AN12324

LPC55Sxx usage of the PUF and Hash Crypt to AES coding

Rev. 0 — February 2019

Application Note

by: NXP Semiconductors

Contents

1 Introduction

This application note describes how to securely generate, store, and retrieve user keys using the root key. The root key is used as a Key Encryption Key (KEK) to protect other user keys. The root key is generated by the Physically Unclonable Function (PUF) periphery. The PUF technology generates a device-unique 256-bit KEK, which is a digital fingerprint of a device. The encrypted keys (key codes) can then be stored in the MCU. This document discusses the key code storage in the flash memory of the Protected Flash Region (PFR) and key provisioning to the system by the ISP and IAP commands. The communication tools (blhost and elftosb) are also described.

2 Physically Unclonable Feature (PUF)

2.1 PUF features

- The PUF periphery generates a 256-bit key. The key is device-unique, unclonable, and used as the Key Encryption Key (KEK), which is a digital fingerprint.
- The PUF can generate 64-bit to 4096-bit keys.
- The PUF encrypts a 64-bit to 4096-bit key to a key code.
- The PUF decrypts a key code back to a key.
- The key can be sent to the hash crypt engine (or PRINCE) through the internal hardware bus.
- The key can be read by the CPU through the register and the AHB bus.
- The PUF offers other features to prevent hacking, such as blocking functionalities (enroll, code output, keylock) or enhanced side-channel attack (keymask).

2.2 PUF unique key principle

The uniqueness of the KEK generated by the PUF is based on naturally occurring variations in the attributes of transistors when chips are fabricated (length, width, thickness). Each time the SRAM block powers on, the cells come up as either 1 or 0. The start-up values create a random and repeatable pattern, which is unique to each chip. It is called SRAM Startup Data (SD).

The SRAM SD, together with the Activation Code (AC), are turned into a digital fingerprint used as a secret key that builds the foundation of a security subsystem. The digital fingerprint is used as a root key (or KEK).

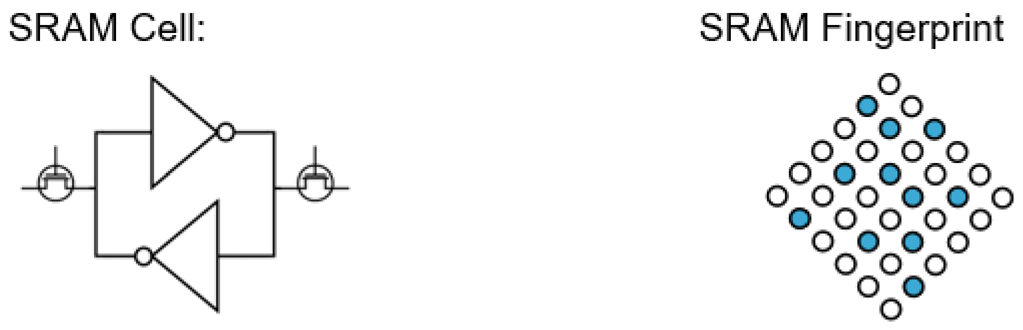


Figure 1. PUF principle based on an array of SRAM cells

The PUF module features embedded error correction, so the probability of a key reconstruction failure is < 10-9 (in worst-case conditions).

2.3 PUF usage

The security of any system depends on how securely the keys are stored. If the key is placed in the inaccessible flash memory, hackers may uncap the chip and read the memory. The same applies for the fuses, which can be read after uncapping.

When you encrypt a key, you must store it somewhere. If hackers read out the whole flash, they can clone your device because the stored key is the same for the whole production. If hackers successfully crack your key, the key is valid for all your devices. All these problems are solved using PUF. Each MCU has its own unique digital fingerprint, which is not stored on the chip and cannot be read when the device is not powered. The following sections show how to use the PUF to generate, store, and retrieve keys.

The PUF controller functionality includes these commands: Zeroize, Enroll, Start, SetKey, GenerateKey, and GetKey. These commands are used to control the PUF key management.

2.4 PUF Enroll

To start working with PUF, you need to enroll it. During enrolling, the SRAM startup data are derived to a digital fingerprint and a corresponding activation code is generated. The activation code is read out through the CODEOUTPUT register.

