

3. Cryptography

3.1 Basic Cryptography

3.1.1 Symmetric Cryptosystems

3.1.2 Public Key Cryptography

3.1.3 Cryptographic Checksums

3.1.4 Digital Signature

3.1.5 Hashing

3.2 Key Management

3.2.1 Session and Interchange Keys

3.2.2 Key Exchange and Generation

3.2.3 Cryptographic Key Infrastructures

3.2.4 Storing and Revoking Keys

3.3 Cipher Techniques

3.3.1 Stream and Block Ciphers

3.3.2 Authenticated Encryption

3.4 Authentication

3.4.1 Authentication Basics

3.4.2 Passwords

3.4.3 Password Selection


3.4.4 Attacking Passwords


3.4.5 Password Aging

3.4.6 Biometrics

3.4.7 Multifactor Authentication

Practical Works

 **Write program to create cipher text**

 **Write program to validate strong**

Password

Cryptography

Cryptography is the **codes to protect data and communications** so only the intended receivers can decode and understand them. Consequently, restricting access to information from outside parties.

"Crypto" indicates "hidden," and "graphy" indicates "writing," respectively. The techniques used in **cryptography to secure data are based on mathematical principles and a set of rule-based calculations known as algorithms** to modify signals in a way that makes them challenging to decode.

These algorithms generate cryptographic keys, create digital signatures, safeguard data privacy, enable online browsing on the Internet, and ensure the confidentiality of private transactions like credit and debit card payments.

3.1 Basic Cryptography

Cryptography aims to keep data and messages private and inaccessible to possible threats or bad actors. It frequently works invisibly to encrypt and decrypt the data you send through email, social media, applications, and website interactions.

There are several uses for symmetric cryptography, including:

- Payment applications and card transactions
- Random number generation
- Verify the sender's signature to be sure they are who they claim they are

There are several uses for asymmetric cryptography, including:

- Email messages
- SIM card authentication
- Web security
- Exchange of private keys

3.1.1 Symmetric Cryptosystems

With the encryption technique, the sender and the recipient use the same shared key to encrypt and decrypt messages.

Although symmetric key systems are quicker and easier to use, they have the drawback of requiring a secure key exchange between the sender and the receiver. Data Encryption System (DES) is the most widely used symmetric key encryption method

3.1.2 Public Key Cryptography

PKC stands for Public Key Cryptography. It is also known as asymmetric cryptography. It is an encryption technique or a framework that uses a pair of keys (public and private key) for secure data communication. This approach uses a set of keys to encrypt and decrypt data. Public keys are used for encryption, whereas private keys are used for decryption.

Symmetric vs. asymmetric encryption

Symmetric encryption



Asymmetric encryption



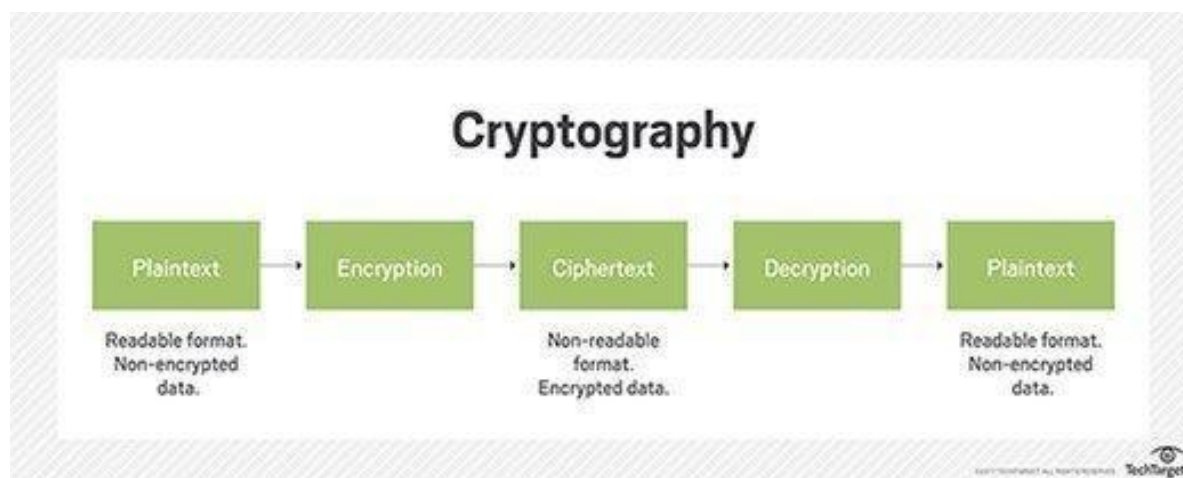
3.1.3 Cryptographic Checksums

What is a cryptographic checksum?

