[Final Year Project]

[The document contains the introduction of Cryptography and algorithm of AES encryption and Decryption along with AEGIS , JUMBO and COPAv2.]

Group Members and Topic

Srilopa Ghosh - AEGIS

Sandip Pal - JUMBO

Md Adil Reza - COPAv2.

12-03-21

What is Cryptography?

⇨ Cryptography is associated with the process of converting ordinary plain text into unintelligible text and vice-versa. It is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. Cryptography not only protects data from theft or alteration, but can also be used for user authentication.

- Earlier cryptography was effectively synonymous with encryption but nowadays cryptography is mainly based on mathematical theory and computer science practice.

Security Service Of Cryptography:-

⇨ *Modern cryptography concerns with:*

Confidentiality:- The Protection of the data from unauthorized disclosure.

Integrity:- The assurance that the data is received exactly sent by an authorized entity.

Authentication:- Sender and receiver can confirm each.

Non-repudiation:- Sender cannot deny his/her intentions in the transmission of the information at a later stage.

Types Of Cryptographic Technique:-

⇨ There are three type of *cryptographic techniques* in general

1) **Symmetric-key cryptography**
2) **Public-key cryptography**
3) **Hash functions.**

Symmetric-key Cryptography:- Both the sender and receiver share a single key. The sender uses this key to encrypt plaintext and send the cipher text to the receiver. On the other side the receiver applies the same key to decrypt the message and recover the plain text.

Example: - AES algorithm

Public-Key Cryptography:- This is the most revolutionary concept in the last

last 300-400 years. In Public-Key Cryptography two related keys (public and private key) are used. Public key may be freely distributed, while its paired private key, remains a secret. The public key is used for encryption and for decryption private key is used.

Hash Functions:- No key is used in this algorithm. A fixed-length hash value is computed as per the plain text that makes it impossible for the contents of the plain text to be recovered. Hash functions are also used by many operating systems to encrypt passwords.

Now we mainly talk about Symmetric key Cryptography which is AES ALGORITHM

Advanced Encryption Standard(AES):-

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.

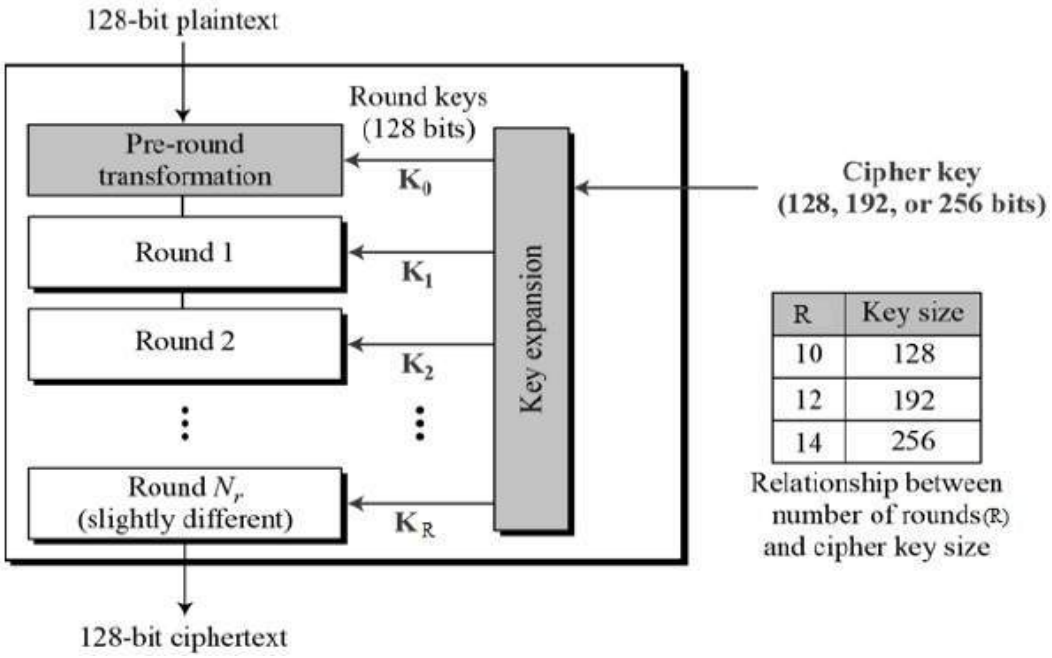
AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix

