Cheet Sheet

Lecture 1

Three Elements of Security

Achieve some ***goal*** against some ***adversary***

- System Goal / Security Service / Policy
  - Confidentiality
    - Information can only be accessed by authorized entity
  - Integrity
    - Information has not been tampered with
  - Availability
    - Information is available to the authorized entities
- Threat models
  - Who are the attackers?
  - What are the attackers capable of?
- Mechanism

Eavesdropping in network security refers to the unauthorized interception and listening of communication as data is transmitted over a network. The goal of eavesdropping attacks is to capture sensitive information, such as usernames, passwords, credit card details, and other private data.

Eavesdropping attacks typically occur during data transmission, with attackers using specific tools and techniques to listen in on network traffic. Common methods include:

1. **Packet Sniffing**: Attackers use packet sniffing tools (like Wireshark) to capture and analyze network traffic, gaining insight into the contents of the data being transmitted.

2. **Man-in-the-Middle (MITM)**: Attackers position themselves between two communicating parties without their knowledge, allowing them to intercept, modify, or even forge messages in the communication.

Ways to Prevent Eavesdropping

1. **Encryption**: Use SSL/TLS to encrypt communications, ensuring that data cannot be easily read during transmission.

2. **Using a VPN**: Virtual Private Networks (VPNs) help encrypt transmitted data, making it harder for attackers to intercept.

3. **Forcing HTTPS**: When accessing websites, use HTTPS instead of HTTP, as HTTPS encrypts the data transmission.

4. **Secure Wireless Networks**: Avoid using open or insecure Wi-Fi networks, and use stronger encryption protocols like WPA3 for wireless networks.

Eavesdropping is a common network attack, especially in cases of unencrypted data transmission. Therefore, it's crucial to ensure data security, particularly when handling sensitive information.

Network Spoof

Network spoofing is a technique used to impersonate a trusted device on a network by falsifying the identity of the attacker's device. The goal is to deceive network devices or users to gain unauthorized access, intercept data, or disrupt network communication. There are several common types of network spoofing, each targeting a different aspect of network communication:

1. IP Spoofing

- **What it is**: IP spoofing involves forging the IP address of a packet to make it appear as if it came from a different device.
- **Purpose**: Often used in **denial-of-service (DoS)** attacks or **man-in-the-middle (MITM)** attacks, IP spoofing can help attackers evade detection or mislead network defenses.
- **How it works**: Attackers replace the source IP address in packet headers with a fake one, often that of a trusted host. This way, the receiving device might trust the packet.

2. MAC Spoofing

- **What it is**: MAC (Media Access Control) spoofing involves changing the hardware (MAC) address of a network interface to impersonate another device on the same local network.
- **Purpose**: Attackers can bypass MAC-based access control lists (ACLs) or redirect traffic meant for another device to themselves.

- **How it works**: Attackers can change the MAC address of their device to that of a trusted device, often to bypass restrictions on certain networks or to intercept traffic.

3. ARP Spoofing (or ARP Poisoning)

- **What it is**: ARP (Address Resolution Protocol) spoofing is an attack that sends fake ARP messages on a local network.
- **Purpose**: Often used for **MITM** attacks, ARP spoofing tricks devices into thinking the attacker's device has the IP address of another legitimate device.
- **How it works**: Attackers send spoofed ARP replies, associating their MAC address with the IP address of a legitimate device. This way, traffic intended for the legitimate device is sent to the attacker.

4. DNS Spoofing (or DNS Cache Poisoning)

- **What it is**: DNS spoofing alters the DNS responses to redirect traffic from a legitimate website to a fake one.
- **Purpose**: Commonly used for phishing attacks, DNS spoofing misleads users into believing they're visiting a legitimate site when they're actually visiting a malicious one.
- **How it works**: Attackers modify DNS entries on the server or cache level, redirecting users to fake sites where they might enter sensitive information.

5. Email Spoofing

- **What it is**: In email spoofing, attackers forge the sender's email address to make it appear as though an email was sent from someone else.
- **Purpose**: Typically used in **phishing** attacks to trick recipients into believing the email is from a trusted source.
- **How it works**: By altering the "From" header in an email, attackers can make it look like the message is coming from a legitimate source, such as a bank or trusted company.

Summary

Network spoofing methods can be used for malicious purposes, like **intercepting data**, **redirecting traffic**, or **launching attacks**. However, network security mechanisms, such as **encryption**, **packet filtering**, **anti-spoofing protocols**, and **network monitoring**, are often implemented to detect and prevent spoofing attempts.

Question

- Can you sniff packet in the network? Why or Why not?

  yes, packet sniffing can be done on many networks if you have access, the right tools, and possibly bypass security mechanisms. However, **no** if the network is well-secured or encrypted, or if you lack the necessary permissions and tools.

- Can you spoof? Why or why not

  In summary, **yes**, packet spoofing can be done if you have the necessary access, tools, and the network lacks certain protections. However, **no**, it is often not feasible on secure networks with anti-spoofing mechanisms, and it's illegal to spoof packets without proper authorization.

- What is smurf attack?

  A **Smurf attack** is a type of **Distributed Denial of Service (DDoS)** attack that overwhelms a target system with large amounts of Internet Control Message Protocol (ICMP) packets, causing it to slow down, crash, or become entirely inaccessible.

  How a Smurf Attack Works

  The attack exploits **ICMP echo requests**, which are commonly used by the ping command to check if a device is online. Here's how it typically works:

  1. **Spoofed Source IP Address**: The attacker sends an ICMP echo request packet, but with the **source IP address spoofed** to be that of the target victim.

  2. **Broadcast to Amplify**: The attacker sends the ICMP packet to a **network broadcast address**, which means all devices on that network will receive the packet.

  3. **Reply Flood to Victim**: Each device on the broadcast network will reply to the ICMP echo request, but since the request appears to come from the victim's IP address, **all replies are sent to the victim**.

  4. **Overload the Victim**: The victim's network becomes overwhelmed with these ICMP responses, which can exhaust its bandwidth and processing resources, causing a **denial of service**.

