NETWORK SECURITY &CRYPTOLOGY

UNIT-2

Data encryption standard (DES):

The Data Encryption Standard (DES) is a symmetric key encryption algorithm that was developed in the early 1970s by IBM in collaboration with the National Bureau of Standards (now the National Institute of Standards and Technology, or NIST). DES became the most widely used encryption algorithm for securing electronic data transmission and storage for several decades.

Here are some key points about DES:

1. Symmetric Encryption: DES is a symmetric encryption algorithm, meaning the same key is used for both encryption and decryption. The encryption and decryption processes are reversible, using the same key.
2. Block Cipher: DES operates on fixed-size blocks of data, specifically 64-bit blocks. It divides the input data into blocks and processes each block separately.
3. Key Length: The key used in DES is 56 bits in length. However, due to the vulnerability of DES to brute-force attacks, it is commonly referred to as having a 56-bit key size, as 8 of the bits are used for parity and not for actual key material. The effective key size is thus considered to be 56 bits.
4. Feistel Cipher Structure: DES is based on a Feistel cipher structure, where the input block is divided into two halves, and multiple rounds of encryption and decryption operations are performed on the data. Each round involves applying various cryptographic functions, including substitution (S-boxes) and permutation (P-boxes).
5. Strength and Vulnerabilities: Over time, DES became vulnerable to brute-force attacks due to advances in computing power. In 1999, a coordinated effort called the "DES Cracker" project demonstrated the ability to crack a DES key in a matter of days. As a result, DES is considered insecure for modern applications.
6. Triple DES (3DES): To enhance security, a variant of DES called Triple DES (3DES or TDEA) was introduced. It applies the DES algorithm three times with different keys in a specific pattern, providing a key size of 168 bits (3 times 56 bits). 3DES is still in use in some legacy systems but is generally being phased out in favor of more modern encryption algorithms.

key recovery attacks on block ciphers:

Key recovery attacks on block ciphers aim to extract the secret encryption key used by the cipher. These attacks exploit vulnerabilities in the cipher's design or implementation to uncover the key and potentially decrypt the encrypted data. Here are a few key recovery attacks commonly discussed in the field of cryptography:

1. **Brute Force Attack:** This attack involves trying all possible combinations of keys until the correct one is found. Brute force attacks are effective against weak keys or small key sizes but become impractical as the key size increases.
2. **Differential Cryptanalysis**: This attack analyzes the differences between pairs of plaintext and corresponding ciphertext to deduce information about the key. It exploits patterns in how the cipher's internal operations propagate differences in the input to differences in the output. Differential cryptanalysis requires a large number of plaintext-ciphertext pairs but can be effective against certain block cipher designs.
3. **Linear Cryptanalysis:** This attack tries to find a linear relationship between the plaintext, ciphertext, and key bits of a block cipher. By analyzing patterns in the cipher's linear approximations, it can deduce parts of the key. Linear cryptanalysis requires known plaintext-ciphertext pairs but can be effective against certain block cipher constructions.
4. **Side-Channel Attacks:** Side-channel attacks exploit information leaked during the execution of a cryptographic algorithm, such as power consumption, electromagnetic radiation, or timing variations. By analyzing these side-channel leakages, an attacker can infer information about the secret key. Common side-channel attacks include power analysis, timing attacks, and electromagnetic attacks.
5. **Meet-in-the-Middle Attack:** This attack involves computing the encryption of certain plaintexts using all possible keys and storing the results. Then, the attacker computes the decryption of certain ciphertexts using all possible keys. By comparing the results, the attacker can identify potential key candidates. Meet-in-the-middle attacks can be effective against block ciphers with certain properties, such as the Data Encryption Standard (DES).

Iterated DES:

Iterated DES (Data Encryption Standard) is a variant of the original DES algorithm that enhances its security through multiple iterations. DES is a symmetric encryption algorithm widely used in the past, though its security is now considered weak due to advances in computing power.

In Iterated DES, the input data is divided into blocks and passed through a series of iterations, with each iteration using a different subkey derived from the original encryption key. The number of iterations depends on the specific implementation, but typically, multiple rounds of encryption and decryption are performed on the data.