Generated by a cryptographic [algorithm](#), a cryptographic checksum is a mathematical value assigned to a file sent through a network for verifying that the data contained in that file is unchanged. The algorithm performs numerous mathematical operations to create a [hash](#) value, or fixed string of digits. This hash value is then used as a [checksum](#) to confirm that the sent file was not changed by an [attacker](#).

A hash value remains unchanged from the time it is created and is considered an "electronic fingerprint" of a file. A cryptographic checksum is assigned to a file and is used to verify that the data in that file has not been tampered with or manipulated, possibly by a malicious entity.

Cryptographic checksums provide the basis of modern [cryptography](#), particularly for signing and [encryption](#), [digital signatures](#), email certificates and website certificates. They are also known as *message [authentication](#) codes*, *integrity check values*, *modification detection codes* or *message integrity codes*.



Cryptographic checksums form the foundation of cryptography today.

How a cryptographic checksum works

A cryptographic checksum is based on hash functions that provide hash values -- also known as *hash codes* -- for every file. The cryptographic hash function takes an input and produces a fixed-length sequence of numbers and letters. The checksum is of the same length, regardless of the original file's size.

When a user creates a file and makes copies of it, the file always has the same hash code. If as much as one [bit](#) of information in the file changes -- say, because it was manipulated by an eavesdropper or data thief -- a different hash code/checksum is generated.

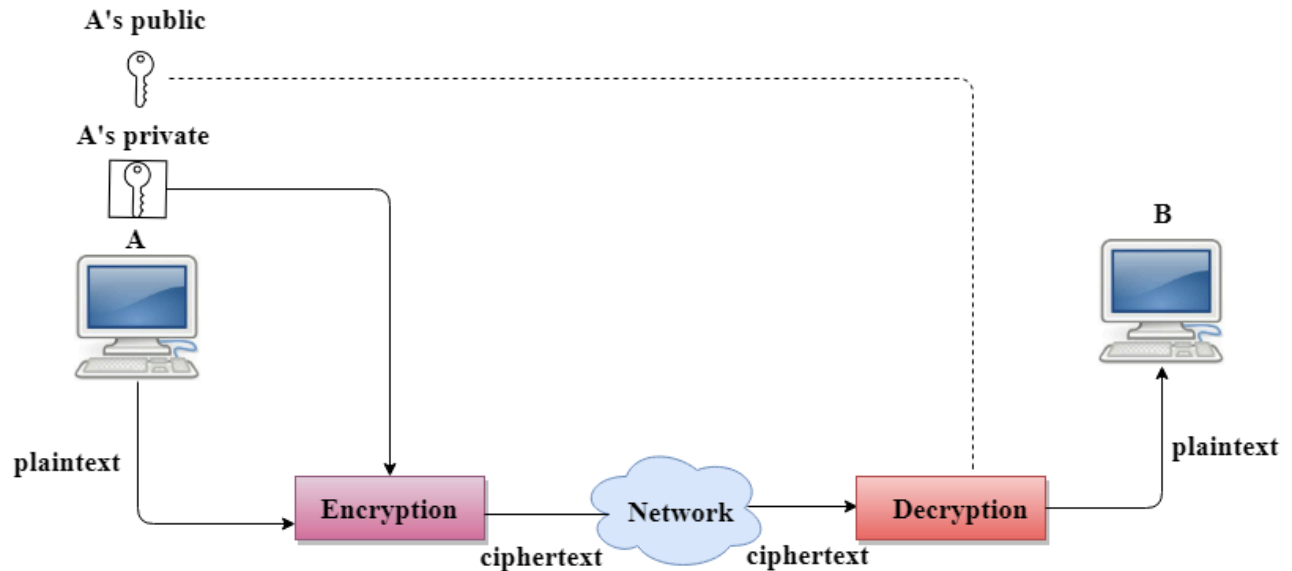
3.1.4 Digital Signature

The Digital Signature is a technique which is used to validate the authenticity and integrity of the message. We know that there are four aspects of security: privacy, authentication, integrity, and non-repudiation. We have already discussed the first aspect of security and other three aspects can be achieved by using a digital signature.

