U R Rao Satellite Centre

Bangalore, Karnataka, India.

Indian Institute of Space Science and Technology

Thiruvananthapuram, Kerala, India

AES - CCMP & GCMP Development for Space Communication

Intermediate Draft

21^{th} July, 2022

Submitted by:

Aravind Potluri, SC19B093

Dept. of Avionics, IIST

Supervised by:

Rama Murthy H, Scientist-SG, URSC.

Lijo P Jose, Scientist-SD, URSC.

Dr. BS Manoj, Sr.Professor, IIST.

Introduction

In today's world, we require communication protocols that guarantee proper secrecy, source authentication, and integrity protection for almost all operations.

Network security protocols strive to keep data in transit over network connections safe and secure. These protocols also outline how the network protects data from unauthorised efforts to inspect or extract it. This helps guarantee that no unauthorised users, services, or devices have access to your network data, and it applies to all data kinds and network media. Typically, network security protocols use encryption and cryptography to safeguard data so that only certain algorithms, formulae, and logical keys have access to it.

In this paper we are going to develop two crucial security protocols of IEEE 802.11 standards for use of space telecommunication link. The protocol we are going to work are:

- Advanced Encryption Standard (AES)
- AES-CCMP
- AES-GCMP

To encrypt and decrypt data, all of the above protocols require cryptographic methods, which uses mathematical ideas and a set of rule-based calculations known as algorithms to change communications in ways that make them difficult to decipher.In the next sections of this paper, all technical concepts will be thoroughly explained.

Cryptography

The study of secure communications systems that allow only the sender and intended recipient of a message to read its contents is known as cryptography.This is the process of scrambling plain text into ciphertext and then back again when it arrives. In addition, cryptography includes techniques such as steganography and merging to obfuscate information in photographs.

The symmetric or "secret key" scheme is the most basic way. Data is encrypted with a secret key, and the encoded message and secret key are then delivered to the receiver for decoding.

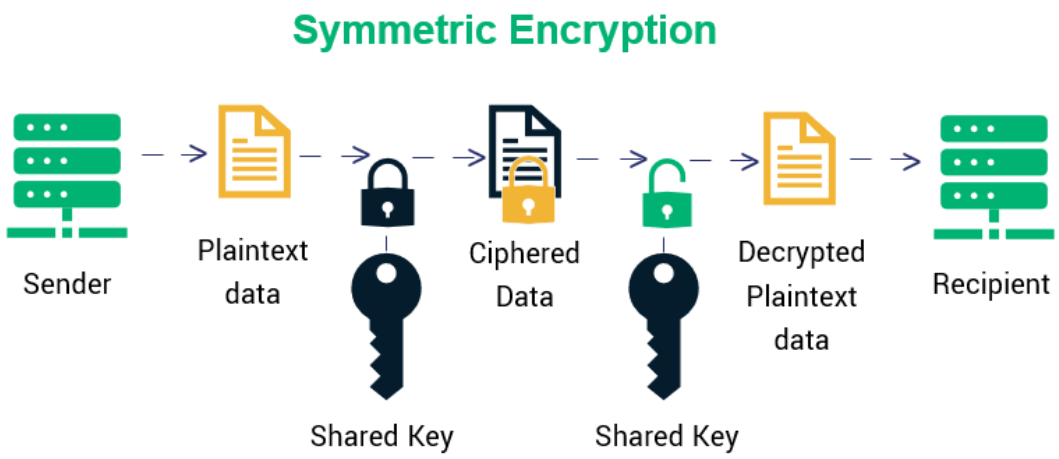

Figure 1: symmetric Key Cryptography

However, because this is insecure, cryptologists created the asymmetric or "public key" scheme. Every user has two keys in this case: one public and one private. Senders encrypt the message and transmit it along after requesting the recipient's public key. Only the recipient's private key can decode the message when it arrives, therefore theft is useless without the associated private key.

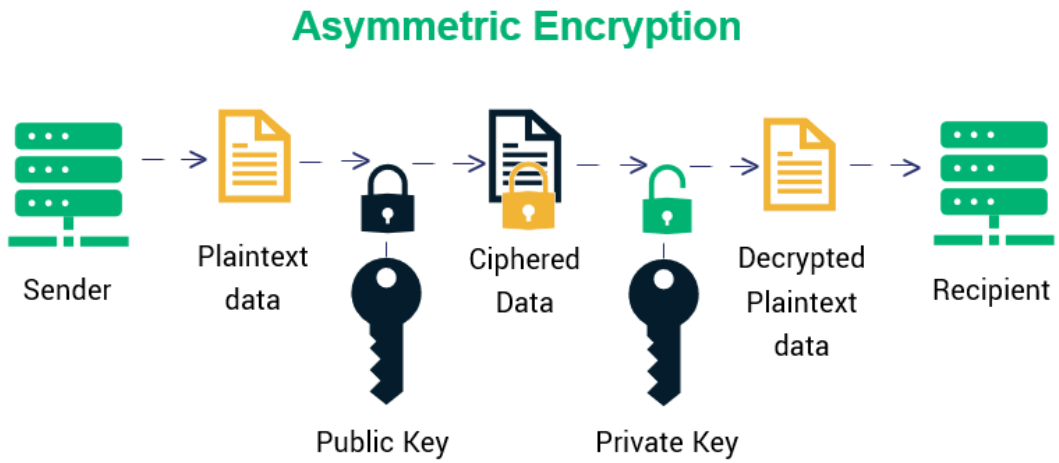

Figure 2: Asymmetric Key Cryptography

Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

It accepts 128 bits as input and produces encrypted ciphertext in 128 bits. AES is based on the substitution-permutation network principle, which entails replacing and shuffling the input data through a series of connected processes.