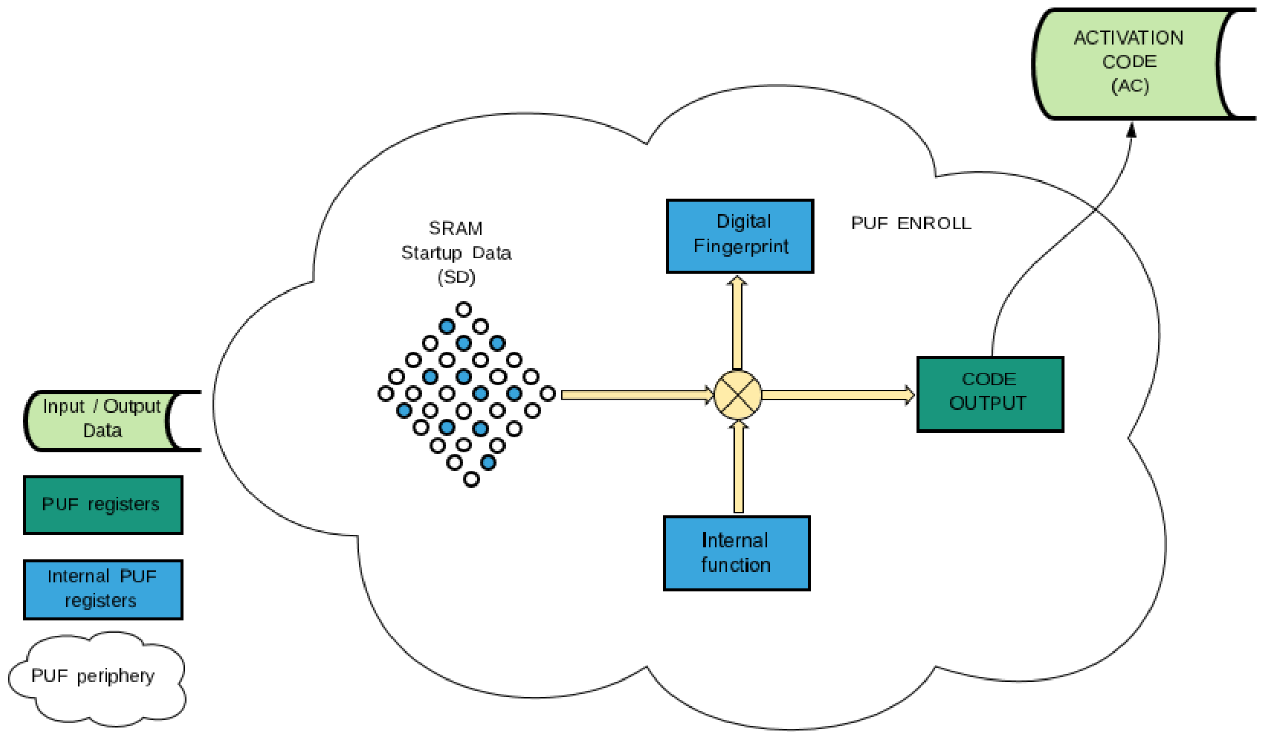


Figure 2. PUF Enroll command

Each time the PUF Enroll is performed, a new and different activation code (and digital fingerprint) is generated. The key code can be decrypted to a key with a corresponding digital fingerprint used for the key encryption to the key code. This is the reason why the AC must be stored to decrypt the key code later. The AC is a 1192-byte data chunk, which must be stored into the non-volatile memory.

After a successful PUF Enroll command, the SetKey command becomes available. The PUF must be power-cycled and started to enable the GetKey command. The keys and the activation code can be also generated using the ISP commands (they use the ROM code to call the PUF functions internally) and stored into the flash.

2.5 PUF Start

After the PUF Enroll command is performed and the AC is stored, the PUF can be used. The PUF must be power-cycled and started by the PUF Start command. The activation code is loaded into the PUF through the CODEINPUT register and the PUF engine combines it together with the SRAM startup data into a digital fingerprint.

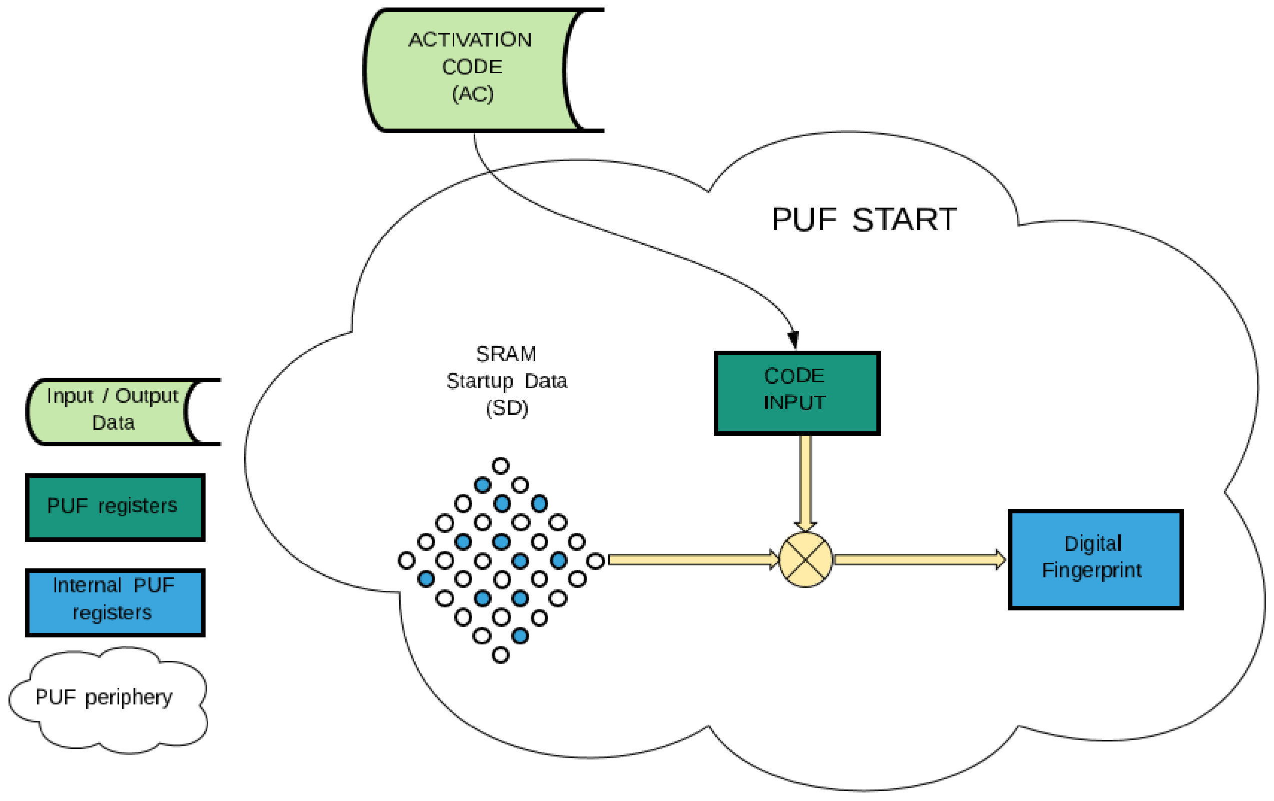


Figure 3. PUF Start command

The digital fingerprint is a 256-bit key used as a root key or a KEK for encryption/decryption of user keys. After a successful Start command, the SetKey and GetKey commands become available. The digital fingerprint is available in the PUF until the next power cycle.

