PROTECTION OF PASSWORDS USING INTEL SGX

TEAM MEMBERS:

- AKSHATHA SHASTRY
- DEEPTHI M K
- INDHU T G
- JAYANTH R
- MOHITH B K

➢ PROBLEM STATEMENT:

Passwords are a fundamental element of digital security, yet traditional methods of storing and handling them are increasingly vulnerable to sophisticated attacks such as memory scraping, phishing, and system compromise. These attacks can lead to unauthorized access and data breaches, as attackers exploit system vulnerabilities to retrieve plaintext passwords or hashes, capture keystrokes, and elevate privileges. The challenge lies in ensuring that passwords are securely stored, processed, and protected from unauthorized access, even if the system is compromised.

One of the most significant issues in password security is the reliance on traditional storage methods, such as plaintext or hashed passwords stored in databases. These methods are highly susceptible to attacks that can compromise the entire system. For instance, memory scraping attacks can capture passwords directly from the system's memory, while phishing attacks trick users into revealing their credentials. System compromise can occur through malware or exploiting software vulnerabilities, giving attackers direct access to stored passwords.

➢ SOLUTION OVERVIEW:

Using Intel Software Guard Extensions (SGX) to create a secure enclave for handling passwords can significantly enhance security. SGX provides a trusted execution environment (TEE) that protects data and code from being accessed or tampered with by unauthorized processes, including those with higher privileges.

1. Password Storage and Retrieval:

•**Secure Storage:** Store hashed passwords inside the SGX enclave using a strong hashing algorithm like scrypt and Argon2

•**Encryption:** Encrypt passwords before they are transmitted or stored outside the enclave.

2. Password Handling Operations:

•**Secure Entry:** Ensure password entry is secure by using techniques like secure input methods or trusted path mechanisms.

•**Validation:** Validate passwords within the SGX enclave to prevent exposure during authentication processes.

3. **<u>Password Management:</u>**

•**Password Generation:** Generate strong, random passwords inside the enclave.

•**Password Update:** Allow users to change their passwords securely, with both the old and new passwords handled within the enclave.

4. <u>Multi-Factor Authentication (MFA):</u>

•**MFA Integration:** Support additional layers of security by integrating multi-factor authentication mechanisms, with sensitive operations performed inside the enclave.

5. <u>Protection Against Attacks:</u>

•**Memory Scraping:** SGX prevents unauthorized processes from reading enclave memory, protecting passwords from memory scraping attacks.

•**Phishing Resistance:** Combine SGX with techniques like secure input and trusted UI to reduce the risk of phishing attacks.

IMPLEMENTATION OUTLINE:

1. <u>Environment Setup</u>:
•Install Intel SGX SDK and PSW on the target system.
•Set up development tools and libraries for SGX application development.

2. <u>Enclave Creation</u>:
•Define the enclave interface using an Enclave Definition Language (EDL) file.
•Implement enclave functions for password storage, retrieval, validation, and management.

3. <u>Secure Communication</u>:
•Establish secure communication channels between the application and the enclave using remote attestation and secure message exchange.

4. <u>Application Integration</u>:
•Modify the existing application to interact with the SGX enclave for password-related operations.
•Ensure that sensitive operations are offloaded to the enclave while maintaining usability and performance.
By leveraging Intel SGX, you can significantly enhance the security of password storage and management in your applications. This approach ensures that passwords are handled within a trusted execution environment, reducing the risk of exposure to various attacks.

➢ FEATURES OFFERED:

Secure Password Storage Using AES-256 Encryption: Advanced Encryption Standard (AES-256):
Utilizes the AES-256 encryption algorithm, which is recognized globally for its security and efficiency. AES-256 uses a 256-bit key to encrypt and decrypt data, providing a high level of security that is currently considered unbreakable by brute force attacks.
Encryption Process: When a user sets or updates their password, the password is immediately encrypted using the AES-256 algorithm. This ensures that the plaintext password is never stored or processed in an unencrypted form, reducing the risk of unauthorized access.
Storage Method: Encrypted passwords are stored in a secure file in the root

