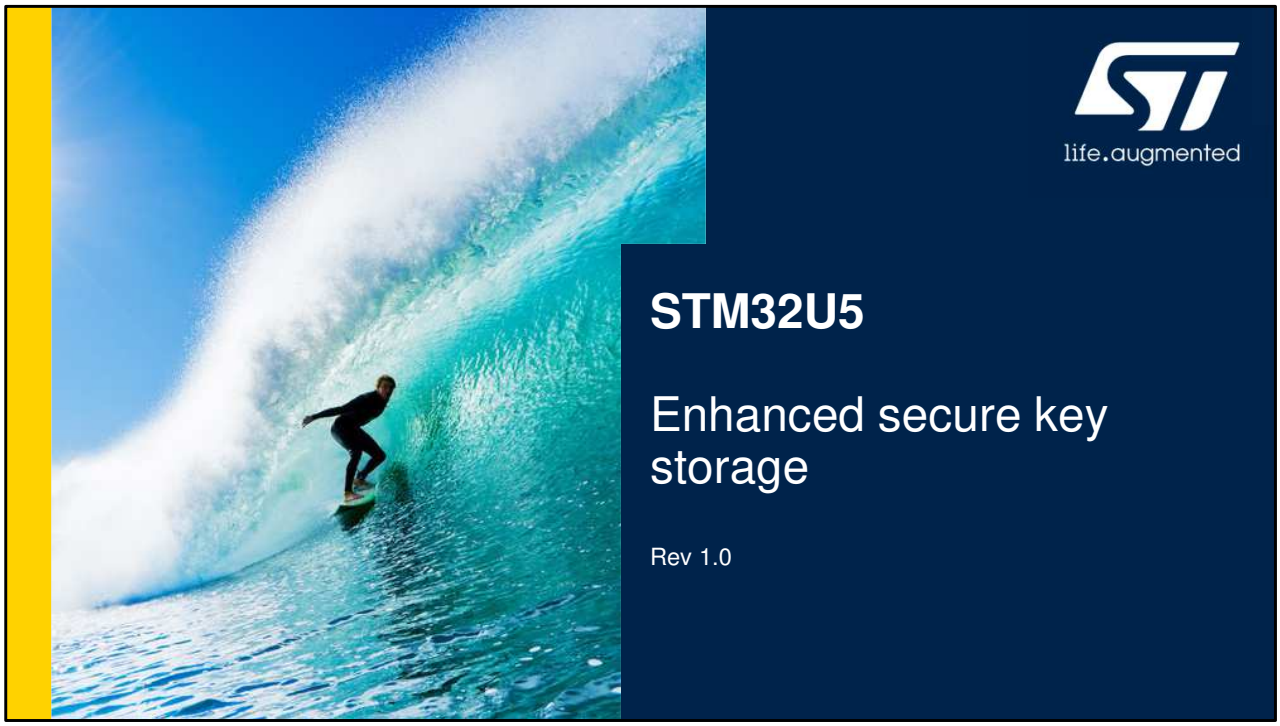
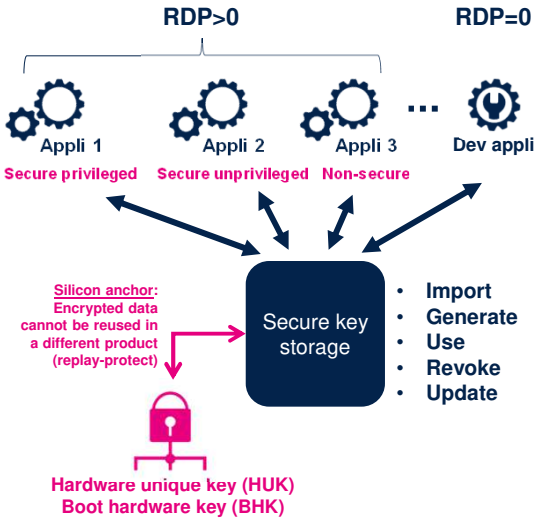


Hello and welcome to this presentation of the STM32U5 enhanced secure key storage functions. It covers the features of the SAES module which are used to secure key storage applications.

