Trusted Firmware-A Tech Forum

Oct 22, 2020

Encrypted FIP Support

Sumit Garg
Linaro Ltd.

Linaro

TrustedFirmware
.org

Overview

- FIP encryption
  - Assets
  - Use-cases
- Challenges
  - Secret key protection?
  - Device unique or class wide key?
  - Play nicely with firmware signature?
  - Firmware updates?
- Implementation

FIP encryption

FIP: Firmware Image Package

FIP encryption allows us to achieve **confidentiality** and in turn **integrity** for a firmware image bundled as part of FIP, using:

- **Symmetric** encryption
  - Reason to not use **asymmetric** encryption: boot time limitation.
- **Authenticated** encryption (eg. AES-GCM)
  - Ensures integrity of **encrypted** firmware blob.

FIP encryption

Assets?

Possible firmware assets to protect:

- **Software IP**
  - Allow confidentiality protection for software IP.
- **Device secrets**
  - Allow firmware image to act as secret store (though unlikely to be suitable for high value secrets).
- **Implementation details**
  - Make it harder to develop exploits for any vulnerabilities in the firmware.

FIP encryption

Use-cases?

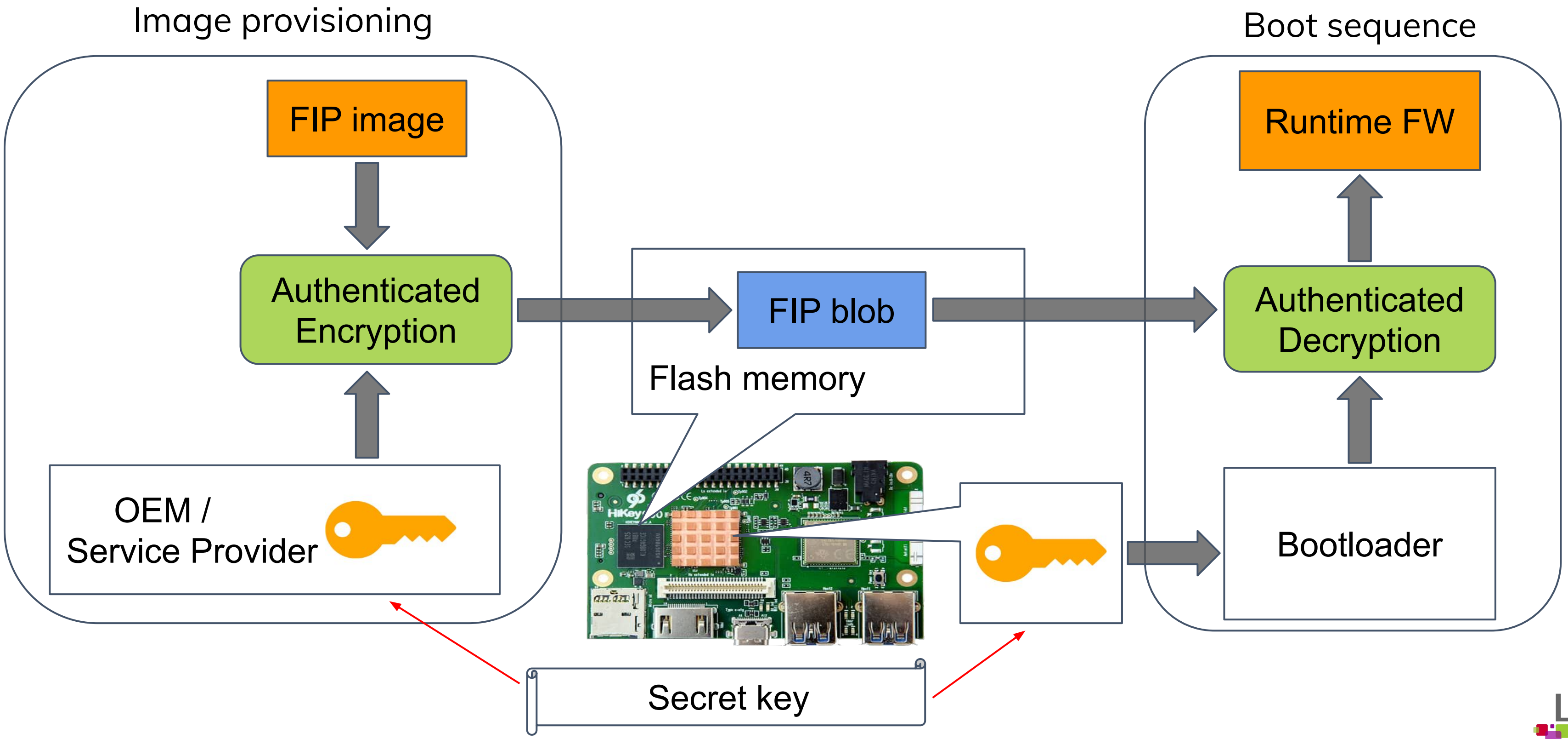
The major drivers for this feature are the emerging robustness requirements for software **Digital Rights Management** (DRM) implementations.

