encrypts** M to generate the **ciphertext** message $C = f_e(M, K_e)$.
- K_e is the **encryption key**.
- C is sent to Bob over the public channel.
- Bob **decrypts** C to recover the plaintext message $M = f_d(C, K_d)$.
- K_d is the **decryption key**.
- Knowledge of K_d is required to retrieve M from C.

Message Encryption

- Required for secure transmission of messages over a public channel.
- Alice wants to send a **plaintext** message M to Bob.
- Alice **encrypts** M to generate the **ciphertext** message $C = f_e(M, K_e)$.
- K_e is the **encryption key**.
- C is sent to Bob over the public channel.
- Bob **decrypts** C to recover the plaintext message $M = f_d(C, K_d)$.
- K_d is the **decryption key**.
- Knowledge of K_d is required to retrieve M from C.
- An eavesdropper (intruder, attacker, adversary, opponent, enemy) cannot decrypt C.

Secret-key or Symmetric Encryption

Secret-key or Symmetric Encryption

- $K_e = K_d$.

Secret-key or Symmetric Encryption

- $K_e = K_d$.
- Algorithms are fast and suitable for software and hardware implementations.

Secret-key or Symmetric Encryption

- $K_e = K_d$.
- Algorithms are fast and suitable for software and hardware implementations.
- The common key has to be agreed upon by Alice and Bob before the actual communication.

Secret-key or Symmetric Encryption

- $K_e = K_d$.
- Algorithms are fast and suitable for software and hardware implementations.
- The common key has to be agreed upon by Alice and Bob before the actual communication.
- Each pair of communicating parties needs a secret key.

Secret-key or Symmetric Encryption

- $K_e = K_d$.
- Algorithms are fast and suitable for software and hardware implementations.
- The common key has to be agreed upon by Alice and Bob before the actual communication.
- Each pair of communicating parties needs a secret key.
- If there are many communicating pairs, the key storage requirement is high.

Public-key or Asymmetric Encryption

Public-key or Asymmetric Encryption

- $K_e \neq K_d$.

Public-key or Asymmetric Encryption

- $K_e \neq K_d$.
- Introduced by Rivest, Shamir and Adleman (1978).

Public-key or Asymmetric Encryption

- $K_e \neq K_d$.
- Introduced by Rivest, Shamir and Adleman (1978).
- K_e is the **public key** known to everybody (even to enemies).

Public-key or Asymmetric Encryption

- $K_e \neq K_d$.
- Introduced by Rivest, Shamir and Adleman (1978).
- K_e is the **public key** known to everybody (even to enemies).
- K_d is the **private key** to be kept secret.

Public-key or Asymmetric Encryption

- $K_e \neq K_d$.
- Introduced by Rivest, Shamir and Adleman (1978).
- K_e is the **public key** known to everybody (even to enemies).
- K_d is the **private key** to be kept secret.
- It is difficult to compute K_d from K_e.

Public-key or Asymmetric Encryption

- $K_e \neq K_d$.
- Introduced by Rivest, Shamir and Adleman (1978).
- K_e is the **public key** known to everybody (even to enemies).
- K_d is the **private key** to be kept secret.
- It is difficult to compute K_d from K_e.
- Anybody can send messages to anybody. Only the proper recipient can decrypt.

Public-key or Asymmetric Encryption

- $K_e \neq K_d$.
- Introduced by Rivest, Shamir and Adleman (1978).
- K_e is the **public key** known to everybody (even to enemies).
- K_d is the **private key** to be kept secret.
- It is difficult to compute K_d from K_e.
- Anybody can send messages to anybody. Only the proper recipient can decrypt.
- No need to establish keys a priori.

Public-key or Asymmetric Encryption

- $K_e \neq K_d$.
- Introduced by Rivest, Shamir and Adleman (1978).
- K_e is the **public key** known to everybody (even to enemies).
- K_d is the **private key** to be kept secret.
- It is difficult to compute K_d from K_e.
- Anybody can send messages to anybody. Only the proper recipient can decrypt.
- No need to establish keys a priori.
- Each party requires only one key-pair for communicating with everybody.