Secure key storage application



- Multiple applications need to access keys stored in the device
- STM32 includes hardware mechanisms that **increase the robustness** of this critical function

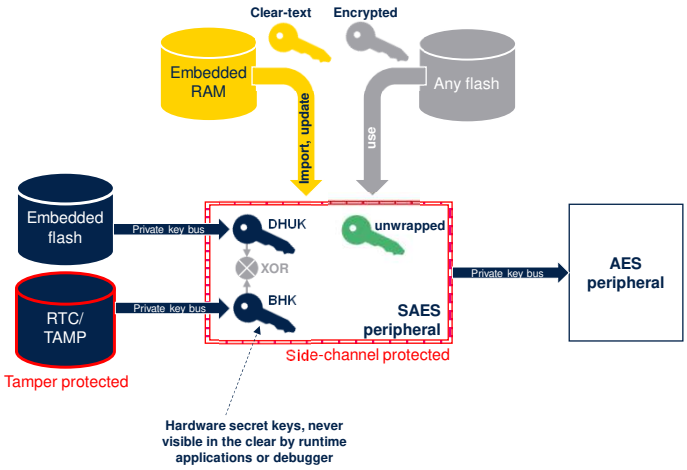
Application benefits

- Protects confidentiality of the keys it manages
- Manages the lifecycle of keys: import, generate, use, revoke, update.
- Makes each key available only to the authorized user (ideally)

2

The Secure key storage application is critical to security. STM32 includes hardware mechanisms to increase the robustness of it. This training module describes them.
The readout protection, or RDP, mechanism, which is a full hardware feature, controls the access to the devices debug, test and provisioned secrets.
The secure key storage manages the life cycle of keys: import, generate, use, revoke and update.

Enhanced secure key storage



Application benefits

- Better protection of keys is increasingly required by the latest security standards (PSA, SESIP…)
- Enables on-chip encrypted storage technology with unique ID per chip
  - Alternative to the Physical Unclonable Function (PUF)
  - ST solution relies on a secret in system flash (RHUK) and a secret in the silicon

life.augmented

3

To better protect keys, the side-channel protected SAES peripheral uses special hardware secret keys DHUK and BHK to make critical application keys that are never visible in the clear to the runtime application or the debugger. Thanks to the derived hardware unique key (DHUK) the user can also implement robust

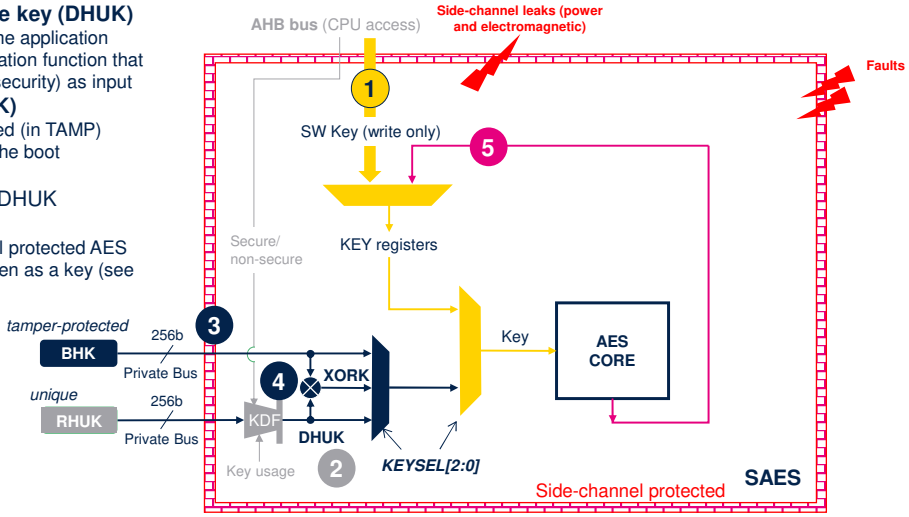
Physical Unclonable Function (PUF).
In this module, the following use cases will be detailed:
- Encrypting keys with hardware secret keys (key wrapping)
- Decrypting keys with hardware secret keys (key unwrapping)
- Provisioning keys for AES peripheral (shared key

wrapping)
- Decrypting keys for AES peripheral (shared key unwrapping)
- Hardware key protection in SAES

Possible selection of keys in SAES

1. **Software key**
   - Written by CPU
2. **Derived hardware unique key (DHUK)**
   - Non-volatile, secret to the application
   - Derived using key derivation function that use key usage (mode, security) as input
3. **Boot hardware Key (BHK)**
   - Volatile, tamper protected (in TAMP)
   - Written then locked by the boot application
4. **XORK**: XOR of BHK and DHUK
5. **Wrapped key**
   - Result of a side-channel protected AES decryption, directly written as a key (see next slide)

- Keys 2 to 4 are never readable by the runtime CPU or debugger
- Keys 5 are also never readable by the runtime CPU or debugger if they are encrypted with keys 2 to 4



life.augmented

4

Multiple keys can be used inside the side-channel protected SAES:

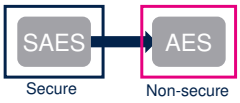
- Software keys, written by the CPU in write-only key registers
- Derived hardware unique key (DHUK), non-volatile and secret to the application. DHUK is derived using an internal key derivation function, see KDF in the figure.
- Boot hardware Key (BHK), volatile and tamper protected in the TAMP peripheral. BHK is written then locked by the boot application.
- XOR of BHK and DHUK
- Wrapped keys, ideally encrypted with hardware secret keys, to make them also hardware secrets.