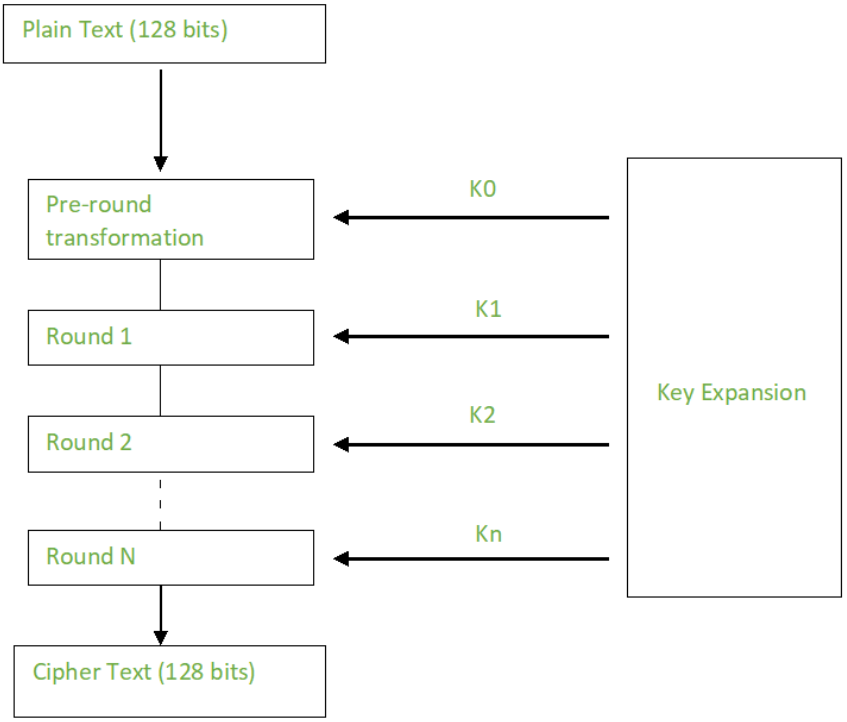

Figure 3: Basic AES Block Diagram

AES - CCMP

The Counter Mode Cipher Block Chaining Message Authentication Code Protocol (Counter Mode CBC-MAC Protocol) or CCM mode Protocol (CCMP) is a wireless LAN encryption protocol that implements the IEEE 802.11i amendment to the original IEEE 802.11 standard.

CCMP is an advanced data cryptographic encapsulation technique for data secrecy that is based on the Advanced Encryption Standard (AES) Counter Mode with CBC-MAC (CCM mode).

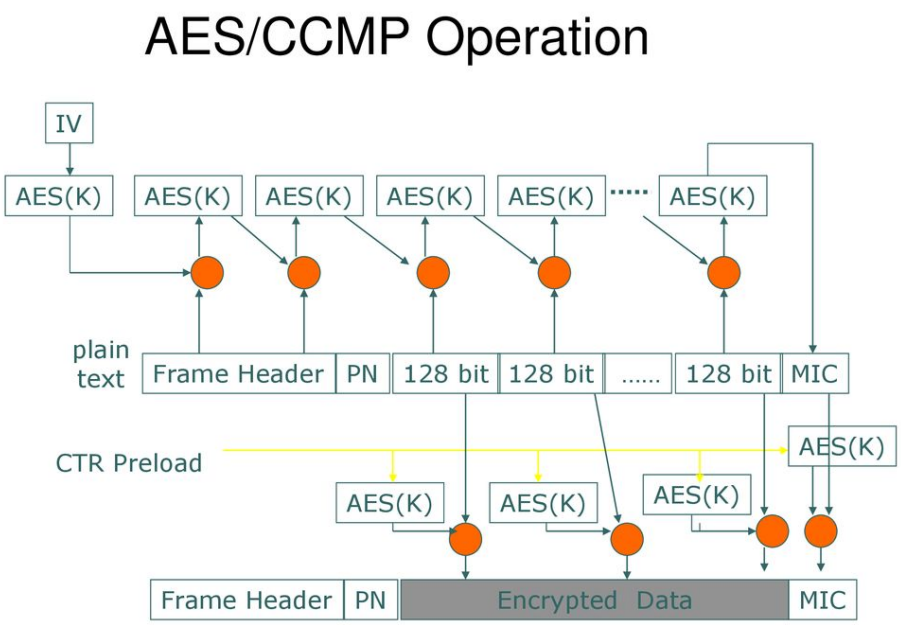

Figure 4: AES-CCMP Block Diagram

AES - GCMP

The Galois/Counter Mode (GCM) is a performance-oriented mode of operation for symmetric-key cryptographic block cipher. With modest hardware resources, GCM throughput rates for state-of-the-art, high-speed communication channels may be reached. The operation is a data integrity (integrity) and confidentiality (confidentiality) authenticated encryption technique. GCM is a block cipher with a block size of 128 bits that is defined. The Galois Message Authentication Code (GMAC) is an authentication-only variation of the Galois Message Authentication Code (GCM) that may be used to generate incremental message authentication codes. Initialization vectors of any length can be accepted by both GCM and GMAC.

Even when using the same block cipher, distinct block cipher modes of operation might have drastically different performance and efficiency characteristics. GCM can make effective use of an instruction pipeline or a hardware pipeline, and it can take full use of parallel processing. The cipher block chaining (CBC) style of operation, on the other hand, causes pipeline delays that reduce efficiency and performance.

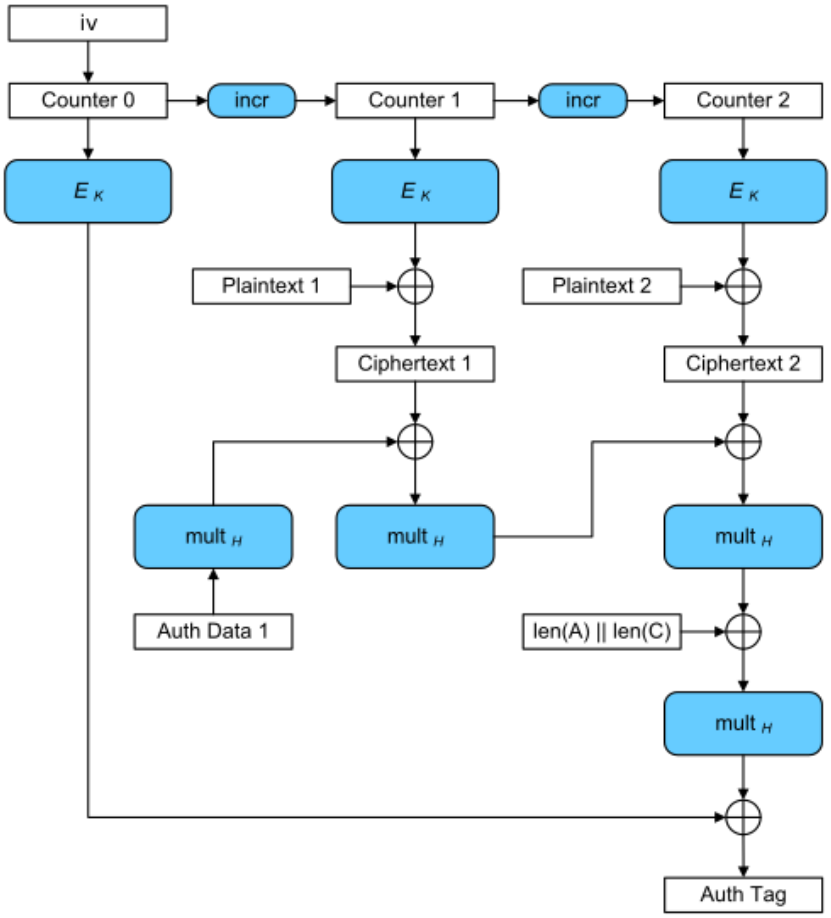

Figure 5: AES-GCMP Block Diagram

Introduction to AES

The Advanced Encryption Standard (AES), sometimes referred to as Rijndael, is an electronic data encryption standard created by the National Institute of Standards and Technology (NIST) of the United States in 2001.