  Protection Against Smurf Attacks

Smurf attacks are relatively rare now due to improved network security practices, but protections include:

1. **Disabling IP-directed broadcasts** on routers.

2. **Filtering out spoofed packets** at the network level, which can be done with anti-spoofing rules.

3. **Configuring firewalls and intrusion prevention systems (IPS)** to detect and block unusual ICMP traffic.

Modern network setups often disable ICMP broadcasts by default, making this type of attack less effective than it once was.

- What is TCP sync flood attack?
  A
  TCP SYN flood attack is a type of **Denial of Service (DoS) attack** that targets a server's resources, aiming to make it unresponsive to legitimate traffic by exhausting its available connection slots.
  A TCP SYN flood attack overloads a server with incomplete TCP connections, preventing legitimate users from accessing it. By carefully managing and limiting connection resources, networks can defend against this common DoS attack vector.

- Will Sync flood on ssh affect telnet ? (Yes)
  Yes, a
  SYN flood attack targeting SSH can indeed affect **Telnet** and other network services on the same server. This is because the attack consumes the server's **TCP connection resources**, which are shared across all services that rely on TCP for communication.

- Is increasing the buffer the right approach to defend
  against TCP sync flood?
  No.

  Increasing the buffer size to defend against a **TCP SYN flood attack** is generally **not an effective solution**. While it may provide a temporary increase in the number of connections a server can handle, it does not address the root of the problem and can lead to other issues. Here's why:

  Why Increasing the Buffer Is Not Effective

  1. **Temporary Relief Only**: A larger buffer will delay the impact of the SYN flood but won't prevent it. Attackers can still overwhelm the server by sending even more SYN requests, especially with a large-scale attack.

2. **Memory Exhaustion**: Increasing the buffer consumes more system memory. If the attack is large enough, the server could eventually run out of memory or slow down significantly, impacting other processes and services.

3. **Doesn't Mitigate the Attack**: Increasing the buffer only increases capacity, not security. The server still allocates resources for malicious connections, meaning it's still vulnerable.

Better Solutions

There are more effective ways to protect against SYN flood attacks:

- **SYN Cookies**: This technique allows the server to respond to SYN requests without allocating resources until the handshake completes, preventing resource exhaustion.
- **Firewalls and Rate Limiting**: Configure firewalls to detect and block suspicious traffic patterns. Rate limiting can help reduce the impact of incoming SYN requests.
- **Intrusion Detection and Prevention Systems (IDPS)**: Use IDPS to monitor traffic for signs of SYN flood attacks and automatically block IPs with abnormal traffic patterns.
- **TCP Connection Limits**: Limit the number of half-open connections and drop older requests if the limit is reached.

In summary, increasing the buffer size is not a viable solution to prevent SYN flood attacks. Implementing SYN cookies, rate limiting, and other network security measures are more effective and sustainable defenses against these attacks.

- What level of network stack is Ping operating on?
  (Network)
  Network layer, ping use ICMP

Lecture 2

Question

- What are the three principles of secure system
  - Confidentiality

    - Integrity
    - Availability
- Systems should always be password (or biometric) protected
  Yes
- If we take all the right steps to construct the system, we can continue to rely on the security mechanisms for years to come, why or why not?
  No.

  While taking the right steps to construct a secure system is essential, it's not sufficient to rely solely on security mechanisms for years to come. Here are several reasons why ongoing vigilance is necessary:

  1. **Evolving Threat Landscape**: Cyber threats are constantly evolving, with attackers developing new techniques and exploits. What may be secure today could become vulnerable tomorrow as new attack vectors emerge.

  2. **Software and Hardware Vulnerabilities**: Systems can have unpatched vulnerabilities that are discovered over time. Regular updates and patch management are critical to address these vulnerabilities as they arise.

  3. **Human Factors**: Users can inadvertently compromise security through actions like poor password management, falling for phishing attacks, or neglecting security best practices. Continuous training and awareness programs are needed to mitigate these risks.

  4. **Changes in Technology**: As technology advances, older security mechanisms may become outdated or less effective. New technologies may introduce new vulnerabilities that need to be addressed.

  5. **Compliance and Regulatory Changes**: Security standards and regulations can evolve, requiring organizations to adapt their security practices to remain compliant and protect sensitive data adequately.

  6. **Operational Changes**: Changes within an organization, such as mergers, acquisitions, or changes in personnel, can introduce new security challenges. Security mechanisms need to be evaluated and updated in light of these changes.

7. **Insider Threats**: Employees or other trusted individuals may intentionally or unintentionally compromise security. Ongoing monitoring and access controls are necessary to mitigate these risks.

In summary, while robust security mechanisms are vital, they require continuous monitoring, updating, and adaptation to remain effective against emerging threats and changing circumstances. Security is an ongoing process rather than a one-time setup.

- What does C.I.A stand for, can you give one example of each, and the techniques to accomplish them.

  C.I.A. stands for **Confidentiality, Integrity, and Availability**, which are the three fundamental principles of information security. Here's a brief overview of each principle, along with an example and techniques to accomplish them:

  1. Confidentiality

  - **Example**: Protecting sensitive financial data from unauthorized access.
  - **Techniques to Accomplish Confidentiality**:
  - **Encryption**: Encrypting sensitive files and communications (e.g., using AES or RSA) ensures that only authorized users can access the information.
  - **Access Controls**: Implementing role-based access control (RBAC) to limit access to sensitive information based on user roles.
  - **Authentication**: Using strong passwords, biometrics, or multi-factor authentication (MFA) to ensure that only authorized users can access the system.

  2. Integrity

  - **Example**: Ensuring that a financial transaction record is not altered or tampered with.
  - **Techniques to Accomplish Integrity**:
  - **Hashing**: Using cryptographic hash functions (e.g., SHA-256) to create a unique hash of data. If the data is modified, the hash will change, indicating potential tampering.
  - **Digital Signatures**: Implementing digital signatures to verify the authenticity and integrity of documents or transactions.

