Cryptography

Sanoj S Vijendra

July 28, 2024

Contents

Chapter 1

Intoduction

So, the forst chapter provides an introductory overview of cryptography.It lays the groundwork for the subsequent chapters by discussing the historical context, fundamental concepts, and the relevance of cryptography in modern society.

1.1 Historical Background

So, lets begin with some historical background:

- **Ancient Cryptography:** Simple substitution ciphers like Caesar cipher used by Julius Caesar.
- **Middle Ages:** The development of more complex ciphers like the Vigenère cipher.
- **20th Century:** The advent of mechanical and electromechanical encryption devices which includes the Enigma machine used during World War II developed by Alan Turing.
- **Modern Cryptography:** The shift towards mathematical and computer-based cryptographic algorithms.

1.2 What is Cryptography?

Cryptography is a technique of securing information and communications through the use of codes so that only those persons for whom the information is intended can understand and process it. Thus preventing unauthorized access to information. The prefix "crypt" means "hidden" and the suffix "graphy" means "writing". In Cryptography, the techniques that are used to protect information are obtained from mathematical concepts and a set of rule-based calculations known as algorithms to convert messages in ways that make it hard to decode them. These algorithms are used for cryptographic key generation, digital signing, and verification to protect data privacy, web browsing on the internet and to protect confidential transactions such as credit card and debit card transactions.. The primary goals of cryptography include:

- **Confidentiality:** Ensuring that information is accessible only to those authorized to view it.
- **Integrity:** Protecting information from being altered by unauthorized parties.
- **Authentication:** Verifying the identity of the parties involved in communication.
- **Non-repudiation:** Preventing entities from denying their involvement in a communication.

1.3 Basic Terminology

Some essential cryptographic terminologies are:

- **Plaintext:** The original and readable message or data.
- **Ciphertext:** The encrypted message which can not be read without decryption.
- **Encryption:** The process of converting plaintext into ciphertext.
- **Decryption:** The process of converting ciphertext back into plaintext.
- **Key:** A piece of information used in the encryption and decryption processes.

1.4 Classical Ciphers

This section will cover several classical ciphers and will provide insights into their operation and historical significance:

- **Caesar Cipher:** It is an subset of substitution cipher, where each letter in the plaintext is shifted by a fixed number of places down the alphabet.
- **Vigenère Cipher:** It is a poly-alphabetic substitution cipher that uses a keyword to vary the shift applied to each letter.
- **One-Time Pad:** It is regarded as theoretically unbreakable cipher that uses a random key that is as long as the message.

1.5 Modern Cryptography

Now, we'll see the modern cryptography and techniques and emphasizing its reliance on mathematical principles and computational algorithms. So, some basic prior informations are:

- **Symmetric-Key Cryptography:** In this, both the sender and receiver use the same key for encryption and decryption For e.g. Advanced Encryption Standard (AES).
- **Public-Key Cryptography:** It uses a pair of keys, one public key and one private key. The public key is used for encryption and the private key is used for decryption For e.g. Rivest–Shamir–Adleman (RSA).

1.6 Applications of Cryptography

Now we'll discuss the importance of cryptography in various areas:

- **Secure Communication:** Cryptography ensures the privacy and security in email, messaging, and online transactions.
- **Digital Signatures:** Cryptography authenticates the identity of the sender and ensures the integrity of the message.
- **Cryptographic Protocols:** Protocols like SSL/TLS for secure internet communication uses cryptography.
- **Data Protection:** Encrypting sensitive data stored on devices and in the cloud.

1.7 Conclusion

So, this chapter provides a comprehensive introduction to the field of cryptography and sets the stage for deeper exploration of cryptographic algorithms, protocols, and applications. By understanding the historical context, basic concepts, and modern applications of cryptography, we can gain a solid foundation for further new techniques and methods.

Chapter 2

Stream Cipher

Now, we'll delve into the mathematical foundations which are essential for understanding modern cryptographic algorithms. This section will cover fundamental concepts in number theory and algebra, which are critical for the development and analysis of cryptographic systems.

2.1 Basic Principles of Number Theory

Number theory provides the mathematical base or backrround for many cryptographic algorithms. The key concepts which we will include are given as follows:

2.1.1 Integers and Divisibility

In this section we'll see about the properties of integers and their divisibility. Some of the essential terms are:

- **Prime Numbers:** Numbers which are greater than 1 and have no divisors other than 1 and number themself.
- **Greatest Common Divisor (GCD):** The largest number which divides the given integers without leaving a remainder is their GCD.