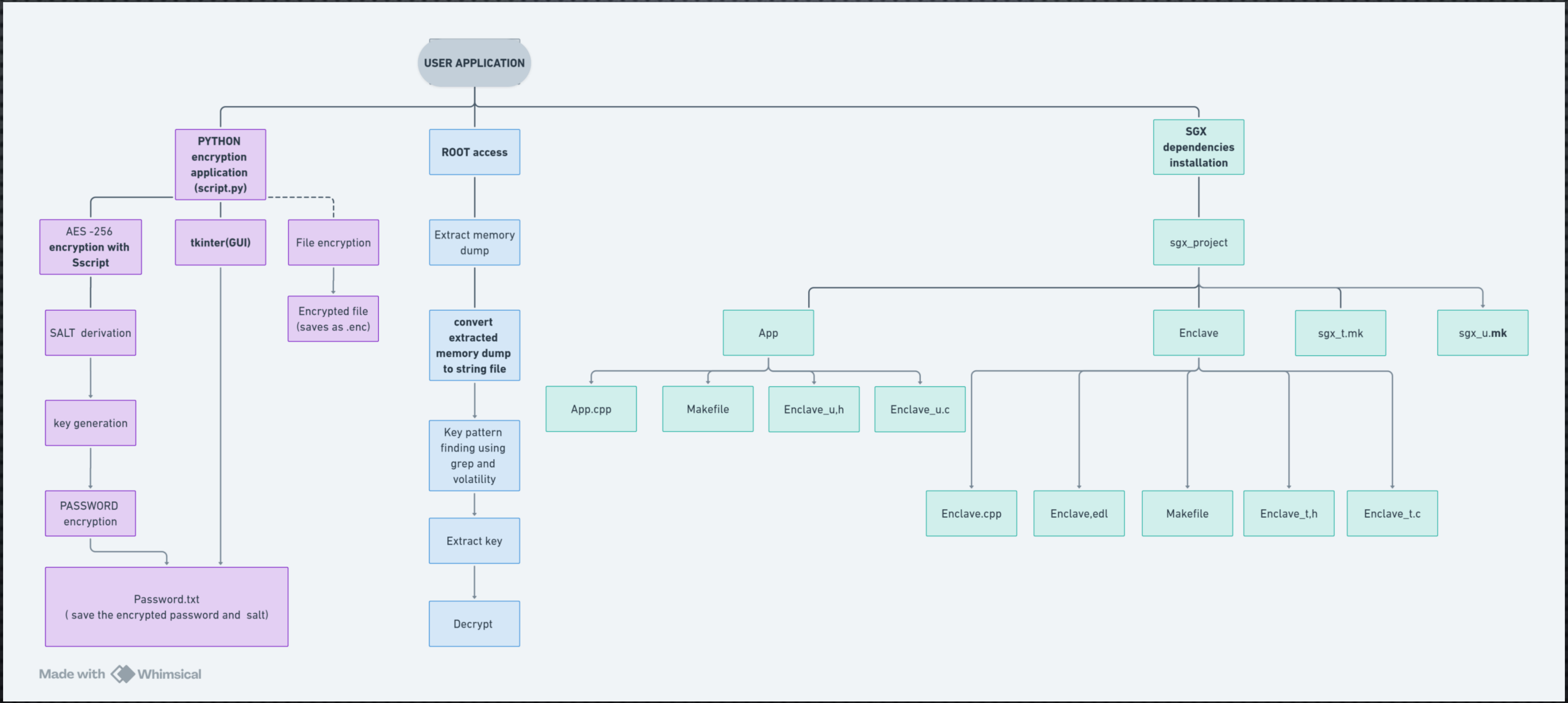
Protection of the Encryption Key and Sensitive Data Within an SGX Enclave:
SGX Enclave: Intel Software Guard Extensions (SGX) provides a protected area of execution known as an enclave. This enclave ensures that even if the operating system is compromised, the sensitive data and encryption keys remain secure and inaccessible.
Key Management: The encryption key used for AES-256 is generated and stored within the SGX enclave. The key is never exposed to the system's memory outside the enclave, preventing any form of key extraction through memory dumps or other attack vectors.
Data Isolation: All sensitive operations, such as encryption and decryption of passwords, are performed within the enclave. This isolation ensures that sensitive data is handled in a secure environment, immune to external threats.