- **Checksums**: Utilizing checksums to verify the integrity of data during transmission and storage.

3. Availability

- **Example**: Ensuring that a website remains accessible to users at all times, even during high traffic or attacks.
- **Techniques to Accomplish Availability**:
- **Redundancy**: Implementing redundant systems (e.g., load balancers, failover servers) to ensure service continuity in case of hardware failure.
- **Backup Solutions**: Regularly backing up data and systems to recover from data loss or corruption.
- **DDoS Protection**: Utilizing Distributed Denial of Service (DDoS) mitigation services to protect against attacks that aim to overwhelm and disrupt services.

By addressing these three principles, organizations can create a more secure and resilient information system.

- What are the differences between ciphertext only attack and known plaintext attack threat model?

  Ciphertext-only attacks and known plaintext attacks are two different types of threat models used to evaluate the security of cryptographic systems. Here are the key differences between them:

  1. Ciphertext-only Attack

  - **Definition**: In a ciphertext-only attack, the attacker has access only to the ciphertext (the encrypted message) and does not have any prior knowledge of the plaintext (the original message) or the encryption key.
  - **Goal**: The attacker aims to derive information about the plaintext or the key by analyzing the ciphertext alone.
  - **Challenges**: This type of attack is often more difficult for the attacker since they lack any reference to the original message. The attacker must rely on patterns, frequency analysis, or other statistical techniques to break the encryption.
  - **Example**: An attacker intercepts an encrypted message and attempts to decipher it without any other information about the plaintext.

  2. Known Plaintext Attack

- **Definition**: In a known plaintext attack, the attacker has access to both the ciphertext and the corresponding plaintext for some messages. This means they know some specific inputs and their encrypted outputs.
- **Goal**: The attacker aims to use the known plaintext and its ciphertext to deduce information about the encryption key or to decrypt other ciphertexts.
- **Challenges**: This type of attack is generally easier for the attacker than a ciphertext-only attack, as the known plaintext provides a basis for analyzing the encryption process and exploiting potential weaknesses.
- **Example**: An attacker has a record of a message that they know is "HELLO," and they also have the corresponding ciphertext. They can use this information to attempt to decipher other encrypted messages.

Summary of Key Differences

Feature	Ciphertext-only Attack	Known Plaintext Attack
Information Available	Only ciphertext	Both ciphertext and corresponding plaintext
Attack Difficulty	Generally more difficult	Generally easier
Goal	Determine plaintext from ciphertext	Use known plaintext to find key or decrypt other messages
Example Scenario	Intercepting encrypted messages	Analyzing a previously known message and its encryption

In summary, the main difference lies in the information available to the attacker, which significantly impacts the difficulty and approach of the attack.

Lecture3

Question

- Cryptography is all about the algorithm, therefore, as long as we use the right cryptographic tool, and random keys, the system is secure.

  While using strong cryptographic algorithms and random keys is crucial for ensuring the security of a cryptographic system, it is not sufficient on its

own to guarantee overall system security. Here are several reasons why relying solely on algorithms and keys may be inadequate:

1. Algorithm Vulnerabilities

- **Flaws in Algorithms**: Even widely accepted algorithms can have vulnerabilities. Cryptographic attacks evolve, and what is considered secure today may be broken in the future due to advancements in cryptanalysis or new mathematical techniques.
- **Implementation Errors**: Even if the algorithm itself is strong, poor implementation can introduce vulnerabilities (e.g., side-channel attacks, padding oracle attacks).

2. Key Management Issues

- **Key Distribution**: Securely distributing cryptographic keys can be challenging. If an attacker can intercept the key during distribution, they can compromise the entire system.
- **Key Storage**: Improper storage of keys can lead to exposure. If keys are not securely stored or protected, they can be stolen or misused.

3. Human Factors

- **User Behavior**: Users may choose weak passwords or fail to follow security best practices, leading to vulnerabilities regardless of the strength of the cryptographic tools used.
- **Social Engineering**: Attackers may exploit human vulnerabilities through phishing or other social engineering tactics to gain access to sensitive information or keys.

4. Environmental Factors

- **Physical Security**: If an attacker can gain physical access to systems, they may be able to bypass cryptographic protections through physical attacks.
- **Network Security**: Cryptography does not address other vulnerabilities in a system, such as those in the network or application layers. An insecure network can expose encrypted data during transmission.

5. Regulatory Compliance

- **Legal and Compliance Issues**: Relying solely on cryptographic tools may not ensure compliance with regulations (e.g., GDPR, HIPAA), which

may require additional security measures.

Conclusion

While strong cryptographic algorithms and random keys are essential components of a secure system, they must be part of a broader security strategy that includes secure implementation, effective key management, user education, and comprehensive security policies. Security is a multi-layered approach, and no single factor can ensure complete protection.

- What is the key space of substitution cipher for English alphabet, how would you crack this?

  26! Thus, the key space for a substitution cipher using the English alphabet is 26! , which is about 4.03×10^{26} .

  Cracking the Substitution Cipher

  Cracking a substitution cipher can be done through various techniques, including:

  1. **Frequency Analysis**:

  • Analyze the frequency of letters in the ciphertext and compare them to the known frequencies of letters in the English language. For example, the letter “E” is the most commonly used letter in English, followed by “T,” “A,” “O,” etc. This can help identify potential substitutions.

  2. **Common Words and Patterns**:

  • Look for common short words (e.g., “the,” “and,” “is”) and common letter patterns (like double letters “ee,” “tt”) to make educated guesses about the substitutions.

  3. **Trial and Error**:

  • Create potential mappings based on frequency analysis and common patterns, then test them against the ciphertext to see if they produce meaningful plaintext.

  4. **Brute Force Attack**:

  • Given the vast key space (26!), a brute force attack would involve systematically trying every possible key until the correct one is found. However, this approach is computationally impractical without further constraints.

  5. **Use of Known Plaintext**:

- If any part of the plaintext is known or can be guessed, this can significantly reduce the complexity of the decryption process.