The basic idea behind the Digital Signature is to sign a document. When we send a document electronically, we can also sign it. We can sign a document in two ways: to sign a whole document and to sign a digest.

Signing the Whole Document

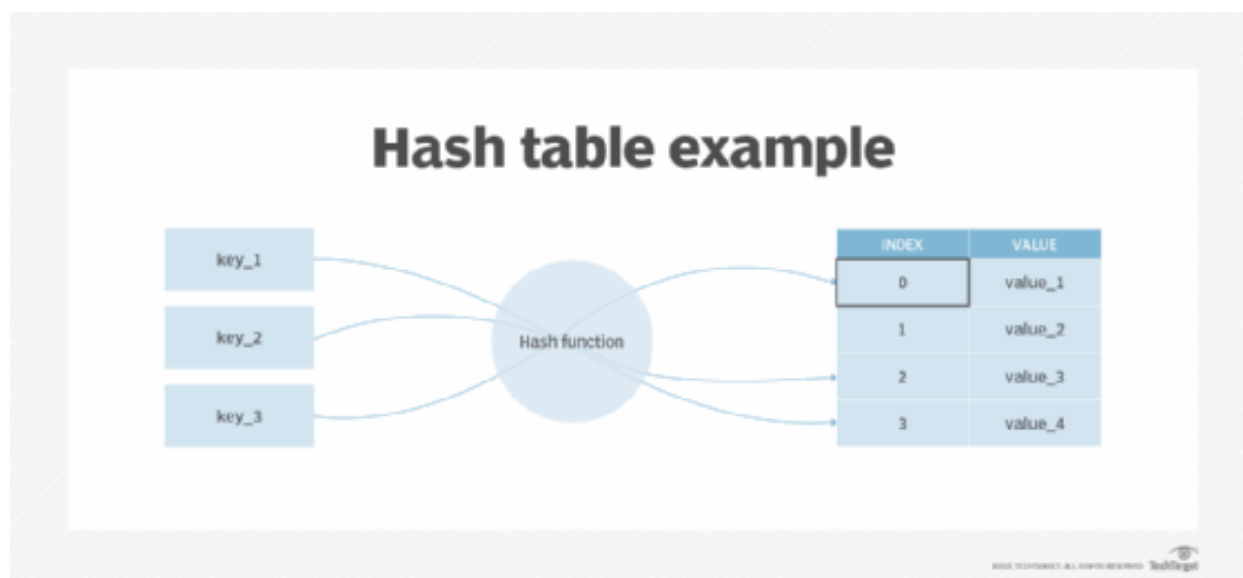
- In Digital Signature, a public key encryption technique is used to sign a document. However, the roles of a public key and private key are different here. The sender uses a private key to encrypt the message while the receiver uses the public key of the sender to decrypt the message.
- In Digital Signature, the private key is used for encryption while the public key is used for decryption.
- Digital Signature cannot be achieved by using secret key encryption.



Digital Signature is used to achieve the following three aspects:

- **Integrity:** The Digital Signature preserves the integrity of a message because, if any malicious attack intercepts a message and partially or totally changes it, then the decrypted message would be impossible.
- **Authentication:** We can use the following reasoning to show how the message is authenticated. If an intruder (user X) sends a message pretending that it is coming from someone else (user A), user X uses her own private key to encrypt the message. The message is decrypted by using the public key of user A. Therefore this makes the message unreadable. Encryption with X's private key and decryption with A's public key results in garbage value.
- **Non-Repudiation:** Digital Signature also provides non-repudiation. If the sender denies sending the message, then her private key corresponding to her public key is tested on the plaintext. If the decrypted message is the same as the original message, then we know that the sender has sent the message.