2.6 PUF SetKey

The SetKey command is used to create a user key and a corresponding key code. The user key and the digital fingerprint are combined to generate a key code. The input parameters are user key, key size, and key index.

- The user key is sent to the PUF through the KEYIN register and must have a correct length, depending on the key size.
- The key size is sent to the PUF through the KEYSIZE register. It must have a number between 0 and 63. It instructs the PUF periphery to generate a corresponding key size between 64 and 4096 bits (0 generates a 4096-bit key; see the user manual).
- The key index is a number from 0 to 15, which is added to the header of a generated key code. It is sent to the PUF through the KEYIDX register. The key index is retrieved back during the GetKey command and accessible in the KEYOUTIDX register.

A special case is using a key index of 0 to instruct the GetKey command to send the decrypted key into one of the peripheries (AES or PRINCEx) through the internal secure bus or to the AHB. The key code is a secure representation of the encrypted key, which must be read out through the CODEOUTPUT register and safely stored into the flash memory. There is also a mechanism to block a key with a given index from being present on the APB register interface and from being read by the CPU. This setting is done through the IDXBLK and IDXBLK_DP registers.

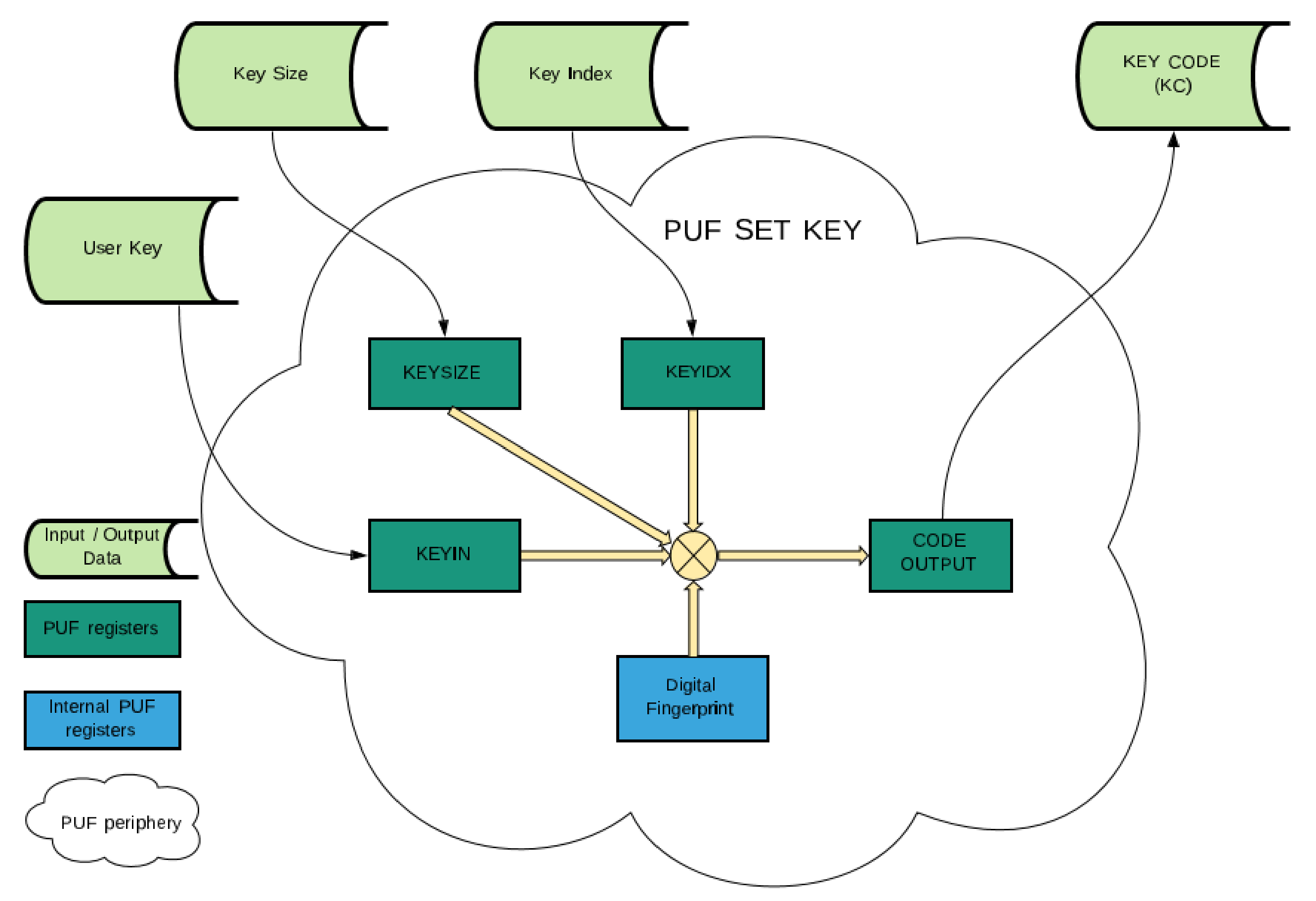


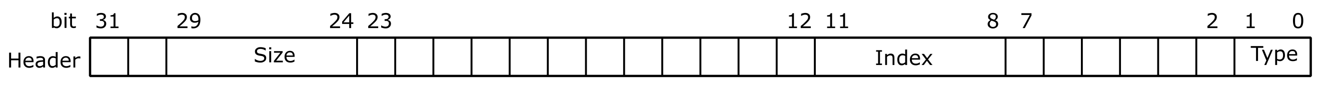
Figure 4. PUF SetKey command—generating user key code

The generated key code is a block of data with a 32-bit key header in the beginning.