6. **Computer Algorithms**:

- Automated tools can use algorithms to perform frequency analysis and other techniques to speed up the process of cracking the cipher.

By employing these techniques, it is possible to break a substitution cipher even without knowing the key, especially given the large amount of ciphertext available for analysis.

- What is the intuitive idea / formal definition of Shanon's idea of perfect secrecy?
  Key has the same space with plaintext.

  Shannon's concept of perfect secrecy is a foundational principle in cryptography, articulated by Claude Shannon in his seminal work during World War II. Here's a breakdown of the intuitive idea and formal definition of perfect secrecy:

  Intuitive Idea

  The intuitive idea behind perfect secrecy is that even if an attacker has access to the ciphertext (the encrypted message), they cannot gain any information about the plaintext (the original message). In other words, knowing the ciphertext does not provide the attacker with any advantage over guessing the plaintext. This can be achieved when the encryption method is such that every possible plaintext is equally likely given a specific ciphertext.

  Formal Definition

  In a more formal sense, perfect secrecy can be defined as follows:

  A cryptographic system is said to achieve **perfect secrecy** if, for every possible plaintext and ciphertext , the following condition holds:

  where:

  - is the probability of plaintext given the ciphertext .
  - is the prior probability of plaintext before any ciphertext is known.

  This means that the distribution of the plaintext remains unchanged regardless of the ciphertext. Consequently, the attacker cannot infer any information about the plaintext based on the ciphertext, making it impossible to distinguish between the different plaintexts.

Implications of Perfect Secrecy

1. **Key Usage**: For a system to achieve perfect secrecy, the key must be at least as long as the plaintext and used only once (as seen in the one-time pad). If the key is shorter or reused, it can lead to patterns that an attacker can exploit.

2. **Randomness of Key**: The key must be chosen uniformly at random from the key space, ensuring that all possible keys are equally likely to be used.

3. **No Information Leakage**: There must be no correlation between the plaintext and the ciphertext that could be exploited by an attacker, ensuring complete confidentiality.

Conclusion

Shannon's idea of perfect secrecy sets a high standard for cryptographic security. While achieving perfect secrecy in practical systems is often not feasible, the principles established by Shannon guide the development of secure cryptographic protocols and highlight the importance of randomness and key management in modern cryptography.

- What is OTP, what's its limitation, and what mathematical property gives it perfect secrecy intuitive?

What is OTP?

One-Time Pad (OTP) is a type of encryption technique that uses a single-use key that is as long as the plaintext message. The key is generated randomly and is combined with the plaintext using a modular addition operation (typically XOR for binary data).

Key Characteristics of OTP:

- **Key Length**: The key must be at least as long as the plaintext.
- **Randomness**: The key must be completely random.
- **Single Use**: Each key can only be used once for a single plaintext message.

Mathematical Property That Gives OTP Perfect Secrecy

The mathematical property that gives the One-Time Pad its perfect secrecy is based on the relationship between the plaintext, ciphertext, and key. The encryption can be expressed as follows:

$C = P \oplus K$

Proof of Perfect Secrecy:

1. **Uniform Distribution of Keys**: Since the key is chosen uniformly at random, every possible key is equally likely.

2. **Ciphertext Independence**: Given the ciphertext , every plaintext can correspond to a valid key such that $C = P \oplus K$. For any ciphertext, there exists exactly one key that can produce any given plaintext, implying that knowing does not provide any information about .

2. **Ciphertext Independence**: Given the ciphertext , every plaintext can correspond to a valid key such that . For any ciphertext, there exists exactly one key that can produce any given plaintext, implying that knowing does not provide any information about .

3. **Probability**: For any plaintext and ciphertext: $P(P|C) = P(P)$

This holds true because for any ciphertext, all plaintexts are equally probable if the key is random and uniformly distributed. This satisfies the condition for perfect secrecy.

Limitations of OTP

Despite its theoretical perfect secrecy, OTP has several practical limitations:

1. **Key Management**: The requirement for a key that is at least as long as the plaintext makes key management cumbersome, especially for long messages. Storing and distributing such keys securely can be impractical.

2. **Single Use**: The key must never be reused. Reusing a key can lead to vulnerabilities and potential leakage of information, as patterns in the plaintext may be discerned.

3. **Random Key Generation**: Generating truly random keys can be challenging. If the key is not perfectly random, it could introduce vulnerabilities that compromise security.

4. **Operational Complexity**: The logistics involved in securely sharing and storing keys, especially in large-scale or real-time communication, can be a significant barrier to the practical implementation of OTP.

Conclusion

While the One-Time Pad provides perfect secrecy when properly implemented, its limitations in key management, generation, and operational complexity have made it difficult to use in most practical applications. As a

result, modern cryptographic systems often rely on different methods that balance security with usability.

- What is the definition of semantic security? Why is it a weaker notion of perfect secrecy?

  Definition of Semantic Security

  Semantic security is a property of a cryptographic system (specifically, an encryption scheme) that ensures that an adversary cannot derive any useful information about the plaintext from the ciphertext. More formally, an encryption scheme is considered semantically secure if, for any two distinct plaintexts and , the probability distribution of the ciphertexts generated by encrypting either plaintext is indistinguishable to any efficient adversary.

  In simpler terms, semantic security implies that even if an attacker knows the ciphertext, they cannot determine which plaintext was used to generate it, except with negligible probability.

  Mathematical Definition

  An encryption scheme is semantically secure if the following holds:

  For any two plaintexts and :

  $$\mathrm{Adv}_{\mathcal{E}}(P_0, P_1) = |P(C|P_0) - P(C|P_1|)| \leq \epsilon$$

  where:

  - is the probability distribution of the ciphertext when encrypting .
  - is a negligible function of the security parameter (typically, a small value that approaches zero).