The features of AES are as follows −

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

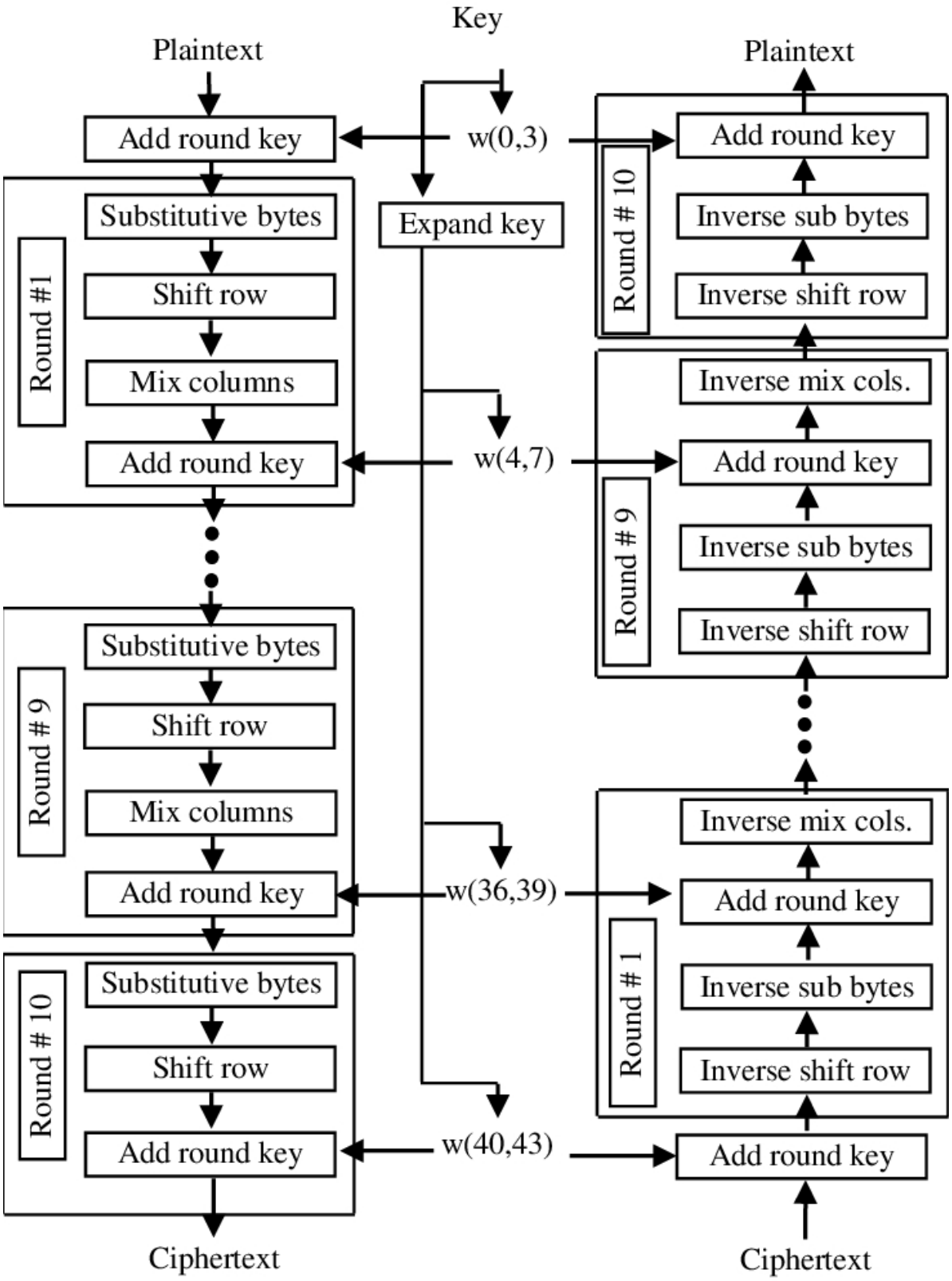
***The schematic of AES structure is given in the following illustration −**



R	Key size
10	128
12	192
14	256

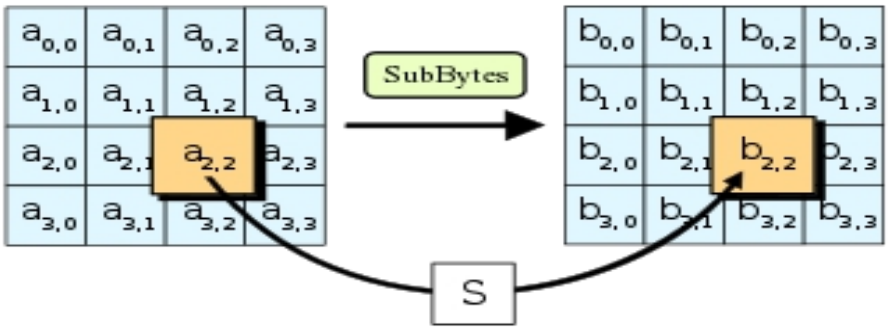
Relationship between number of rounds(R) and cipher key size

