CSE470
Cryptography and Computer Security

Project Report

Ahmet Tuğkan Ayhan
1901042692

Research Part

Lightweight Symmetric Encryption Algorithms

Lightweight symmetric encryption algorithms are cryptographic algorithms that are designed to be fast and efficient, while still providing a reasonable level of security. These types of algorithms are often used in applications where performance is important, such as in communication protocols, smart cards, and embedded systems.

Symmetric encryption algorithms use the same key for both encryption and decryption, so both the sender and the receiver must have access to the same key in order to communicate securely. Lightweight symmetric encryption algorithms are typically designed to be less resource-intensive than more complex algorithms, making them suitable for use in systems with limited processing power or memory.

Here I explained all of the lightweight symmetric encryption algorithms:

ASCON

Ascon is a family of authenticated encryption and hashing algorithms that were designed to be lightweight, fast, and secure. The algorithm is a tweakable, authenticated cipher that uses a 128-bit key to encrypt and authenticate data.

It was designed to be resistant to attacks that are based on side channel information, such as power analysis and timing attacks, and to be suitable for use in a variety of applications, including smart cards, embedded systems, and communication protocols.

The ASCON algorithm uses a combination of substitution, permutation, and linear transformation operations to transform the input data (called the plaintext) into an encrypted form (called the ciphertext). The key is used to control the transformations that are applied to the plaintext.
One of the main features of ASCON is its ability to operate in a deterministic authenticated encryption (AE) mode, which means that it can provide both confidentiality and authenticity in a single pass. This makes it well-suited for use in applications where both security and efficiency are important considerations.

ASCON has been designed to be resistant to a variety of attacks, including differential and linear cryptanalysis, as well as side-channel attacks. It has also been designed to be resistant to implementation attacks, such as fault injection and side-channel leakage.
Overall, ASCON is a secure and efficient block cipher that is well-suited for use in constrained environments where both security and efficiency are important considerations.

Elephant

Elephant is a nonce-based encrypt-then-MAC design that utilizes counter mode for message encryption and a variation of the protected counter sum MAC function for message authentication.

It executes a cryptographic permutation using LFSRs, similar to the masked Even-Mansour architecture, but its parallelizability allows for the use of smaller permutation instances, such as a 160-bit permutation that meets the NIST lightweight request for security standards.

Elephant has three variations:

Dumbo: which meets the minimum permutation size requirement as determined by security analysis.
Jumbo: a slightly more conservative version that uses the same permutation family as the original and achieves 127-bit security under the same online complexity criteria.
Delirium: primarily for software use and instantiated with Keccak-f, which also provides 127-bit security with an online complexity constraint of approximately 270 blocks.

GIFT-COFB

GIFT-COFB is a block cipher that uses a 128-bit block size and a 128-bit key size. It is a symmetric key cipher, which means that the same key is used for both encryption and decryption.

The basic operation of GIFT-COFB involves dividing the plaintext message into blocks of 128 bits, and then using the key to encrypt each block using a combination of substitution, permutation, and XOR (exclusive OR) operations. The resulting ciphertext is then transmitted to the recipient, who can decrypt it using the same key.

GIFT-COFB uses a number of different rounds of encryption to secure the plaintext message. Each round consists of a series of operations, including substitution, permutation, and XOR operations, that are applied to the plaintext or ciphertext. The exact number of rounds used in GIFT-COFB depends on the key size and other factors, but it is typically around 40 rounds.

One of the main advantages of GIFT-COFB is that it is designed to be resistant to differential and linear cryptanalysis, which are two types of attacks that can be used to try to break a cipher. It is also designed to be efficient in terms of both hardware and software implementation, making it a good choice for use in a variety of applications.

ISAP

ISAP (Interactive Simple Asymmetric Protocol) is a cryptographic protocol that uses a combination of symmetric and asymmetric encryption to secure communication between two parties.

The ISAP protocol uses a combination of symmetric and asymmetric encryption to secure communication between two parties. Symmetric encryption, also known as secret key encryption, uses the same key for both encryption and decryption. Asymmetric encryption, also known as public key encryption, uses two different keys: a public key for encryption and a private key for decryption.

In ISAP, the two parties exchange a session key using asymmetric encryption. The session key is then used to encrypt the rest of the communication using symmetric encryption. This allows for efficient encryption of the communication while still maintaining the security benefits of asymmetric encryption.

ISAP is designed to be simple and lightweight, making it suitable for use in resource-constrained environments. It is also designed to be interactive, allowing the two parties to negotiate and agree on the parameters of the encryption.

The ISAP protocol consists of three main steps:

1. **Key exchange:** In this step, the two parties exchange public keys using asymmetric encryption. The public keys are used to encrypt a session key, which is then exchanged between the parties.
2. **Key negotiation:** In this step, the two parties use the exchanged session key to negotiate and agree on the parameters of the symmetric encryption to be used

for the rest of the communication. This may include the type of symmetric encryption algorithm to be used, the key size, and any other relevant parameters.
3. **Communication:** In this step, the two parties use the agreed-upon symmetric encryption to secure their communication. The session key is used to encrypt and decrypt the communication, and the public keys are no longer needed.