Example: Finding the GCD

Given two integers $a = 56$ and $b = 98$, we can find their GCD using the Euclidean algorithm:

$$\begin{aligned} 98 &= 56 \cdot 1 + 42, \\ 56 &= 42 \cdot 1 + 14, \\ 42 &= 14 \cdot 3 + 0. \end{aligned}$$

Hence, the $\gcd(56, 98) = 14$.

2.1.2 Modular Arithmetic

A system of arithmetic for integers where numbers "wrap around" after reaching a certain value, called the modulus, is one of the most important stuff in cryptography.

Definition

For two integers, a and b, and positive integer n, a is called congruent to b modulo n if $a \equiv b \pmod{n}$. In simple terms, this means that n divides the difference $a - b$.

Example: Modular Addition

$$17 + 19 \equiv 36 \pmod{10} \equiv 6 \pmod{10}.$$

2.1.3 Modular Exponentiation

Computation of large powers efficiently in modular arithmetic is vital for various cryptographic algorithms such as RSA.

Example: Modular Exponentiation

Compute $3^{200} \mod 7$:

$$\begin{aligned}
&3^2 \equiv 9 \pmod{7} \equiv 2 \pmod{7},\\
&3^4 \equiv 2^2 \equiv 4 \pmod{7},\\
&3^8 \equiv 4^2 \equiv 16 \pmod{7} \equiv 2 \pmod{7},\\
&3^{16} \equiv 2^2 \equiv 4 \pmod{7},\\
&3^{32} \equiv 4^2 \equiv 16 \pmod{7} \equiv 2 \pmod{7}.
\end{aligned}$$

By continuing this process, we will find that $3^{200} \equiv 1 \pmod{7}$.

2.2 Important Algorithms in Cryptography

Several algorithms leverage the principles of number theory. Two significant ones which will be mentioned are:

2.2.1 Euclidean Algorithm

The Euclidean algorithm is used to find the GCD of two integers. The algorithm involves repeated division and taking remainders.

Example: Euclidean Algorithm

Find the GCD of 252 and 105:

$$\begin{aligned}
&252 = 105 \cdot 2 + 42,\\
&105 = 42 \cdot 2 + 21,\\
&42 = 21 \cdot 2 + 0.
\end{aligned}$$

Thus, $\gcd(252, 105) = 21$.

2.2.2 Extended Euclidean Algorithm

The extended Euclidean algorithm, unlike the above one, not only finds the GCD of two integers but also expresses it as a linear combination of the two integers, $ax + by = \gcd(a, b)$.

Example: Extended Euclidean Algorithm

Find integers x and y such that $240x + 46y = 2$:

$$\begin{aligned} 240 &= 46 \cdot 5 + 10, \\ 46 &= 10 \cdot 4 + 6, \\ 10 &= 6 \cdot 1 + 4, \\ 6 &= 4 \cdot 1 + 2, \\ 4 &= 2 \cdot 2 + 0. \end{aligned}$$

Rewriting the steps:

$$\begin{aligned} 2 &= 6 - 1 \cdot 4, \\ 4 &= 10 - 1 \cdot 6, \\ 6 &= 46 - 4 \cdot 10, \\ 10 &= 240 - 5 \cdot 46. \end{aligned}$$

Substituting back, we find:

$2 = 6-(10-1\cdot 6) = 2\cdot 6-10 = 2\cdot(46-4\cdot 10)-10 = 2\cdot 46-9\cdot 10 = 2\cdot 46-9\cdot(240-5\cdot 46) = 47\cdot 46-9\cdot 240.$

Thus, $x = -9$ and $y = 47$.

2.3 Groups, Rings, and Fields

So, we have to know about the algebraic structures of groups, rings, and fields well enough for many cryptographic algorithms:

2.3.1 Groups

A group is a set equipped with a single binary operation satisfying closure, associativity, identity, and invertibility.

Example: Multiplicative Group

The set of non-zero integers modulo n, denoted $(\mathbb{Z}/n\mathbb{Z})^*$, forms a group under multiplication.

2.3.2 Rings

A ring is a set equipped with two binary operations (addition and multiplication) satisfying the properties of a group under addition and closure under multiplication.