AES Encryption Process



Byte Substitution (Sub Bytes)

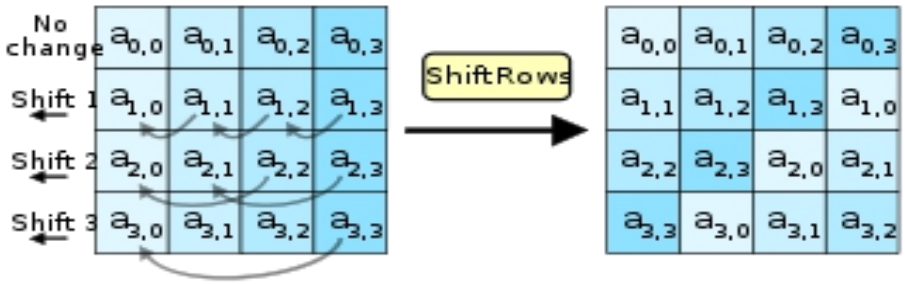
The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.



Shift Rows:-

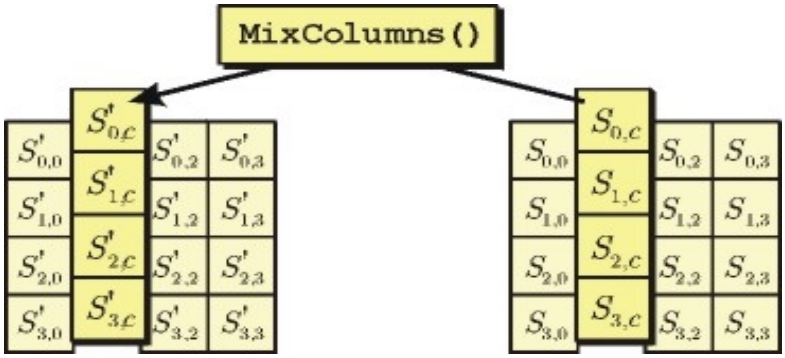
Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows −

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.



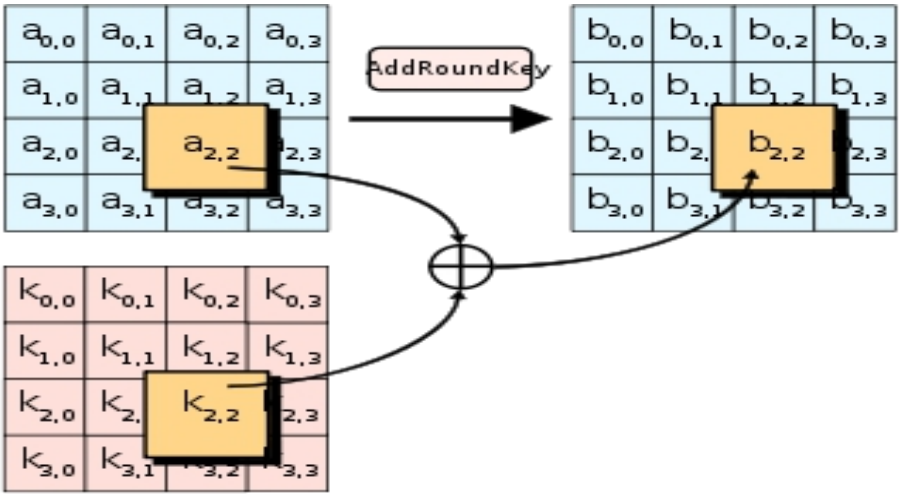
Mix Columns:-

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.