  Why Semantic Security is a Weaker Notion than Perfect Secrecy

  1. **Indistinguishability versus Perfect Secrecy**:

  - Perfect secrecy guarantees that knowing the ciphertext gives no information about the plaintext whatsoever. It requires that the distribution of plaintexts remains unchanged after observing the ciphertext. This means that for any plaintext , the conditional probabilities are equal, regardless of the ciphertext.
  - Semantic security, on the other hand, requires that the ciphertext does not allow the adversary to distinguish between any two specific plaintexts. However, this does not prevent the adversary from making some probabilistic inference about the plaintext.

2. **Key Requirements**:

- Perfect secrecy necessitates that the key be as long as the message, truly random, and used only once. This makes it impractical for most real-world applications.
- Semantic security can be achieved with shorter, reusable keys, which are often more manageable in practice. Many modern encryption schemes (like those based on block ciphers) can be semantically secure even though they do not achieve perfect secrecy.

3. **Practical Implementation**:

- Achieving perfect secrecy is theoretically appealing but very challenging in real-world scenarios due to the strict requirements on key generation and management.
- Semantic security is more achievable with current cryptographic techniques and is widely accepted in practice, making it a preferred goal for many encryption schemes.

Conclusion

In summary, semantic security provides a more practical approach to encryption, focusing on ensuring that ciphertexts do not leak information about plaintexts, while perfect secrecy sets an ideal but often impractical standard that ensures complete information hiding.

- Can stream cipher have perfect secrecy?

  Yes, a stream cipher can achieve perfect secrecy, but only under specific conditions. Let's explore how this can occur and the requirements necessary for a stream cipher to provide perfect secrecy.

  Perfect Secrecy and Stream Ciphers

  Perfect secrecy means that the ciphertext does not reveal any information about the plaintext, and the probability distribution of the plaintext remains unchanged regardless of the ciphertext observed. This can be achieved if the following conditions are met:

  1. **Key Length**: The key used in the stream cipher must be at least as long as the plaintext message. In practice, this means using a key that is random and of sufficient length to cover the entirety of the plaintext being encrypted.

2. **Randomness of Key**: The key must be generated completely at random and must not follow any predictable patterns. Each bit of the key must be independent of other bits.

3. **Single Use of Keys**: The key must be used only once (similar to a one-time pad). Reusing keys can introduce vulnerabilities and allow for statistical analysis that could lead to the recovery of the plaintext.

Stream Ciphers and Perfect Secrecy

The classic example of a stream cipher that can achieve perfect secrecy is the **one-time pad** (OTP) implemented as a stream cipher. In this case:

- Each bit of the plaintext is XORed with a corresponding bit of a random key that is as long as the plaintext.
- Because the key is random, equally likely, and used only once, the ciphertext produced provides no information about the plaintext.

Limitations and Practical Considerations

While it is theoretically possible for a stream cipher to achieve perfect secrecy, there are practical challenges:

1. **Key Management**: Generating and securely managing random keys that are as long as the plaintext makes the implementation of a stream cipher like the one-time pad impractical for most real-world applications.

2. **Key Reuse**: In practical stream ciphers (like RC4 or Salsa20), keys are often reused or are not of sufficient length relative to the plaintext, leading to vulnerabilities and the inability to achieve perfect secrecy.

3. **Security Protocols**: Most modern cryptographic protocols use stream ciphers in ways that do not allow for perfect secrecy. Instead, they provide computational security, ensuring that breaking the encryption is computationally infeasible, rather than guaranteeing absolute secrecy.

Conclusion

In summary, while a stream cipher can theoretically achieve perfect secrecy if it adheres to the principles of using a truly random, single-use key that is at least as long as the plaintext, practical implementations often compromise these conditions. As a result, most modern stream ciphers focus on providing computational security rather than perfect secrecy.

- Name two attacks against stream cipher? What can an adversary achieve with these two attacks?

Lecture-4

Diff between Block and Stream

Block ciphers and stream ciphers are both fundamental types of symmetric key encryption, each with distinct characteristics, advantages, and use cases. Here's a breakdown of their differences and similarities:

Differences Between Block Ciphers and Stream Ciphers

1. **Encryption Unit**:

- **Block Cipher**: Encrypts data in fixed-size blocks. Common block sizes include 64 bits (e.g., DES) or 128 bits (e.g., AES). If the plaintext is shorter than the block size, it may need to be padded.
- **Stream Cipher**: Encrypts data one bit or byte at a time, effectively allowing for continuous encryption of data streams.

2. **Processing**:

- **Block Cipher**: Processes the entire block of data at once, which may involve multiple rounds of transformation, substitution, and permutation within the block.
- **Stream Cipher**: Operates in real-time, typically using a pseudorandom keystream generated from a key to encrypt each bit of plaintext independently.

3. **Performance**:

- **Block Cipher**: Generally requires more computational resources due to the complexity of processing entire blocks and performing multiple transformations.
- **Stream Cipher**: Usually faster and more efficient for applications requiring low-latency encryption, as it can start encrypting data as soon as it is available.

4. **Key Management**:

- **Block Cipher**: Often uses a single key for a set of blocks; key length can vary (e.g., AES supports 128, 192, or 256-bit keys).
- **Stream Cipher**: Typically generates a keystream from a short key, making it important to manage the key securely to prevent keystream reuse.

5. **Use Cases**:

- **Block Cipher**: Suitable for file encryption, data at rest, and when data can be processed in chunks (e.g., disk encryption, secure messaging).
- **Stream Cipher**: Ideal for scenarios where data is continuous and needs to be encrypted in real-time, such as secure communications (e.g., video or voice streaming).

Similarities Between Block Ciphers and Stream Ciphers

1. **Symmetric Key Encryption**:
   - Both block ciphers and stream ciphers use the same key for both encryption and decryption, which means that both parties must securely share the key before communication.
2. **Confidentiality**:
   - Both types of ciphers aim to provide confidentiality, ensuring that unauthorized parties cannot access the plaintext data without the key.
3. **Security Principles**:
   - Both rely on similar cryptographic principles, such as the use of strong keys, the importance of key randomness, and the necessity for proper implementation to avoid vulnerabilities.
4. **Resistance to Attacks**:
   - Both types of ciphers are designed to be secure against various cryptographic attacks, including brute force attacks, provided they are implemented correctly and the keys are managed securely.
5. **Application in Protocols**:
   - Both block ciphers and stream ciphers can be utilized in network protocols, providing security for data in transit, albeit in different contexts and configurations.