Example: Ring of Integers Modulo n

The set of integers modulo n, denoted $\mathbb{Z}/n\mathbb{Z}$, forms a ring.

2.3.3 Fields

A field is a ring in which division is possible (except by zero), meaning every non-zero element has a multiplicative inverse.

Example: Finite Field

The set of integers modulo a prime p, denoted $\mathbb{Z}/p\mathbb{Z}$, forms a finite field.

2.4 Conclusion

Hence, we covered a comprehensive overview of the mathematical foundations necessary for understanding modern cryptographic techniques. These were very important concept which will be required further in detailed manner.

Chapter 3

Data Encryption Standard (DES)

Now, we will read more about the Data Encryption Standard (DES), one of the most significant symmetric key cryptographic algorithms in the history of cryptography. We will cover the design principles, structure, and security of DES, as well as its role in the development of modern cryptographic systems.

3.1 Data Encryption Standard (DES)

Data Encryption Standard or DES is a symmetric key block cipher that was developed back in 1970s and became a standard for data encryption. This algorithm operates on 64-bit blocks and uses a 56-bit key.

3.1.1 DES Structure

DES consists of 16 rounds of the Feistel network which is a structure used to build many symmetric block ciphers. Each round involves several operations which includes permutations and substitutions.

Initial and Final Permutations

DES begins with an initial permutation (IP) and ends with a final permutation (FP), which are fixed and inverses of each other.

Example: Initial Permutation

The initial permutation reorders the 64 input bits according to a predefined table. For instance, if the input bits are $x_1, x_2, \ldots, x_{64}$, the permutation might produce $x_{58}, x_{50}, \ldots, x_7$.

3.1.2 Round Function

Each of the 16 rounds of DES uses a round function f that combines the data with a sub-key. The round function involves following things:

- **Expansion or (E)**: It involves expanding the 32-bit half-block to 48 bits blocks.
- **Key Mixing**: It involves XORing the expanded half-block with a 48-bit sub-key.
- **Substitution or (S-boxes)**: It involves substituting the 48-bit result with 32 bits using the S-boxes.

- **Permutation (P)**: It involves permuting the 32-bit output from the S-boxes.

Example: S-box Substitution

Let us consider, the first S-box, which maps 6-bit input to 4-bit output. Suppose, the input to this S-box is 011011:

Row: 01 (first and last bits),

Column: 1101 (middle four bits).

If the S-box maps row 1, column 13 to 9, then the 4-bit output is 1001.

3.1.3 Key Schedule

DES generates 16 subkeys, each of them are 48 bits long, from the original 56-bit key. The key schedule involves:

- **Permutation Choice 1 or (PC-1)**: This means selecting and permuting 56 bits from the 64-bit key (excluding 8 parity bits).
- **Left Shifts**: Means rotating the halves of the key.
- **Permutation Choice 2 (PC-2)**: Includes selecting 48 bits from the permuted key halves to form each subkey.

Example: Key Generation

Assume we are given a 64-bit key, suppose the initial permutation (PC-1) produces the 56-bit key:

$$K = k_1, k_2, \ldots, k_{56}.$$

After left shifts and PC-2, the first subkey K_1 might be:

$$K_1 = k_3, k_{17}, \ldots, k_{48}.$$

3.2 Security of DES

The security of DES has been a subject of extensive analysis. Various factors affect its security are:

3.2.1 Brute Force Attacks

One of the primary weakness of DES is its relatively short key length of 56 bits, which makes it vulnerable to brute force attacks. If we have enough computational power, we can can try all possible keys to decrypt a cipher-text.

Example: Brute Force Attack Time

If an attacker can test 10^6 keys per second, the total time to test all 2^{56} keys is:

$$\frac{2^{56}}{10^6 \times 60 \times 60 \times 24 \times 365} \approx 228 \text{ years}.$$

However, this time can be reduced significantly with modern parallel computing and specialized hardware.

3.2.2 Differential Cryptanalysis

Differential cryptanalysis is a chosen plaintext attack that analyzes the effect of specific differences in plaintext pairs on the differences of the resultant cipher-text pairs.

Example: Differential Cryptanalysis

Suppose, an attacker knows the plaintexts P and P' and their corresponding ciphertexts C and C'. By analyzing the differences $\Delta P = P \oplus P'$ and $\Delta C = C \oplus C'$, the attacker can gain the basic information about the key.

