SAI AKASH
20BEC0405

Assessment – 3

Cryptography Analysis and Implementa on

Objec ve: The objec ve of this assignment is to analyze cryptographic algorithms and implement them in a prac cal scenario.

DES

A symmetric encryp on technique that uses 64-bit data blocks is called the Data Encryp on Standard (DES). Here is a quick explana on of the DES algorithm's opera on:

Key Genera on: DES employs a 56-bit encryp on key. However, the effec ve key size is decreased to 56 bits due to security concerns by using 8 of the 64 bits for parity tests. An algorithm that incorporates human input and a random component creates the key.

Key Expansion: For each encryp on cycle, a separate 48-bit subkey is created from the 56-bit key. This expansion comprises a series of mathema cal procedures that permute and transform the key.

Ini al Permuta on (IP): The bits in the 64-bit plaintext block are rearranged in accordance with a predetermined pa ern.

Rounds of Encryp on: DES uses 16 rounds of encryp on. The following steps are involved in each round:

a. Expansion: Using a specified expansion func on, the 32-bit right half of the data from the preceding round is enlarged to 48 bits.

b. Key Mixing: The relevant round subkey is XORed with the extended 48-bit data.

c. Subs tu on: Eight 6-bit blocks make up the final product. A predetermined S-box (subs tu on box), which converts each block's six bits into four bits using a lookup table, is used to subs tute each block.

d. Permuta on: Using a predetermined permuta on pa ern, the subs tuted data is then rearranged.

e. XOR and Swap: The data that has been permuted is XORed with the 32-bit le half of the data from the previous round. The next round's right half is the 32-bit block that was produced, and the le half is the preceding right half.

A er the 16 cycles, the data is subjected to a final permuta on (FP), which is the opposite of the ini al permuta on.

The same procedure is used to decrypt the ciphertext, but the subkeys are used in reverse.

SAI AKASH
20BEC0405

Strengths:

1. Simplicity: DES has a rela vely simple structure, making it easy to implement and understand.
2. Performance: DES was designed to be efficient in hardware implementa ons, which made it suitable for many applica ons at the me.

Weaknesses:

1. Key Length: The key length of 56 bits is considered short by today's standards, and it can be brute-forced within a reasonable amount of me using modern compu ng resources.
2. Vulnerability to A acks: DES has been subjected to various cryptanaly c a acks over the years. The most significant a ack is the exhaus ve search, where all possible keys are tested un l the correct one is found.
3. Security Margin: Due to advances in compu ng power, the security margin of DES has significantly decreased over the years. It is now recommended to use more robust encryp on algorithms.

Common Use Cases:

1. Legacy Systems: DES may s ll be encountered in legacy systems that have not been updated or where compa bility with older systems is required.
2. Educa onal Purposes: DES is o en used in academic se ngs to teach the fundamentals of cryptography and encryp on algorithms.

SAI AKASH
20BEC0405

Ellip c Curve Cryptography ECC

Key genera on, key exchange, encryp on, and decryp on are all components of the ellip c curve cryptography (ECC) algorithm. The ECC algorithm is described in the following general terms:

Genera ng a Key:

Pick a base point on an ellip c curve that is specified over a finite field.

Pick a private key, which is a chance number that falls within a certain range.

By employing ellip c curve scalar mul plica on, mul ply the base point by the private key to create the public key.

Exchanging keys

Alice and Bob, two par es, concur on an ellip c curve and its base point.

Both Alice's public and private keys are generated.

Bob uses both his private key and public key in the same way.

Public keys are exchanged between Alice and Bob.

To obtain a shared secret point on the curve, each party mul plies their respec ve private key and the public key they got.

Encryp on:

Bob is the recipient of a communica on from Alice.

Alice creates a session key at random.

By dividing the base point by the session key, she calculates the coordinates of an ephemeral public key.

The shared secret point is calculated by Alice by dividing Bob's public key by her session key.

Alice uses the shared secret point to derive a symmetric encryp on key.

She uses the symmetric encryp on key to encrypt the communica on.

Decryp on:

Alice gives Bob the temporary public key and the encrypted message.

Bob mul plies his private key with Alice's transitory public key to determine the shared secret point.

Bob and Alice arrive at the same symmetric encryp on key.

SAI AKASH
20BEC0405

He uses the symmetric encryp on key to decrypt the communica on.

Strengths:

1. Security: ECC provides a high level of security with smaller key sizes compared to other public-key algorithms such as RSA. This makes ECC more efficient in terms of computa onal resources and bandwidth.
2. Key Size Efficiency: ECC provides an equivalent level of security with significantly smaller key sizes compared to other asymmetric algorithms, reducing the computa onal and storage requirements.

Weaknesses:

1. Implementa on Complexity: ECC requires careful implementa on and parameter selec on to ensure security. Improper implementa on can introduce vulnerabili es.
2. Patented Algorithms: Some ECC algorithms may be subject to patents, which can limit their use and adop on.