Conclusion

In summary, while block ciphers and stream ciphers differ fundamentally in how they process data and their typical use cases, they share key characteristics as symmetric encryption methods aimed at providing data confidentiality. Understanding the strengths and weaknesses of each type allows for informed choices in cryptographic implementations based on the specific needs of an application.

Feature	Block Cipher	Stream Cipher

Encryption Unit	Fixed-size blocks (e.g., 64 bits, 128 bits)	Individual bits or bytes
Processing	Processes entire blocks at once	Processes data in real-time, bit by bit
Performance	Generally slower due to complexity	Typically faster and more efficient
Key Management	Uses a single key for a set of blocks	Generates a keystream from a short key
Use Cases	File encryption, data at rest	Real-time encryption, secure communications
Symmetric Key	Yes	Yes
Confidentiality	Yes	Yes
Security Principles	Similar cryptographic principles	Similar cryptographic principles
Resistance to Attacks	Designed to resist various cryptographic attacks	Designed to resist various cryptographic attacks
Examples	AES, DES	RC4, Salsa20

ECB

The Electronic Codebook (ECB) mode of the Advanced Encryption Standard (AES) is known for several vulnerabilities, primarily due to its fundamental approach to data encryption. Here are the root causes behind the vulnerabilities in ECB mode:

1. Deterministic Encryption

- **Description**: In ECB mode, the same plaintext block will always produce the same ciphertext block when encrypted with the same key. This determinism means that identical plaintext blocks result in identical ciphertext blocks.
- **Impact**: If an attacker observes the ciphertext, they can infer patterns and make assumptions about the underlying plaintext. This property can lead to various types of attacks, especially when the same data is repeated in the plaintext.

2. Lack of Diffusion

- **Description**: ECB mode does not provide good diffusion, which means that changes in the plaintext do not significantly affect the ciphertext. This is because each block is encrypted independently without any feedback from previous blocks.

- **Impact**: An attacker can manipulate the ciphertext by altering specific blocks without affecting the rest of the message. This can lead to vulnerabilities where an attacker can replace or insert blocks of ciphertext and control the corresponding plaintext.

3. Pattern Recognition

- **Description**: Due to the deterministic nature of ECB, patterns in the plaintext can be directly observed in the ciphertext. For example, in images or structured data, identical blocks will show up as identical ciphertext blocks.
- **Impact**: This can lead to the identification of repeated sequences, which can compromise confidentiality. An attacker can recognize repeated blocks and infer information about the structure of the plaintext, such as in the case of encrypting images or text with repeated phrases.

4. Lack of Integrity Protection

- **Description**: ECB mode does not inherently provide any integrity protection mechanisms. It does not detect or prevent tampering with ciphertext.
- **Impact**: An attacker can alter the ciphertext, resulting in changes to the decrypted plaintext without detection. This could lead to unauthorized actions or data corruption, especially in applications where integrity is critical (e.g., financial transactions, data integrity checks).

5. Inability to Handle Larger Messages

- **Description**: Since ECB encrypts blocks independently, it is not suitable for encrypting larger messages without proper padding or handling.
- **Impact**: If the message length is not a multiple of the block size, the final block will need padding, which can introduce additional complexity and vulnerabilities if not managed carefully.

Conclusion

Due to these vulnerabilities, ECB mode is generally not recommended for use in secure applications. Instead, modes like Cipher Block Chaining (CBC), Galois/Counter Mode (GCM), or other authenticated encryption modes are preferred, as they provide better security through techniques like chaining, randomness, and integrity protection. These modes address the weaknesses of ECB by ensuring that identical plaintext blocks do not produce identical ciphertext blocks and that alterations in the ciphertext can be detected.

Here are the answers to your true/false questions:

1. **DES is still secure**

Answer: **False**

Explanation: The Data Encryption Standard (DES) is considered insecure today due to its short key length of 56 bits, which makes it vulnerable to brute-force attacks.

2. **The key length of block cipher needs to be the same as the length of the block**

Answer: **False**

Explanation: The key length does not need to match the block length. For example, AES has a block size of 128 bits but supports key lengths of 128, 192, or 256 bits.

3. **When the file is not a multiple of block size, we pad it with random bytes to secure it, since the goal is to have the output as random as possible**

Answer: **False**

Explanation: Padding is typically done using specific schemes (like PKCS#7) to fill the remaining space in the final block, not random bytes. The purpose of padding is to ensure the final block is the correct size, not necessarily to maximize randomness.

4. **The entries in the S-box have to be non-linear, therefore we just randomly generate it**

Answer: **False**

Explanation: While S-boxes should be non-linear to provide security, they cannot simply be randomly generated. S-boxes must be carefully designed to meet specific cryptographic properties, such as diffusion, confusion, and resistance to certain types of attacks. Random generation without consideration of these properties can lead to weak S-boxes.

1. How is block cipher different from stream cipher, how is it similar to stream cipher?

- **Differences**:
- **Data Processing**: Block ciphers encrypt data in fixed-size blocks (e.g., 128 bits), while stream ciphers encrypt data one bit or byte at a time.

- **Operation Mode**: Block ciphers often require modes of operation (like CBC, ECB, etc.) to handle data longer than one block, whereas stream ciphers can directly encrypt data of any length without needing a mode.
- **Performance**: Stream ciphers are generally faster for streaming data since they process data on-the-fly. Block ciphers might introduce latency due to the need to wait for a complete block.
- **Similarities**:
- **Key Use**: Both block and stream ciphers rely on symmetric keys for encryption and decryption.
- **Confidentiality**: Both types of ciphers aim to provide confidentiality by transforming plaintext into ciphertext, making it unreadable to unauthorized parties.
- **Cryptographic Foundations**: Both are based on mathematical algorithms and can use similar cryptographic techniques (like permutations and substitutions).