3.2.3 Linear Cryptanalysis

Linear cryptanalysis is another attack method that uses linear approximations to describe the behavior of the block cipher. This method requires a large number of known plaintext-ciphertext pairs to work.

Example: Linear Cryptanalysis

Given a linear approximation:

$$P[i] \oplus P[j] \oplus C[k] \approx K[m],$$

where P is the plaintext, C is the ciphertext, and K is the key. By analyzing a large number of such approximations, the attacker can determine key bits with high probability.

3.3 Triple DES

DES was very vulnerable, to address its vulnerabilities Triple DES or 3DES was developed which involves applying DES three times with using either two or three different keys.

3.3.1 Operation Modes

- **EDE (Encrypt-Decrypt-Encrypt)**: Encrypt with K_1, decrypt with K_2, and encrypt again with K_1.
- **EEE (Encrypt-Encrypt-Encrypt)**: Encrypt with K_1, encrypt with K_2, and encrypt again with K_3.

Example: Triple DES Encryption

Given a plaintext block P and keys K_1, K_2, and K_3:

$$C = E_{K_3}(E_{K_2}(E_{K_1}(P))).$$

3.4 Conclusion

So, we saw the detailed overview on DES algorithm, its structure and its security. By understanding the design and weaknesses of DES, we can appreciate the evolution of crypto-graphic standards and the development of more secure algorithms like Triple-DES and AES.

Chapter 4

Advanced Encryption Standard (AES)

Now we'll see the Advanced Encryption Standard (AES), a symmetric key block cipher that has become the global standard for data encryption and security. We will know about the design principles, structure, and security of AES.

4.1 The Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a symmetric block cipher chosen by the U.S. government to protect classified information.
AES is a symmetric key block cipher that operates on 128-bit blocks and supports key sizes of 128, 192, and 256 bits. It was established by the National Institute of Standards and Technology (NIST) in 2001, replacing DES as the encryption standard.

4.1.1 AES Structure

AES uses a substitution-permutation network (SPN) structure rather than a Feistel network. The number of rounds depends on the key size:

- 10 rounds for 128-bit keys.
- 12 rounds for 192-bit keys.
- 14 rounds for 256-bit keys.

Each round consists of four main operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

4.1.2 SubBytes

The SubBytes step is a non-linear substitution step where each byte in the state is replaced with its corresponding value from an substitution box (S-box).

Example: SubBytes Transformation

Suppose, we are given a byte $b = 0x53$ which is hexadecimal. The S-box maps $0x53$ to $0xED$. Therefore, b is replaced by $0xED$.

4.1.3 ShiftRows

ShiftRows is a transposition step where the rows of the state are shifted cyclically.

Example: ShiftRows Transformation

For a 4x4 matrix state:

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

So, the transformation would be:

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{bmatrix}$$

4.1.4 MixColumns

The MixColumns step is a mixing operation that operates on the columns of the state by combining the four bytes in each column.

Example: MixColumns Transformation

Let the state be:

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \\ c_0 & c_1 & c_2 & c_3 \\ d_0 & d_1 & d_2 & d_3 \end{bmatrix}$$

MixColumns multiplies each column by a fixed polynomial in the Galois Field $GF(2^8)$:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

For example, if the first column is a_0, b_0, c_0, d_0, the transformation would be:

$$\begin{aligned} \text{New } a_0 &= (02 \cdot a_0) \oplus (03 \cdot b_0) \oplus c_0 \oplus d_0, \\ \text{New } b_0 &= a_0 \oplus (02 \cdot b_0) \oplus (03 \cdot c_0) \oplus d_0, \\ \text{New } c_0 &= a_0 \oplus b_0 \oplus (02 \cdot c_0) \oplus (03 \cdot d_0), \\ \text{New } d_0 &= (03 \cdot a_0) \oplus b_0 \oplus c_0 \oplus (02 \cdot d_0). \end{aligned}$$

4.1.5 AddRoundKey

In the AddRoundKey step, the state is combined with a su-bkey using bitwise XOR.