Resistance to Memory Scraping and Root-Level Attacks:

Memory Scraping Protection: Traditional systems are vulnerable to memory scraping attacks, where an attacker with root privileges can dump the system memory to extract sensitive information. By running the password management application within an SGX enclave, the memory used by the application is encrypted and protected.
Root-Level Attack Mitigation: Even if an attacker gains root access to the system, the SGX enclave's protections ensure that the encryption keys and sensitive data remain inaccessible. The enclave's memory is encrypted and can only be decrypted within the secure execution environment, rendering root-level attacks ineffective.

User-Friendly Registration and Login Process with Secure Password Handling

PROCESS FLOW:



➢ TECHNOLOGIES USED:

1. *Hardware*:

•**Intel XEON 3rd Gen (Icelake) or later generation CPU with SGX enabled SKU:**

- SGX is a set of security-related instruction codes that are built into modern Intel CPUs. It allows applications to run in a protected area known as an enclave.
- **Note:** SGX is not supported on Intel Core Class CPUs; Xeon is required.

2. *Operating System*:

•**Ubuntu 22.04 LTS or 23.04 LTS:**

- These are the recommended Linux distributions for developing and running SGX applications.

3. *Software*:

•**SGX Software Packages:**

- **SGX SW packages to run SGX applications:** Required to enable the execution of SGX-enabled applications.
- **SGX SW packages to build SGX applications:** Necessary for developing applications that utilize SGX, including using frameworks like Gramine.

•***Docker*:**

- Containerization technology used to package and run the application.

4. *Programming Languages*:

•C, C++, Python:

- These languages are suited for system software development and are used to implement the authorization module and SGX integration.

5. *Libraries/Tools*:

•Linux Libraries for Encryption:

- **AES-256:** A widely used encryption standard. The application uses AES-256 to encrypt passwords with a generated 256-bit Password Encryption Key.

6. *SGX Enclave Framework*:

•Gramine:

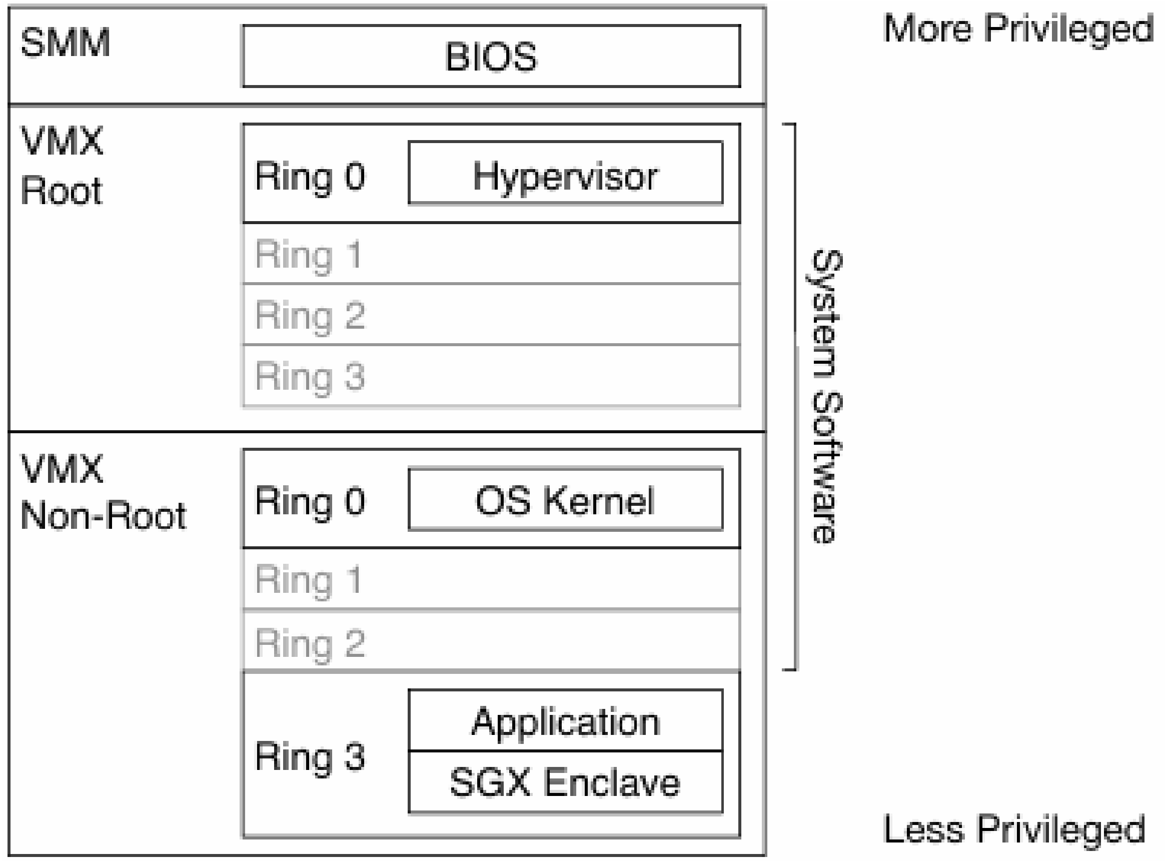
- A framework that helps in converting applications to run inside an SGX enclave. It ensures that sensitive parts of the application are executed in a secure enclave, protecting against various attacks.