The 32-bit key header has these features:

- Type:
  - 0—generated key.
  - 1—user key.
- Index: from 0 to 15. 0 is a special case to send the key internally to the AES or PRINCE engines. Otherwise, it can be used as a user's tag.
- Size: from 0 to 63 for keys ranging from 64 bits to 4096 bits (0 means 4096 bits).

2.7 PUF GenerateKey



The difference between the GenerateKey and SetKey commands is that the generated key is a random number generated by the PUF periphery in contrary to the key defined and provided by the user. The only required parameter (key size) must be entered into the KEYSIZE register and the key index must be entered into the KEYIDX register. The output is read out from the CODEOUTPUT register and must be stored for the key reconstruction later on.

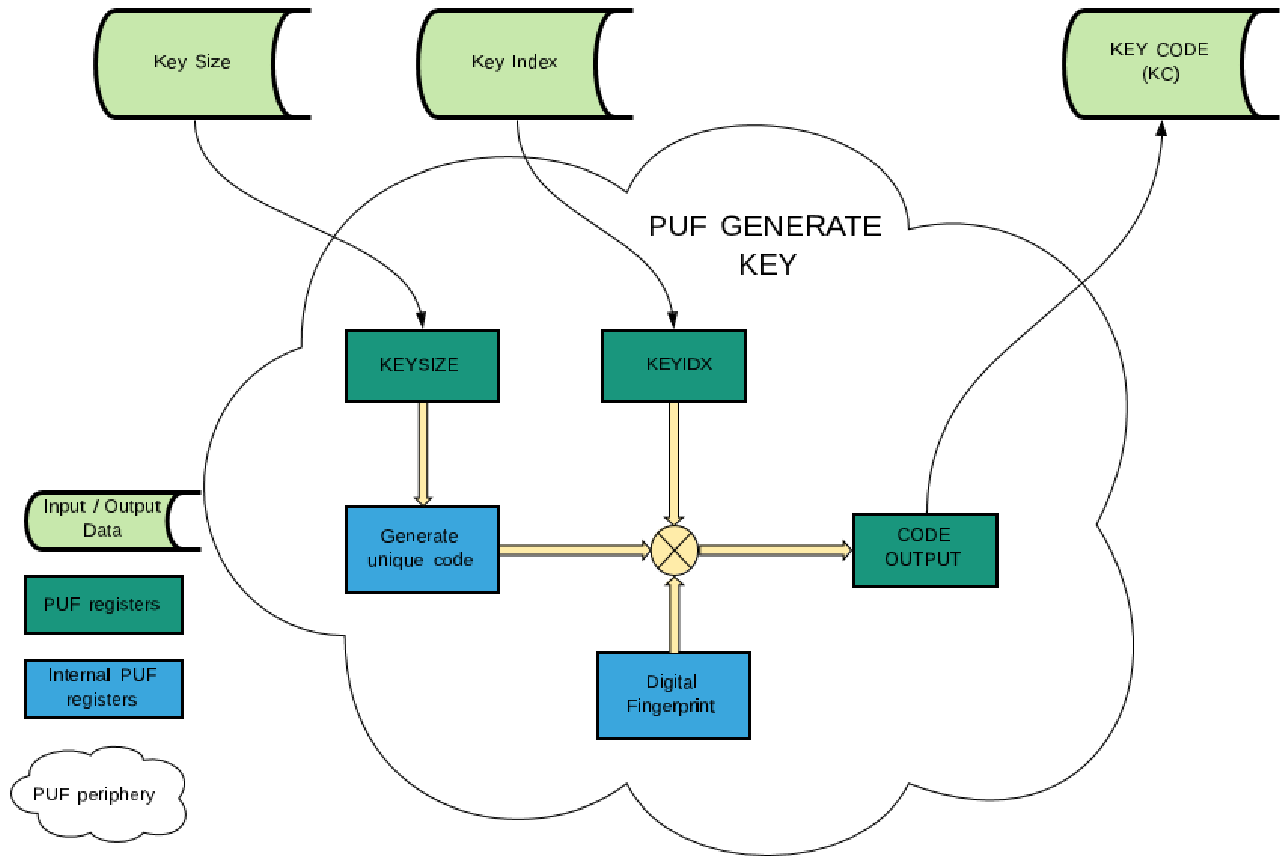


Figure 5. PUF GenerateKey command—generating key code

2.8 PUF GetKey

The GetKey command is used to get a key out of a key code. The key code is input into the CODEINPUT register after the PUF is started with a corresponding activation code (the GetKey command is not available after the Enroll command).

After combining the key code with the digital fingerprint, the key is retrieved. Depending on the value of the key index in the key code header, the key is sent to the internal secret hardware bus (key index = 0) or the CODEOUTPUT register (key index > 0). The user code must read out the key from this register, as well as the key index from the KEYIDX register.

The internal hardware bus may feed the key to hardware encryption units AES, PRINCE1, PRINCE2, or PRINCE3 (depending on the setting of the KEYENABLE register).

The KEYMASK[0..3] register must be loaded with a random number. It is used to obscure the key value stored in the key hold registers. A random value must be loaded into this register. This protects from the side channel analysis.