Example: AddRoundKey Transformation

Suppose the state is:

$$\begin{bmatrix} d4 & bf & 5d & 30 \\ e0 & b4 & 52 & ae \\ b8 & 41 & 11 & f1 \\ 1e & 27 & 98 & e5 \end{bmatrix}$$

And the subkey is:

$$\begin{bmatrix} a0 & fa & 22 & 24 \\ c8 & 66 & 99 & ff \\ e0 & 88 & 0b & 2d \\ 06 & 20 & c9 & 9e \end{bmatrix}$$

The result of AddRoundKey will be:

$$\begin{bmatrix} d4 \oplus a0 & bf \oplus fa & 5d \oplus 22 & 30 \oplus 24 \\ e0 \oplus c8 & b4 \oplus 66 & 52 \oplus 99 & ae \oplus ff \\ b8 \oplus e0 & 41 \oplus 88 & 11 \oplus 0b & f1 \oplus 2d \\ 1e \oplus 06 & 27 \oplus 20 & 98 \oplus c9 & e5 \oplus 9e \end{bmatrix} = \begin{bmatrix} 74 & 45 & 7f & 14 \\ 28 & d2 & cb & 51 \\ 58 & c9 & 1a & dc \\ 18 & 07 & 51 & 7b \end{bmatrix}$$

4.2 Key Expansion

AES uses a key expansion process to generate a series of round keys from the initial key. This process involves several steps:

- **Key Splitting:** Splitting the initial key into four 32-bit words.
- **RotWord:** Rotating the bytes within a word.
- **SubWord:** Applying the SubBytes transformation to a word.
- **Rcon:** Adding a round constant.

Example: Key Expansion for AES-128

Given a 128-bit key:

$$K = \{2b7e151628aed2a6abf7158809cf4f3c\}.$$

The key expansion process generates 44 32-bit words:

$$W_0 = 2b7e1516, W_1 = 28aed2a6, W_2 = abf71588, W_3 = 09cf4f3c.$$

For $i \geq 4$:

$$W_i = W_{i-4} \oplus \text{SubWord}(\text{RotWord}(W_{i-1})) \oplus \text{Rcon}[i/4] \text{ if } i \text{ is a multiple of 4.}$$

4.3 Security of AES

AES is considered secure due to its robust design and resistance to various cryptographic attacks. However, its security relies on proper implementation and key management.

4.3.1 Brute Force Attacks

Due to the large key sizes (128, 192, and 256 bits), AES is resistant to brute force attacks. The number of possible keys are :

$$2^{128}, 2^{192}, \text{ and } 2^{256}.$$

Example: Brute Force Attack Time

Even if we suppose that the attacker can test 10^{15} keys per second, the time which will be required to test all 2^{128} keys is:

$$\frac{2^{128}}{10^{15} \times 60 \times 60 \times 24 \times 365} \approx 10^{18} \text{ years.}$$

Even after using modern parallel computing the still remains very high.

4.3.2 Known Attacks

Although AES is secure against known practical attacks, some theoretical attacks which exists are:

- **Linear and Differential Cryptanalysis:** These attacks are infeasible due to the large number of required plaintext-ciphertext pairs.
- **Side-Channel Attacks:** These are the attacks that exploit physical implementations of AES, such as power consumption or electromagnetic leaks.

4.4 Conclusion

Hence, we saw a comprehensive overview of AES, detailing its structure, operations, and security. By understanding the intricacies of AES, we can appreciate its role as a milestone of modern crypto-graphic practices.

Chapter 5

Different Modes of Operations

Now, we'll see a mode of operation called block cipher mode. This mode is used to encrypt data larger than their block size which helps in enabling practical applications in real world apps.We'll see several modes like Electronic Codebook Mode (ECB), Cipher BLock Chaining Mode (CBC), Output Feedback Mode (OFB), Cipher Feedback Mode (CFB), Counter Mode (CTR) and Galois Counter Mode (GCM) and their usage in outside world.

5.1 Block Cipher Modes of Operation

It is the mode of operation which defines how to apply a block cipher algorithm multiple times to encrypt or decrypt data that exceeds the block size.

5.1.1 Electronic Codebook Mode (ECB)

In ECB mode, each plaintext block is encrypted independently using the same key. This mode is way to simple to execute but comes with lots of security concerns.

Encryption and Decryption

$$C_i = E_K(P_i),$$
$$P_i = D_K(C_i),$$

Here, C_i is i^{th} cipher-text block, P_i is i^{th} plain-text block, E_K is encryption function and D_K is decryption function.

Example: ECB Mode

Let us assume the following plain-text blocks P_1, P_2, P_3 and a key K:

$$C_1 = E_K(P_1),$$
$$C_2 = E_K(P_2),$$
$$C_3 = E_K(P_3).$$

For, decryption above will be followed.