Common Use Cases:

1. Secure Communica on: ECC is widely used in protocols such as Transport Layer Security (TLS) to secure communica on channels over the internet.
2. Internet of Things (IoT): ECC's efficiency and security make it suitable for resourceconstrained devices in IoT environments.
3. Digital Signatures: ECC is used for genera ng and verifying digital signatures, ensuring the integrity and authen city of digital documents.

SAI AKASH
20BEC0405

MD5 (Message Digest Algorithm 5)

is a widely used cryptographic hash func on that produces a 128-bit (16-byte) hash value. It takes an input message of any length and produces a fixed-size output, which is commonly represented as a 32-digit hexadecimal number.

The algorithm follows the following steps:

1. Padding: The input message is padded to make its length a mul ple of 512 bits (64 bytes). The padding is done in such a way that the resul ng message length is congruent to 448 modulo 512. The padding starts with a single bit "1" followed by a sequence of "0" bits un l the length requirement is met. A er that, the original message length is appended as a 64bit representa on.
2. Ini aliza on: MD5 uses four 32-bit state variables (A, B, C, D) and a 64-element table (T[1..64]) containing precomputed values. The ini al values of A, B, C, and D are fixed and serve as the ini al state.
3. Message Processing: The padded message is divided into blocks of 512 bits. Each block is further divided into 16 words of 32 bits each (M[0..15]). The message is processed in a loop for each block.
4. Round Func on: MD5 uses four basic func ons (F, G, H, I) that operate on the state variables A, B, C, and D, as well as the current block's words. Each func on takes three inputs (X, Y, Z) and produces a 32-bit output. The func ons and their opera ons are as follows:
   - F(X, Y, Z) = (X AND Y) OR ((NOT X) AND Z)
   - G(X, Y, Z) = (X AND Z) OR (Y AND (NOT Z))
   - H(X, Y, Z) = X XOR Y XOR Z
   - I(X, Y, Z) = Y XOR (X OR (NOT Z))
5. Round Opera ons: Each block goes through four rounds of processing. In each round, a different func on is used, and the results are combined with the state variables using modular addi on. The block's 16 words are used as inputs to the func ons in a

predetermined order.

6. Output: A er processing all the blocks, the resul ng values of A, B, C, and D are concatenated, usually in li le-endian format. The resul ng 128-bit hash value represents the message's fingerprint.

Strengths:

1. Speed: MD5 is fast and efficient in terms of computa on, making it suitable for applica ons where performance is cri cal.
2. Compa bility: MD5 is supported by many so ware systems and programming languages, making it easy to implement and use.

SAI AKASH
20BEC0405

Weaknesses:

1. Collision Vulnerability: MD5 is considered weak against collision a acks, where two different inputs produce the same hash value. This weakness makes it unsuitable for security-cri cal applica ons.
2. Preimage Vulnerability: MD5 is also suscep ble to preimage a acks, where an a acker can find an input that produces a specific hash value, compromising the integrity of the system. Common Use Cases:

1. Checksums: MD5 checksums are used to verify the integrity of files. By comparing the MD5 hash of a downloaded file with the provided hash

IMPLEMENTING DES
CODE

SAI AKASH
20BEC0405

```
#include <iostream>
#include <cstring>
#include <openssl/des.h>

// Function to encrypt using DES algorithm
void encryptDES(const unsigned char* plaintext, const unsigned char* key, unsigned char*
ciphertext) {     DES_cblock keyEncrypt;
    DES_key_schedule schedule;

    // Prepare the key for encryption     memcpy(keyEncrypt, key, 8);
    DES_set_odd_parity(&keyEncrypt);
    DES_set_key_checked(&keyEncrypt, &schedule);

    // Encrypt the plaintext
    DES_ecb_encrypt((DES_cblock*)plaintext, (DES_cblock*)ciphertext,
&schedule, DES_ENCRYPT);
}

// Function to decrypt using DES algorithm void decryptDES(const unsigned char* ciphertext, const
unsigned char* key, unsigned char* plaintext) {     DES_cblock keyEncrypt;
    DES_key_schedule schedule;

    // Prepare the key for decryption     memcpy(keyEncrypt, key, 8);
DES_set_odd_parity(&keyEncrypt);
```

```
    DES_set_key_checked(&keyEncrypt, &schedule);

    // Decrypt the ciphertext
    DES_ecb_encrypt((DES_cblock*)ciphertext, (DES_cblock*)plaintext,
&schedule, DES_DECRYPT);
}  int main() {
    // Input plaintext and key     const unsigned char plaintext[] = "Hello, DES!";
const unsigned char key[] = "SecretKey";

    // Buffer to store the encrypted and decrypted data     unsigned char ciphertext[32];
unsigned char decryptedtext[32];

    // Encrypt the plaintext     encryptDES(plaintext, key, ciphertext);
    // Decrypt the ciphertext     decryptDES(ciphertext, key, decryptedtext);

    // Print the encrypted and decrypted messages     std::cout << "Plaintext: " << plaintext <<
std::endl;     std::cout << "Ciphertext: " << ciphertext << std::endl;     std::cout << "Decrypted text: "
<< decryptedtext << std::endl;
     return 0;
}
```