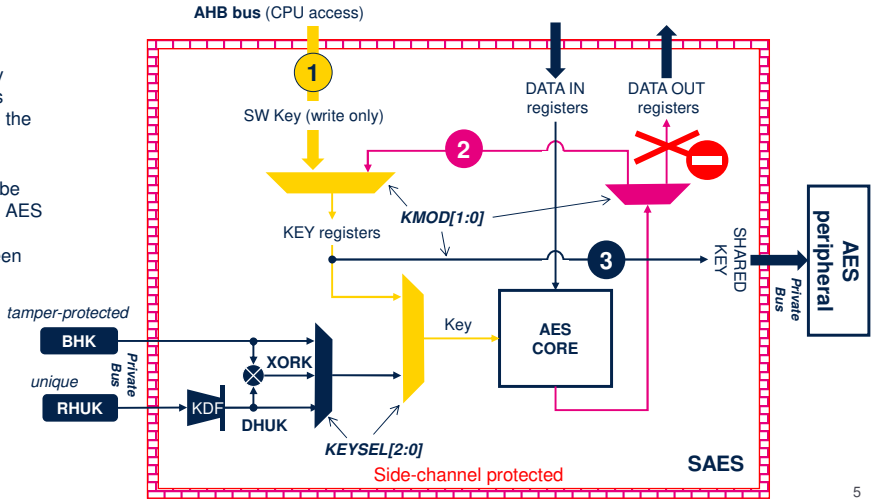
The SAES peripheral can therefore wrap and unwrap

application keys using hardware-secret keys DHUK, XOR-ed or not with the application key BHK.
Wrapping / unwrapping is equivalent to encrypting / decrypting the key.
With this feature, AES keys can be made usable by the application software without being exposed in clear-text.

Possible usage for keys in SAES

1. **Normal key mode**
   - Written by the CPU in write-only registers
2. **Wrapped key mode**
   - Written by the CPU encrypted
   - When decrypted key is directly loaded into write-only registers
   - Never disclosed in the clear to the application!
3. **Shared key mode**
   - Special wrapped key that can be loaded by faster, not protected AES peripheral
   - Easy way to share keys between secure and non-secure worlds



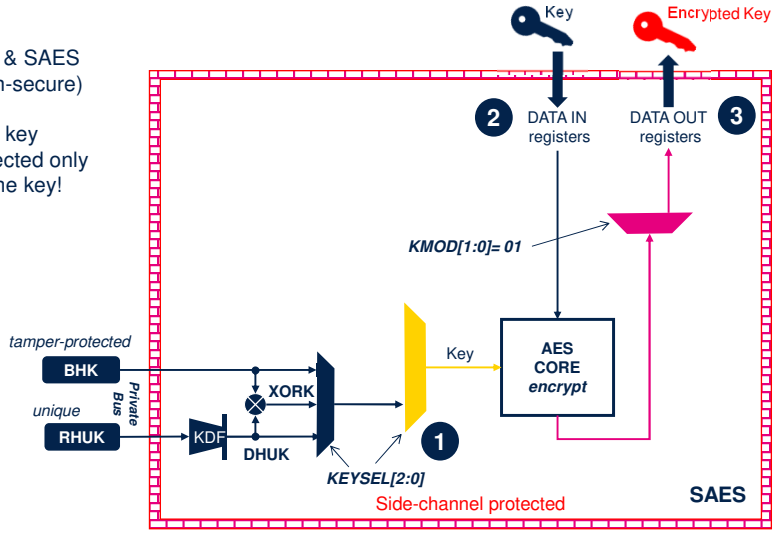


Application has three options when loading keys in SAES:

- Write key registers directly
- Write an encrypted key and decrypt it directly into the key registers
- Write an encrypted key and decrypt it directly into the key registers, then allow the AES peripheral to load it through a private key bus.