Security Weaknesses

This technique reveals patterns in the plaintext because identical plaintext blocks produce identical ciphertext blocks due to which it becomes unsuitable for encrypting data with repeating patterns.

Example: ECB Weakness

If a bitmap image is encrypted with ECB mode, the structure of the image may still be discernible in the ciphertext.

5.1.2 Cipher Block Chaining (CBC) Mode

Each of the plaintext block is XORed with the ciphertext block before it being encrypted. For the first block an initialization vector is used.

Encryption and Decryption

$$\begin{aligned} C_0 &= IV, \\ C_i &= E_K(P_i \oplus C_{i-1}), \\ P_i &= D_K(C_i) \oplus C_{i-1}, \end{aligned}$$

where IV is the initialization vector.

Example: CBC Mode

Consider plaintext blocks P_1, P_2, P_3 and a key K, with $IV = C_0$:

$$\begin{aligned} C_1 &= E_K(P_1 \oplus IV), \\ C_2 &= E_K(P_2 \oplus C_1), \\ C_3 &= E_K(P_3 \oplus C_2). \end{aligned}$$

Security Properties

This mode provides high confidentiality and diffusion which makes it more fool proof than ECB. But, it requires a unique IV for each encryption session to maintain security.

5.1.3 Cipher Feedback (CFB) Mode

This mode allows block ciphers to operate as stream ciphers. It encrypts small segments of plaintext, typically one byte or one bit at a time.

Encryption and Decryption

$$\begin{aligned} C_0 &= IV, \\ C_i &= E_K(C_{i-1}) \oplus P_i, \\ P_i &= E_K(C_{i-1}) \oplus C_i, \end{aligned}$$

where IV is the initialization vector.

Example: CFB Mode

Consider plaintext bytes P_1, P_2, P_3 and a key K, with $IV = C_0$:

$$\begin{aligned} C_1 &= E_K(IV) \oplus P_1, \\ C_2 &= E_K(C_1) \oplus P_2, \\ C_3 &= E_K(C_2) \oplus P_3. \end{aligned}$$

Security Properties

For encrypting data of arbitary length and confidentiality, CFB mode is suitable. Major disadvantages is that it can propagate errors in various applications.

5.1.4 Output Feedback (OFB) Mode

This mode transforms a block cipher into a synchronous stream cipher. It generates keystream blocks independent of the plaintext and ciphertext.

Encryption and Decryption

$$
\begin{aligned}
O_0 &= IV, \\
O_i &= E_K(O_{i-1}), \\
C_i &= P_i \oplus O_i, \\
P_i &= C_i \oplus O_i,
\end{aligned}
$$

where IV is the initialization vector.

Example: OFB Mode

Consider plaintext bytes P_1, P_2, P_3 and a key K, with $IV = O_0$:

$$
\begin{aligned}
O_1 &= E_K(IV), \\
C_1 &= P_1 \oplus O_1, \\
O_2 &= E_K(O_1), \\
C_2 &= P_2 \oplus O_2, \\
O_3 &= E_K(O_2), \\
C_3 &= P_3 \oplus O_3.
\end{aligned}
$$

Security Properties

OFB mode provides confidentiality and allows for pre-computation of the key-stream, which can improve performance. One of the major shortcoming is that is prone to synchronization errors.

5.1.5 Counter (CTR) Mode

CTR mode transforms a block cipher into a stream cipher by encrypting a counter value, which is then XORed with the plaintext.

Encryption and Decryption

$$
\begin{aligned}
O_i &= E_K(\text{Counter}_i), \\
C_i &= P_i \oplus O_i, \\
P_i &= C_i \oplus O_i,
\end{aligned}
$$

where Counter_i is a unique counter value for each block.

Example: CTR Mode

Consider plaintext bytes P_1, P_2, P_3 and a key K:

$$
\begin{aligned}
O_1 &= E_K(\text{Counter}_1),\\
C_1 &= P_1 \oplus O_1,\\
O_2 &= E_K(\text{Counter}_2),\\
C_2 &= P_2 \oplus O_2,\\
O_3 &= E_K(\text{Counter}_3),\\
C_3 &= P_3 \oplus O_3.
\end{aligned}
$$

Security Properties

CTR mode provides confidentiality, supports parallel processing, and allows random access to encrypted data. It requires a unique counter value for each encryption session to maintain security.