Add Round key:-

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

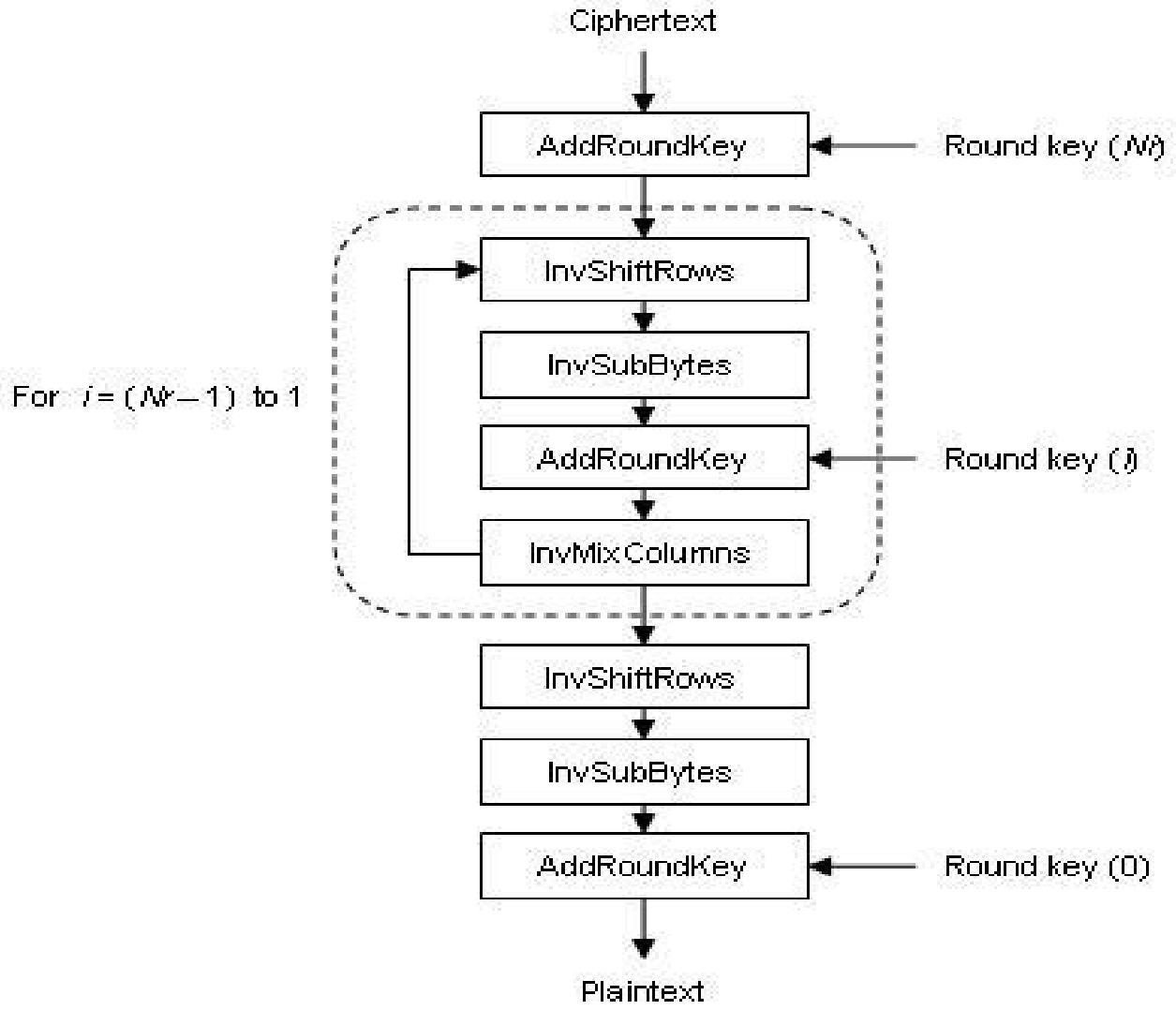


Decryption Process:-

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.



AEGIS: A Fast Authenticated Encryption Algorithm

The protection of a message typically requires the protection of both confidentiality and authenticity. There are two main approaches to authenticate and encrypt a message. One approach is to treat the encryption and authentication separately. The plaintext is encrypted with a block cipher or stream cipher, and a MAC algorithm is used to authenticate the ciphertext.

AEGIS is constructed from the AES encryption round function (not the last round). AEGIS-128L uses eight AES round functions to process a 32-byte message block (one step). AEGIS-128 processes a 16-byte message block with 5 AES round functions, and AEGIS-256 uses 6 AES round functions. The computational cost of AEGIS is about half that of AES. AEGIS is very fast.

The speed of AEGIS-128L is much faster than that of AES in counter (CTR) mode, and are about 8 times that of AES encryption in CBC mode. AEGIS offers a very high security. it is impossible to recover the AEGIS state and key faster than exhaustive key search. AEGIS is suitable for network communication since AEGIS can protects a packet while leaving the packet header (associated data) unencrypted.

AEGIS Algorithm:

*variables and constants are used in AEGIS:

AD : associated data. ADi : a 16-byte associated data block .

adlen : bit length of the associated data with 0 ≤ adlen < 2 64 .

C : ciphertext. Ci : a 16-byte ciphertext block

const : a 32-byte constant in the hexadecimal format; = 00 ‖ 01 ‖ 01 ‖ 02 ‖ 03 ‖ 05 ‖ 08 ‖ 0d ‖ 15 ‖ 22 ‖ 37 ‖ 59 ‖ 90 ‖ e9 ‖ 79 ‖ 62 ‖ db ‖ 3d ‖ 18 ‖ 55 ‖ 6d ‖ c2 ‖ 2f ‖ f1 ‖ 20 ‖ 11 ‖ 31 ‖ 42 ‖ 73 ‖ b5 ‖ 28 ‖ dd. const0 : first 16 bytes of const. const1 : last 16 bytes of const.