2. What are PRP and PRF, what constructions will allow one to construct a PRP from PRF?

- **PRP (Pseudorandom Permutation)**: A function that takes an input and produces an output that appears random, but it is reversible with a secret key. For every input, a unique output is produced, making it indistinguishable from a truly random permutation.
- **PRF (Pseudorandom Function)**: A function that produces output that appears random and is indistinguishable from random. It is not required to be reversible. A PRF takes a secret key and an input and outputs a value that looks random.
- **Construction**:
- You can construct a PRP from a PRF using a technique called **Feistel Construction** or through **Universal Hashing**.
- In a Feistel Construction, the input is split into two halves, and the PRF is applied to one half in combination with a subkey derived from the main key. This is repeated for several rounds to create a reversible function (the PRP).

3. What are the four key design principles of block cipher?

1. **Substitution-Permutation Network (SPN)**: The design should use a combination of substitution (replacing bits) and permutation (rearranging bits)

to achieve diffusion and confusion, which help in obscuring the relationship between the plaintext, ciphertext, and key.

2. **Diffusion**: The design should ensure that a small change in the plaintext (or key) produces a significant change in the ciphertext, meaning that bits in the ciphertext should depend on many bits of the plaintext and key.

3. **Confusion**: The design should obscure the relationship between the key and the ciphertext, making it difficult to derive the key from the ciphertext alone.

4. **Key Strength**: The design should allow for a sufficiently large key space to prevent brute-force attacks. This often involves using long keys and ensuring the key schedule (the process of deriving round keys from the main key) is secure.

4. What is the root cause behind the vulnerability in ECB mode of AES?

- **Root Cause**: The root cause behind the vulnerability in **Electronic Codebook (ECB)** mode of AES is that it encrypts identical plaintext blocks into identical ciphertext blocks. This means that patterns in the plaintext are preserved in the ciphertext, making it susceptible to various attacks, including pattern analysis and frequency analysis. An attacker can potentially infer information about the plaintext by examining the structure of the ciphertext.

5. What are the two approaches we studied in class to address the problem of the one-time-key?

1. **Stream Ciphers**: Instead of using a one-time key, a stream cipher generates a key stream (a sequence of bits derived from a shorter key) that is combined with the plaintext using an XOR operation. This approach can efficiently encrypt long streams of data while maintaining confidentiality.

2. **Key Derivation Functions (KDFs)**: Use a key derivation function to derive multiple keys from a single key or a seed value. This approach allows for the generation of unique keys for each session or message without requiring completely new random keys.

6. What are the requirements for IVs in block cipher modes of operation?

- **Randomness**: Initialization Vectors (IVs) should be random or unpredictable to ensure that the same plaintext encrypted multiple times with the same key produces different ciphertexts.

- **Uniqueness**: IVs should be unique for each encryption operation with the same key. Reusing IVs with the same key can lead to vulnerabilities, as it can

allow attackers to identify patterns in the ciphertext.

- **Length**: IVs should be the same length as the block size of the cipher (e.g., for AES with a block size of 128 bits, the IV should also be 128 bits).
- **Transmission**: IVs must be transmitted along with the ciphertext, as they are needed for decryption. However, they do not need to be kept secret.

Lecture-4

1. What is MAC, name one property of MAC

MAC (Message Authentication Code): A MAC is a short piece of information used to authenticate a message and provide integrity and authenticity assurances. It is generated by applying a secret key to the message through a cryptographic algorithm.

- **Property of MAC**: **Integrity**: A MAC ensures that the message has not been altered during transmission. If even a single bit of the message is changed, the MAC will also change, allowing the recipient to detect tampering.

2. What is a Hash Function, name the most important function of a hash

Hash Function: A hash function is a mathematical algorithm that takes an input (or "message") and produces a fixed-size string of bytes, typically a digest that is unique to each unique input.

- **Most Important Function of Hash**: **Integrity Verification**: Hash functions are primarily used for verifying the integrity of data. By comparing the hash of the original message with the hash of the received message, one can check whether the data has been altered.

3. Why should there be two keys in MAC design

- **Reason for Two Keys in MAC**: The use of two keys (a secret key and an additional key or nonce) in MAC design enhances security by providing a means to create different MACs for different contexts or sessions, helping to prevent replay attacks and ensuring that an attacker cannot derive the key through analysis of multiple MACs. This approach can also improve resilience against certain cryptographic attacks, such as key recovery or brute-force attacks.

4. What was the construction that allows hash functions to handle very long messages

- **Construction**: **Merkle-Damgård Construction**: This construction allows hash functions to process inputs of arbitrary length by breaking the input message into fixed-size blocks. Each block is hashed sequentially, and the output of each block is fed into the hash function for the next block, creating a chained hash that can accommodate very long messages.

5. If I have a message that I want to send to the bank, but I don't care who can read it, what can I do?

- **What to Do**: If you want to send a message to the bank and you don't care who can read it (i.e., confidentiality is not a concern), you can send the message in **plaintext**. However, if you want to ensure the integrity of the message and verify that it has not been altered, you can:

1. **Use a Hash Function**: Compute a hash of the message and send it along with the message. The bank can then verify the hash to ensure the message integrity.

2. **Use a MAC**: If you want to authenticate the message as well, you can use a Message Authentication Code (MAC) by generating it with a shared secret key. This way, the bank can verify both the integrity and authenticity of the message.

In both cases, plaintext messages can be read by anyone, but integrity can still be maintained through hashing or MACs.

Lecture-5

1. What is authenticated encryption?

Authenticated Encryption (AE) is a cryptographic scheme that simultaneously provides both confidentiality (ensuring that information remains secret) and integrity/authentication (ensuring that the information is not altered in transit). In AE, a single operation produces a ciphertext that is both encrypted and verified through an authentication tag.

2. In real systems, is it always possible to have confidentiality without integrity?

Answer: **No**, it is not advisable to have confidentiality without integrity in real systems.