Two Belgian cryptographers, Joan Daemen and Vincent Rijmen, developed the AES, a Rijndael block cipher version, and submitted a proposal to NIST as part of the AES selection process. A cipher family with a variety of key and block sizes is Rijndael.

It accepts 128 bits as input and generates 128 bits of encrypted cipher text as output. AES uses a series of connected operations that include replacing and rearranging the input data in order to function. This is known as the substitution-permutation network principle. AES does actions on data in bytes rather than bits. Because of the 128 bit block size, the cipher only analyses 128 bits (or 16 bytes) of the incoming data at once.

Features of AES:

- AES follows Block cipher.
- Key sizes range from 128 to 256 bits.
- And data is encrypted in chunks of 128 bits.

As opposed to DES, which only supports block and key sizes of 64 and 56 bits, AES supports a wide range of block and key sizes. However, Three members of the Rijndael family were selected by NIST for use with AES; each had a block size of 128 bits but three different key lengths: 128, 192, and 256 bits. The name of the standard is changed to AES-128, AES-192, or AES256 depending on which version is being utilised.

No. of Rounds based on key size:

1. 128 bit key = 10 rounds.
2. 192 bit key = 12 rounds.
3. 256 bit key = 14 rounds.

Working of AES

To generate ciphertext, the AES algorithm employs a substitution permutation (SP) network with many rounds. The number of rounds is determined on the key size. Each of these rounds requires a round key, however because the method only accepts one key, this key must be expanded to receive keys for each round.

There are a few more blocks and techniques we need to comprehend before delving into the specifics of key expansion.

1. SubBytes Transformation:

The SubBytes() transformation is a non-linear byte substitution that utilises a substitution table to work independently on each byte of the State (S-box). By combining two transformations, this invertible S-box is created.

It uses a straightforward 16 by 16 byte matrix, or "s-box," for a basic table lookup. This matrix includes all of the potential configurations for an 8-bit sequence. There is a clear process for producing the s-box tables, so the s-box is not simply a random permutation of these values. In contrast to the s-boxes in DES, where no explanation was provided, the inventors of Rijndael demonstrated how this was accomplished.

Each byte in a round is converted into a new byte in the following manner: the leftmost nibble of the byte is used to define a specific row of the s-box, and the rightmost nibble identifies a column. For instance, the byte "A5" chooses row 11 column 5, which turns out to have the value "2A". The state matrix is subsequently updated using this.

2. InvSubBytes Transformation:

An inverse s-box is used in the Inverse replacement byte transformation (known as InvSubBytes). In this situation, the intended result is to choose the value '2A' and receive the value 'A5'.

The s-box is built to withstand known crypt analytic attacks. The Rijndael developers specifically wanted a design with a low correlation between input and output bits, as well as the feature that the output cannot be represented as a simple mathematical function of the input.

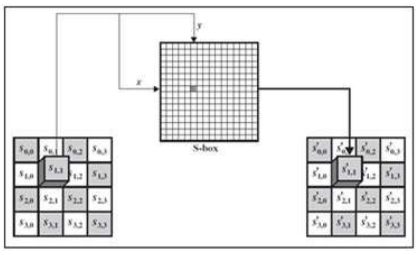

Figure 6: SubBytes/InvSubBytes

3. ShiftRows Transformation:

The ShiftRows transformation cyclically shifts the bytes in each row of the State across different numbers of bytes dependent on offsets. Each row is rotated a certain number of times as follow:

1. The first row is not shifted (offset = 0)
2. The second row is shifted by 1 to the left (offset = 1).
3. The third row is shifted by 2 to the left (offset = 2).
4. The fourth row is shifted by 3 to the left (offset = 3).

4. InvShiftRows Transformation:

The Inverse Shift Rows transformation (also known as InvShiftRows) reverses these circular shifts for rows with the same offset.

This action may not appear to do much, but when you consider how the bytes are organised inside state, it becomes clear that it has significantly more of an influence.

Remember that state is handled as a four-byte array. A one-byte shift corresponds to a four-byte linear distance. The transformation also guarantees that the four bytes of one column are distributed throughout four columns.

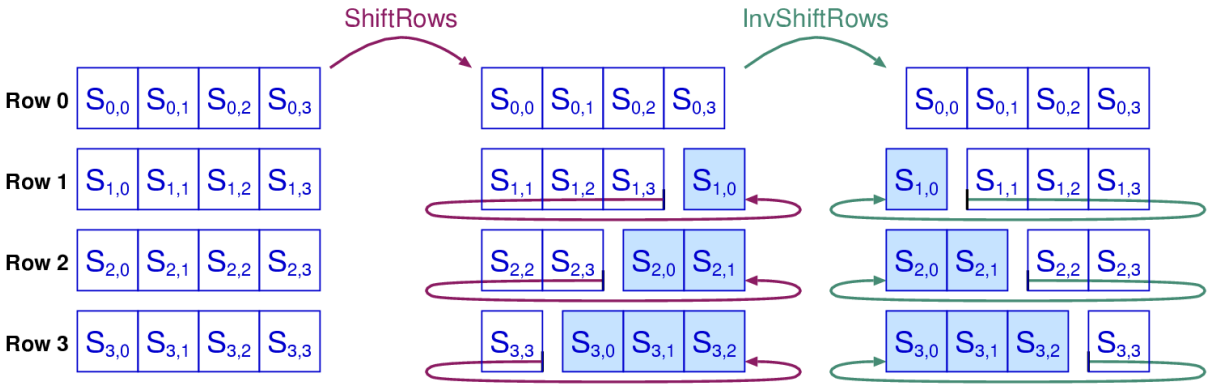

Figure 7: ShiftRows

5. MixColumns Transformation:

MixColumns is just a column substitution performed individually, except it employs Galois field arithmetic. Each byte in a column is mapped to a new value that is a function of the column's four bytes. The following matrix multiplication on state determines the transformation:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} S_{(0,0)} & S_{(0,1)} & S_{(0,2)} & S_{(0,3)} \\ S_{(1,0)} & S_{(1,1)} & S_{(1,2)} & S_{(1,3)} \\ S_{(2,0)} & S_{(2,1)} & S_{(2,2)} & S_{(2,3)} \\ S_{(3,0)} & S_{(3,1)} & S_{(3,2)} & S_{(3,3)} \end{bmatrix} = \begin{bmatrix} S'_{(0,0)} & S'_{(0,1)} & S'_{(0,2)} & S'_{(0,3)} \\ S'_{(1,0)} & S'_{(1,1)} & S'_{(1,2)} & S'_{(1,3)} \\ S'_{(2,0)} & S'_{(2,1)} & S'_{(2,2)} & S'_{(2,3)} \\ S'_{(3,0)} & S'_{(3,1)} & S'_{(3,2)} & S'_{(3,3)} \end{bmatrix} \quad (1)$$