IV 128 : 128-bit initialization vector of AEGIS-128.

K128 : 128-bit key of AEGIS-128.

Msglen : bit length of the plaintext/ciphertext with 0 ≤ msglen < 2 64 .

mi : a 16-byte data block. P : plaintext. Pi : a 16-byte plaintext block

Si : state at the beginning of the ith step. Si,j : j-th 16-byte element of the state Si

T : authentication tag. t : bit length of the authentication tag with 64 ≤ t ≤ 128.

u : u = ⌈ adlen /128 ⌉ . v : v = ⌈ msglen/ 128 ⌉ uL : uL = ⌈ adlen /256 ⌉ . vL : vL = ⌈ msglen/ 256 ⌉ .

*Functions in AEGIS

The AES encryption round function (not the last round) is used in AEGIS: 6AESRound(A, B): A is the 16-byte state, B is the 16-byte round key. This function mapping 2 16-byte inputs to a 16-byte output .

The state update function of AEGIS-128

The state update function updates the 80-byte state Si with a 16-byte message block mi .

Si+1 = StateUpdate128(Si , mi) is given as follows:

Si+1,0 = AESRound(Si,4, Si,0 ⊕ mi);

Si+1,1 = AESRound(Si,0, Si,1);

Si+1,2 = AESRound(Si,1, Si,2);

Si+1,3 = AESRound(Si,2, Si,3);

Si+1,4 = AESRound(Si,3, Si,4);

The initialization of AEGIS-128:

1. Load the key and IV into the state as follows:
   S−10,0 = K128 ⊕ IV 128;
   S−10,1 = const1;
   S−10,2 = const0;
   S−10,3 = K128 ⊕ const0;
   S−10,4 = K128 ⊕ const1;

2. For i = −5 to −1, m2i = K128; m2i+1 = K128 ⊕ IV 128;

3. For i = −10 to −1, Si+1 = StateUpdate128(Si , mi) ;

*Processing the authenticated data

After the initialization, the associated data AD is used to update the state.

1. If the last associated data block is not a full block, use 0 bits to pad it to 128 bits, and the padded full block is used to update the state. Note that if adlen = 0, the state will not be updated.

2. For i = 0 to ⌈ adlen /128 ⌉ − 1, we update the state: Si+1 = StateUpdate128(Si , ADi) ;

*The encryption of AEGIS-128

After processing the associated data, at each step of the encryption, a 16-byte plaintext block Pi is used to update the state, and Pi is encrypted to Ci .

1. If the last plaintext block is not a full block, use 0 bits to pad it to 128 bits, and the padded full block is used to update the state. Note that for the last block, only the

original plaintext bits will be encrypted. Note that if msglen = 0, the state will not be updated, and there is no encryption.

2. Let u = ⌈ adlen 128 ⌉ and v = ⌈ msglen 128 ⌉. For i = 0 to v − 1, we perform encryption and update the state:

Ci = Pi ⊕ Su+i,1 ⊕ Su+i,4 ⊕ (Su+i,2&Su+i,3) ;

Su+i+1 = StateUpdate128(Su+i , Pi) ;

*The finalization of AEGIS-128

After encrypting all the plaintext blocks, we generate the authentication tag using seven more steps. The length of the associated data and the length of the message are used to update the state.

1. Let tmp = Su+v,3 ⊕ (adlen ‖ msglen), where adlen and msglen are represented as 64-bit integers.
2. 2. For i = u + v to u + v + 6, we update the state:
   Si+1 = StateUpdate128(Si , tmp) ;
3. 3. We generate the authentication tag from the state Su+v+7 as follows:
   $T' = \bigoplus_{i=0}^{4} Su+v+7,i$;
   The authentication tag T consists of the first t bits of T ′

*The decryption and verification of AEGIS-128

The exact values of key size, IV size, and tag size should be known to the decryption and verification processes. The decryption starts with the initialization and the processing of authenticated data. Then the ciphertext is decrypted as follows:

1. If the last ciphertext block is not a full block, decrypt only the partial ciphertext block. The partial plaintext block is padded with 0 bits, and the padded full plaintext block is used to update the state.

2. For i = 0 to v − 1, we perform decryption and update the state.

   Pi = Ci ⊕ Su+i,1 ⊕ Su+i,4 ⊕ (Su+i,2&Su+i,3) ;
   Su+i+1 = StateUpdate128(Su+i , Pi) ;

The finalization in the decryption process is the same as that in the encryption process. We emphasize that if the verification fails, the ciphertext and the newly generated authentication tag should not be given as output; otherwise, the state of AEGIS-128 is vulnerable to known-plaintext or chosen-ciphertext attacks.