The process of Iterated DES can be summarized as follows:

1. **Key Generation:** The original encryption key is expanded and transformed to generate a set of subkeys to be used in each iteration. This key expansion process enhances the security of the algorithm.
2. **Initial Permutation (IP):** The input block is permuted using an initial permutation table.
3. **Rounds:** The main encryption process consists of multiple rounds, with each round including the following steps:

a. Expansion: The right half of the previous round's output is expanded to match the size of the subkey.

b. Subkey Mixing: The expanded right half is XORed with a subkey generated from the main encryption key.

c. Substitution: The XOR output is divided into smaller chunks, and each chunk is substituted using predefined substitution boxes (S-boxes) that introduce non-linearity.

d. Permutation: The output of the substitution step is permuted using a fixed permutation table.

e. Swapping: The left and right halves of the output from the current round are swapped to become the input for the next round.

4. Final Permutation (FP): After the final round, a final permutation is applied to the output to generate the encrypted block.

Iterated DESX:

Iterated DESX (Data Encryption Standard Extended) is a cryptographic algorithm that combines the principles of the Data Encryption Standard (DES) with additional security features to enhance its strength. DESX was proposed by Ronald Rivest in 1996 as a modification to DES to address some of its weaknesses.

DESX operates by encrypting data in multiple rounds using a combination of DES and exclusive OR (XOR) operations. It uses a secret key that is typically 128 bits long, similar to DES. The algorithm consists of the following steps:

1. **Key Generation:** A 128-bit key is derived from the original user-supplied key using a secure key generation algorithm. This derived key is used as the actual encryption key.
2. **Initial XOR:** The input data (plaintext) is XORed with a random value called the initialization vector (IV) of the same length. The IV is different for each encryption operation and is typically generated using a secure random number generator.
3. **Iterative Encryption:** The XORed result from the previous step is encrypted using the DES algorithm with the derived key. The resulting ciphertext is then XORed with the next block of plaintext and the process is repeated for a specified number of rounds.
4. **Final XOR:** After completing the specified number of rounds, the final ciphertext is obtained. It is then XORed with the IV to produce the final encrypted output.

To decrypt the ciphertext, the same process is applied in reverse. The final ciphertext is XORed with the IV, and then the iterative decryption process is performed using the derived key.

Advanced encryption standards:

Advanced Encryption Standard (AES) is a symmetric encryption algorithm widely used to secure sensitive data. It was selected by the U.S. National Institute of Standards and Technology (NIST) in 2001 to replace the aging Data Encryption Standard (DES). AES has become the de facto standard for encryption and is used by governments, organizations, and individuals around the world.

Key features of AES:

1. **Symmetric encryption:** AES is a symmetric encryption algorithm, which means the same key is used for both encryption and decryption. This key must be kept secret and shared between the sender and the intended recipient.
2. **Block cipher:** AES operates on fixed-size blocks of data, with the most common block size being 128 bits. It encrypts and decrypts data in blocks rather than individual bits.
3. **Key sizes:** AES supports key sizes of 128, 192, and 256 bits. The larger the key size, the stronger the encryption. AES-128 is generally considered secure for most applications, while AES-256 provides a higher level of security.
4. **Substitution-Permutation Network:** AES uses a series of substitution and permutation operations to transform the input data. These operations include byte substitution, shift rows, mix columns, and add round keys. The number of rounds performed depends on the key size: 10 rounds for AES-128, 12 rounds for AES-192, and 14 rounds for AES-256.
5. **Security and resistance:** AES is designed to be resistant against various cryptographic attacks, such as brute force, differential cryptanalysis, and linear cryptanalysis. When used with a sufficiently strong key, AES is considered secure and has been extensively analyzed by the cryptographic community.

AES has been implemented in various software and hardware platforms, making it widely accessible and compatible across different systems. It is used in many applications, including secure communication protocols (e.g., SSL/TLS), disk encryption, file encryption, and securing sensitive data in databases.

Pseudorandom Functions:

Pseudorandom functions (PRFs) are mathematical functions that generate seemingly random outputs based on an input, but are actually deterministic and predictable. They are widely used in cryptography and computer science for various purposes, such as generating cryptographic keys, creating secure message authentication codes (MACs), and constructing secure encryption algorithms.

The key characteristic of a PRF is that it appears random to an observer who does not possess the secret key or internal state used in its computation. PRFs are designed to be computationally indistinguishable from true random functions, meaning that even with

significant computational resources, an attacker should not be able to distinguish the outputs of a PRF from random values.