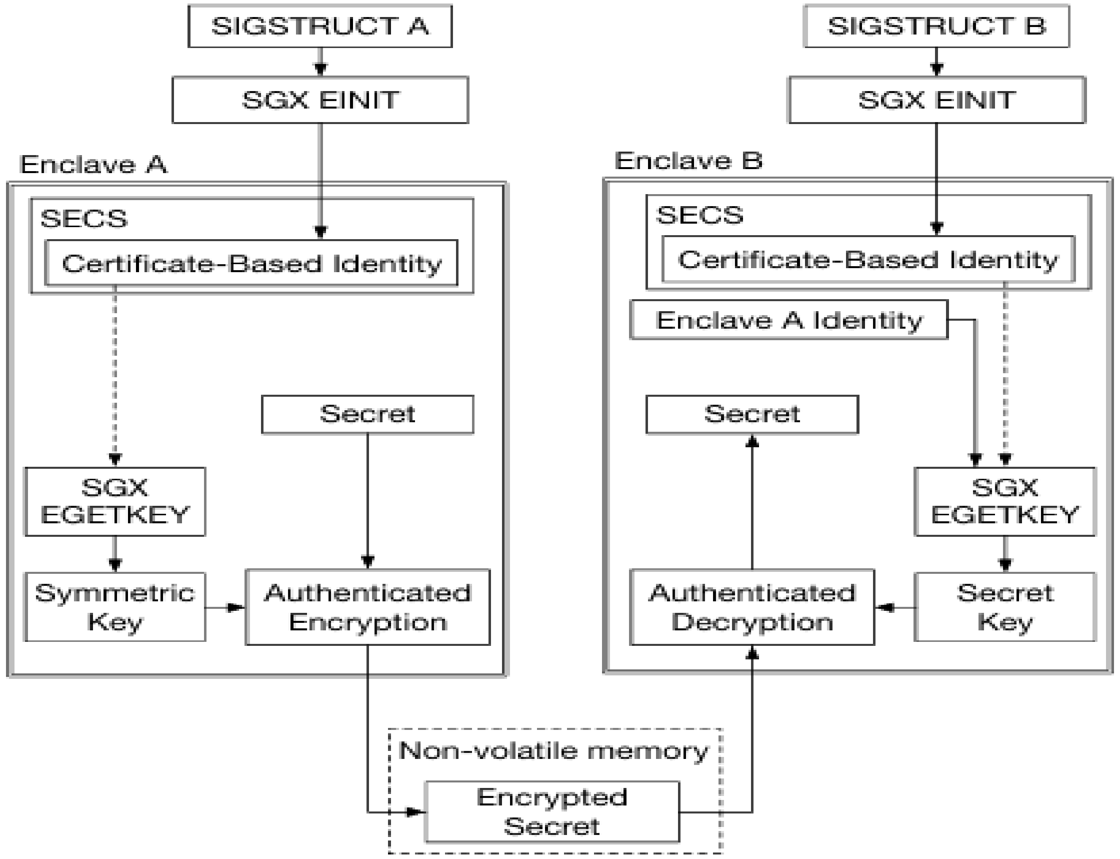
***Summary of Technologies*:**

- **Hardware:** Intel XEON 3rd Gen or later with SGX enabled SKU.
- **Operating System:** Ubuntu 22.04 LTS or 23.04 LTS.
- **SGX Software Packages:** For running and building SGX applications.
- **Containerization:** Podman.
- **Programming Languages:** C, C++, Python.