Make it even harder to reverse engineer Trusted Execution Environment (TEE) and therefore would like to see that **Trusted OS** is not just signed, but also **encrypted**.

TEE assets:

- DRM software IP.
- DRM implementation details.

FIP encryption



Challenge: Secret key protection?

Secret key protection may vary from one platform to another depending on use-case and hardware capabilities like:

- Key is derived from **device secrets** like OTP or such.
- Key is provisioned into **secure fuses** on the device.
- Key is provisioned into **hardware crypto accelerator**.
- Key is provisioned into platform **secure storage** like non-volatile SRAM etc.

Solution: Secret key protection

In order to address this varying requirement, we need to provide an **abstraction layer** to retrieve **secret key / secret key handle** and platform can provide underlying implementation.

TF-A provides:

```
int plat_get_enc_key_info(enum fw_enc_status_t fw_enc_status,
                          uint8_t *key, size_t *key_len,
                          unsigned int *flags, const uint8_t *img_id,
                          size_t img_id_len);
```

Challenge: Device unique or class wide key?

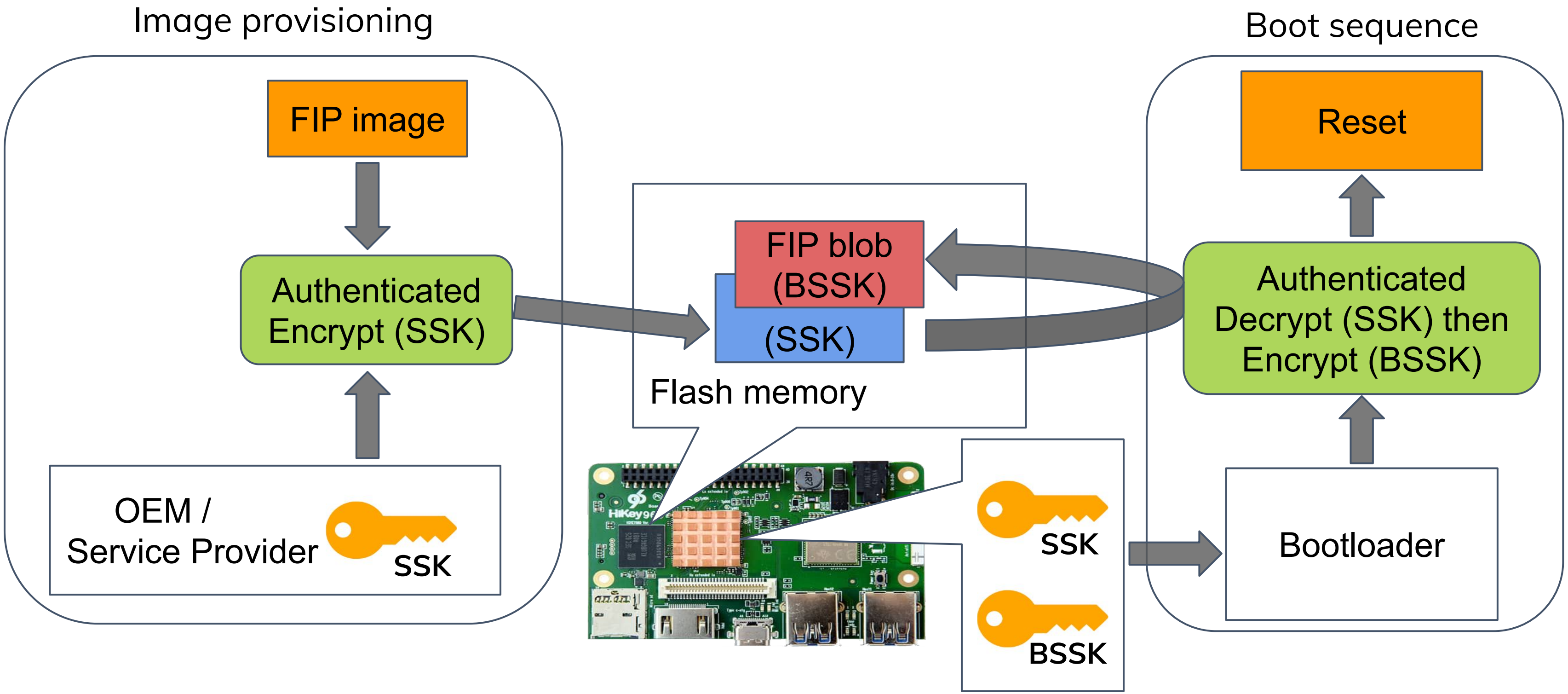
Secret key type?