$$S'_{(0,j)} = (2 * S_{(0,j)} \oplus (3 * S_{(1,j)}) \oplus S_{(2,j)} \oplus S_{(3,j)} \quad (2)$$

$$S'_{(1,j)} = S_{(0,j)} \oplus (2 * S_{(1,j)}) \oplus (3 * S_{(2,j)}) \oplus S_{(3,j)} \quad (3)$$

$$S'_{(2,j)} = S_{(0,j)} \oplus S_{(1,j)} \oplus (2 * S_{(2,j)}) \oplus (3 * S'_{(3,j)}) \quad (4)$$

$$S'_{(3,j)} = (3 * S_{(0,j)}) \oplus S_{(1,j)} \oplus S_{(2,j)} \oplus (2 * S'_{(3,j)}) \quad (5)$$

Multiplication by other numbers can be considered as a repeating use of this procedure. What is crucial to observe is that a multiplication operation has been reduced to a shift and an XOR operation. This is one of the reasons why AES is a relatively efficient algorithm to implement.

6. InvMixColumns Transformation:

The inverse of the MixColumns transformation is InvMixColumns. InvMixColumns performs column-by-column operations on the State, considering each column as a four term polynomial. Columns are treated as polynomials over Galois field and multiplied by a fixed polynomial. As a result of this multiplication, the four bytes in a column are replaced with the following:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \cdot \begin{bmatrix} S_{(0,0)} & S_{(0,1)} & S_{(0,2)} & S_{(0,3)} \\ S_{(1,0)} & S_{(1,1)} & S_{(1,2)} & S_{(1,3)} \\ S_{(2,0)} & S_{(2,1)} & S_{(2,2)} & S_{(2,3)} \\ S_{(3,0)} & S_{(3,1)} & S_{(3,2)} & S_{(3,3)} \end{bmatrix} = \begin{bmatrix} S'_{(0,0)} & S'_{(0,1)} & S'_{(0,2)} & S'_{(0,3)} \\ S'_{(1,0)} & S'_{(1,1)} & S'_{(1,2)} & S'_{(1,3)} \\ S'_{(2,0)} & S'_{(2,1)} & S'_{(2,2)} & S'_{(2,3)} \\ S'_{(3,0)} & S'_{(3,1)} & S'_{(3,2)} & S'_{(3,3)} \end{bmatrix} \quad (6)$$

$$S'_{(0,j)} = (0E * S_{(0,j)} \oplus (0B * S_{(1,j)}) \oplus (0D * S_{(2,j)}) \oplus (09 * S_{(3,j)}) \quad (7)$$

$$S'_{(1,j)} = (09 * S_{(0,j)}) \oplus (0E * S_{(1,j)}) \oplus (0B * S_{(2,j)}) \oplus (0D * S_{(3,j)}) \quad (8)$$

$$S'_{(2,j)} = (0D * S_{(0,j)}) \oplus (09 * S_{(1,j)}) \oplus (0E * S_{(2,j)}) \oplus (0B * S'_{(3,j)}) \quad (9)$$

$$S'_{(3,j)} = (OB * S_{(0,j)}) \oplus ()D * S_{(1,j)}) \oplus (09 * S_{(2,j)}) \oplus (0E * S'_{(3,j)}) \quad (10)$$

7. AddRoundKey Transformation:

In AddRoundKey, the 128 bits of state are bit wise XORed with the 128 bits of the round key. The operation is considered as a column-wise action between the four bytes of a state column and one word of the round key. This transition is as basic as possible, which improves efficiency, but it affects every state.

The most important stage in the AES algorithm is AddRoundKey. AddRoundKey can give significantly greater security while encrypting data. This procedure is based on establishing a link between the key and the cipher text. The encrypted text originates from the preceding stage. The AddRoundKey result is entirely dependent on the key that users specify.

Using Rijndael's key scheduling, the main key is utilised to generate the round keys for each round. The round key is created by bit wise XORing each byte of the state with the corresponding byte of the round key, This will be addressed in further depth in the key expansion.

Key Expansion:

In key Expansion, the AES algorithm takes the Cipher Key and runs a procedure to produce round keys in terms of words. For AES-128, The Key Expansion takes four words as input and outputs 44 words for 11 rounds, each round consists of four words. Each word is 32 bytes long, hence each round key is 128 bits long.

The key is duplicated into the first four words of the extended key. The remainder of the extended key is filled in four words at a time. Each additional word w[i] is dependent on the word immediately preceding it, $w_{[i1]}$, and the word four positions back, $w_{[i4]}$. In three out of four examples, a basic XOR is used.

A more sophisticated function is employed for words whose position in the w array is a multiple of 4. Generation of the first eight words of the expanded key with the symbol g to denote that complicated function. The function g is made up of the following sub functions:

1. RotWord alters a word by performing a one-byte circular left shift.
2. SubWord uses the s-box to do byte substitutions on each byte of its input word.
3. Steps 1 and 2 are XORed with the round constant, $Rcon_{[j]}$.

The round constant is a word with three rightmost bytes that are always zero. Thus, XORing a word with Rcon has the effect of merely XORing the word's leftmost byte. The round constant varies with each round and is defined below and varies as well as with multiplication defined over the Galois field.

$$Rcon_{[j]} = (RC_{[J]}, 0, 0, 0), where, RC_{[1]} = 1, RC[j] = 2 * RC_{[j-1]}$$