- **Encryption Libraries:** Linux libraries implementing AES-256.
- **SGX Enclave Framework:** Gramine.

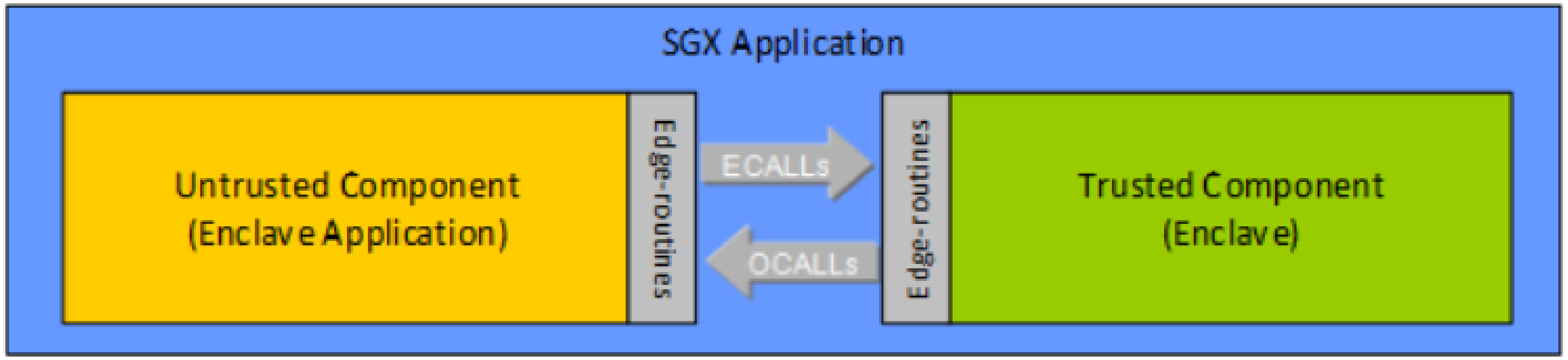
These technologies work together to create a secure environment for password protection, ensuring that sensitive data remains protected even in the presence of potential attacks.