Provisioning usage: key wrapping

- Encrypting a key using **Wrap Mode**
  1. Select a hardware secret key (DHUK, BHK, XORK)
     - KDF depends on KMOD & SAES protection (secure or non-secure)
  2. Write the clear text key
  3. Read the encrypted, wrapped key
     - If DHUK or XORK is selected only this silicon can decrypt the key!
  4. Save the key in any flash

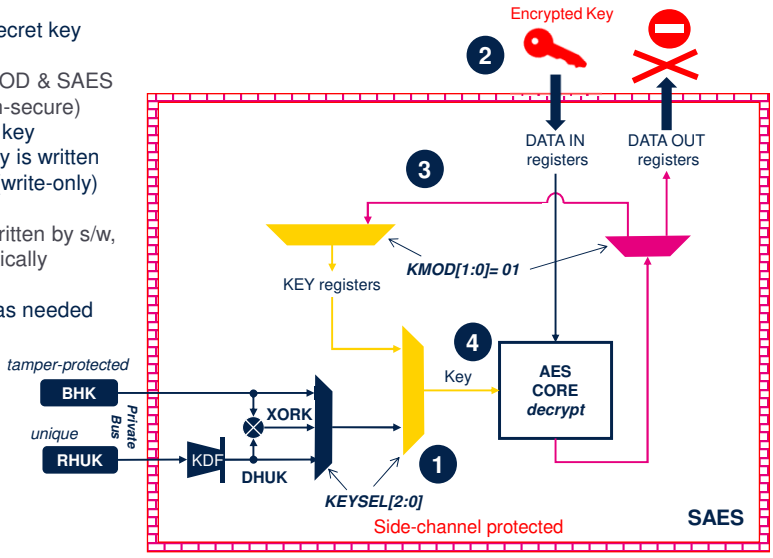


life.augmented

This slide details how to encrypt a key using the wrap key mode. The Resulting encrypted key can be stored in any flash.
The first step consists in selecting a hardware secret key. The Key Derivation Function depends on the TrustZone state and the key use state programmed in the KMOD field.
Then software writes a clear text key.
Finally, the AES core reads the resulting encrypted key.

Provisioning usage: key unwrapping

- Decrypting a key using **Wrap Mode**
  1. Select the correct hardware secret key (DHUK, BHK, XORK)
     - KDF depends on the KMOD & SAES protection (secure or non-secure)
  2. Write the wrapped, encrypted key
  3. The decrypted, unwrapped key is written by SAES in the key registers (write-only)
     - DOUT returns zero
     - If the key registers are written by s/w, the whole key is automatically erased!
  4. The application uses the key as needed



life.augmented

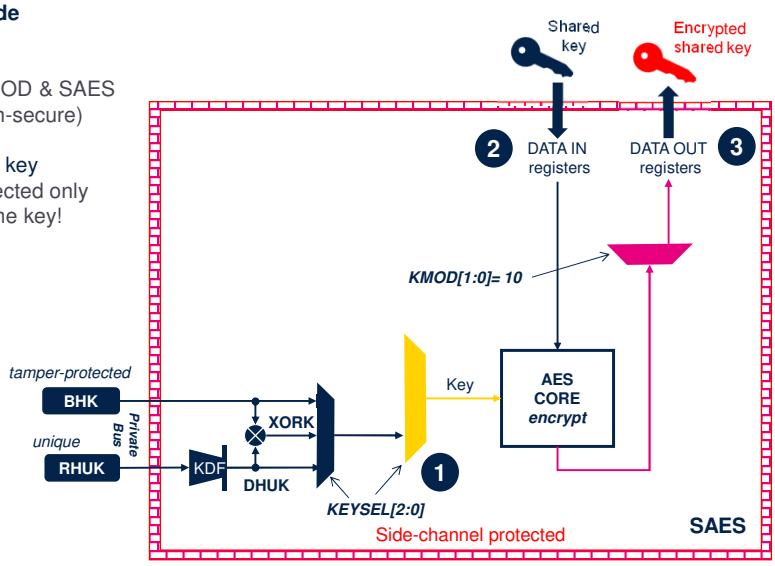
7

Any key encrypted with the wrap key mode must be decrypted using the same mode, with the correct key. The result is automatically stored in the write-only key registers.
The encrypted key generated in the previous slide has to be decrypted prior to being used for encryption and decryption purposes.
The first step consists in selecting a hardware secret key.
In step 2, the software writes the wrapped key to the DATA IN registers.
In step 3, the AES core unwraps this key and stores the original key in the write registers present in the SAES module, which are not accessible by the software.
Then the application can use the unwrapped key as

needed.
Note that this unwrapped key is automatically erased when the software writes the key registers.