Public-key or Asymmetric Encryption

- $K_e \neq K_d$.
- Introduced by Rivest, Shamir and Adleman (1978).
- K_e is the **public key** known to everybody (even to enemies).
- K_d is the **private key** to be kept secret.
- It is difficult to compute K_d from K_e.
- Anybody can send messages to anybody. Only the proper recipient can decrypt.
- No need to establish keys a priori.
- Each party requires only one key-pair for communicating with everybody.
- Algorithms are slow, in general.

Real-life Analogy

Real-life Analogy

Symmetric encryption

- Alice locks the message in a box by a key.
- Bob uses a copy of the same key to unlock.

Real-life Analogy

Symmetric encryption

- Alice locks the message in a box by a key.
- Bob uses a copy of the same key to unlock.

Asymmetric encryption

- Alice presses a self-locking padlock in order to lock the box. The locking process does not require a real key.
- Bob has the key to open the padlock.

Symmetric and Asymmetric Encryption Together

Symmetric and Asymmetric Encryption Together

- Alice reads Bob's public key K_e.

Symmetric and Asymmetric Encryption Together

- Alice reads Bob's public key K_e.
- Alice generates a random secret key K.

Symmetric and Asymmetric Encryption Together

- Alice reads Bob's public key K_e.
- Alice generates a random secret key K.
- Alice encrypts M by K to generate $C = f_e(M, K)$.

Symmetric and Asymmetric Encryption Together

- Alice reads Bob's public key K_e.
- Alice generates a random secret key K.
- Alice encrypts M by K to generate $C = f_e(M, K)$.
- Alice encrypts K by K_e to generate $L = f_E(K, K_e)$.

Symmetric and Asymmetric Encryption Together

- Alice reads Bob's public key K_e.
- Alice generates a random secret key K.
- Alice encrypts M by K to generate $C = f_e(M, K)$.
- Alice encrypts K by K_e to generate $L = f_E(K, K_e)$.
- Alice sends (C, L) to Bob.

Symmetric and Asymmetric Encryption Together

- Alice reads Bob's public key K_e.
- Alice generates a random secret key K.
- Alice encrypts M by K to generate $C = f_e(M, K)$.
- Alice encrypts K by K_e to generate $L = f_E(K, K_e)$.
- Alice sends (C, L) to Bob.
- Bob recovers K as $K = f_D(L, K_d)$.

Symmetric and Asymmetric Encryption Together

- Alice reads Bob's public key K_e.
- Alice generates a random secret key K.
- Alice encrypts M by K to generate $C = f_e(M, K)$.
- Alice encrypts K by K_e to generate $L = f_E(K, K_e)$.
- Alice sends (C, L) to Bob.
- Bob recovers K as $K = f_D(L, K_d)$.
- Bob decrypts C as $M = f_d(C, K)$.

Key Agreement or Key Exchange

Real-life analogy

Key Agreement or Key Exchange

Real-life analogy

- Alice procures a lock L with key K. Alice wants to send K to Bob for a future secret communication.

Key Agreement or Key Exchange

Real-life analogy

- Alice procures a lock L with key K. Alice wants to send K to Bob for a future secret communication.
- Alice procures another lock L_A with key K_A.

Key Agreement or Key Exchange

Real-life analogy

- Alice procures a lock L with key K. Alice wants to send K to Bob for a future secret communication.
- Alice procures another lock L_A with key K_A.
- Bob procures a lock L_B with key K_B.

Key Agreement or Key Exchange

Real-life analogy

- Alice procures a lock L with key K. Alice wants to send K to Bob for a future secret communication.
- Alice procures another lock L_A with key K_A.
- Bob procures a lock L_B with key K_B.
- Alice puts K in a box, locks the box by L_A using K_A, and sends the box to Bob.

Key Agreement or Key Exchange

Real-life analogy

- Alice procures a lock L with key K. Alice wants to send K to Bob for a future secret communication.
- Alice procures another lock L_A with key K_A.
- Bob procures a lock L_B with key K_B.
- Alice puts K in a box, locks the box by L_A using K_A, and sends the box to Bob.
- Bob locks the box by L_B using K_B, and sends the doubly-locked box back to Alice.

Key Agreement or Key Exchange

Real-life analogy

- Alice procures a lock L with key K. Alice wants to send K to Bob for a future secret communication.
- Alice procures another lock L_A with key K_A.
- Bob procures a lock L_B with key K_B.
- Alice puts K in a box, locks the box by L_A using K_A, and sends the box to Bob.
- Bob locks the box by L_B using K_B, and sends the doubly-locked box back to Alice.
- Alice unlocks L_A by K_A and sends the box again to Bob.