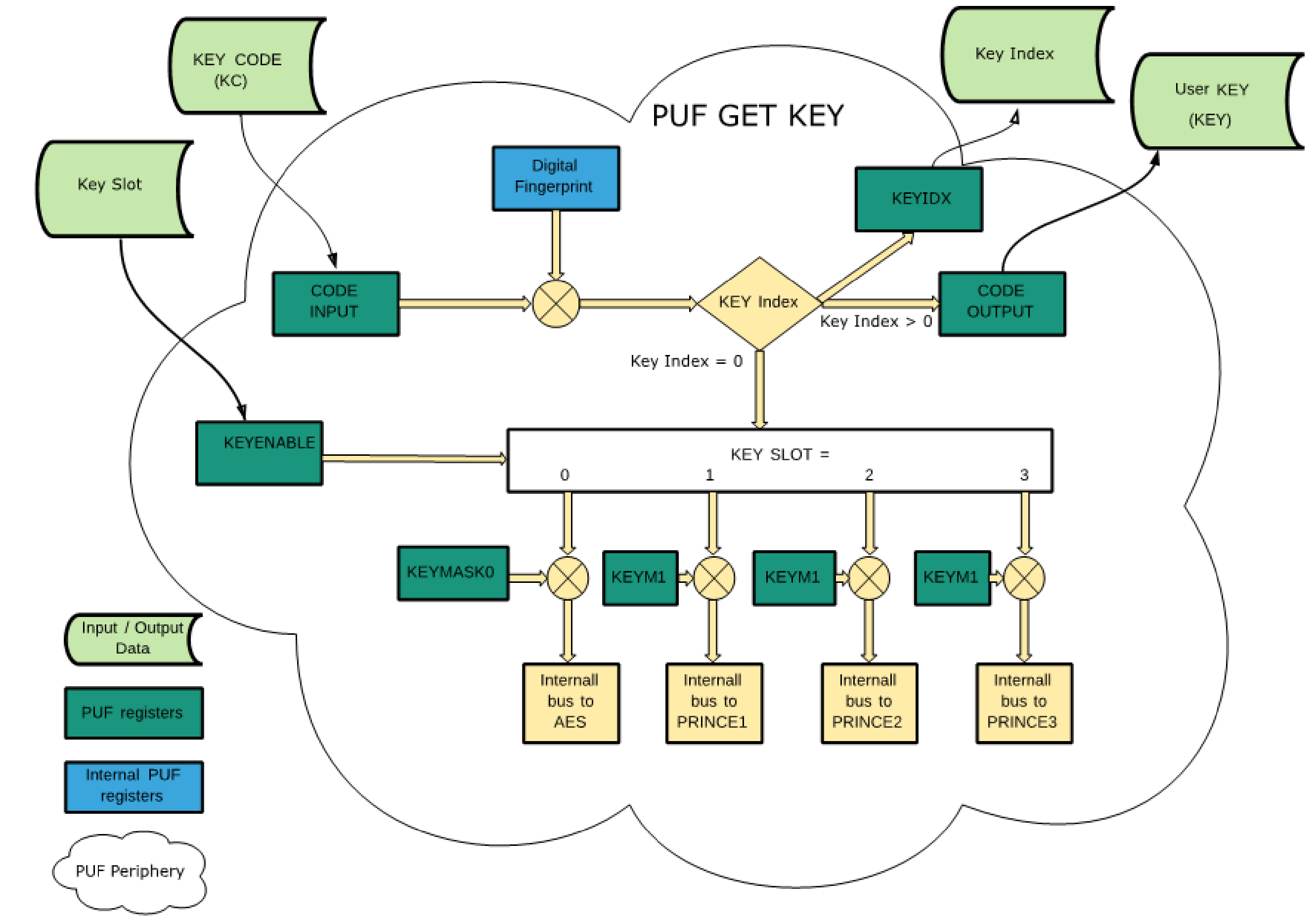


Figure 6. PUF GetKey command to get a key out of a key code

2.9 PUF Zeroize

When the Zeroize command is called, all the internal, critical, and security parameters are erased and the PUF controller goes to the error state. No new operations can be performed until the device is re-powered.

2.10 PUF DisableEnroll

Calling this command causes the PUF periphery to block the enroll operation.

2.11 PUF DisableSetKey

This command blocks the key output data. If the bit is set to 1, the PUF periphery blocks the key output (keyout data = 0) when the key code index is equal to 15. If the bit is set to 0, the key output is not blocked (even if key output index = 15).

3 Key management

3.1 Key storage in the Protected Flash Region (PFR)

Physically Unclonable Feature (PUF) on page 1 shows that it is possible to securely store a key code into the MCU flash memory (or any other external memory), because the master key is not physically accessible on the MCU. The place to store the key codes is in the Protected Flash Region (PFR) on the MCU.

The PFR is protected from being accessed from the bus and has three areas:

- CFPA (Customer Field Programmable Area) holds the monotonic counters, key revocation, and PRINCE IV codes.
- CMPA (Customer Manufacturing floor Programable Area) holds the boot configurations, security policies, user-defined data, and PUF key storage.