5.2 Conclusion

Hence, we saw here various block cipher modes of operations, their structures, security properties and practical applications. These modes can be used appropriately for getting desired levels of accuracy and efficiency.

Chapter 6

Public Key Crypto

6.1 Introduction

Now, let's jump to the ElGamal cryptosystem, a public key cryptosystem based on the Diffie-Hellman key exchange. We'll briefly discuss the mathematical foundation, key generation, encryption, decryption, and security aspects of ElGamal with its practical examples to illustrate its operation.

6.2 The ElGamal Crypto-system

The ElGamal cryptosystem is a public key cryptosystem based on the hardness of the discrete logarithm problem in a finite field. It comprises three main steps: key generation, encryption, and decryption.

6.2.1 Mathematical Foundation

The ElGamal cryptosystem relies on the difficulty of solving the discrete logarithm problem, which involves finding x such that $g^x \equiv h \mod p$ for given g, h, and p.

Discrete Logarithm Problem

Given a prime p and a generator g of the multiplicative group of integers modulo p, the discrete logarithm problem is finding x such that:

$$g^x \equiv h \mod p$$

6.2.2 Key Generation

Key generation involves selecting a large prime number, a generator, and computing public and private keys.

Steps of Key Generation

- Choose a large prime p and a generator g of the multiplicative group $\mathbb{Z}_p^*$.
- Select a private key a such that $1 < a < p - 1$.
- Compute the public key $h = g^a \mod p$.

The public key is (p, g, h) and the private key is a.

Example: Key Generation

Let $p = 23$ and $g = 5$. Choose a private key $a = 6$:

$$h = g^a \mod p = 5^6 \mod 23 = 8$$

Public key: $(23, 5, 8)$, Private key: 6.

6.2.3 Encryption

To encrypt a message M using the public key (p, g, h), the sender performs the following steps:

- Choose a random integer k such that $1 < k < p - 1$.
- Compute $c_1 = g^k \mod p$.
- Compute $c_2 = M \cdot h^k \mod p$.

The ciphertext is (c_1, c_2).

Example: Encryption

Given $M = 15$ and public key $(23, 5, 8)$, choose $k = 10$:

$$c_1 = 5^{10} \mod 23 = 9$$

$$c_2 = 15 \cdot 8^{10} \mod 23 = 15 \cdot 6 \mod 23 = 21$$

Ciphertext: $(9, 21)$.

6.2.4 Decryption

To decrypt the ciphertext (c_1, c_2) using the private key a, the receiver computes the message M as:

$$M = c_2 \cdot (c_1^a)^{-1} \mod p$$

Example: Decryption

Given ciphertext $(9, 21)$ and private key 6:

$$c_1^a \mod p = 9^6 \mod 23 = 13$$

$$(c_1^a)^{-1} \mod p = 13^{-1} \mod 23 = 16$$

$$M = 21 \cdot 16 \mod 23 = 15$$

6.3 Security of ElGamal

The security of ElGamal is based on the computational difficulty of the discrete logarithm problem.

6.3.1 Discrete Logarithm Problem

The difficulty of solving the discrete logarithm problem in a finite field provides the foundation for the security of ElGamal.

Example: Discrete Logarithm Challenge

Given $p = 23$, $g = 5$, and $h = 8$, solving for x in $5^x \equiv 8 \mod 23$ is infeasible for large p.

6.3.2 Chosen Ciphertext Attacks

ElGamal can be vulnerable to chosen ciphertext attacks if not properly implemented. CCA involves an attacker being able to decrypt chosen ciphertexts to gather information about the private key.

Example: CCA Vulnerability

If an attacker can modify ciphertext (c_1, c_2) and get the corresponding plaintext M, they can exploit mathematical properties of ElGamal to gather information about the private key.

6.4 Conclusion

We saw an overview on the ElGamal cryptosystem, detailing its mathematical foundation, key generation, encryption, decryption, and security. We also saw examples which made our understanding easier about the concept of ElGamal.

Chapter 7

Digital Signatures

7.1 Introduction

In this chapter, we shall explore about the Digital Signature Algorithm (DSA), a standard for digital signatures. We'll discuss about the mathematical foundations, key generation, signing process, verification process and security aspects of DSA, along with practical examples to illustrate its operations.

