Cacti

University at Buffalo

Xi Tan, MD Armanuzzaman, Zheyuan Ma, Qiqing Huang
Advised by: Professor Ziming Zhao, Professor Hongxing Hu

Design Overview

1. For integrity protection of the Firmware and Configuration files, we implemented **AES-GCM** mode encryption with two different keys. AES-GCM produces a tag along with ciphertext which ensures authenticity. For Readback authentication we used asymmetric crypto.

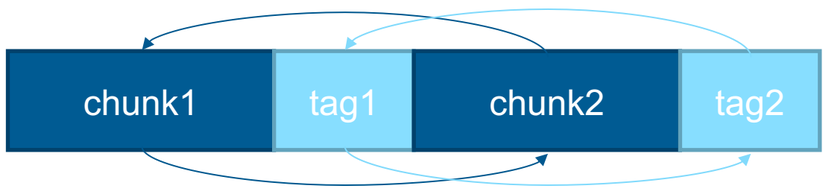
2. We utilized the **EEPROM** hardware feature to store the keys at build time. In addition, EEPROM memory is used to store the tags of firmware and configuration, after each update and load operation.

Offensive Highlight

Vulnerability: data integrity.

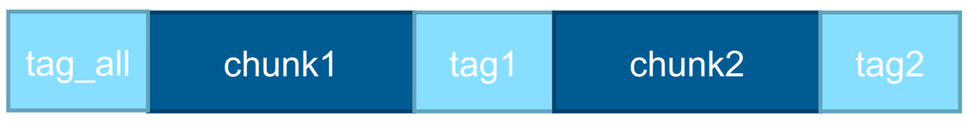
Case 1: Lack of integrity checking for whole data.

a. Attack: exchange chunks along with its tag



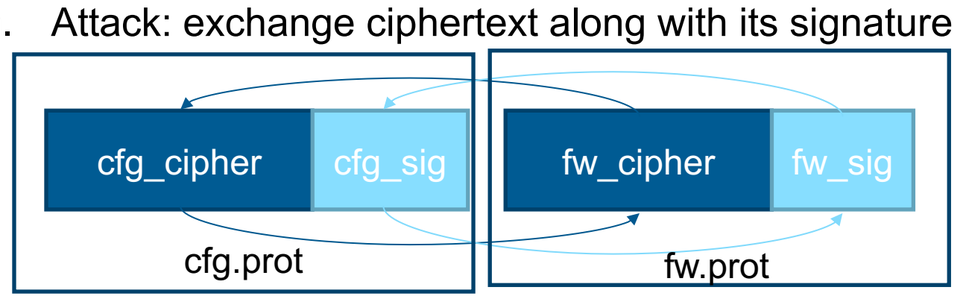
b. Defeat: add sequence number for each chunk and integrity checking for all tags

tag_all = sig(tag1, …, tagn)



Case2: Same crypto key for firmware and configuration protection and no integrity checking for generated signatures.

a. Attack: exchange ciphertext along with its signature



a. Defeat: Use different keys for different purposes

Case3: Non-cryptographic algorithm for configuration protection

a. Attack: decode directly

cfg_plaintext = base64.decode(cfg_cipher)

b. Defeat: use more secure crypto algorithm

e.g., cfg_cipher = AES.enc(cfg_plaintext)

Defensive Highlight

Measure point: Data integrity
Reason: To prevent the firmware and configuration from modifying by attackers.
Supposed defensive features: AES-GCM, EEPROM key storage, and MPU access control.

Feature description:
AES-GCM is good to protect the files as it produces a tag along with ciphertext, which ensures integrity and authenticity.
We divide the data into 4KB chunks (Figure 2). Each chunk of 4KB generates a tag, so there is a maximum of 16 tags for the configuration file and a maximum of 4 tags for the firmware file. For firmware files we do two separate encryptions, the first one is encryption of firmware data, and the second is encryption of generated tags and version number. The tag of the second encryption will be put as plaintext in file format. Thus, the bootloader can verify the integrity of firmware without waiting for the whole encrypted ciphertext. For the configuration file, there is only one encryption step, and the tags are plaintext.
Issue: i) we did not include any block sequence number, ii) our AES-GCM library cannot prevent a side-channel attack, iii) we did not authenticate the version number with valid ciphertext.

EEPROM is used for key and tag storage (Figure 1). IV storage memory will not be accessible by the flash trojan.
Issue: some pieces of sensitive data (e.g. version number) are stored on FLASH rather than EEPROM.

MPU is used to constrain the access rights of flash and SRAM.
Issue: when implementing, the result is not consistent with the physical environment and emulator environment. Due to the time, we did not spend much time fixing it.

Future plan: i) add sequence number for each cipher block; ii) authenticate the version number ciphertext combined with file ciphertext, iii) all sensitive data should store on EEPROM, iv) figure out the right way to use MPU correctly, v) find appropriate crypto libraries.

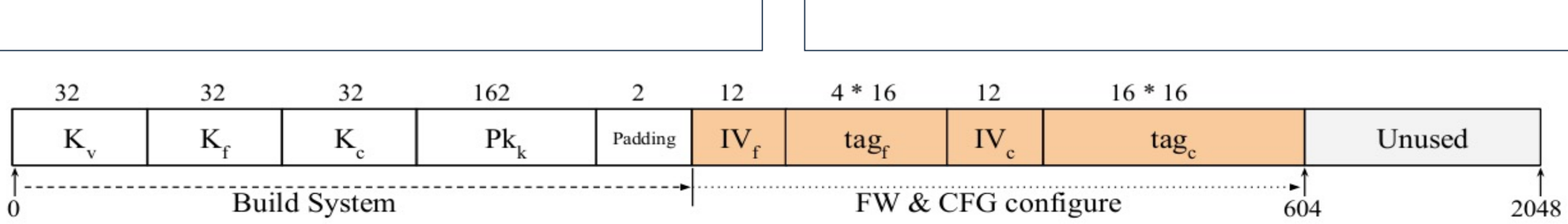


Figure 1: EEPROM Key and other data storage format

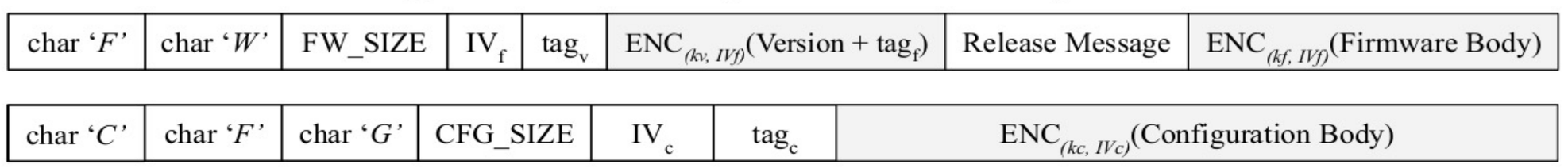


Figure 2: Protected Firmware and Configuration Format

References

AES-GCM	RSA	