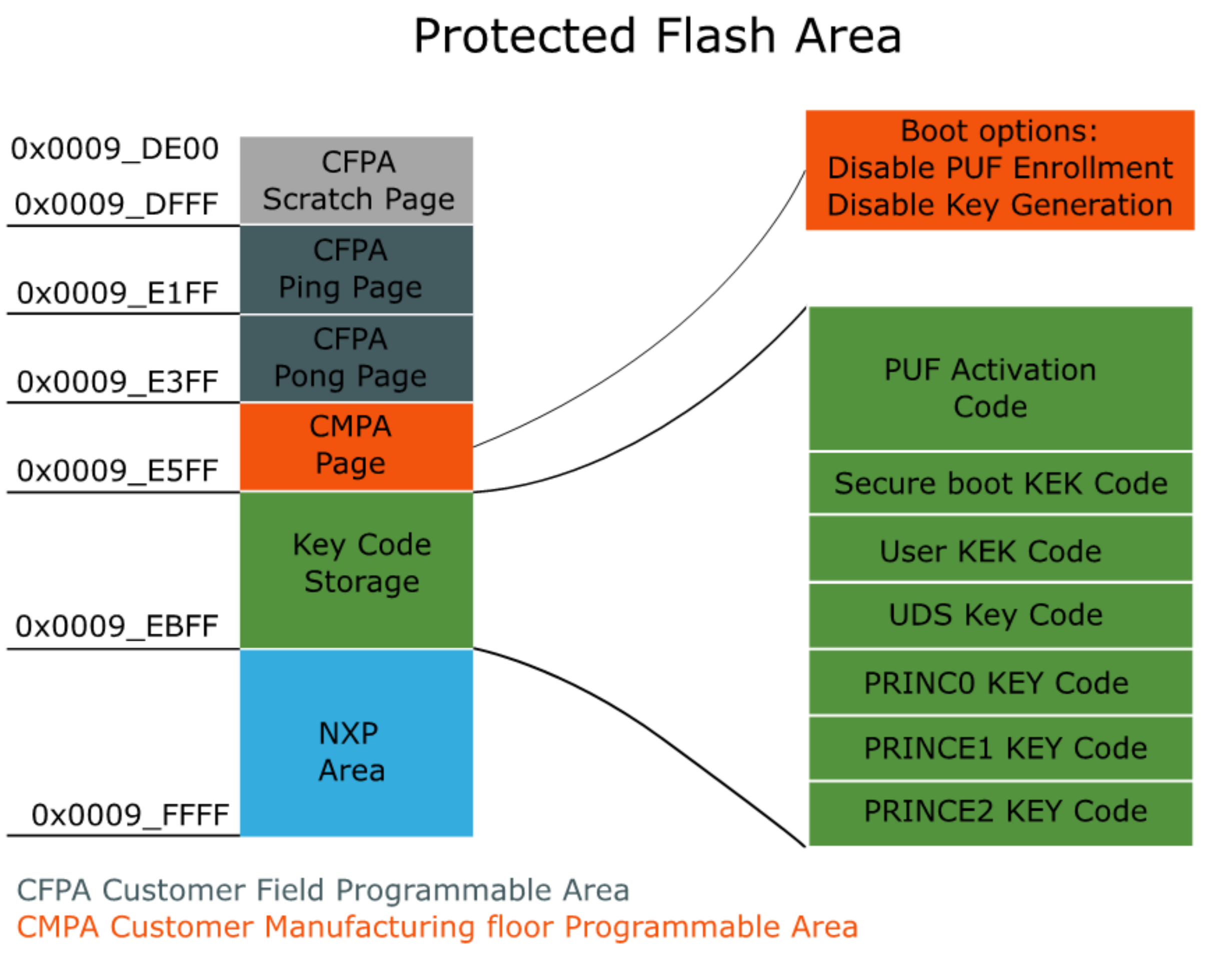


Figure 7. Keys and PUF boot option in the PFR

NOTE

The key storage is used and managed mainly by the boot code.

The secure boot uses the secure-boot KEK code. The flash encryption by PRINCE uses the PRINCE1 to PRINCE3 key codes for its protected regions. When correct key codes are found, they are applied to PRINCE by the boot process. The boot process checks the key code storage for existing code validity. If the AC (or KC) is corrupt, booting stops and blocks the MCU forever.

This region is also protected against data corruption or modification. When locked by the ROM, all regions are protected from the erase/write commands. Each region has a SHA256 hash digest field used in the deployment lifecycle state to cross-check the integrity of the region.

Figure 7. on page 8 shows the positions for key codes in the PFR memory. There is enough place for the activation code and five key codes—secure boot key code, user key code (can be used for AES), and three PRINCE key codes. One position is secured for the Unique Device Secret (UDS). Each key code position has the maximum size limited to 52 bytes, which gives the maximum key size of 256 bits. This may seem low, but the internal encrypting engines accept a maximum key size of 256 bits. The AES maximum allowed key size is 128/196/256 bits. The maximum PRINCE allowed key size is 128 bits.

From the PUF periphery use point of view, the important part is also placed in the CMPA page. There are the "PUF Enrollment Disable" and "PUF Code Generation Disable" options. These options are copied to the PUF periphery during the boot process. Remember that these settings may be changed only after a power cycle. If the options are loaded during the boot process, the PUF may never be enrolled or a new key code may never be generated after that in the user code. However, you may read out the existing activation code and key codes from the PFR and use them.

The CFPA region may be updated through the ROM API—both ISP and IAP. The “In System Programming” function supports the key-provisioning commands to generate the key codes and store them into the PFR. Another option is to use the flash IAP commands to manage the key storage area in the PFR from the user code.

The following sections describe the IAP and the ISP commands dedicated to work with the key storage area.

3.2 ISP KeyProvision commands

The “In System Programming (ISP) is a bunch of functions to support image programming through serial interfaces (UART, I^2C, SPI, USB HID). There is a security-related command (KeyProvision) to install the pre-shared keys, generate random keys, and save them into the Protected Flash Region (PFR) key store.

There are three possible parameters for the KeyProvision command: key operation, key type, and key size.

- Key operation—required to specify the KeyProvision command behavior and may have these values: Enroll, SetUserKey, SetIntrinsicKey, WriteNonVolatile, ReadNonVolatile, WriteKeyStore, ReadKeyStore.
- Key type—this parameter defines what the generated key code will be used for.

Key Type	**Value**
UserKEK	11
SBKEK	3
PRINCE0	7
PRINCE1	8
PRINCE2	9
UDS	12

- Key size—The key size parameter is defined in bytes.