A PRF takes an input, often called a "key," and produces an output that appears random. This output is usually indistinguishable from random data unless the same input (key) is used again. In other words, the same key will always produce the same output, making the function deterministic.

The security of a PRF relies on the secrecy of the key and the computational hardness of inverting the function. If an attacker can easily predict the output of a PRF given the key, or if they can efficiently determine the key from the output, then the PRF is considered insecure.

PRFs are commonly used in cryptographic protocols. For example, in symmetric encryption algorithms, a PRF may be used to derive subkeys from a master key. In MAC algorithms, a PRF can be used to generate a tag that provides data integrity and authenticity. Additionally, PRFs are employed in key derivation functions (KDFs) to generate multiple cryptographic keys from a single secret key.

The birthday attack:

The birthday attack is a cryptographic phenomenon that deals with the probability of collisions in hashing algorithms. It is named after the "birthday paradox," which states that in a group of just 23 people, there is a 50% chance that two people will share the same birthday. This probability seems counterintuitive because it is much higher than what most people expect.

In the context of cryptography, a hashing algorithm is used to convert an input of any size into a fixed-size output, known as a hash value or digest. The purpose of a hashing algorithm is to provide a unique representation of the input data, allowing for efficient data retrieval, integrity checks, and various security applications.

The birthday attack takes advantage of the fact that the probability of collisions (i.e., two different inputs producing the same hash value) increases as the number of hashed inputs grows. Specifically, it exploits the birthday paradox to demonstrate that the number of possible collisions is much larger than one might expect.

To illustrate the attack, let's consider a simplified scenario with a hash function that produces a 16-bit output (for brevity). In this case, there are 2^16 (65,536) possible hash values. The birthday attack aims to find two different inputs that generate the same hash value.

Using the birthday paradox, the attack calculates the expected number of inputs required to find a collision. It turns out that with a 16-bit hash, the expected number of inputs needed to find a collision is approximately 2^8 (256). This means that after hashing around 256 different inputs, there is a 50% chance of finding a collision.

The attack can be generalized to any hash function, and the expected number of inputs needed to find a collision depends on the size of the hash value. As the hash size increases, the number of inputs required for a collision to become likely increases exponentially.

To mitigate the birthday attack, cryptographic hash functions are designed with larger hash sizes, such as 128 bits (MD5), 160 bits (SHA-1), or even 256 bits (SHA-256), to make the probability of collisions negligible for practical purposes.

Number theoretic primitives:

Number theoretic primitives refer to fundamental concepts or operations in number theory, which is a branch of mathematics that deals with properties and relationships of numbers. These primitives form the building blocks for more advanced concepts and algorithms in number theory. Some important number theoretic primitives include:

1. **Prime numbers:** Prime numbers are positive integers greater than 1 that have no divisors other than 1 and themselves. They play a crucial role in many number theoretic algorithms, such as primality testing, factorization, and cryptography.
2. **Modular arithmetic:** Modular arithmetic is a system of arithmetic where numbers "wrap around" after reaching a certain value called the modulus. It involves operations like addition, subtraction, multiplication, and exponentiation performed on integers modulo a given modulus. Modular arithmetic is extensively used in cryptography, error correction codes, and various algorithms in number theory.
3. **Euclidean algorithm:** The Euclidean algorithm is an efficient method for finding the greatest common divisor (GCD) of two integers. It is based on the observation that the GCD of two numbers remains the same if the larger number is replaced by its difference with the smaller number. The Euclidean algorithm has applications in many areas, including modular inverse computation, solving linear Diophantine equations, and primality testing.
4. **Modular exponentiation:** Modular exponentiation is the computation of an exponentiation operation modulo a given modulus. It is used in various cryptographic algorithms, such as RSA and Diffie-Hellman, to ensure efficient and secure encryption, key exchange, and digital signatures.
5. **Chinese remainder theorem:** The Chinese remainder theorem is a result in number theory that provides a solution to a system of simultaneous congruences. It states that if the moduli of the congruences are pairwise coprime, then there exists a unique solution that satisfies all the congruences. The Chinese remainder theorem has applications in cryptography, error correction, and solving linear congruences.
6. **Euler's totient function:** Euler's totient function, denoted as $\phi(n)$, is a number theoretic function that counts the positive integers less than or equal to a given number n that are coprime with n. It has applications in modular exponentiation, primality testing, and cryptography, particularly in the RSA algorithm.