JAMBU LIGHTWEIGHT AUTHENTICATION ENCRYPTION MODE

Authenticated Encryption:- *Authenticated Encryption* (AE) provides the **confidentiality** and **data integrity** simultaneously. However AE scheme also satisfy **Nonrepudiation, Authentication** and **Access Control** properties. An AE scheme is usually more complicated than confidentiality-only or authenticity-only schemes. It is easier to use, because it usually needs only a single key, and is more robust, because there is less freedom for the user to do something wrong.

As a separate feature, an authenticated encryption scheme may authenticate, but not encrypt, a part of its input, which is called ***associated data*.**

- For example, we may wish to encrypt the contents of an Internet packet, but we have to leave its header unencrypted but still bound to the internal data.

✓ *What is JAMBU?*

⇨ JAMBU is a nonce-based **authenticated encryption** operating **mode** proposed by Wu and Huang, that can be instantiated with any block cipher. As JAMBU internally takes the help of AES algorithm (AES-128) that's why it is called as **AES-JAMBU.**

✓ *Why JAMBU is called as lightweight Authenticated Encryption Mode?*

⇨ The main advantage of JAMBU mode is its low memory requirement (Only 3 register in use (3n)), which places it in the group of lightweight Authenticated Encryption Mode. When JAMBU instantiated with 2n Block cipher and without counting the memory needed to store the secret key, JAMBU will require to maintain a 3n bit internal state, where classical authenticated encryption modes like OCB would require a 6n-bit internal state or even more. That's why JAMBU is reasonably fast.

✓ ***Security Claims Of JAMBU:-***

⇨ The Security claims of JAMBU are given in the CAESAR competition submission document. When instantiated with 2n block cipher, JAMBU process plaintext block of n-bit and output an n-bit of Tag (T). When the nonce is not reused, JAMBU is claimed to provide 2*n*-bit security for confidentiality and *n*-bit security for authentication. But when the nonce is misused (means if associated data is repeated) the confidentiality of JAMBU is supposed to be partially compromised.

DESCRIPTION OF JAMBU

Padding:-

⇨ For associated data, a '1' bit is padded followed by the least number of `0' bits to make the length of padded associated data a multiple of n-bit. And for the Plaintext same Padding method is applied.

Initialization:-

⇨ JAMBU uses an n-bit initialization Vector(IV). The Initialization Vector is Public. Each key/IV pair should be used only once to achieve the maximum security of the scheme .Let X and Y each represent n bit states which result in a state of 2n bit. The initial state(S_{-1}) set as :-

$$S_{-1} = (0^n, IV)$$

The following Operation are used for initialization:-

$$(X_{-1}, Y_{-1}) = E_K(S_{-1});$$
$$R_0 = X_{-1};$$
$$S_0 = (X_{-1}, Y_{-1} \wedge 5).$$

Processing Associated Data:-

⇨ The associated data is divided into n-bit block and Processed Sequentially. For the last block only Padding Scheme is applied only to make it full block. A Padded block of initialization vector $1 \,||\, 0^{n-1}$ will be processed. If N_A be the number of Associated data (AD) blocks after padding. AD processed as:-

For $i = 0$ to $N_A - 1$

$$(X_i, Y_i) = E_K(S_i);$$
$$U_{i+1} = X_i \wedge A_i;$$
$$V_{i+1} = Y_i \wedge R_i \wedge 1;$$
$$S_{i+1} = (U_{i+1}, V_{i+1});$$
$$R_{i+1} = R_i \wedge U_{i+1}:$$

Programming Interface:-

❖ **Encryption:-** JAMBU uses a *k*-bit secret key *K* and an *n*-bit public nonce value *IV* to authenticate a variable length associated data *AD* and to encrypt and authenticate a variable length plaintext *P.* It produces a ciphertext *C*, which has the same bit length with plaintext, and an *n*-bit tag *T*. let N_P the number of Plain text blocks after Padding.