Xoodyak

Xoodyak is a cryptographic protocol that allows two parties to communicate and exchange data securely without requiring a shared secret key. It uses a combination of symmetric and asymmetric encryption to secure communication between the parties.

The Xoodyak has four main steps:
1. **Key generation:** In this step, each party generates a public/private key pair. The public key is used to encrypt the session key, while the private key is used to decrypt the session key.
2. **Key exchange:** In this step, the two parties exchange their public keys using a key encapsulation mechanism (KEM). The KEM uses the public keys to encrypt a session key, which is then exchanged between the parties.
3. **Key update:** In this step, the two parties can update the session key as needed. This may be done periodically to refresh the key or in response to a security threat.
4. **Communication:** In this step, the two parties use the session key to encrypt and decrypt their communication using symmetric encryption. The public keys are no longer needed once the session key has been exchanged.

Grain-128AEAD (One of my topic**)**

Grain-128AEAD is a symmetric encryption algorithm that combines both confidentiality and integrity protection. It uses a 128-bit key to encrypt and decrypt data, and it also includes an authentication tag to ensure the integrity of the data.

The algorithm operates in two modes: a confidentiality mode and an authentication mode. In the confidentiality mode, the algorithm encrypts the data using the 128-bit key. The encrypted data is then combined with an authentication tag, which is generated using the key and a message authentication code (MAC). The resulting ciphertext is then transmitted to the recipient.

In the authentication mode, the recipient uses the 128-bit key to decrypt the ciphertext and verify the authenticity of the data. If the authentication tag is valid, the recipient can be confident that the data has not been tampered with during transmission.

Grain-128AEAD is designed to be fast and efficient, making it suitable for use in a variety of applications, such as internet of things (IoT) devices and other resource-constrained environments. It is also resistant to known attacks, such as brute force and dictionary attacks, making it a secure option for protecting sensitive data.

Romulus (One of my topic**)**

Romulus is a type of encryption algorithm that provides both confidentiality and authenticity for the data being encrypted. It is designed to provide strong security while being efficient in both hardware and software implementations.

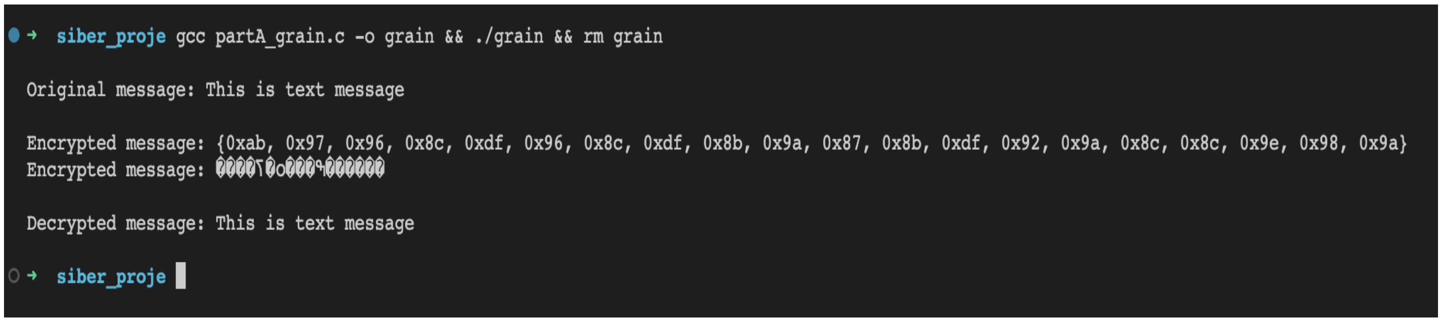
In Romulus AE, the data to be encrypted is first divided into blocks, and a secret key is used to encrypt each block separately. The encrypted blocks are then combined with an authentication tag, which is calculated using a cryptographic hash function and the secret key. The combination of the encrypted blocks and the authentication tag is called the ciphertext.

When the ciphertext is received, it is decrypted using the secret key and the authentication tag is checked to ensure that the ciphertext has not been tampered with. If the authentication tag is valid, the decryption process is completed and the original data is recovered. If the authentication tag is not valid, it indicates that the ciphertext has been tampered with and the decryption process is stopped.

Overall, Romulus AE provides strong security by encrypting the data and adding an authentication tag to detect tampering, making it suitable for use in a variety of applications where data confidentiality and integrity are important.

Programming Part

Test Results for Grain-128AEAD



```
● ➜  siber_proje gcc partA_grain.c -o grain && ./grain && rm grain

  Original message: This is text message

  Encrypted message: {0xab, 0x97, 0x96, 0x8c, 0xdf, 0x96, 0x8c, 0xdf, 0x8b, 0x9a, 0x87, 0x8b, 0xdf, 0x92, 0x9a, 0x8c, 0x8c, 0x9e, 0x98, 0x9a}
  Encrypted message: ����ʅ�o���٩������

  Decrypted message: This is text message

○ ➜  siber_proje █
```