3.1.5 Hashing



Cryptographic checksums are based on hashing.

The hash function checksum procedure ensures that the files sent during communication return the same hash code for the sender and the receiver. If the hash code changes, any damage or manipulation can be easily identified.

If the checksum of the original file is known, an authorized user can run a checksum/ hashing [utility](#) on the file to match the resulting checksum to the original checksum. If these two checksums match, the file is identical. However, if they don't match, the user can identify a fake version of the original file.

Checksums are used to check files and other data for errors or manipulation that might have occurred during data transmission or [storage](#).

3.2 Key Management

Key management define as managing cryptographic keys within a cryptosystem. It can manage with generating, exchanging, saving, using and replacing keys as required at the user level.

A key management system will also contains key servers, user process and protocols, including cryptographic protocol design. The security of the cryptosystem is based upon successful key management.

Cryptographic keys play an essential role in protecting sensitive data, avoiding data breaches, and understanding with regulations. Unfortunately, a lost or stolen key can lead to costly losses of systems and information, which is each security-aware company should provide strong key management protocols.

3.2.1 Session and Interchange Keys

Session keys and interchange keys are both cryptographic keys used in secure communication protocols, but they serve different purposes and have distinct characteristics.

1. Session Key:

- A session key is a temporary cryptographic key used to encrypt data exchanged during a specific communication session between two or more parties.
- It is typically generated dynamically for each session and is used exclusively for that session.
- Session keys are often derived from a more secure long-term key, such as a master key or a key exchanged during a key establishment protocol (e.g., Diffie-Hellman key exchange).
- They provide confidentiality and sometimes integrity for the data transmitted during the session.
- Once the session ends, the session key is usually discarded, which helps limit the potential impact of key compromise.

2. Interchange Key:

- An interchange key, also known as a long-term key or static key, is a cryptographic key used for long-term communication or data protection needs.
- Unlike session keys, interchange keys are typically not generated dynamically for each session; instead, they are established and used over an extended period.
- Interchange keys are often used for key exchange mechanisms, such as RSA or Elliptic Curve Cryptography (ECC), to securely exchange session keys or other sensitive data.
- They are also used for digital signatures, certificate signing, and other cryptographic operations requiring long-term key stability.
- Interchange keys require careful management and protection because compromising them can lead to widespread security breaches.

3.2.2 Key Exchange and Generation

Diffie-Hellman algorithm is one of the most important algorithms used for establishing a shared secret. At the time of exchanging data over a public network, we can use the shared secret for secret communication. We use an elliptic curve for generating points and getting a secret key using the parameters.

1. We will take four variables, i.e., **P (prime)**, **G (the primitive root of P)**, and **a and b (private values)**.
2. The variables **P** and **G** both are publicly available. The sender selects a private value, either **a** or **b**, for generating a key to exchange publicly. The receiver receives the key, and that generates a secret key, after which the sender and receiver both have the same secret key to encrypt.

Let's understand the process step by step for user1 (sender) and user2 (receiver):

Steps	User1	User2
1.	P, G => available public keys.	P, G => available public keys.
2.	a is selected as a private key.	b is selected as a private key.
3.	Eq. to generate key: $x = G^a \text{ mod } P$	Eq. to generate key: $y = G^b \text{ mod } P$
4.	After exchanging keys, user1 receives key y.	After exchanging keys, user2 receives key x.
5.	User1 generates a secret key by using the received key y: $k_a = y^a \text{ mod } P$	User2 generates a secret key by using the received key x: $k_b = x^b \text{ mod } P$

Algebraically, 5th step can be shown as follows:

$$k_a = k_b$$

It means that both the users have the symmetric secret key to encrypt.

