IMPLEMENTATION OF AES BASED CIPHER

Lakshay Bansal (2021059)
Sahil Goswami (2021281)
Ekansh (2021044)

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Goals

- **Goal 01**: Implement the two algorithms, Rocca and Snow V. These ciphers are proposed for use in 5G and potentially in 6G technologies.

- **Goal 02**: Compare their performance based on the time taken to encrypt plaintexts of varying sizes: 128 bits, 256 bits, 1024 bits, 8192 bits, and 81,920 bits.

ROCCA

The Rocca algorithm is a lightweight cryptographic cipher designed for high-performance, secure communication in modern technologies, including 5G networks. It focuses on efficiency and robustness, making it suitable for resource-constrained environments such as IoT devices and mobile networks. Its design is optimized for encrypting data with a balance of speed, security, and energy efficiency, addressing the demands of next-generation communication systems.

Background

- **Keys**: K0 -> 128 bit, K1 -> 128 bit
- **Nonce**: 128 bit value to ensure encryption uniqueness
- **Associated Data (AD)**: Used for Authentication
- **Plain Text**: Data to encrypt
- **State S**: Combination of keys, nonce and constants

Methodology

- **Initialization**: Perform 20 AES-based state transformation rounds using keys, a nonce, and predefined constants.
- **Padding**: Align the input to 32-byte blocks.
- **Associated Data**: Authenticate associated data through state updates.
- **Plaintext Encryption**: Encrypt the plaintext using XOR operations, with state updates applied to each block.
- **Tag Generation**: Generate an authentication tag from the final state to ensure data integrity.

Implementation

Operations:

XOR: Facilitates secure data mixing.
AES Transformations: Enhance non-linearity and ensure robust security.
State Updates: Enable continuous evolution of the encryption state.

Key Functions:

- **rocca_encrypt:** Handles the primary encryption and authentication process.
- **rocca_initialization:** Configures the initial state using keys and a nonce.
- **rocca_processAD:** Authenticates any supplementary data.
- **rocca_encryption:** Performs block-by-block encryption of plaintext.
- **rocca_finalization:** Produces the authentication tag to ensure data integrity.

Conclusion

Rocca is much faster than other efficient schemes with 256-bit key length algorithms. Rocca is the first dedicated cryptographic algorithm targeting 6G systems, i.e., 256-bit key length and the speed of more than 100 Gbps.

ROCCA 1

----------Test Case 0----------
K0 : 00000000000000000000000000000000
K1 : 00000000000000000000000000000000
N : 00000000000000000000000000000000
AD :
0000000000000000000000000000000000000000000000000000000000000000
Message Length[in bytes] : 64

Time Elapsed: 0.11505484580993652
Speed: 4.450051594052738 Mbps

ROCCA 2

----------Test Case 1----------
K0 : 00000000000000000000000000000000
K1 : 00000000000000000000000000000000
N : 00000000000000000000000000000000
AD :
0000000000000000000000000000000000000000000000000000000000000000
Message Length[in bytes] : 256

Time Elapsed: 0.12309837341308594
Speed: 16.637100419901145 Mbps

ROCCA 3

----------Test Case 2----------
K0 : 00000000000000000000000000000000
K1 : 00000000000000000000000000000000
N : 00000000000000000000000000000000
AD :
0000000000000000000000000000000000000000000000000000000000000000
Message Length[in bytes] : 1024

Time Elapsed: 0.2166590171813965
Speed: 37.81983747912793 Mbps

ROCCA 4

----------Test Case 3----------
K0 : 00000000000000000000000000000000
K1 : 00000000000000000000000000000000
N : 00000000000000000000000000000000
AD :
0000000000000000000000000000000000000000000000000000000000000000
Message Length[in bytes] : 8192

Time Elapsed: 0.999748706817627
Speed: 65.55247288952484 Mbps

ROCCA 5

----------Test Case 4----------
K0 : 00000000000000000000000000000000
K1 : 00000000000000000000000000000000
N : 00000000000000000000000000000000
AD :
00000000000000000000000000000000000000000000000000000000000000000000000000000000
Message Length[in bytes] : 81920

Time Elapsed: 10.573662281036377
Speed: 61.98041724629065 Mbps

SNOW-V

- The **SNOW-V algorithm** is a lightweight **stream cipher** designed for efficient hardware and software implementation, particularly in constrained environments like IoT devices and embedded systems.
- SNOW-V, a successor to SNOW 3G, is a high-performance stream cipher designed for 5G networks. It offers faster speeds, a larger internal state, and stronger security, exceeding **AES-256** standards. This evolution addresses the increased security demands of 5G while building on SNOW 3G’s legacy in LTE systems.

Background

SNOW-V maintains the core design structure of its predecessors in the SNOW family.

The cipher is built on two primary components:

- **Linear Feedback Shift Register (LFSR):** A circular arrangement featuring 32 shift registers.
- **Finite State Machine (FSM):** Incorporating three 128-bit registers alongside two AES encryption round functions.

The LFSR-A and LFSR-B are generated using distinct polynomials:

- **LFSR-A:**

  g^A (x)=x^16+x^15+x^12+x^4+x^3+x^2+x+1
- **LFSR-B:**

  g^B (x)=x^16+x^15+x^14+x^11+x^8+x^6+x^2+x+1
- Each register has 16 cells, with each cell of size 16 bit and the outputs are circularly shifted.