- **Explanation**: While it is technically possible to implement systems that provide confidentiality without integrity (for example, by only encrypting data

without checking its authenticity), doing so is dangerous. If an attacker can modify encrypted data without detection, they can exploit this vulnerability, leading to unauthorized access or manipulation of sensitive information. Thus, both confidentiality and integrity are essential for secure communication.

3. What are the different ways to combine MAC and Symmetric Cipher? Which one is always correct?

There are several ways to combine Message Authentication Codes (MACs) with symmetric ciphers:

1. **Encrypt-Then-MAC (EtM)**:

- First, the plaintext is encrypted using a symmetric cipher.
- Then, a MAC is computed over the ciphertext (and possibly other data, like the plaintext or associated data).
- Finally, the MAC is appended to the ciphertext.

Security: This is the preferred method and is considered secure because it ensures that the integrity check is applied to the encrypted data.

2. **MAC-Then-Encrypt (MtE)**:

- First, the MAC is computed over the plaintext.
- Then, the plaintext and the MAC are encrypted together.

Security: This method is less secure because if an attacker modifies the ciphertext, the MAC will not protect the integrity of the plaintext after decryption.

3. **Encrypt-and-MAC (E&M)**:

- Both the plaintext and the ciphertext are separately authenticated using their respective MACs.
- This involves encrypting the plaintext and computing a MAC on both the plaintext and the ciphertext.

Security: While this method can provide security, it is often more complex and can lead to inefficiencies.

Always Correct: **Encrypt-Then-MAC (EtM)** is the method that is always considered correct and secure, as it properly ensures both confidentiality and integrity.

4. Given a network protocol, what is a common mistake that was shown in the last class, and how would you defend against them?

Common Mistake: A typical mistake in network protocols is **lacking protection against replay attacks**. In replay attacks, an adversary captures valid data transmissions and resends them, potentially leading to unauthorized actions or data manipulation.

Defense Mechanism:

- **Use of Nonces**: Implement nonces (a number used once) in the protocol. Each request or transaction should include a unique nonce that the server verifies to ensure that it has not been reused.
- **Timestamps**: Include timestamps in the messages to ensure that old messages cannot be reused. The server checks the timestamp to see if the request is within an acceptable time window.
- **Session Tokens**: Use session tokens that expire after a certain period or after use. Tokens can help ensure that requests are fresh and valid.
- **Replay Detection**: Maintain a database of recently seen nonces or message identifiers and reject any requests that reuse previously seen identifiers.

By incorporating these defenses, protocols can significantly enhance their resilience against replay and other types of attacks.

Lecture-6

1. How many symmetric keys does it take to support secure communications among 4 peers? What are the possible approaches to mitigate this issue?

- **Number of Keys**: To support secure communications among peers, the number of unique symmetric keys required is given by the formula . For 4 peers, this results in symmetric keys.
- **Possible Approaches to Mitigate the Issue**:

1. **Key Agreement Protocols**: Use protocols like Diffie-Hellman to agree on a shared key without the need to exchange keys directly.
2. **Hierarchical Key Management**: Implement a hierarchy of keys where a single key can be used to derive session keys for secure communication among peers.
3. **Public Key Infrastructure (PKI)**: Use asymmetric keys to encrypt symmetric keys, allowing peers to communicate securely without needing separate symmetric keys for each pair.

4. **Key Derivation Functions**: Use a shared master key to derive unique session keys for each communication session.

2. What is Diffie-Hellman key exchange, what attack is it vulnerable to, how do you launch that attack?

- **Diffie-Hellman Key Exchange**: It is a method for two parties to securely exchange cryptographic keys over a public channel. It allows them to create a shared secret key that can be used for symmetric encryption.
- **Vulnerability**: The Diffie-Hellman key exchange is vulnerable to a **Man-in-the-Middle (MitM) attack**. In this attack, an adversary intercepts the key exchange messages and can establish separate keys with each party.
- **Launching the Attack**: To execute a MitM attack:

1. The attacker intercepts the public keys exchanged between the two parties.
2. The attacker sends their own public key to each party pretending to be the other party.
3. Both parties now believe they are communicating directly with each other, but they are actually communicating through the attacker, who can read and modify the messages.

3. What is public key crypto and how is it different from symmetric key crypto? What key does Alice use if she wants to deliver a secret to Bob?

- **Public Key Crypto**: Public key cryptography (asymmetric cryptography) involves a pair of keys: a public key and a private key. The public key can be shared openly, while the private key is kept secret. Data encrypted with one key can only be decrypted by the other key in the pair.
- **Differences from Symmetric Key Crypto**:
- **Key Pair**: Public key crypto uses a pair of keys (public and private), while symmetric key crypto uses a single shared secret key.
- **Key Distribution**: Public key crypto simplifies key distribution since the public key can be freely distributed, while symmetric key crypto requires secure sharing of the secret key.
- **Performance**: Public key cryptography is generally slower than symmetric key cryptography.
- **Key Alice Uses**: If Alice wants to deliver a secret to Bob, she would encrypt the message using Bob's public key. Only Bob can decrypt the

message using his private key.

4. In practice, can we use RSA to directly encrypt the secret key for communication?

- **Answer**: **Yes**, RSA can be used to encrypt a symmetric key for secure communication. However, due to its computational intensity and size limitations, it is typically not used to encrypt large messages directly. Instead, RSA is often used to securely transmit symmetric keys (e.g., AES keys) which are then used for encrypting the actual data.

5. What is a digital signature? What key does Alice use to sign a file that she wants to authenticate and why?

- **Digital Signature**: A digital signature is a cryptographic mechanism that allows a person to prove the authenticity and integrity of a message or document. It ensures that the message has not been altered in transit and verifies the identity of the sender.
- **Key Alice Uses**: Alice uses her **private key** to sign the file. The reason for using her private key is that only she possesses it, ensuring that the signature can only be generated by her. Anyone with Alice's public key can verify the signature, confirming the authenticity of the file and that it was signed by her.