Example:

1. User1 and User2 get public keys $P = 33$ and $G = 8$.
2. User1 selects a as a private key, i.e., 3, and User2 selects b as a private key, i.e., 2.
3. User1 calculate the public value:
 $x = (8^3 \text{ mod } 33) = 512 \text{ mod } 33 = 17$
4. User2 calculate the public value:
 $y = (8^2 \text{ mod } 33) = 64 \text{ mod } 33 = 31$
5. User1 and User2 exchange public keys, i.e., 17 and 31.
6. User1 receives public key $y = 31$ and User2 receives public key $x = 17$.
7. User1 and User2 calculate symmetric keys:
User1: $k_a = y_a \text{ mod } P = 31_3 \text{ mod } 33 = 29791 \text{ mod } 33 = 25$
User2: $k_b = x_b \text{ mod } P = 17_2 \text{ mod } 33 = 289 \text{ mod } 33 = 25$
8. 25 is the shared secret.

3.2.3 Cryptographic Key Infrastructures

A Cryptographic Key Infrastructure (CKI), often referred to as a Public Key Infrastructure (PKI), is a framework that manages the creation, distribution, usage, storage, and revocation of cryptographic keys and digital certificates. It provides a secure way for entities to communicate over insecure channels, ensuring confidentiality, integrity, authentication, and non-repudiation.

The main components of public key infrastructure include the following:

- **Certificate authority (CA):** The CA is a trusted entity that issues, stores, and signs the digital certificate. The CA signs the digital certificate with their own private key and then publishes the public key that can be accessed upon request.
- **Registration authority (RA):** The RA verifies the identity of the user or device requesting the digital certificate. This can be a third party, or the CA can also act as the RA.
- **Certificate database:** This database stores the digital certificate and its metadata, which includes how long the certificate is valid.

- **Central directory:** This is the secure location where the cryptographic keys are indexed and stored.
- **Certificate management system:** This is the system for managing the delivery of certificates as well as access to them.
- **Certificate policy:** This policy outlines the procedures of the PKI. It can be used by outsiders to determine the PKI's trustworthiness.

3.2.4 Storing and Revoking Keys

In cryptography, the secure storage and management of keys are crucial aspects to ensure the security of encrypted data. Here's a brief overview of storing and revoking keys:

Storing Keys:

1. **Use Secure Key Management Systems (KMS):** Implement dedicated systems designed to securely generate, store, and manage cryptographic keys. These systems often employ hardware security modules (HSMs) for added protection.
2. **Encryption of Keys:** Keys themselves should be encrypted when stored, using a master key or passphrase. This adds an extra layer of security, ensuring that even if the storage is compromised, the keys remain inaccessible.
3. **Access Controls:** Implement strict access controls to limit who can retrieve or modify keys. Role-based access control (RBAC) and multi-factor authentication (MFA) can enhance security.
4. **Regular Key Rotation:** Periodically rotate keys to limit the exposure window in case of a compromise. This practice reduces the impact of key leakage.
5. **Secure Backup:** Backup copies of keys should be stored securely, preferably in a different location or on a different medium. This ensures availability in case of disasters but prevents unauthorized access.

Revoking Keys:

1. **Key Expiration:** Assign expiration dates to keys. When a key is expired, it's considered revoked. Users or systems relying on these keys must switch to newer ones.
2. **Key Compromise:** If a key is compromised or suspected to be compromised, it should be immediately revoked. This involves updating access controls to deny further use of the compromised key.
3. **Certificate Revocation Lists (CRLs):** In systems that use digital certificates, maintain a list of revoked certificates known as Certificate Revocation Lists (CRLs). These lists are regularly published and checked by relying parties to ensure revoked certificates (and their associated keys) are not used.
4. **Key Management Systems:** Many key management systems provide mechanisms for revoking keys. This can involve marking a key as revoked within the system, triggering automated processes to propagate this information to relevant systems.
5. **Communicate Revocations:** Ensure that relevant stakeholders are informed about key revocations. This includes users, administrators, and any systems or services that rely on the revoked keys.
6. **Auditing:** Regularly audit key usage and revocation processes to ensure effectiveness and compliance with security policies.