Key Agreement or Key Exchange

Real-life analogy

- Alice procures a lock L with key K. Alice wants to send K to Bob for a future secret communication.
- Alice procures another lock L_A with key K_A.
- Bob procures a lock L_B with key K_B.
- Alice puts K in a box, locks the box by L_A using K_A, and sends the box to Bob.
- Bob locks the box by L_B using K_B, and sends the doubly-locked box back to Alice.
- Alice unlocks L_A by K_A and sends the box again to Bob.
- Bob unlocks L_B by K_B and obtains K.

Key Agreement or Key Exchange

Real-life analogy

- Alice procures a lock L with key K. Alice wants to send K to Bob for a future secret communication.
- Alice procures another lock L_A with key K_A.
- Bob procures a lock L_B with key K_B.
- Alice puts K in a box, locks the box by L_A using K_A, and sends the box to Bob.
- Bob locks the box by L_B using K_B, and sends the doubly-locked box back to Alice.
- Alice unlocks L_A by K_A and sends the box again to Bob.
- Bob unlocks L_B by K_B and obtains K.
- A third party always finds the box locked either by L_A or L_B or both.

Key Agreement or Key Exchange (contd.)

Key Agreement or Key Exchange (contd.)

- Alice generates a key pair (A_e, A_d).

Key Agreement or Key Exchange (contd.)

- Alice generates a key pair (A_e, A_d).
- Bob generates a key pair (B_e, B_d).

Key Agreement or Key Exchange (contd.)

- Alice generates a key pair (A_e, A_d).
- Bob generates a key pair (B_e, B_d).
- Alice sends her public-key A_e to Bob.

Key Agreement or Key Exchange (contd.)

- Alice generates a key pair (A_e, A_d).
- Bob generates a key pair (B_e, B_d).
- Alice sends her public-key A_e to Bob.
- Bob sends his public-key B_e to Alice.

Key Agreement or Key Exchange (contd.)

- Alice generates a key pair (A_e, A_d).
- Bob generates a key pair (B_e, B_d).
- Alice sends her public-key A_e to Bob.
- Bob sends his public-key B_e to Alice.
- Alice computes $K_{AB} = f(A_e, A_d, B_e)$.

Key Agreement or Key Exchange (contd.)

- Alice generates a key pair (A_e, A_d).
- Bob generates a key pair (B_e, B_d).
- Alice sends her public-key A_e to Bob.
- Bob sends his public-key B_e to Alice.
- Alice computes $K_{AB} = f(A_e, A_d, B_e)$.
- Bob computes $K_{BA} = f(B_e, B_d, A_e)$.

Key Agreement or Key Exchange (contd.)

- Alice generates a key pair (A_e, A_d).
- Bob generates a key pair (B_e, B_d).
- Alice sends her public-key A_e to Bob.
- Bob sends his public-key B_e to Alice.
- Alice computes $K_{AB} = f(A_e, A_d, B_e)$.
- Bob computes $K_{BA} = f(B_e, B_d, A_e)$.
- The protocol insures $K_{AB} = K_{BA}$ to be used by Alice and Bob as a shared secret.

Key Agreement or Key Exchange (contd.)

- Alice generates a key pair (A_e, A_d).
- Bob generates a key pair (B_e, B_d).
- Alice sends her public-key A_e to Bob.
- Bob sends his public-key B_e to Alice.
- Alice computes $K_{AB} = f(A_e, A_d, B_e)$.
- Bob computes $K_{BA} = f(B_e, B_d, A_e)$.
- The protocol insures $K_{AB} = K_{BA}$ to be used by Alice and Bob as a shared secret.
- An intruder cannot compute this secret using A_e and B_e only.

Digital Signatures

Digital Signatures

- Alice establishes her binding to a message M by digitally signing it.

Digital Signatures

- Alice establishes her binding to a message M by digitally signing it.
- **Signing:** Only Alice has the capability to sign M.

Digital Signatures

- Alice establishes her binding to a message M by digitally signing it.
- **Signing:** Only Alice has the capability to sign M.
- **Verification:** Anybody can verify whether Alice's signature on M is valid.