Fermat's theorem:

Fermat's theorem, also known as Fermat's Last Theorem, is a famous mathematical conjecture formulated by the French mathematician Pierre de Fermat in the 17th century. Fermat stated this theorem in the margin of his copy of a book in 1637, claiming that he had discovered a "truly marvelous proof" but that the margin was too narrow to contain it.

The theorem states that there are no three positive integers a, b, and c that satisfy the equation an + bn = cn for any integer value of n greater than 2. In other words, there are no whole number solutions to the equation when the exponent (n) is greater than 2.

Fermat's Last Theorem remained unproven for over 350 years, becoming one of the most famous unsolved problems in mathematics. Many mathematicians attempted to prove or disprove the conjecture, but it resisted proof until 1994.

In 1994, the British mathematician Andrew Wiles, building upon the work of previous mathematicians, finally presented a proof for Fermat's Last Theorem. Wiles' proof involved advanced mathematical techniques and relied heavily on elliptic curves and modular forms. His proof was groundbreaking and highly complex, spanning over 100 pages. It was later verified and accepted by the mathematical community.

Euler's theorem:

Euler's theorem, also known as Euler's totient theorem, is a fundamental result in number theory discovered by the Swiss mathematician Leonhard Euler. The theorem establishes a relationship between the values of the Euler's totient function and exponentiation modulo an integer.

Euler's totient function, denoted as ϕ(n), is a function that counts the positive integers less than or equal to n that are relatively prime to n (i.e., they share no common factors other than 1). For example, ϕ(10) = 4 because there are four positive integers less than or equal to 10 (1, 3, 7, and 9) that are coprime to 10.

Euler's theorem states that for any positive integer n and any integer a that is coprime to n, the following congruence holds:

a^ϕ(n) ≡ 1 (mod n)

Here, "a^ϕ(n)" denotes the exponentiation of a to the power of ϕ(n), and "≡" denotes congruence (equality modulo n). In other words, when a is coprime to n, raising a to the power of ϕ(n) and taking the remainder modulo n will result in 1.

Euler's theorem has several important consequences and applications. One of the most notable applications is in the field of cryptography, particularly in the RSA algorithm, which relies on the

difficulty of factoring large composite numbers. Euler's theorem plays a crucial role in the proof of the correctness and security of the RSA algorithm.

RSA Algorithm:

The RSA algorithm is a widely used asymmetric encryption algorithm named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman. It is one of the most popular and secure methods for secure communication and digital signatures.

RSA relies on the use of two keys: a public key and a private key. The public key is used for encryption, while the private key is used for decryption. Here's a simplified overview of how the RSA algorithm works:

1. Key Generation:

- Choose two distinct prime numbers, p and q.
- Calculate their product, n = p * q. This is the modulus for both the public and private keys.
- Calculate Euler's totient function of n, $\phi(n) = (p-1) * (q-1)$.
- Choose an integer e ($1 < e < \phi(n)$) that is coprime with $\phi(n)$. e will be the public exponent.
- Calculate the modular multiplicative inverse of e modulo $\phi(n)$ to find d. d will be the private exponent.

2. Public Key Encryption:

- To encrypt a message M, the sender uses the recipient's public key (n, e).
- The sender converts the message into a numeric representation (usually using a specific encoding like ASCII or Unicode).
- The sender raises the numeric representation of the message to the power of e and takes the result modulo n.
- The resulting ciphertext is sent to the recipient.

3. Private Key Decryption:

- To decrypt the received ciphertext C, the recipient uses their private key exponent d.
- The recipient raises the ciphertext to the power of d and takes the result modulo n.
- The resulting numeric value is converted back into the original message using the chosen encoding.

The security of RSA is based on the difficulty of factoring large composite numbers into their prime factors. The larger the prime numbers used in the key generation process, the more secure the encryption becomes.

It's worth noting that while RSA is a robust encryption algorithm, its efficiency decreases with the increase in the size of the numbers involved. As a result, RSA is typically used for encrypting small amounts of data, such as symmetric encryption keys, and not for encrypting large messages directly.

CODECHAMP
CREATED WITH ARBOK