3.3 Cipher Techniques

Cipher techniques, also known as encryption techniques, are methods used to encode information in such a way that only authorized parties can read it. These techniques are crucial for ensuring the confidentiality and integrity of sensitive data. Here are some common cipher techniques:

1. **Substitution Ciphers:** In substitution ciphers, each letter in the plaintext is replaced with another letter or symbol according to a predetermined scheme.
2. **Transposition Ciphers:** Transposition ciphers involve rearranging the order of letters in the plaintext to form the ciphertext without changing the actual letters themselves.
3. **Polyalphabetic Ciphers:** These ciphers use multiple alphabets to encode the plaintext, making them more secure than simple substitution ciphers.
4. **Stream Ciphers:** Stream ciphers encrypt plaintext one bit or byte at a time, using a stream of pseudorandom or truly random characters. They are often used for real-time encryption.
5. **Block Ciphers:** Block ciphers encrypt blocks of plaintext of fixed size, typically 64 or 128 bits.
6. **Public Key Cryptography:** Public key cryptography uses a pair of keys, a public key for encryption and a private key for decryption.

3.3.1 Stream and Block Ciphers

Stream ciphers and block ciphers are two fundamental types of symmetric encryption algorithms used to encrypt data. They differ in how they process the plaintext and produce ciphertext.

1. **Stream Ciphers:**
 - **Description:** Stream ciphers encrypt plaintext one bit or byte at a time, typically using a keystream generated from a relatively short key. The keystream is combined with the plaintext using bitwise XOR (exclusive OR) operation.
 - **Operation:** The key stream generator produces a stream of pseudo-random bits based on the key. This stream is then XOR with the plaintext to produce the ciphertext.
 - **Advantages:**
 - Well-suited for real-time encryption and communication applications.
 - Often more efficient in terms of computational resources compared to block ciphers.
 - **Disadvantages:**
 - Vulnerable to certain attacks, especially if the same keystream is used more than once (known as a "stream reuse" attack).
 - Key management can be challenging, as the keystream must be as long as the plaintext and should never be reused with the same key.
2. **Block Ciphers:**
 - **Description:** Block ciphers encrypt fixed-size blocks of plaintext (e.g., 64 or 128 bits) at a time. They use a key to transform each block independently into ciphertext.
 - **Operation:** The plaintext is divided into fixed-size blocks, and each block is processed through multiple rounds of substitution and permutation using the key.
 - **Advantages:**
 - Generally more secure than stream ciphers against certain types of attacks.

- Provide a wider range of cryptographic modes, allowing for diverse applications.
- **Disadvantages:**
 - Can be less efficient for real-time applications compared to stream ciphers, especially when dealing with large volumes of data.
 - Block size limitations may pose challenges for encrypting data of arbitrary lengths.

3.3.2 Authenticated Encryption

Authenticated Encryption (AE) is a cryptographic process that combines encryption with authentication in a single operation. It ensures both the confidentiality and integrity of data being transmitted or stored. The primary goal of authenticated encryption is to prevent unauthorized modification of encrypted data and to detect any attempts at tampering.

There are two main types of authenticated encryption:

1. **Encrypt-then-MAC (EtM):**
 - In this approach, the plaintext is first encrypted using a symmetric encryption algorithm, and then a message authentication code (MAC) is generated over the ciphertext and attached to it.
 - The MAC ensures the integrity and authenticity of the ciphertext by using a cryptographic hash function along with a secret key.
 - Decrypting the ciphertext involves first verifying the MAC before decrypting the ciphertext. If the MAC verification fails, decryption is not performed, preventing the use of tampered data.
2. **Authenticated Encryption with Associated Data (AEAD):**
 - AEAD schemes provide a more integrated approach by allowing additional associated data (AAD) to be authenticated along with the plaintext.
 - AAD may include metadata or contextual information that needs to be authenticated but not encrypted. For example, protocol headers or timestamps.
 - The encryption process incorporates both the plaintext and the AAD into the generation of ciphertext, ensuring their integrity and confidentiality.
 - Popular AEAD schemes include GCM (Galois/Counter Mode) and CCM (Counter with CBC-MAC).

3.4 Authentication

Authentication is a fundamental concept in computer security and cryptography. It refers to the process of verifying the identity of a user, system, or entity to ensure that they are who they claim to be. Authentication mechanisms are crucial for controlling access to resources, protecting sensitive information, and preventing unauthorized activities.