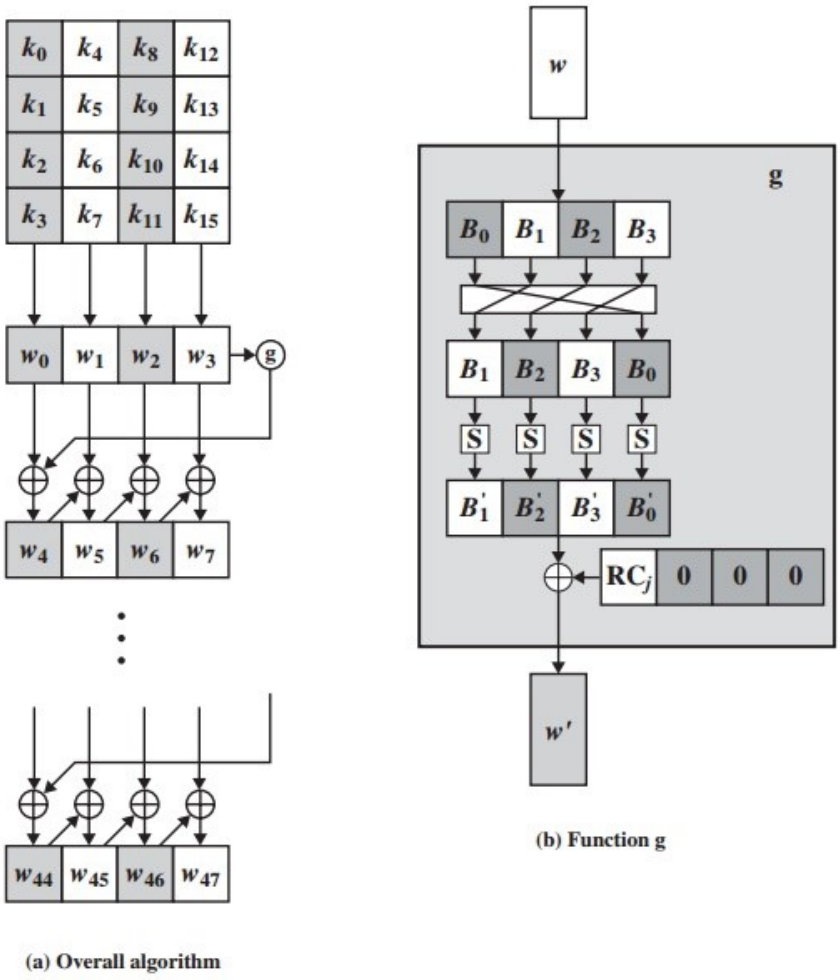

Figure 8: Key Expansion

Encryption

The AES algorithm encrypts data using a specific structure to offer the highest level of security. It uses several rounds to do it, with four sub-processes inside each round. The following four procedures are used in each round to encrypt a 128 bit block.

1. SubBytes
2. ShiftRows
3. MixColumns
4. AddRoundKey

MixColumns is not included in the final round. The substitution and permutation operations of the method are carried out by SubBytes, ShiftRows, and MixColumns, respectively.

Each round's key is obtained from a key expansion block, and a respective key is employed in that particular round. The presence of a round-dependent round constant removes the symmetry, or resemblance, between how round keys are created in various rounds.

Decryption

Decryption is the procedure used to recover the encrypted data's original form. The decryption process is the encryption process done in reverse.

The steps in the rounds are simply undone since they each have an opposite that, when used, reverses the alterations. Depending on the key size, each of the 128 blocks goes through 10, 12, or 14 rounds.

Following are the phases of each round of decryption:

1. AddRoundKey
2. InvMixColumns
3. InvShiftRows
4. InvSubBytes

Both the transmitter and the receiver use the same key to encrypt and decrypt data with an AES encryption algorithm. The decryption procedure is similar to the encryption process in reverse order.

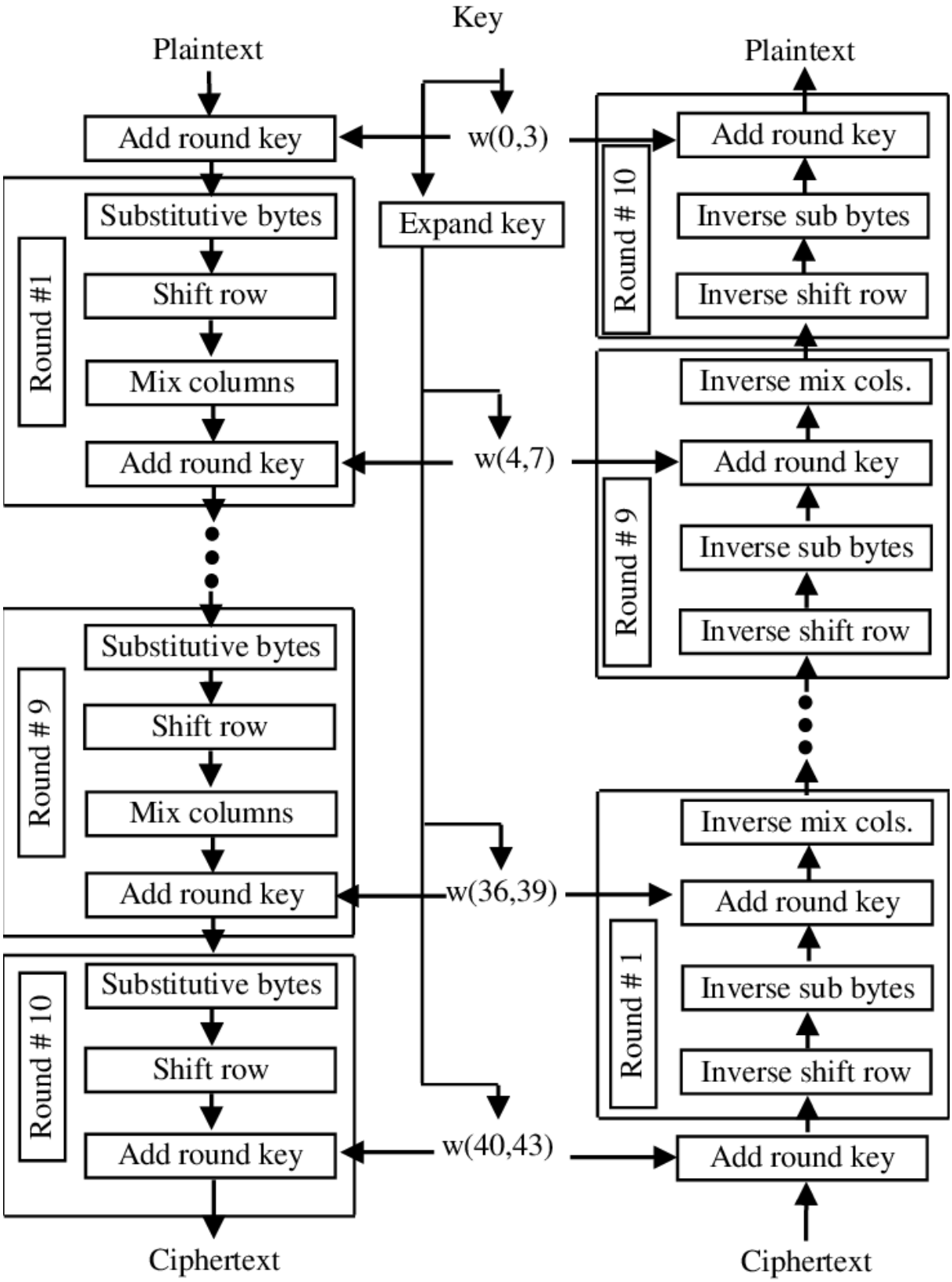

Figure 9: Working of AES

Introduction to CCMP

The Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP) employs the Counter Mode with CBC-MAC (CCM) mode of operation and is based on the Advanced Encryption Standard (AES) algorithm used by the U.S. federal government.