Digital Signatures

- Alice establishes her binding to a message M by digitally signing it.
- **Signing:** Only Alice has the capability to sign M.
- **Verification:** Anybody can verify whether Alice's signature on M is valid.
- **Forging:** Nobody can forge signatures on behalf of Alice.

Digital Signatures

- Alice establishes her binding to a message M by digitally signing it.
- **Signing:** Only Alice has the capability to sign M.
- **Verification:** Anybody can verify whether Alice's signature on M is valid.
- **Forging:** Nobody can forge signatures on behalf of Alice.
- Digital signatures are based on public-key techniques.

Digital Signatures

- Alice establishes her binding to a message M by digitally signing it.
- **Signing:** Only Alice has the capability to sign M.
- **Verification:** Anybody can verify whether Alice's signature on M is valid.
- **Forging:** Nobody can forge signatures on behalf of Alice.
- Digital signatures are based on public-key techniques.
- Signature generation $\equiv$ Decryption (uses private key), and Signature verification $\equiv$ Encryption (uses public key).

Digital Signatures

- Alice establishes her binding to a message M by digitally signing it.
- **Signing:** Only Alice has the capability to sign M.
- **Verification:** Anybody can verify whether Alice's signature on M is valid.
- **Forging:** Nobody can forge signatures on behalf of Alice.
- Digital signatures are based on public-key techniques.
- Signature generation $\equiv$ Decryption (uses private key), and Signature verification $\equiv$ Encryption (uses public key).
- **Non-repudiation:** An entity should not be allowed to deny valid signatures made by him.

Signature With Message Recovery

Signature With Message Recovery

Generation

Signature With Message Recovery

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.

Signature With Message Recovery

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice signs M by her private key to obtain the signed message $S = f_s(M, K_d)$.

Signature With Message Recovery

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice signs M by her private key to obtain the signed message $S = f_s(M, K_d)$.

Verification

- Recover M from S by using Alice's public key: $M = f_v(S, K_e)$.

Signature With Message Recovery

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice signs M by her private key to obtain the signed message $S = f_s(M, K_d)$.

Verification

- Recover M from S by using Alice's public key: $M = f_v(S, K_e)$.

Forging

- $K_d' \neq K_d$ generates forged signature $S' = f_s(M, K_d')$. Verification yields $M' = f_v(S', K_e) \neq M$.

Signature With Message Recovery

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice signs M by her private key to obtain the signed message $S = f_s(M, K_d)$.

Verification

- Recover M from S by using Alice's public key: $M = f_v(S, K_e)$.

Forging

- $K_d' \neq K_d$ generates forged signature $S' = f_s(M, K_d')$. Verification yields $M' = f_v(S', K_e) \neq M$.

Drawback

- Algorithms are slow, not suitable for long messages.

Signature With Appendix

Signature With Appendix

Generation

Signature With Appendix

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.

Signature With Appendix

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice generates a short representative $m = H(M)$ of M.

Signature With Appendix

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice generates a short representative $m = H(M)$ of M.
- Alice uses her private-key: $s = f_s(m, K_d)$.

Signature With Appendix

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice generates a short representative $m = H(M)$ of M.
- Alice uses her private-key: $s = f_s(m, K_d)$.
- Alice publishes (M, s) as the signed message.

Signature With Appendix

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice generates a short representative $m = H(M)$ of M.
- Alice uses her private-key: $s = f_s(m, K_d)$.
- Alice publishes (M, s) as the signed message.

Verification

Signature With Appendix

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice generates a short representative $m = H(M)$ of M.
- Alice uses her private-key: $s = f_s(m, K_d)$.
- Alice publishes (M, s) as the signed message.

Verification

- Compute the representative $m = H(M)$.

Signature With Appendix

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice generates a short representative $m = H(M)$ of M.
- Alice uses her private-key: $s = f_s(m, K_d)$.
- Alice publishes (M, s) as the signed message.

Verification

- Compute the representative $m = H(M)$.
- Use Alice's public-key to generate $m' = f_v(s, K_e)$.

Signature With Appendix

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice generates a short representative $m = H(M)$ of M.
- Alice uses her private-key: $s = f_s(m, K_d)$.
- Alice publishes (M, s) as the signed message.