3.3 Configure key store area by the blhost PC application

As a part of the SDK package, there is a command-line application called blhost. The blhost application is used on a host computer to issue commands to an MCU running an implementation of the MCU bootloader. The blhost application, in conjunction with the

MCU bootloader, allows to program a firmware application onto the MCU device without a programming tool. The blhost application supports also several key-provisioning-related commands described in these examples:

- Enroll the PUF—this command calls a ROM code that configures, starts, and enrolls the PUF periphery:

  ```
  .\blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning enroll
  ```

- Generate UDS-—this command calls the ROM code that uses the PUF to generate an intrinsic 32-byte key and its key code and stores them to the CMPA key code storage [UDS key code] position (RAM version):

  ```
  .\blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning set_key 12 32
  ```

- Generate user Secure Bootloader Code SBKEK—this command calls the ROM code that generates a key code based on the key stored in the *tempSbkek.bin* file and stores it to the CMPA key code storage [secure boot KEK] position (RAM version), where tempSbkek.bin is a plain text file with the SB KEK:

  ```
  .\blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning set_user_key 3 ".\temp\tempSbkek.bin"
  ```

- Generate User Key Code SBKEK—this command sets the user key code and stores it to the CMPA key code storage [user key code SB KEK] position (RAM version), where *tempSbkek.bin* is a plain-text file with the SB KEK:

  ```
  \blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning set_user_key 11 ".\temp\UserKek.bin"
  ```

- Generate intrinsic key code for PRINCE0-—this command generates a 128-bit intrinsic key code for PRINCE0 and stores it to the CMPA key code storage [PRINCE0 key code] (RAM version):

  ```
  .\blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning set_key 7 16
  ```

- Generate intrinsic key code for PRINCE1—this command generates a 128-bit intrinsic key code for PRINCE1 and stores it to the CMPA key code storage [PRINCE1 key code] (RAM version):

  ```
  .\blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning set_key 8 16
  ```

- Generate 16-byte intrinsic key code for PRINCE 2—this command generates a 128-bit intrinsic key code for PRINCE2 and stores it to the CMPA key code storage [PRINCE2 key code] (RAM version):p

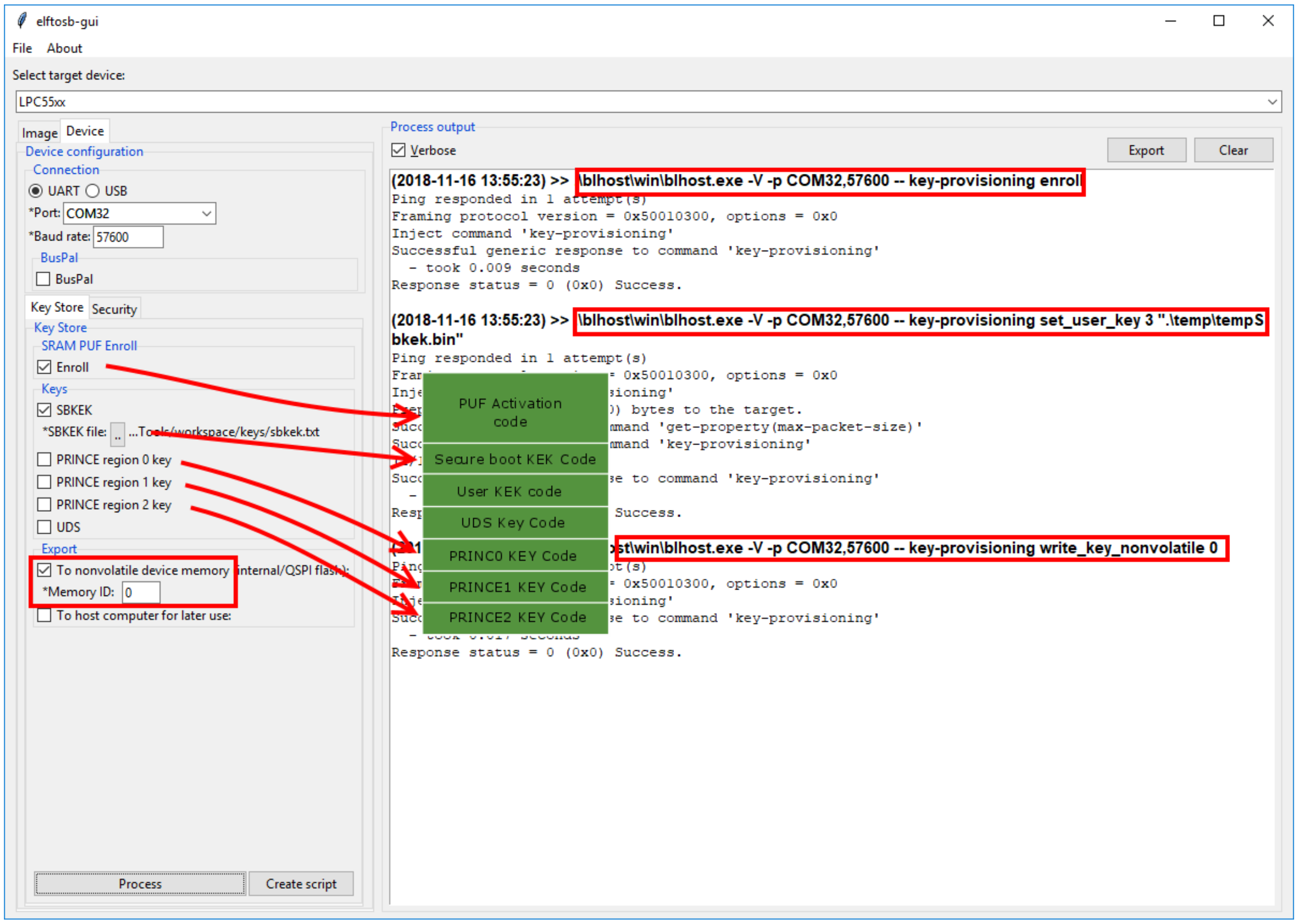
  ```
  .\blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning set_key 9 16
  ```