Rivest Cipher 4 is replaced by CCMP in Wired Equivalent Privacy (WEP) and Temporal Key Integrity Protocol (TKIP).CCMP is a component of the 802.11i standard for wireless local area networks (WLANs). It applies modified versions of the original 802.11 standard. The 802.11i task group created this protocol in response to the WLAN industry's expansion and the requirement for stronger encryption standards. To overcome the flaws in the current WEP protocol, CCMP was created.

Sensitive data is encrypted by CCMP using the AES algorithm. To prevent replays and reduce vulnerability to replay attacks, it uses 128-bit keys and a 48-bit initialization vector (IV), often referred to as a CCM nonce block.

Counter Mode and CBC-MAC are CCMP's two primary components. While CBC-MAC offers data integrity and authentication, the Counter Mode component ensures data privacy. CCM is a block cipher mode for authorised encryption that may be used with any block-oriented encryption technique.

Characteristics of CCMP:

1. Only 128-bit block ciphers are supported per its definition.
2. Both the CCMP key and the block size are 128 bits.
3. CCM requires a new temporal key for each session since it persists only for the length of a transaction.
4. CCM requires a unique nonce value and a 48-bit packet identifier for each frame protected by a specific temporal key.
5. The reuse of a packet number with the same temporal key renders security assurances null and void.

Working of CCMP

There are five parts in a CCMP MPDU (Medium Access Control Protocol Data Unit). The MAC header, which includes the source and destination addresses of the data packet, is the first.

The second is the 8-octet CCMP header, which contains the key ID, the Ext IV, and the packet number (PN). The 48-bit packet number is spread across 6 octets. The CCMP header's first two and last four octets are the PN codes, which are increased with each succeeding packet. There is a Key ID octet and a reserved octet in between the PN codes. Ext IV bit (bit 5), Key ID bits (bits 6-7), and a reserved sub field bits (0–4) make up the Key ID octet.These numbers are used by CCMP to encrypt the MIC and the data unit. The data unit, or third portion, contains the data being conveyed in the packet. The message integrity code (MIC), which guards the packet's validity and integrity, comes in at number four. The frame check sequence (FCS), which is utilised for mistake detection and repair, comes in fifth. Only the data unit and MIC are encrypted among these components.

CCMP is the standard encryption mechanism for use with the Wi-Fi Protected Access II (WPA2) standard, and it is far more secure than the Wired Equivalent Privacy (WEP) protocol and Wi-Fi Protected Access Temporal Key Integrity Protocol (TKIP) (WPA).

The following security services are provided by CCMP:

1. Data confidentiality guarantees that only authorised parties have access to the data.
2. Authentication validates the user's identity.
3. Layer management in combination with access control

CCMP is safe against attacks to the 264 steps of operation since it is a block cipher mode with a 128-bit key. There are generic meet-in-the-middle attacks that may be used to limit the theoretical strength of the key to 2n/2 (where n is the number of bits in the key) operations.

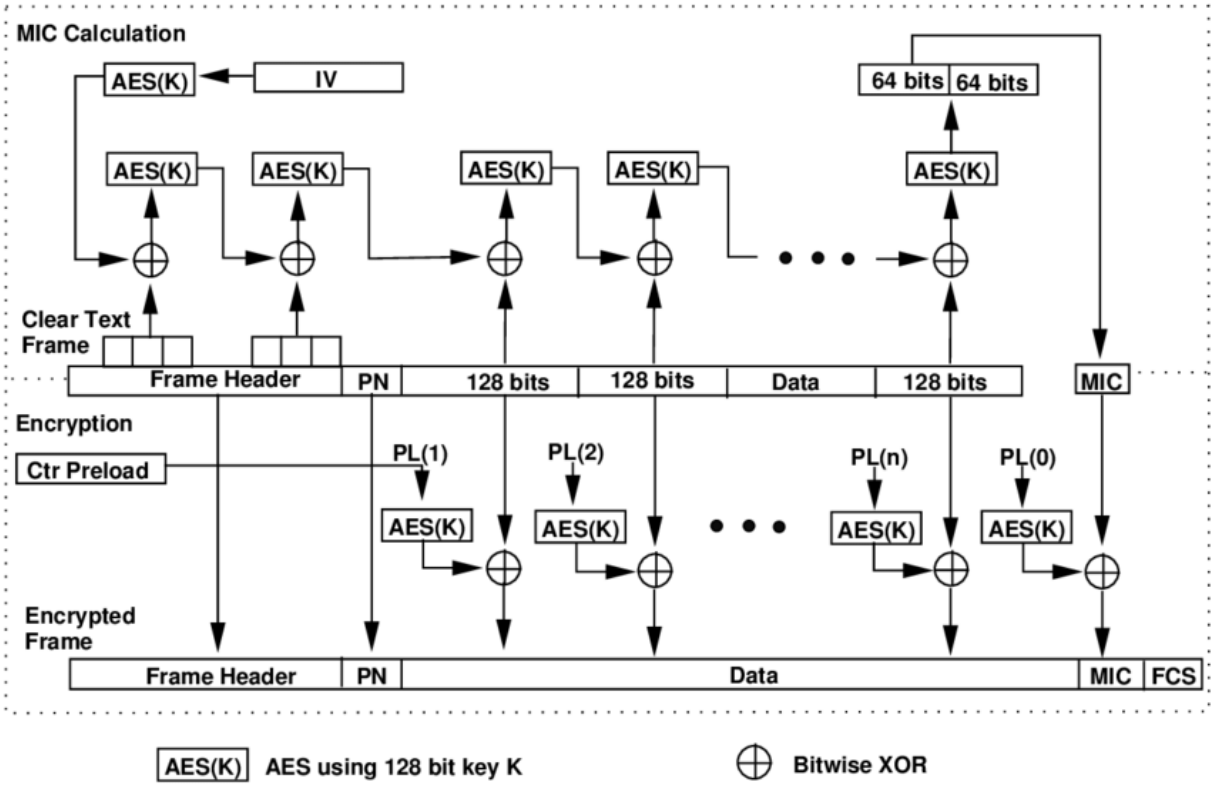

Figure 10: Working of CCMP

Introduction GCMP

Galois/Counter Mode (GCM) is a popular mode of operation for symmetric-key cryptographic block ciphers. GCM throughput rates for cutting-edge, high-speed communication channels may be reached using low-cost technology. The operation is an authenticated encryption procedure meant to offer both data authenticity (integrity) and secrecy. GCM is specified for 128-bit block ciphers. The Galois Message Authentication Code (GMAC) is an authentication-only variation of the GCM that may be used to generate an incremental message authentication code. Both GCM and GMAC support initialization vectors of any length.