- **Device unique key:** Unique per device, aka Binding Secret Symmetric Key (**BSSK**)
  - **Pros**: limits attacks surface to per device, provides protection against software cloning.
  - **Cons**: scalability issue to manage per device unique firmware images.
- **Class wide key:** Common shared key for a class of devices, aka Shared Secret Key (**SSK**)
  - **Pros**: single firmware image, easy to deploy and update.
  - **Cons**: comparatively larger attack surface, class wide attacks.

How about leveraging benefits of both key types?

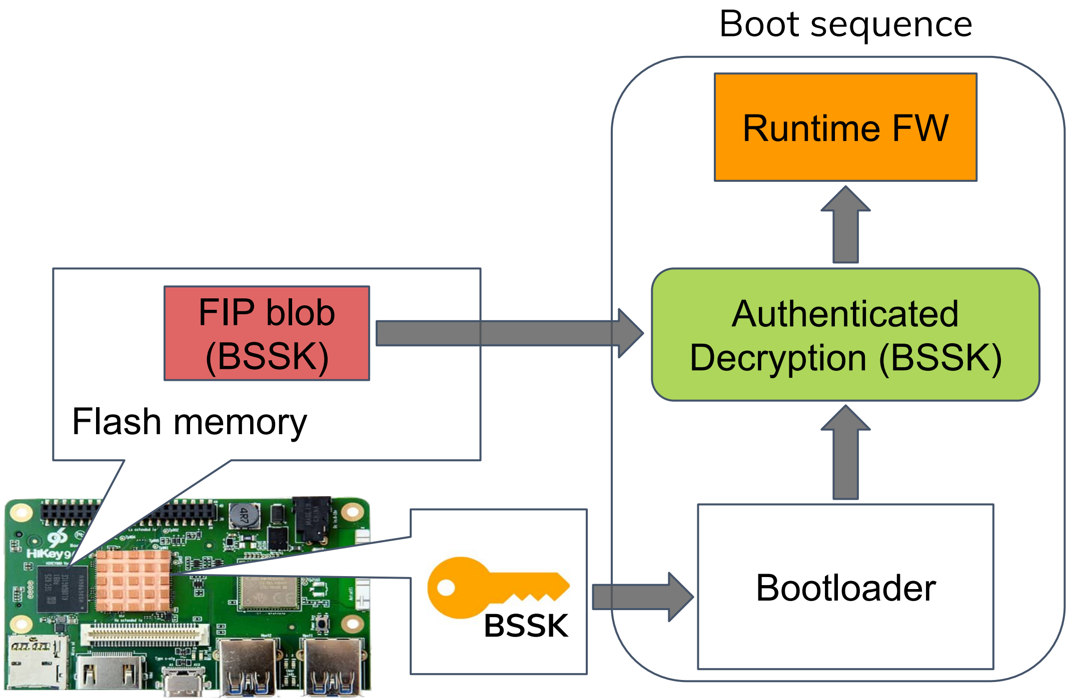
Linaro

FIP encryption: first boot (firmware binding)



Linaro

FIP encryption: subsequent boot



Challenge: encryption + signature?

Encryption and **signature** schemes are well known cryptographic constructs but when their combination is to be used:

- Proper attention is required towards **achievable** security properties

Possible combinations:

- **Encrypt-then-sign**
- **Sign-then-encrypt**
- **Sign-then-encrypt-then-MAC**

Challenge: encryption + signature?

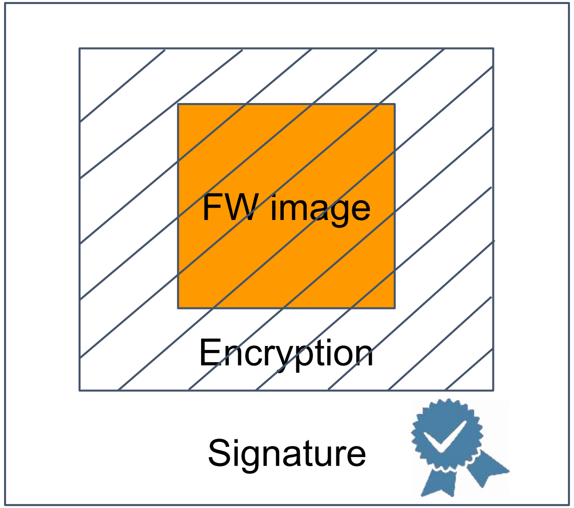
Encrypt-then-sign

Security properties:

- Confidentiality
- Integrity
- Authentication
- Authorization

Shortcomings:

- Only encrypted firmware blob is **non-repudiable** to OEM / SP.
- Signing encrypted blob makes it **immutable**
    - Doesn't allow **re-encryption** on device, aka firmware binding.