Methodology

The FSM takes two blocks (T1 and T2) from the LFSR outputs:

- T1: From LFSR-B
- T2: From LFSR-A

FSM produces a 128-bit keystream output using:

- AES encryption rounds
- XOR operations

Initialization Step:

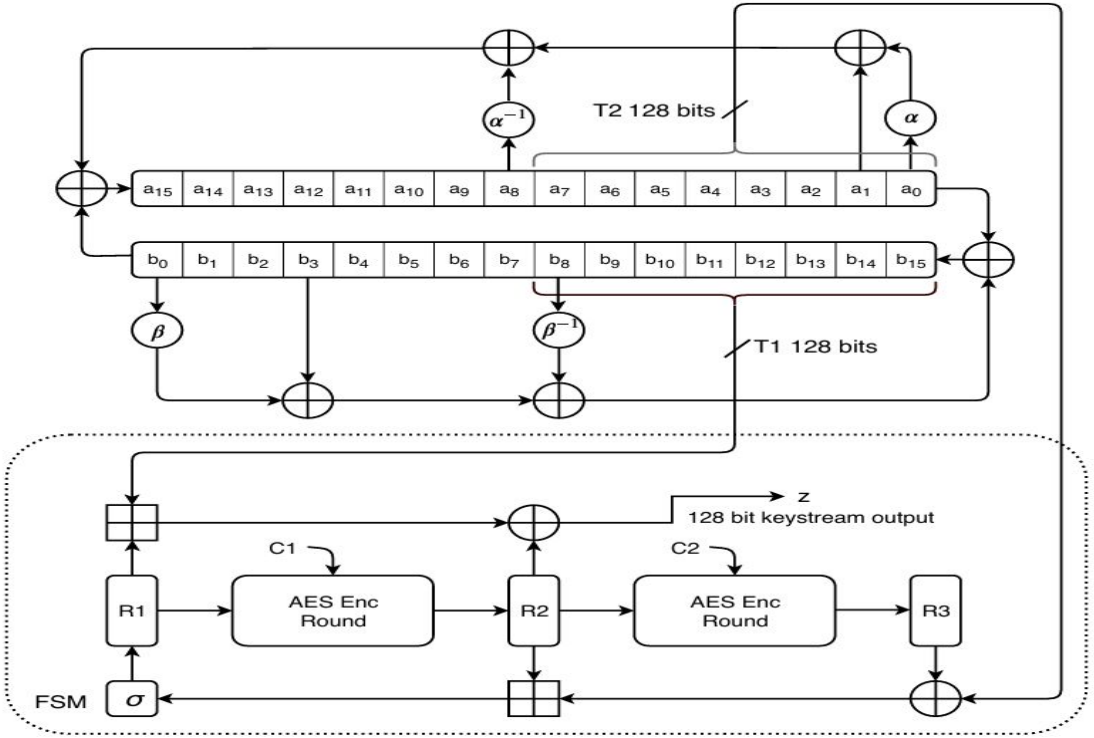
Initialization consists of loading the key and IV into the LFSRs:

- LFSR-A: Key and IV
- LFSR-B: Key only

16 iterations are performed to generate the initial keystream, with two special conditions:

- Iteration 14: R1 XOR with last 128 bits of key
- Iteration 15: R1 XOR with first 128 bits of key

Architecture



SNOW-V 1

----------Test Case 0----------

K : 00000000000000000000000000000000000000000000000000000000000000000000000000000000

IV : 00000000000000000000000000000000

Message Length[in bytes] : 64

Time Elapsed: 0.0144758224487304 69

Speed: 35.369320245075436 Mbps

SNOW-V 2

----------Test Case 1----------

K : 00000000000000000000000000000000000000000000000000000000000000000000000000000000

IV : 00000000000000000000000000000000

Message Length[in bytes] : 256

Time Elapsed: 0.03887677192687988

Speed: 52.679270898620764 Mbps

SNOW-V 3

----------Test Case 2----------
K : 00000000000000000000000000000000000000000000000000000000000000000000000000000000
IV : 00000000000000000000000000000000
Message Length[in bytes] : 1024

Time Elapsed: 0.1554093360900879
Speed: 52.71240587020622 Mbps

SNOW-V 4

----------Test Case 3----------
K : 00000000000000000000000000000000000000000000000000000000000000000000000000000000
IV : 00000000000000000000000000000000
Message Length[in bytes] : 8192

Time Elapsed: 1.1994881629943848
Speed: 54.636637544131226 Mbps

SNOW-V 5

----------Test Case 4----------
K : 00000000000000000000000000000000000000000000000000000000000000000000000000000000
IV : 00000000000000000000000000000000
Message Length[in bytes] : 81920

Time Elapsed: 13.324339866638184
Speed: 49.185175893096726 Mbps

Comparison

Algorithm	Size of input plaintexts (bits)				
	81920	8192	1024	256	64
Rocca(Mbps)	61.98	65.55	37.81	16.63	4.45
Snow-V(Mbps)	49.18	54.6	52.71	52.67	35.37

For smaller input sizes, such as 256 bits, the performance difference between Rocca and Snow V may not be very noticeable. However, as input sizes grow larger, like 1024 bits or 8192 bits, Rocca's simpler structure enables it to achieve better speed performance. Rocca scales more efficiently with larger inputs, whereas Snow V experiences a more significant performance decline due to its complex operations.

Thank You