7.2 The Digital Signature Algorithm (DSA)

The Digital Signature Algorithm is a Federal Information Processing Standard for digital signatures. DSA is based on the discrete logarithm problem and is used to ensure the authenticity and integrity of digital messages or documents.

7.2.1 Mathematical Foundation

DSA relies on the difficulty of the discrete logarithm problem in the multiplicative group of integers modulo a prime.

Discrete Logarithm Problem

Given a prime p and a generator g of the multiplicative group $\mathbb{Z}_p^*$, the discrete logarithm problem involves finding x such that:

$$g^x \equiv y \mod p$$

7.2.2 Key Generation

Key generation involves selecting domain parameters and generating public and private keys.

Steps of Key Generation

- Choose a large prime p and q such that q divides $p-1$.
- Choose a generator g of order q in $\mathbb{Z}_p^*$.
- Select a private key x such that $0 < x < q$.
- Compute the public key $y = g^x \mod p$.

The public key is (p, q, g, y) and the private key is x.

Example: Key Generation

Let $p = 23$, $q = 11$, and $g = 2$. Choose a private key $x = 6$:

$$y = g^x \mod p = 2^6 \mod 23 = 9$$

Public key: $(23, 11, 2, 9)$, Private key: 6.

7.2.3 Signing Process

For signing a message M using the private key x, the signer will perform following steps:

- A random number k will be chosen such that $0 < k < q$.
- Now, he will compute $r = (g^k \mod p) \mod q$.
- Then, $s = k^{-1}(H(M) + xr) \mod q$.

The required signature will be (r, s).

Example: Signing Process

Lets, take an example. We have given M, a private key $x = 6$ and random $k = 7$:

$$r = (2^7 \mod 23) \mod 11 = 13 \mod 11 = 2$$

$$H(M) = 5 \quad \text{(assume a hash value for } M\text{)}$$

$$s = 7^{-1}(5 + 6 \cdot 2) \mod 11 = 8 \cdot (5 + 12) \mod 11 = 8 \cdot 17 \mod 11 = 8 \cdot 6 \mod 11 = 4$$

Hence, signature: $(2, 4)$.

7.2.4 Verification Process

To verify the signature (r, s) of a message M using the public key (p, q, g, y), the verifier will perform the following steps:

- Firstly, compute $w = s^{-1} \mod q$.
- Then, he will do $u_1 = H(M)w \mod q$ and $u_2 = rw \mod q$.
- Then, $v = ((g^{u_1} y^{u_2}) \mod p) \mod q$.
- Finally, the signature will be accepted if $v = r$.

Example: Verification Process

Let a digital signature is given which is $(2, 4)$, a message M, and the public key $(23, 11, 2, 9)$:

$$w = 4^{-1} \mod 11 = 3$$

$$u_1 = 5 \cdot 3 \mod 11 = 15 \mod 11 = 4$$

$$u_2 = 2 \cdot 3 \mod 11 = 6$$

$$v = ((2^4 \cdot 9^6) \mod 23) \mod 11 = (16 \cdot 13 \mod 23) \mod 11 = (208 \mod 23) \mod 11 = 2$$

Since, $v = r$, so the signature is verified.

7.3 Security of DSA

The security of DSA is based on the hardness of the discrete logarithm problem and the properties of the hash function used in the signing process.

7.3.1 Discrete Logarithm Problem

The security of DSA relies on the difficulty of solving the discrete logarithm problem in a finite field.

Example: Discrete Logarithm Challenge

Given $p = 23$, $g = 2$, and $y = 9$, solving for x in $2^x \equiv 9 \mod 23$ is infeasible for large p.

7.3.2 Hash Function Security

The hash function used in DSA must be secure to ensure the integrity of the signed message.

Example: Hash Function Role

If the hash function $H(M)$ is weak, an attacker could find collisions, leading to forgery of signatures.

7.4 Conclusion

So, we learnt about digital signatures, its mathematical aspects and applications.

Chapter 8

References

- Understanding Cryptography by Crystof Paar and Jan Pelzl
- Youtube link: $https://youtube.com/@introductiontocryptography4223?si=vd2IFozBeYB-xEc2$
- Youtube link: $https://youtu.be/-UrdExQW0cs?si=QTxtd5gAztbFg3\_z$
- GeekforGeeks
- AES link: $https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard$
- Various other websites and youtube videos