Cipher(C) = JAMBU_ENCRYPTION(P, AD, IV, Key);

Algorithm described as follows :-

For $i = N_A$ to $N_A + N_P - 1$

$$(X_i, Y_i) = E_K(S_i);$$
$$U_{i+1} = X_i \wedge P_{i-NA};$$

$V_{i+1} = Y_i \wedge R_i$;
$S_{i+1} = (U_{i+1}, V_{i+1})$;
$R_{i+1} = R_i \wedge U_{i+1}$;
$C_{i-NA} = (P_{i-NA} \wedge V_{i+1})$ if $i < N_A + N_P - 1$ or the last plaintext block is a partial block;

- **Tag Generation:-** After all padded plaintext blocks are processed, The state will be S_{N+1} and R_{N+1} ($N = N_A + N_P - 1$). We use following steps to generate authentication tag-

$(X_{N+1}, Y_{N+1}) = E_K(S_{N+1})$;
$U_{N+2} = X_{N+1}$;
$V_{N+2} = Y_{N+1} \wedge R_{N+1} \wedge 3$;
$R_{N+2} = R_{N+1} \wedge X_{N+1}$;
$S_{N+2} = (X_{N+2}, Y_{N+2})$;

Authentication tag is generated as $T = R_{N+2} \wedge X_{N+2} \wedge Y_{N+2}$.

- **Decryption and Verification:-** The decryption and verification are similar to the encryption and authentication, except that the ciphertext block is XORed with the sub-state V to compute the plaintext block. For the final block, the ciphertext is padded using the same scheme as the plaintext before the XOR operation.

Plaintext = Decryption (C , Key, R_{N+2});

A tag T' is generated after the decryption and is compared to the tag T. If the two tags match, the plaintext is outputted.

.

AES-COPA

- **What is AES-COPA?**

AES-COPA is an Authenticated Encryption mode which has two parameters: the key length **κ** and the tag length **τ** . The key length can be either 16 bytes (128 bits), 24 bytes (192 bits), or 32 bytes (256 bits). The tag length is between 8 bytes (64 bits) and 16 bytes (128 bits).

The **nonce**, also called the public message number, is an input of length 16 bytes (128 bits).

AES-COPA does not support a secret message number. Each key size of AES-COPA corresponds to a key size of AES. AES-COPA supports variable length associated data and plaintexts. To comply with our security claims from Sect. 2, the length of the associated data together with the plaintext data is at most ≈ 2 64 · 16 bytes. Recommended parameter set: 16 byte (128 bits) key length and 16 byte (128 bits) tag length

AES-COPA ALGORITHM -

Variables :-

A block cipher $E : K \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a function that takes as input a key $k \in K$ and a plaintext $M \in \{0, 1\}^n$, and produces a ciphertext $C = E(k, M)$.

If X is a string with length a multiple of n, by $X[i]$ we denote the ith n-bit block of X. The length of a string X is denoted by $|X|$. By $b\ Xc\ j$ we denote the j most signifificant bits of X.

$E : \{0, 1\}^\kappa \times \{0, 1\}^n \times \{0, 1\}^* \times \{0, 1\}^+ \rightarrow \{0, 1\}^+ \times \{0, 1\}^\tau$ - *AES-COPA Encryption function.*

$N \in \{0, 1\}^n$ - public message number

$A \in \{0, 1\}^*$ - associated data

$M \in \{0, 1\}$ - *a* message

$C \in \{0, 1\}$ - ciphertext

The function E takes as input a public message number $N \in \{0, 1\}^n$, associated data $A \in \{0, 1\}^*$, and a message $M \in \{0, 1\}^+$. It returns a ciphertext $C \in \{0, 1\}^+$, where $|C| = |M|$, and tag $T \in \{0, 1\}^\tau$: $(C, T) \leftarrow E(N, A, M)$.

Padding -

If the last bit of the message M is incomplete i.e is not equal to 128 bit then we apply padding by adding a series of 10* after the message to make the length of M equal to 128 bit

$D : \{0,1\}^{\kappa} \times \{0,1\}^{n} \times \{0,1\}^{*} \times \{0,1\}^{+} \times \{0,1\}^{\tau} \times \{0,1\} \rightarrow \{0,1\}^{+} \cup \{\perp\}$- The decryption function D takes as input a public message number $N \in \{0,1\}^{n}$, associated data $A \in \{0,1\}^{*}$, ciphertext $C \in \{0,1\}^{+}$, tag $T \in \{0,1\}^{\tau}$, and a bit P indicating whether the last block of M was

Incomplete The algorithm D outputs $M \in \{0,1\}^{+}$ if the tag is correct .