Challenge: encryption + signature?

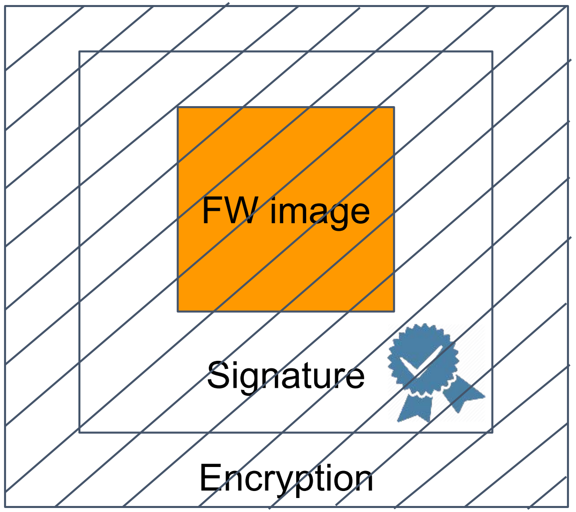
Sign-then-encrypt

Security properties:

- Confidentiality
- Authentication
- Authorization
- Non-repudiation

Shortcomings:

- **Plain** encryption doesn't assure integrity of encrypted blob.
    - Vulnerable to **Chosen Ciphertext Attacks** (CCAs).



Solution: encryption + signature

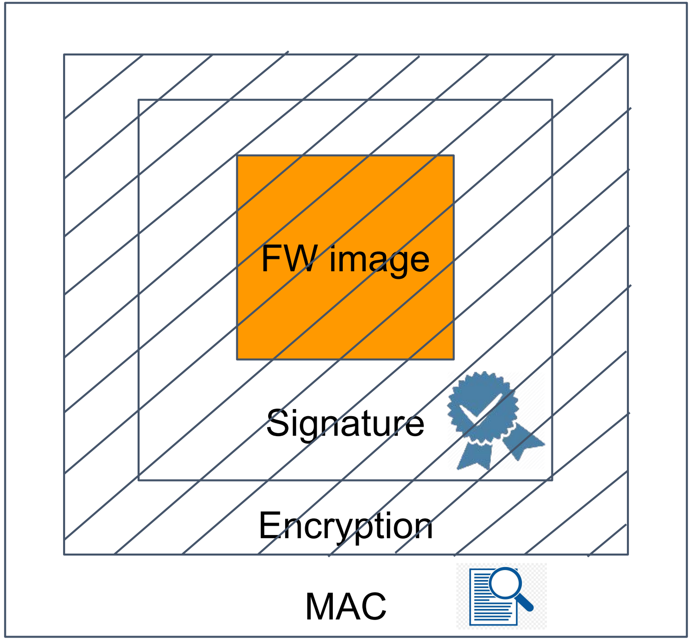
Sign-then-encrypt-then-MAC

Security properties:

- Confidentiality
- Integrity
- Authentication
- Authorization
- Non-repudiation

Concerns addressed:

- MAC tag assures **integrity** of encrypted blob.
- Allows firmware **re-encryption**.



Challenge: Firmware updates?

Generally, following approaches are used to apply firmware updates:

- Update complete FIP **partition**
  - Encryption **doesn’t** adds any complexity
    - Updater could verify overall FIP partition signature.
- Update **individual** images in FIP
  - Encryption **adds** complexity:
    - Updater needs to verify each individual image, requires access to encryption key.
    - Either updater needs to be a secure world entity or leverages secure world decrypt and verify service.

Implementation

Trusted Firmware-A (TF-A) supports an **I/O encryption layer** (drivers/io/io_encrypted.c):

- Layered on top of any base I/O layer (eg. drivers/io/io_fip.c)
  - To allow loading of corresponding encrypted firmware payload.
- Approach used: **sign-then-encrypt-then-MAC**
  - Leveraging existing **TBBR** Chain of Trust.
- Uses **encrypt_fw** tool (tools/encrypt_fw/) to encrypt firmwares during build.
- Build options:
  - **DECRYPTION_SUPPORT**: enables firmware decryption layer (values: **aes_gcm** or **none**)
  - **FW_ENC_STATUS**: firmware encryption key flag (values: 0 -> **SSK**, 1 -> **BSSK**)
  - **ENC_KEY**: 32-byte (256-bit) symmetric key
  - **ENC_NONCE**: 12-byte (96-bit) encryption nonce or Initialization Vector (IV)
  - **ENCRYPT_{BL31/BL32}**: flag to enable BL31/BL32 encryption

Future work...

- Let's champion open source security frameworks
    - Reduces efforts to maintain custom solutions
- FIP encryption framework: contributions are welcome, adding:
    - Framework improvement
    - Platform support

Thank you

Linaro