Even when using the same block cipher, distinct block cipher modes of operation might have drastically different performance and efficiency characteristics. GCM may fully use parallel processing, and GCM implementation can make good use of an instruction pipeline or a hardware pipeline. The cipher block chaining (CBC) style of operation, on the other hand, suffers from pipeline delays, which reduces efficiency and performance.

Blocks are numbered sequentially, as in standard counter mode, and then this block number is paired with an initialization vector (IV) and encrypted with a block cipher E, often AES. The ciphertext is created by XORing the output of this encryption with the plain text. Because this is really a stream cipher, like with all counter modes, a separate IV must be used for each stream that is encrypted.

The ciphertext blocks are treated as coefficients of a polynomial, which is then evaluated using finite field arithmetic at a key-dependent point H. The result is then encrypted, yielding an authentication tag that may be used to validate the data's integrity. The IV, ciphertext, and authentication tag are then included in the encrypted text.

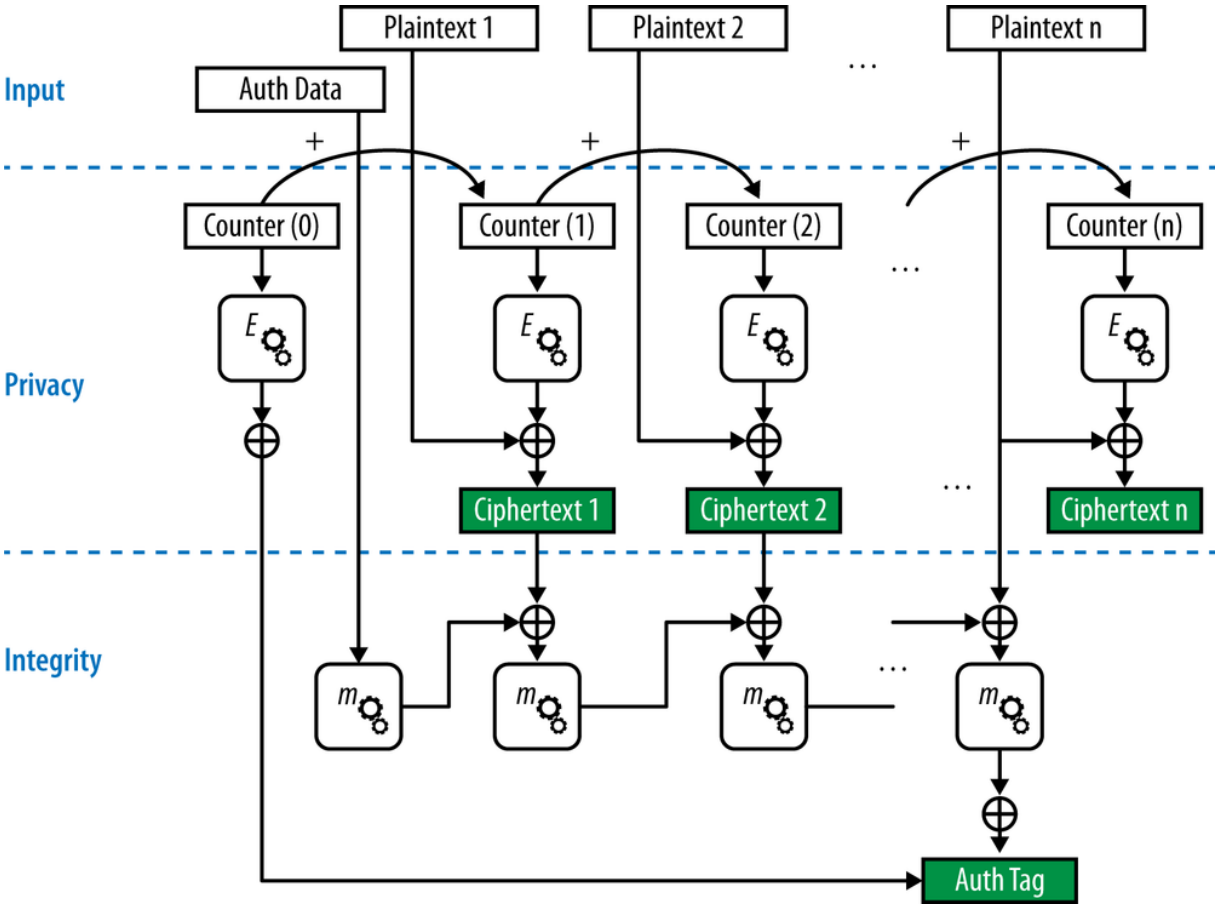

Figure 11: working of GCMP

Results of Simulation:

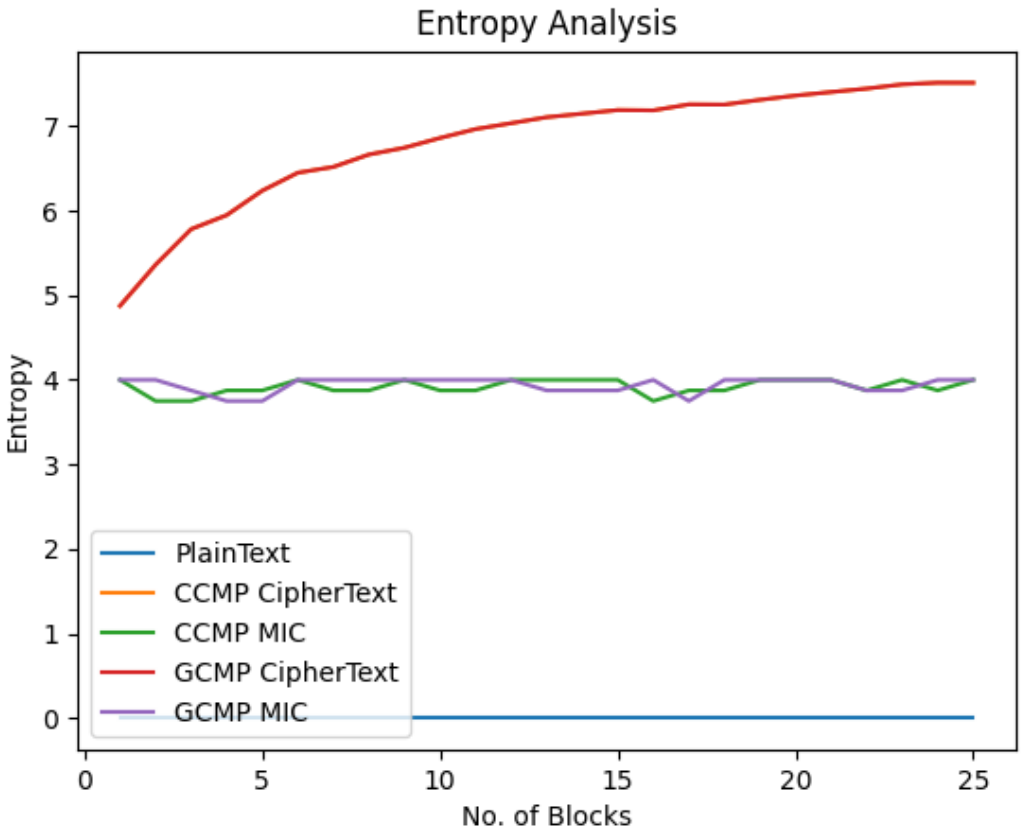

Figure 12: Entropy Analysis

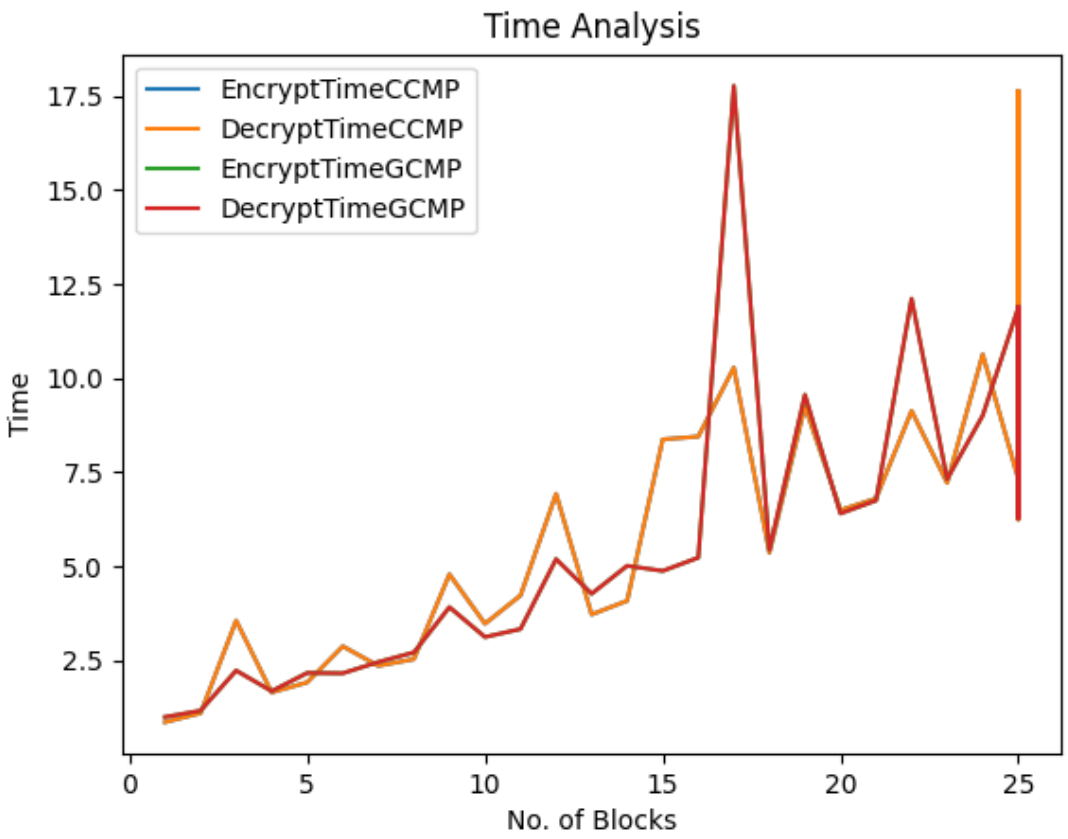

Figure 13: Time Analysis

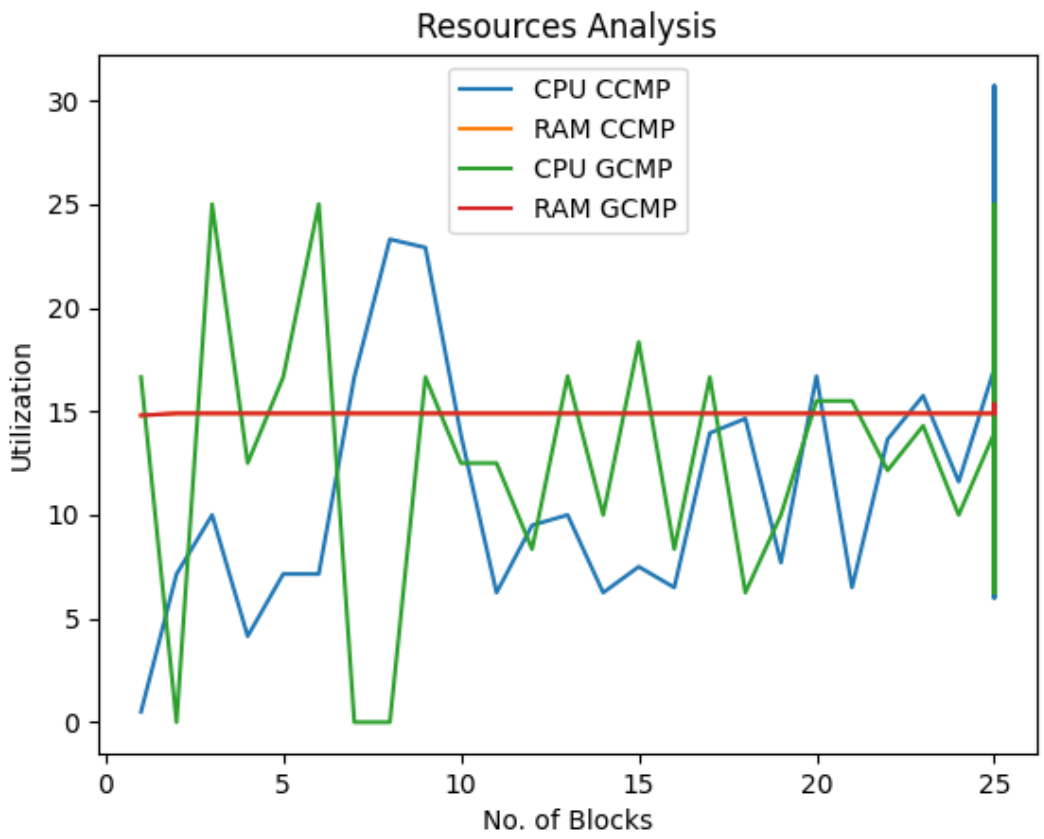

Figure 14: Resources Analysis