The functions for the following encryption and decryption are defined as-

AES-COPA - Encrypt:

$P \leftarrow 0$ **if** $|M0[d]| = 128$ **else** 1

$V \leftarrow$ PMAC1'($Ak\ N$)

$(C, S) \leftarrow$ Encrypt(V, M, P)

$\Sigma \leftarrow M[1] \oplus M[2] \oplus \cdots \oplus M[d]$

$T \leftarrow Ek(\ Ek(\ \Sigma \oplus 2^{d-1} 3^{2} 7^{p} L)\ \oplus 2^{d} 7L$

where the subroutines are defined as:

PMAC1' (*X*):

$X[1]X[2] \cdots X[x] \leftarrow X$

$\Delta 0 \leftarrow 33L$, $U \leftarrow 0$

for $i = 1, \ldots, x\ x\ 1$ **do**

$U \leftarrow U \oplus Ek\ (X[i] \oplus \Delta 0)$

$\Delta 0 \leftarrow 2\Delta 0$

end for

if $X[x] = n$ **then**

$V \leftarrow Ek(U \oplus X[x] \oplus 3\Delta 0)$

else

$V \leftarrow Ek(U \oplus X[x]||k\ 10^{*} \oplus 32\Delta 0)$

end if

Output V

Encrypt(*V, M, P*):

$V[0] \leftarrow V \oplus L$, $\Delta 0 \leftarrow 3L$, $\Delta 1 \leftarrow 2L$

for $i = 1, \ldots, d$ **do**

$V[i] \leftarrow Ek(M[i] \oplus \Delta 0) \oplus V[i-1]$

$C[i] \leftarrow Ek(V[i] \oplus \Delta 1$

$\Delta 0 \leftarrow 2\Delta 0$, $\Delta 1 \leftarrow 2\Delta 1$

$\Delta 0 \leftarrow 7\Delta 0$ **if** $i = d - 1$ **and** $P = 1$

end for

$C \leftarrow C[1]C[2] \cdots C[d]$, $S \leftarrow V[d]$

Output (C, S)

AES-COPA-Decrypt:

$V \leftarrow$ PMAC1$0$ $(Ak\ N)$
$(M, S) \leftarrow$ Decrypt(V, C, P)
if P **then ->** B incomplete last block
$M0[d]10* \leftarrow M[d]$
$\Sigma \leftarrow M[1] \oplus M[2] \oplus \ldots\ldots M[\text{d -1}] \oplus M'[d]\ 10*$
else
$M0[d] \leftarrow M[d]$
$\Sigma \leftarrow M[1] \oplus M[2] \oplus \cdots \oplus M[d]$
end if
if $[Ek$ **S(** $\oplus\ Ek\ (\Sigma \oplus 2^{d-1} 3^2 7^p L)) \oplus 2d7L]\ \tau = T$ **then**
Else
Output $\perp$
end if

Decrypt(*V, C, P*):

$V[0] \leftarrow V \oplus L$, $\Delta 0 \leftarrow 3L$, $\Delta 1 \leftarrow 2L$

for $i = 1, \ldots, d$ **do**

$V[i] \leftarrow E_k^{-1}(C[i] \oplus \Delta 1$

$M[i] \leftarrow E_k^{-1}(V[i] \oplus V[i-1] \oplus \Delta 0$

$\Delta 0 \leftarrow 2\Delta 0$, $\Delta 1 \leftarrow 2\Delta 1$

$\Delta 0 \leftarrow 7\Delta 0$ **if I= d-1 and** $P = 1$

end for

$M \leftarrow M[1]M[2] \cdots M[d]$, $S \leftarrow V[d]$

Output (M, S)

Security Claims

we specify the security levels with respect to the recommended key length of 16 bytes, tag length of 16 bytes, nonce length of 16 bytes. For these parameters AES-COPA achieves the following security levels (in log2 of number of AES calls):

	AES-COPA
confidentiality for the plaintext	64
integrity for the plaintext	64
integrity for the associated data	64
integrity for the public message number	64

Advantages of AES- COPA:-

• AES-COPA is designed for efficiency for both short and long messages. Besides the AES key schedule, the overhead for short messages basically only amounts to two AES calls for the tag generation. On Intel's recent Haswell microarchitecture, AES COPA achieves a performance of up to 1.29 cycles/byte (cpb) for longer messages (2048 bytes) and 1.44 cpb for shorter messages (128 bytes).

• Key agility (computational cost under distinct keys): AES-COPA requires one extra AES call every time a new key is used.

• Nonce agility (computational cost under use of nonces): Since nonces are appended to the associated data *A*, changing the nonce requires one extra AES call.

• Ability to efficiently preprocess *A*: Associated data (excluding the nonce) can be preprocessed independently of the message or the value of the nonce.

• Ability to effiffifficiently preprocess plain text: In the same way, the message (or parts thereof) can be preprocessed without seeing *A* in the first layer of AES calls. This also applies if the nonce is used.

Acknowledgement

We would like to Express my Special thanks of gratitude to our Faculty Member “ Mr . Anit Ghosal “ for his able guidance and support in

Completing our project.

We would also like to extend our gratitude to our Principal sir “Mr.Bibek Chakroborty “ for providing us with all the facility required.

DATE :

12-03-2021