Provisioning usage: shared key wrapping

- Encrypting a key using **Shared Mode**
  1. Select a hardware secret key (DHUK, BHK, XORK)
     - KDF depends on the KMOD & SAES protection (secure or non-secure)
  2. Write the clear text key
  3. Read the encrypted, wrapped key
     - If DHUK or XORK is selected only this silicon can decrypt the key!
  4. Save the key in any flash

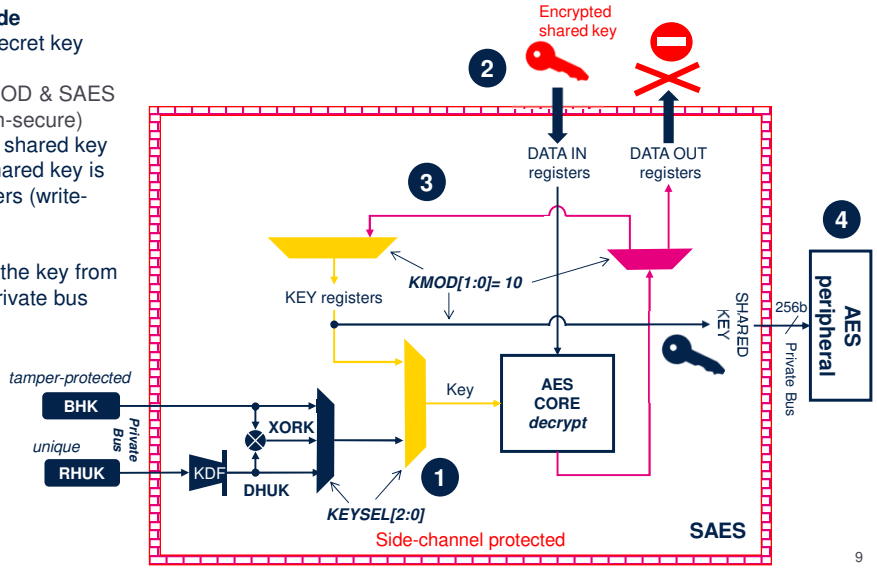


life.augmented

This slide details how to encrypt a shared key using the shared key mode. The resulting encrypted key can be stored in any flash.
Before SAES can share a key with the AES peripheral, the key must be encrypted once.
The encryption sequence of a shared key is the same as for a wrapped key, with KMOD set to 10 instead of 01.

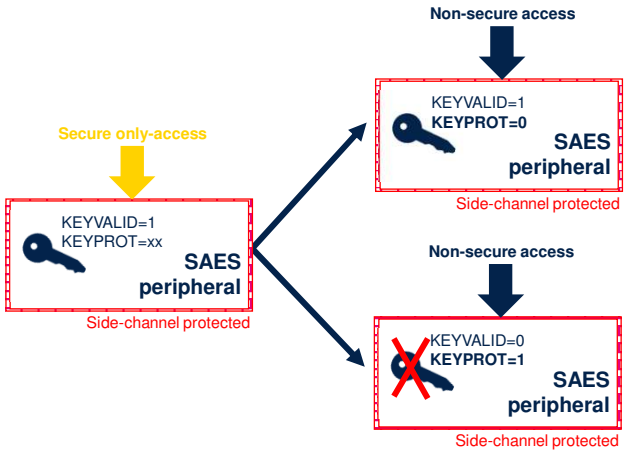
Shared key unwrapping

- Decrypting a key using **Shared Mode**
  1. Select the correct hardware secret key (DHUK, BHK, XORK)
     - KDF depends on the KMOD & SAES protection (secure or non-secure)
  2. Write the wrapped, encrypted shared key
  3. The decrypted, unwrapped shared key is written by SAES in key registers (write-only)
     - DOUT returns zero
  4. Use the key in SAES, or load the key from the AES peripheral, using a private bus



Each time SAES needs to share a key with the AES peripheral, the shared encrypted key must be decrypted in SAES, then loaded by AES.
The decryption sequence of a shared key is the same as for a wrapped key, with KMOD set to 10 instead of 01.
The result is automatically stored in the write-only key registers and can be loaded by the AES peripheral while KEYVALID is equal to 1 in the SAES status register.

SAES key protection (KEYPROT)



Application benefits

- When loading a key, the application can make sure that it can only be used by authorized applications (for example secure applications)

life.augmented

DHUK, BHK and XORK are *always* protected with KEYPROT

10

When loading a key in SAES, the application can make sure that this key can only be used by the authorized applications (for example secure applications), by setting the KEYPROT bit to 1.
When KEYPROT is equal to 1, SAES erases the key data and clears KEYVALID, as soon as it detects an access that does not match the security level of the loaded key.
DHUK, BHK and XORK are always protected with KEYPROT.

Thank you

life.augmented

For more details on SAES module please refer to the symmetric crypto training module.