Verification

- Compute the representative $m = H(M)$.
- Use Alice's public-key to generate $m' = f_v(s, K_e)$.
- Accept the signature if and only if $m = m'$.

Signature With Appendix

Generation

- Alice generates a key-pair (K_e, K_d), publishes K_e, and keeps K_d secret.
- Alice generates a short representative $m = H(M)$ of M.
- Alice uses her private-key: $s = f_s(m, K_d)$.
- Alice publishes (M, s) as the signed message.

Verification

- Compute the representative $m = H(M)$.
- Use Alice's public-key to generate $m' = f_v(s, K_e)$.
- Accept the signature if and only if $m = m'$.

Forging

- Verification is expected to fail if a key $K'_d \neq K_d$ is used to generate s.

Entity Authentication

Entity Authentication

- Alice proves her identity to Bob.

Entity Authentication

- Alice proves her identity to Bob.
- Alice demonstrates to Bob her knowledge of a secret piece of information.

Entity Authentication

- Alice proves her identity to Bob.
- Alice demonstrates to Bob her knowledge of a secret piece of information.
- Alice may or may not reveal the secret itself to Bob.

Entity Authentication

- Alice proves her identity to Bob.
- Alice demonstrates to Bob her knowledge of a secret piece of information.
- Alice may or may not reveal the secret itself to Bob.
- Both symmetric and asymmetric techniques are used for entity authentication.

Entity Authentication

- Alice proves her identity to Bob.
- Alice demonstrates to Bob her knowledge of a secret piece of information.
- Alice may or may not reveal the secret itself to Bob.
- Both symmetric and asymmetric techniques are used for entity authentication.

- Simplest Example: **Passwords**

Entity Authentication

- Alice proves her identity to Bob.
- Alice demonstrates to Bob her knowledge of a secret piece of information.
- Alice may or may not reveal the secret itself to Bob.
- Both symmetric and asymmetric techniques are used for entity authentication.

- Simplest Example: **Passwords**
  - Time-invariant

Entity Authentication

- Alice proves her identity to Bob.
- Alice demonstrates to Bob her knowledge of a secret piece of information.
- Alice may or may not reveal the secret itself to Bob.
- Both symmetric and asymmetric techniques are used for entity authentication.

- Simplest Example: **Passwords**
  - Time-invariant
  - Secret revealed to the verifier

Entity Authentication

- Alice proves her identity to Bob.
- Alice demonstrates to Bob her knowledge of a secret piece of information.
- Alice may or may not reveal the secret itself to Bob.
- Both symmetric and asymmetric techniques are used for entity authentication.

- Simplest Example: **Passwords**
  - Time-invariant
  - Secret revealed to the verifier
  - Weak authentication

Challenge-response Authentication

Challenge-response Authentication

- Alice does not reveal her secret directly to Bob.

Challenge-response Authentication

- Alice does not reveal her secret directly to Bob.
- Bob generates a challenge C and sends C to Alice.

Challenge-response Authentication

- Alice does not reveal her secret directly to Bob.
- Bob generates a challenge C and sends C to Alice.
- Alice responds to C by sending a response R back to Bob.

Challenge-response Authentication

- Alice does not reveal her secret directly to Bob.
- Bob generates a challenge C and sends C to Alice.
- Alice responds to C by sending a response R back to Bob.
- Bob determines whether the response R is satisfactory.

Challenge-response Authentication

- Alice does not reveal her secret directly to Bob.
- Bob generates a challenge C and sends C to Alice.
- Alice responds to C by sending a response R back to Bob.
- Bob determines whether the response R is satisfactory.
- Generating R from C requires the knowledge of the secret.

Challenge-response Authentication

- Alice does not reveal her secret directly to Bob.
- Bob generates a challenge C and sends C to Alice.
- Alice responds to C by sending a response R back to Bob.
- Bob determines whether the response R is satisfactory.
- Generating R from C requires the knowledge of the secret.
- Absence of the knowledge of the secret fails to generate a satisfactory response with a good probability p.

Challenge-response Authentication

- Alice does not reveal her secret directly to Bob.
- Bob generates a challenge C and sends C to Alice.
- Alice responds to C by sending a response R back to Bob.
- Bob determines whether the response R is satisfactory.
- Generating R from C requires the knowledge of the secret.
- Absence of the knowledge of the secret fails to generate a satisfactory response with a good probability p.
- The above protocol may be repeated more than once.