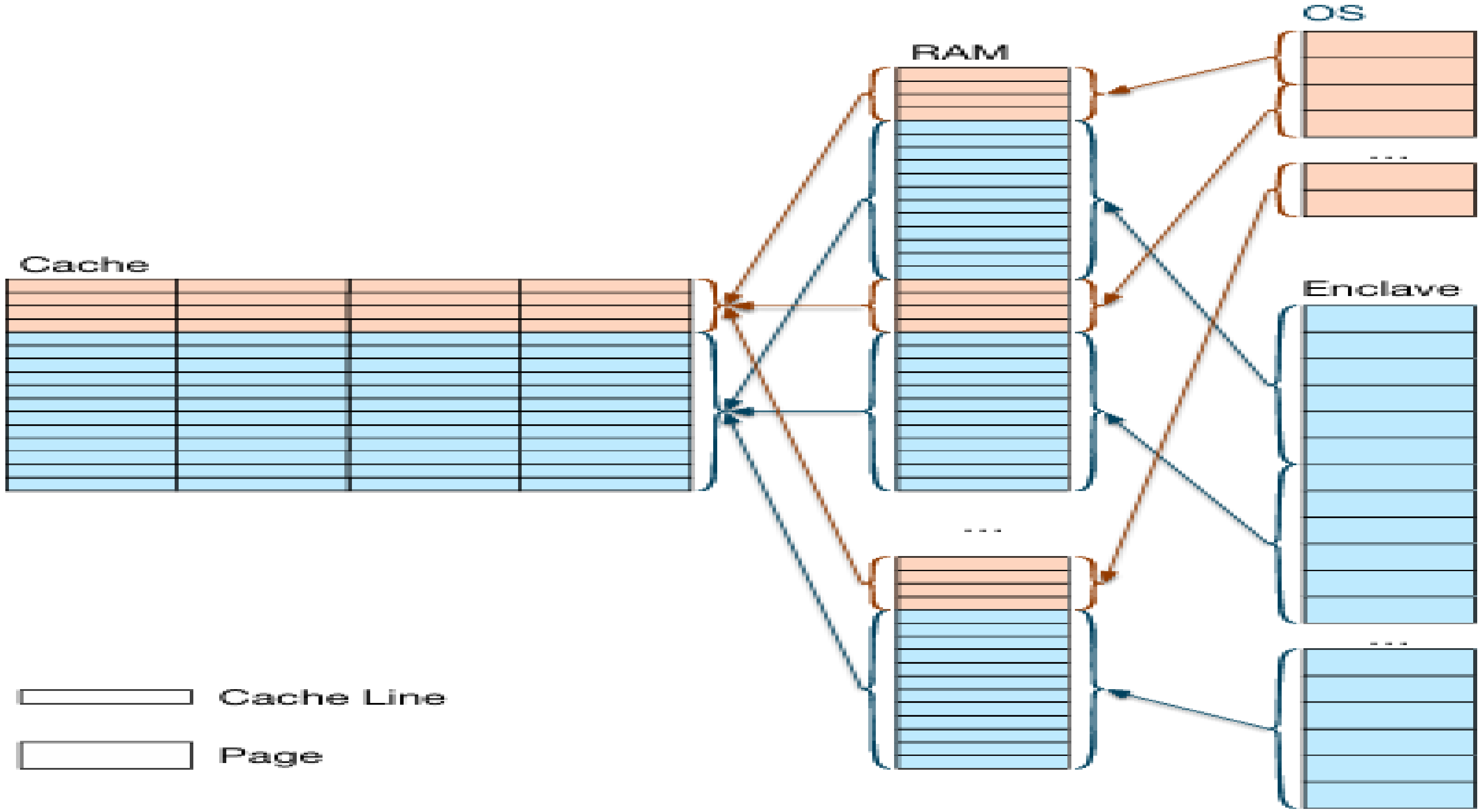
➢ ARCHITECHTURE:

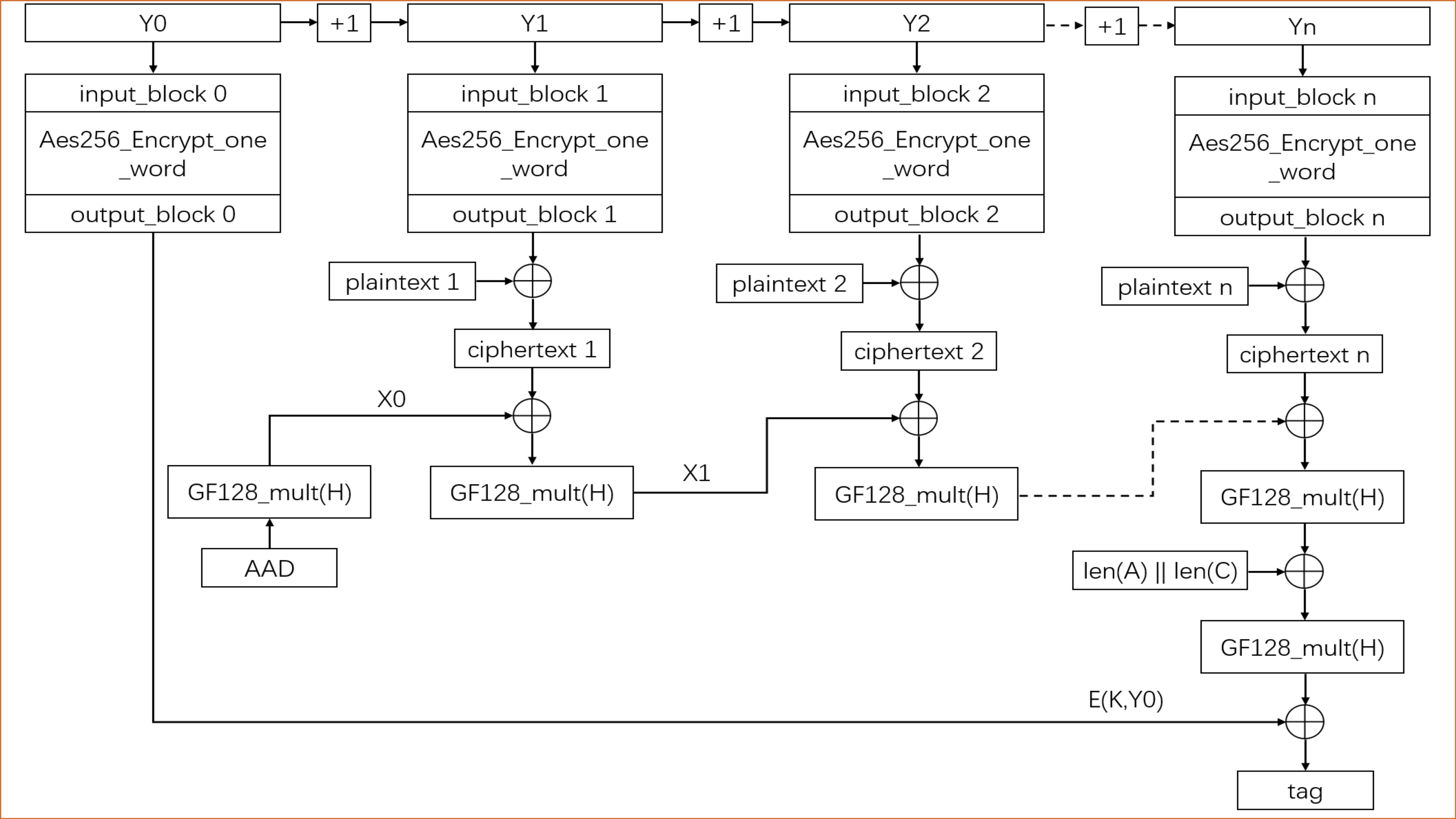


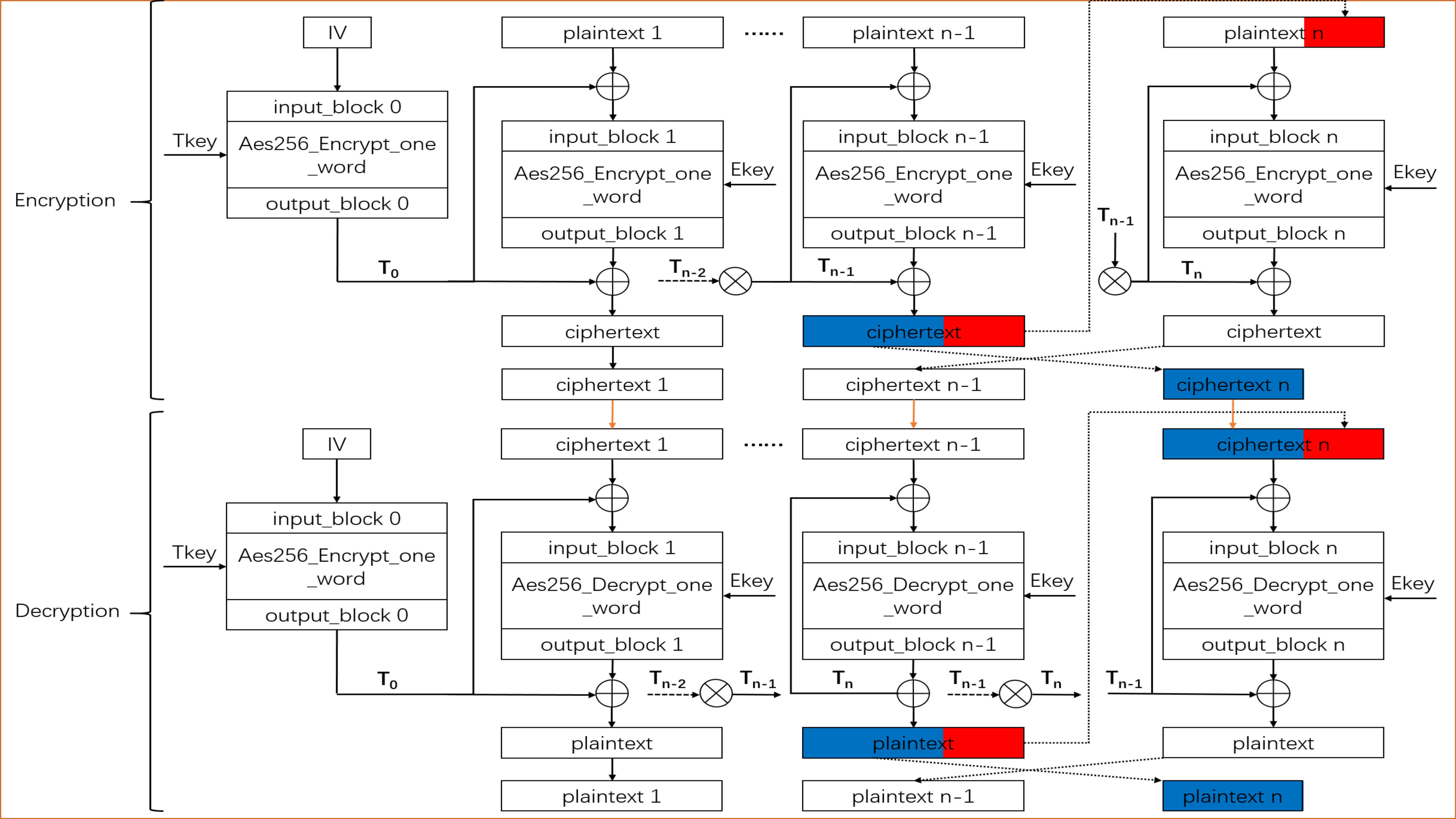


Intel SGX Application






Y0
+1
Y1
+1
Y2
+1
Yn
input_block 0
Aes256_Encrypt_one_word
output_block 0
input_block 1
Aes256_Encrypt_one_word
output_block 1
input_block 2
Aes256_Encrypt_one_word
output_block 2
input_block n
Aes256_Encrypt_one_word
output_block n
plaintext 1
plaintext 2
plaintext n
ciphertext 1
ciphertext 2
ciphertext n
X0
X1
GF128_mult(H)
GF128_mult(H)
GF128_mult(H)
GF128_mult(H)
AAD
len(A) || len(C)
GF128_mult(H)
E(K,Y0)
tag


Encryption
Decryption
IV
Tkey
input_block 0
Aes256_Encrypt_one_word
output_block 0
T_0
plaintext 1
input_block 1
Aes256_Encrypt_one_word
output_block 1
Ekey
ciphertext
ciphertext 1
......
plaintext n-1
input_block n-1
Aes256_Encrypt_one_word
output_block n-1
T_{n-2}
T_{n-1}
ciphertext
ciphertext n-1
plaintext n
input_block n
Aes256_Encrypt_one_word
output_block n
T_n
ciphertext
ciphertext n
ciphertext 1
Aes256_Decrypt_one_word
plaintext
plaintext 1
ciphertext n-1
Aes256_Decrypt_one_word
plaintext n-1
ciphertext n
Aes256_Decrypt_one_word
plaintext n

TEAM MEMBERS

AKSHATHA S SHASTRY
Encryption and memory extraction from root

INDHU T G
Decryption

MOHITH B K
SGX dependencies installation and docker containerization

DEEPTHI M K
Key pattern extraction

JAYANTH R
SGX enclave creation