Poten al Threats for DES algo

When implemen ng the DES algorithm, there are several poten al threats and vulnerabili es that could be exploited. Here are some of them:

1. Key Security: The security of the algorithm heavily relies on the secrecy and strength of the encryp on key. If the key is weak or compromised, the en re system becomes vulnerable. Therefore, it is crucial to use a strong and random key and protect it using secure key management prac ces.

Countermeasure: Implement a robust key management system that includes strong key genera on techniques, secure storage, and appropriate key rota on policies. Consider using longer key lengths and more advanced encryp on algorithms, such as AES, for stronger security.

2. Brute Force A acks: DES has a rela vely small key length (56 bits), which makes it suscep ble to brute force a acks. An a acker could systema cally try all possible keys un l the correct one is found, especially with the increasing computa onal power available today.

Countermeasure: Use stronger encryp on algorithms, such as AES, which provide longer key lengths and be er resistance against brute force a acks. AES is the recommended replacement for DES.

3. Cryptanalysis A acks: Over me, new a acks and vulnerabili es against cryptographic algorithms are discovered. DES, being an older algorithm, has known weaknesses and vulnerabili es. Advanced cryptanalysis techniques, such as differen al and linear cryptanalysis, can exploit these weaknesses.

Countermeasure: Upgrade to more modern and secure encryp on algorithms, such as AES. AES is widely adopted and extensively analyzed, providing be er resistance against known cryptanalysis a acks.

4. Implementa on Flaws: Errors or vulnerabili es in the implementa on of the algorithm can compromise the security. These flaws could be related to key handling, input valida on, or insecure cryptographic opera ons.

Countermeasure: Follow secure coding prac ces and rely on well-tested cryptographic libraries or frameworks. Regularly update and patch the implementa on to address any discovered vulnerabili es or weaknesses.

5. Side-Channel A acks: Side-channel a acks target informa on leaked during the execu on of the algorithm, such as ming, power consump on, or electromagne c emissions. These a acks can poten ally reveal sensi ve informa on, including the encryp on key.

Countermeasure: Implement countermeasures against side-channel a acks, such as using constantme algorithms, ensuring consistent execu on me regardless of input, and applying physical protec ons like tamper-resistant hardware.

6. Regulatory and Compliance Issues: Depending on the context and the data being encrypted, there may be legal or compliance requirements to consider. Using an outdated encryp on algorithm like DES may not meet the necessary security standards and could lead to compliance viola ons.

Countermeasure: Understand the relevant legal and compliance requirements for your specific use case. Ensure that your encryp on algorithms align with the recommended standards and industry best prac ces. Upgrade to modern and approved encryp on algorithms to meet compliance obliga ons.

SAI AKASH
20BEC0405

Trade Offs

During the implementa on process of the DES algorithm, several limita ons and trade-offs were encountered. Here are some of them:

1. Key Length: One significant limita on of DES is its rela vely short key length of 56 bits. This limited key length provides weaker security compared to modern encryp on algorithms. Brute force a acks can become feasible as computa onal power increases. As a result, the key length of DES is a trade-off between security and efficiency.
2. Speed: DES was designed for efficient hardware implementa on in the 1970s when computa onal power was limited. While DES is rela vely fast compared to other encryp on algorithms of that me, it is slower than modern encryp on algorithms like AES. When performance is a cri cal factor, the speed of DES can be a limita on.
3. Security: Over me, DES has been subjected to various cryptanalysis a acks, and several vulnerabili es have been discovered. Although DES is s ll considered secure for certain legacy applica ons, it is no longer considered secure for protec ng sensi ve data against advanced a acks. Therefore, the security provided by DES is a significant limita on compared to modern algorithms like AES.
4. Compa bility: DES is an older encryp on algorithm that may not be widely supported by modern systems and frameworks. Some programming languages or pla orms may lack builtin support for DES, requiring addi onal effort to integrate or implement the algorithm.
5. Regulatory Compliance: Depending on the industry or jurisdic on, there may be specific regula ons or compliance requirements that mandate the use of stronger encryp on algorithms like AES. Implemen ng DES may not meet these requirements, necessita ng an upgrade to a more secure algorithm.
6. Key Management: As DES uses a fixed key length, key management can be challenging. With a limited key space, the risk of key collisions and key reuse increases. This limita on can impact the overall security of the system if proper key management prac ces are not implemented.

THANK YOU