Challenge-response Authentication

- Alice does not reveal her secret directly to Bob.
- Bob generates a challenge C and sends C to Alice.
- Alice responds to C by sending a response R back to Bob.
- Bob determines whether the response R is satisfactory.
- Generating R from C requires the knowledge of the secret.
- Absence of the knowledge of the secret fails to generate a satisfactory response with a good probability p.
- The above protocol may be repeated more than once.
- If Bob receives satisfactory response in every iteration, he accepts Alice's identity.

Challenge-response Authentication

- Alice does not reveal her secret directly to Bob.
- Bob generates a challenge C and sends C to Alice.
- Alice responds to C by sending a response R back to Bob.
- Bob determines whether the response R is satisfactory.
- Generating R from C requires the knowledge of the secret.
- Absence of the knowledge of the secret fails to generate a satisfactory response with a good probability p.
- The above protocol may be repeated more than once.
- If Bob receives satisfactory response in every iteration, he accepts Alice's identity.

Drawback

- C and R may reveal to Bob or an eavesdropper some knowledge about Alice's secret.

Zero-knowledge Protocol

Zero-knowledge Protocol

- A special class of challenge-response techniques.

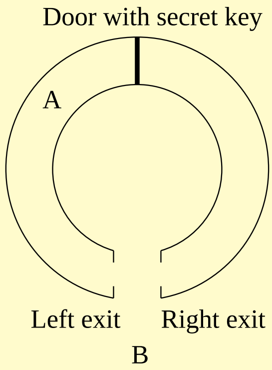
Zero-knowledge Protocol

- A special class of challenge-response techniques.
- No information is leaked to Bob or to any third party.

Zero-knowledge Protocol

- A special class of challenge-response techniques.
- No information is leaked to Bob or to any third party.

A real-life example



Secret Sharing

Secret Sharing

- A secret is distributed to n parties.

Secret Sharing

- A secret is distributed to n parties.
- All of these n parties should cooperate to reconstruct the secret.

Secret Sharing

- A secret is distributed to n parties.
- All of these n parties should cooperate to reconstruct the secret.
- Participation of only $\leqslant n-1$ parties should fail to reconstruct the secret.

Secret Sharing

- A secret is distributed to n parties.
- All of these n parties should cooperate to reconstruct the secret.
- Participation of only $\leqslant n-1$ parties should fail to reconstruct the secret.

Generalization

Secret Sharing

- A secret is distributed to n parties.
- All of these n parties should cooperate to reconstruct the secret.
- Participation of only $\leqslant n-1$ parties should fail to reconstruct the secret.

Generalization

- Any m (or more) parties can reconstruct the secret (for some $m \leqslant n$).

Secret Sharing

- A secret is distributed to n parties.
- All of these n parties should cooperate to reconstruct the secret.
- Participation of only $\leqslant n - 1$ parties should fail to reconstruct the secret.

Generalization

- Any m (or more) parties can reconstruct the secret (for some $m \leqslant n$).
- Participation of only $\leqslant m - 1$ parties should fail to reconstruct the secret.

Cryptographic Hash Functions

Cryptographic Hash Functions

- Used to convert strings of any length to strings of a fixed length.

Cryptographic Hash Functions

- Used to convert strings of any length to strings of a fixed length.
- Used for the generation of (short) representatives of messages.

Cryptographic Hash Functions

- Used to convert strings of any length to strings of a fixed length.
- Used for the generation of (short) representatives of messages.
- Symmetric techniques are typically used for designing hash functions.

Cryptographic Hash Functions

- Used to convert strings of any length to strings of a fixed length.
- Used for the generation of (short) representatives of messages.
- Symmetric techniques are typically used for designing hash functions.

Modification detection code (MDC)

- An unkeyed hash function is used to guard against unauthorized/accidental message alterations. Signature schemes also use MDC's.

Cryptographic Hash Functions

- Used to convert strings of any length to strings of a fixed length.
- Used for the generation of (short) representatives of messages.
- Symmetric techniques are typically used for designing hash functions.

Modification detection code (MDC)

- An unkeyed hash function is used to guard against unauthorized/accidental message alterations. Signature schemes also use MDC's.

Message authentication code (MAC)

- A keyed hash function is used to authenticate the source of messages.