3.4.1 Authentication Basics

Here are some basic concepts related to authentication:

1. **Identification:** Users or entities typically provide a unique identifier, such as a username, email address, or account number, to indicate who they are.
2. **Authentication Factors:**
 - **Knowledge factors:** Something the user knows, such as a password, PIN, passphrase, or answers to security questions.
 - **Possession factors:** Something the user possesses, such as a physical token (smart card, USB token) or a virtual token (software-based token, mobile device).
 - **Inherence factors:** Something inherent to the user, such as biometric characteristics (fingerprint, facial recognition, iris scan).
 - **Location factors:** The geographic location or network address of the user can also be used for authentication.
3. **Authentication Methods:**
 - **Single-factor authentication:** Relies on only one authentication factor, such as a password or biometric scan.
 - **Multi-factor authentication (MFA):** Requires two or more authentication factors to verify identity, providing an additional layer of security.
 - **Biometric authentication:** Uses unique biological characteristics to verify identity.
 - **Token-based authentication:** Involves the use of physical or virtual tokens.
 - **Certificate-based authentication:** Utilizes digital certificates issued by a trusted authority.
4. **Authentication Protocols and Standards:**
 - **Kerberos:** A network authentication protocol used for secure authentication in a distributed environment.
 - **OAuth:** An authorization framework allowing third-party applications limited access to a user's resources without sharing their credentials.
 - **OpenID Connect:** An authentication layer built on top of OAuth 2.0, enabling authentication and single sign-on capabilities.
 - **LDAP (Lightweight Directory Access Protocol):** Used for accessing and managing directory information, often for centralized authentication.
5. **Session Management:** After successful authentication, a session is established, allowing the user or system to interact with the resources or services. Session management includes maintaining the session state, handling session timeouts, and securely managing session tokens.

3.4.2 Passwords

As an AI developed by OpenAI, I don't have access to specific passwords, and it's important to maintain the security of your passwords by not sharing them with anyone, including AI assistants like myself. However, I can offer guidance on creating strong and secure passwords.

3.4.3 Password Selection

Selecting strong passwords is crucial for maintaining the security of accounts and systems. Here are some guidelines and best practices for choosing secure passwords:

1. **Length:** Choose passwords that are long enough to provide sufficient security. Longer passwords are generally more resistant to brute force attacks. Aim for a minimum length of 12-16 characters.
2. **Complexity:** Use a combination of uppercase and lowercase letters, numbers, and special characters (e.g., !, @, #, \$, %). Avoid using common words, phrases, or predictable patterns.
3. **Randomness:** Generate passwords that are random and unpredictable. Avoid using easily guessable information such as birthdays, names, or sequential characters.
4. **Passphrases:** Consider using passphrases instead of passwords. Passphrases are longer phrases or sentences composed of random words or meaningful phrases. They are easier to remember and often more secure than traditional passwords. For example, "correct horse battery staple" is a passphrase recommended by security experts.
5. **Avoid Dictionary Words:** Avoid using common words found in dictionaries or easily guessable phrases. Attackers often use dictionary attacks to crack passwords by systematically trying words from a list.
6. **Unique Passwords:** Use unique passwords for each account or service. Reusing passwords across multiple accounts increases the risk of compromise if one account is breached.

3.4.4 Attacking Passwords

Attacking passwords is a common method used by malicious actors to gain unauthorized access to accounts, systems, or data. There are several techniques and strategies employed in password attacks. Here are some common methods used to attack passwords:

1. **Brute Force Attack:**
 - In a brute force attack, the attacker systematically tries all possible combinations of characters until the correct password is found.
 - Brute force attacks can be time-consuming and resource-intensive, especially for longer and more complex passwords.
 - Attackers may use automated tools or scripts to speed up the process of trying different combinations.
2. **Dictionary Attack:**
 - A dictionary attack involves using a precompiled list of commonly used passwords, words from dictionaries, or previously breached passwords.
 - Attackers try each word or password from the dictionary list until they find a match.
 - Dictionary attacks are more efficient than brute force attacks and can be effective against weak or easily guessable passwords.
3. **Rainbow Table Attack:**
 - Rainbow table attacks exploit weaknesses in password hashing algorithms by precomputing hashes for a large number of possible passwords and storing them in a table.
 - When attacking a hashed password, the attacker looks up the hash value in the rainbow table to find the corresponding plaintext password.
 - Rainbow table attacks are effective against unsalted hashes and can quickly recover passwords if the hash value is found in the table.
4. **Phishing:**