Cryptographic Hash Functions: Properties

Cryptographic Hash Functions: Properties

- A **collision** for a hash function H is a pair of two distinct strings x, y with $H(x) = H(y)$. Collisions must exist for any hash function.

Cryptographic Hash Functions: Properties

- A **collision** for a hash function H is a pair of two distinct strings x, y with $H(x) = H(y)$. Collisions must exist for any hash function.

First pre-image resistance

- For most hash values y, it should be difficult to find a string x with $H(x) = y$.

Cryptographic Hash Functions: Properties

- A **collision** for a hash function H is a pair of two distinct strings x, y with $H(x) = H(y)$. Collisions must exist for any hash function.

First pre-image resistance

- For most hash values y, it should be difficult to find a string x with $H(x) = y$.

Second pre-image resistance

- Given a string x, it should be difficult to find a different string x' with $H(x') = H(x)$.

Cryptographic Hash Functions: Properties

- A **collision** for a hash function H is a pair of two distinct strings x, y with $H(x) = H(y)$. Collisions must exist for any hash function.

First pre-image resistance

- For most hash values y, it should be difficult to find a string x with $H(x) = y$.

Second pre-image resistance

- Given a string x, it should be difficult to find a different string x' with $H(x') = H(x)$.

Collision resistance

- It should be difficult to find two distinct strings x, x' with $H(x) = H(x')$.

Digital Certificates

Digital Certificates

- A **public-key certificate** insures that a public key actually belongs to an entity.

Digital Certificates

- A **public-key certificate** insures that a public key actually belongs to an entity.
- Certificates are issued by a trusted **Certification Authority** (CA).

Digital Certificates

- A **public-key certificate** insures that a public key actually belongs to an entity.
- Certificates are issued by a trusted **Certification Authority** (CA).
- A certificate consists of a public key and other additional information about the owner of the key.

Digital Certificates

- A **public-key certificate** insures that a public key actually belongs to an entity.
- Certificates are issued by a trusted **Certification Authority** (CA).
- A certificate consists of a public key and other additional information about the owner of the key.
- The authenticity of a certificate is achieved by the digital signature of the CA on the certificate.

Digital Certificates

- A **public-key certificate** insures that a public key actually belongs to an entity.
- Certificates are issued by a trusted **Certification Authority** (CA).
- A certificate consists of a public key and other additional information about the owner of the key.
- The authenticity of a certificate is achieved by the digital signature of the CA on the certificate.
- Compromised certificates are revoked and a **certificate revocation list** (CRL) is maintained by the CA.

Digital Certificates

- A **public-key certificate** insures that a public key actually belongs to an entity.
- Certificates are issued by a trusted **Certification Authority** (CA).
- A certificate consists of a public key and other additional information about the owner of the key.
- The authenticity of a certificate is achieved by the digital signature of the CA on the certificate.
- Compromised certificates are revoked and a **certificate revocation list** (CRL) is maintained by the CA.
- If a certificate is not in the CRL, and the signature of the CA on the certificate is verified, one gains the desired confidence of treating the public-key as authentic.

Attacks on Cryptosystems

Attacks on Cryptosystems

- **Partial breaking of a cryptosystem**
  The attacker succeeds in decrypting some ciphertext messages, but without any guarantee that this capability would help him break new ciphertext messages in future.

Attacks on Cryptosystems

- **Partial breaking of a cryptosystem**
  The attacker succeeds in decrypting some ciphertext messages, but without any guarantee that this capability would help him break new ciphertext messages in future.
- **Complete breaking of a cryptosystem**
  The attacker possesses the capability of decrypting any ciphertext message. This may be attributed to a knowledge of the decryption key(s).

Attacks on Cryptosystems

- **Partial breaking of a cryptosystem**
  The attacker succeeds in decrypting some ciphertext messages, but without any guarantee that this capability would help him break new ciphertext messages in future.
- **Complete breaking of a cryptosystem**
  The attacker possesses the capability of decrypting any ciphertext message. This may be attributed to a knowledge of the decryption key(s).
- **Passive attack**
  The attacker only intercepts messages meant for others.