- Phishing attacks involve tricking users into revealing their passwords or sensitive information through deceptive emails, websites, or messages.
 - Attackers may create fake login pages that mimic legitimate websites to steal users' credentials when they enter their passwords.
 - Phishing attacks rely on social engineering techniques to manipulate users into disclosing their passwords willingly.
5. **Keylogging:**
- Keylogging involves capturing keystrokes typed by users, including their passwords, using malicious software installed on the victim's computer or device.
 - Keyloggers record all keyboard input, allowing attackers to capture passwords as users type them.
6. **Social Engineering:**
- Social engineering attacks exploit human psychology to manipulate individuals into divulging their passwords or sensitive information.
 - Attackers may impersonate trusted individuals or use persuasive tactics to convince users to disclose their passwords voluntarily.
7. **Credential Stuffing:**
- Credential stuffing attacks involve using automated scripts or tools to systematically test large collections of username-password pairs obtained from previous data breaches on various websites or services.
 - Attackers exploit the tendency of users to reuse passwords across multiple accounts, hoping to find matches and gain unauthorized access.

3.4.5 Password Aging

Password aging, also known as password expiration or password rotation, is a security practice that involves setting a predefined period after which users are required to change their passwords. The primary goal of password aging is to enhance security by reducing the risk of unauthorized access resulting from compromised or guessed passwords. Here are some key aspects of password aging:

1. **Frequency:** Password aging policies typically specify a specific time interval, such as 30, 60, or 90 days, after which users must change their passwords.
2. **Enforcement:** Password aging policies are enforced by the system or application, which prompts users to change their passwords when the expiration period is reached. Users may receive notifications or reminders to update their passwords before they expire.
3. **Password Complexity:** When changing passwords, users are often required to adhere to password complexity requirements, such as using a minimum length, including a mix of uppercase and lowercase letters, numbers, and special characters.

3.4.6 Biometrics

Biometrics refers to the measurement and analysis of unique physical or behavioral characteristics of individuals for the purpose of identification or authentication. Biometric authentication relies on the premise that these characteristics are distinctive and difficult to forge, making them an effective means of verifying a person's identity. Here are some key aspects of biometrics:

Types of Biometric Characteristics:

- **Physiological Biometrics:** These are physical characteristics inherent to an individual's body. Examples include:
 - Fingerprint: Patterns of ridges and valleys on the fingertips.
 - Iris: Unique patterns in the colored portion of the eye.
 - Retina: Blood vessel patterns on the back of the eye.
 - Face: Distinctive features and proportions of the face.
 - Hand Geometry: Measurements of the size and shape of the hand.
 - DNA: Unique genetic information encoded in an individual's DNA.
 - Voice: Distinctive vocal characteristics, including pitch, tone, and cadence.
- **Behavioral Biometrics:** These are traits related to an individual's behavior or actions. Examples include:
 - Typing Rhythm: Patterns and timing of keystrokes when typing on a keyboard.
 - Gait: Unique walking patterns and movements.
 - Signature: Characteristics of an individual's handwritten signature.
 - Voice: Behavioral aspects of speech, such as accent and pronunciation.

3.4.7 Multifactor Authentication

Multi-factor authentication (MFA), also known as two-factor authentication (2FA) or multi-step verification, is a security process that requires users to provide multiple forms of identification to verify their identity before granting access to a system, application, or service. MFA adds an extra layer of security beyond traditional password-based authentication by combining two or more authentication factors from different categories.

Two-factor Authentication (2FA):

- Requires users to provide two different authentication factors to verify their identity. Common combinations include:
 - Password (knowledge factor) + One-time passcode sent via SMS, email, or authenticator app (possession factor).
 - Password (knowledge factor) + Biometric scan (inherence factor).
- 2FA is widely used in online banking, email services, and other sensitive applications.