Attacks on Cryptosystems

- **Partial breaking of a cryptosystem**
  The attacker succeeds in decrypting some ciphertext messages, but without any guarantee that this capability would help him break new ciphertext messages in future.
- **Complete breaking of a cryptosystem**
  The attacker possesses the capability of decrypting any ciphertext message. This may be attributed to a knowledge of the decryption key(s).
- **Passive attack**
  The attacker only intercepts messages meant for others.
- **Active attack**
  The attacker alters and/or deletes messages and even creates unauthorized messages.

Types of Passive Attack

Types of Passive Attack

- **Ciphertext-only attack:** The attacker has no control/knowledge of the ciphertexts and the corresponding plaintexts. This is the most difficult (but practical) attack.

Types of Passive Attack

- **Ciphertext-only attack:** The attacker has no control/knowledge of the ciphertexts and the corresponding plaintexts. This is the most difficult (but practical) attack.
- **Known plaintext attack:** The attacker knows some plaintext-ciphertext pairs. Easily mountable in public-key systems.

Types of Passive Attack

- **Ciphertext-only attack:** The attacker has no control/knowledge of the ciphertexts and the corresponding plaintexts. This is the most difficult (but practical) attack.
- **Known plaintext attack:** The attacker knows some plaintext-ciphertext pairs. Easily mountable in public-key systems.
- **Chosen plaintext attack:** A known plaintext attack where the plaintext messages are chosen by the attacker.

Types of Passive Attack

- **Ciphertext-only attack:** The attacker has no control/knowledge of the ciphertexts and the corresponding plaintexts. This is the most difficult (but practical) attack.
- **Known plaintext attack:** The attacker knows some plaintext-ciphertext pairs. Easily mountable in public-key systems.
- **Chosen plaintext attack:** A known plaintext attack where the plaintext messages are chosen by the attacker.
- **Adaptive chosen plaintext attack:** A chosen plaintext attack where the plaintext messages are chosen adaptively by the attacker.

Types of Passive Attack (contd.)

Types of Passive Attack (contd.)

- **Chosen ciphertext attack:** A known plaintext attack where the ciphertext messages are chosen by the attacker. Mountable if the attacker gets hold of the victim's decryption device.

Types of Passive Attack (contd.)

- **Chosen ciphertext attack:** A known plaintext attack where the ciphertext messages are chosen by the attacker. Mountable if the attacker gets hold of the victim's decryption device.
- **Adaptive chosen ciphertext attack:** A chosen ciphertext attack where the ciphertext messages are chosen adaptively by the attacker.

Attacks on Digital Signatures

Attacks on Digital Signatures

- **Total break:** An attacker knows the signing key or has a function that is equivalent to the signature generation transformation.

Attacks on Digital Signatures

- **Total break:** An attacker knows the signing key or has a function that is equivalent to the signature generation transformation.
- **Selective forgery:** An attacker can generate signatures (without the participation of the legitimate signer) on a set of messages chosen by the attacker.

Attacks on Digital Signatures

- **Total break:** An attacker knows the signing key or has a function that is equivalent to the signature generation transformation.
- **Selective forgery:** An attacker can generate signatures (without the participation of the legitimate signer) on a set of messages chosen by the attacker.
- **Existential forgery:** The attacker can generate signatures on certain messages over which the attacker has no control.

Attacks on Digital Signatures (contd.)

Attacks on Digital Signatures (contd.)

- **Key-only attack:** The attacker knows only the verification (public) key of the signer. This is the most difficult attack to mount.

Attacks on Digital Signatures (contd.)

- **Key-only attack:** The attacker knows only the verification (public) key of the signer. This is the most difficult attack to mount.
- **Known message attack:** The attacker knows some messages and the signatures of the signer on these messages.

Attacks on Digital Signatures (contd.)

- **Key-only attack:** The attacker knows only the verification (public) key of the signer. This is the most difficult attack to mount.
- **Known message attack:** The attacker knows some messages and the signatures of the signer on these messages.
- **Chosen message attack:** This is similar to the known message attack except that the messages for which the signatures are known are chosen by the attacker.

Attacks on Digital Signatures (contd.)

- **Key-only attack:** The attacker knows only the verification (public) key of the signer. This is the most difficult attack to mount.
- **Known message attack:** The attacker knows some messages and the signatures of the signer on these messages.
- **Chosen message attack:** This is similar to the known message attack except that the messages for which the signatures are known are chosen by the attacker.
- **Adaptive chosen message attack:** The messages to be signed are adaptively chosen by the attacker.