- This command is used to update the key storage in the PFR with the RAM version of key codes created in the previous operation (including the activation code created during the enroll operation):

  ```
  .\blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning write_key_nonvolatile 0
  ```

3.4 Key provisioning through the blhost—elftosb-gui tool

There is a window-based PC tool called elftosb-gui, built on the command-line blhost application, which helps to generate an image and adds all the settings and encryption needed. The tool can be used to tell the PUF to enroll and store an activation code, generate a key/key code, and store it to the PFR. At this time, the tool does not support the generation of a user KEK code.



Select target device: LPC55xx

Device configuration

Connection: [✓ UART] USB

*Port: COM32

*Baud rate: 57600

BusPal: ☐ BusPal

Key Store

SRAM PUF Enroll: ☑ Enroll

Keys:
- ☑ SBKEK
- *SBKEK file: ...Tools/workspace/keys/sbkek.txt
- ☐ PRINCE region 0 key
- ☐ PRINCE region 1 key
- ☐ PRINCE region 2 key
- ☐ UDS

Export:
- ☑ To nonvolatile device memory (internal/QSPI flash):
- *Memory ID: 0
- ☐ To host computer for later use:

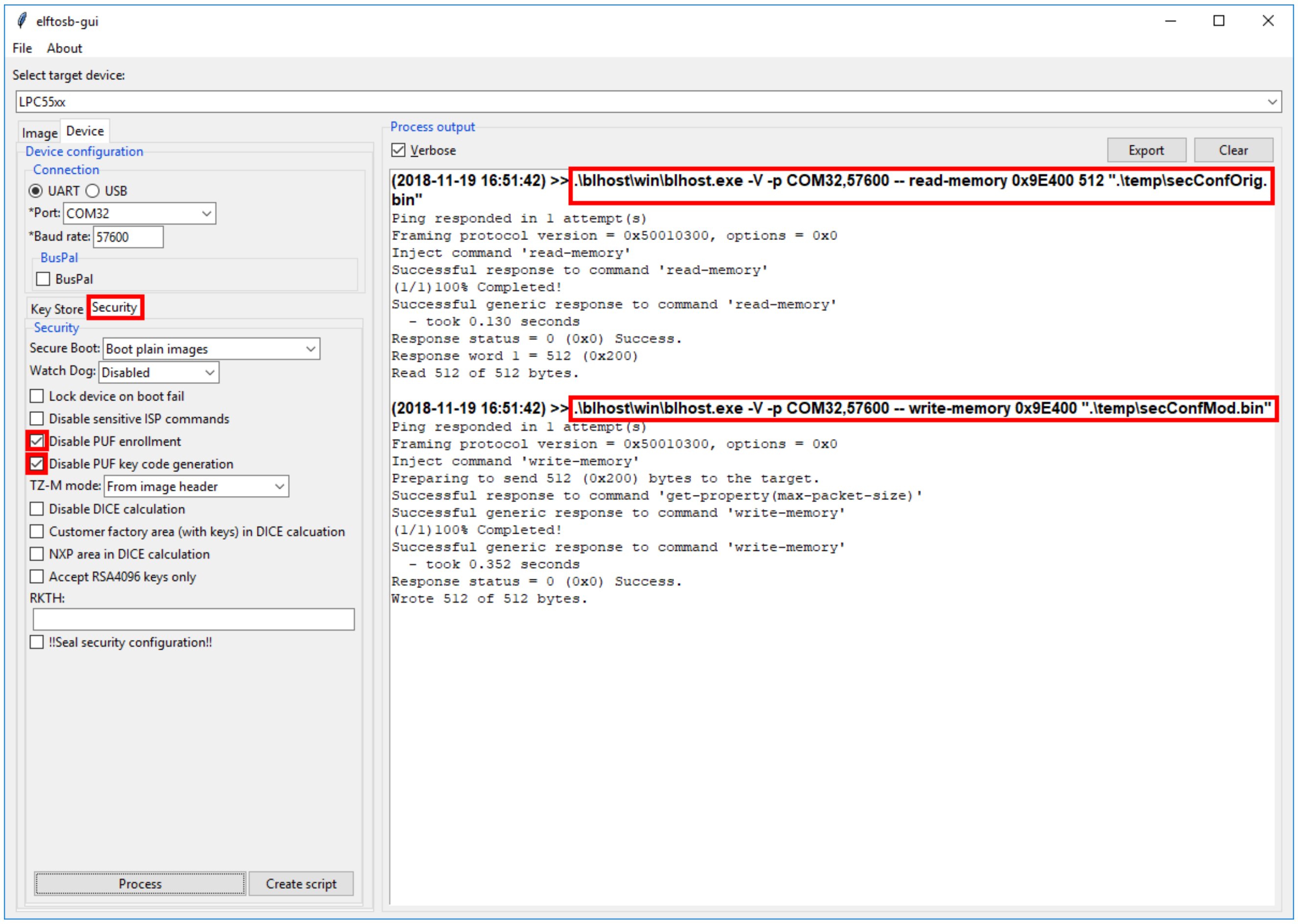
Process output

☑ Verbose

```
(2018-11-16 13:55:23) >> \blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning enroll
Ping responded in 1 attempt(s)
Framing protocol version = 0x50010300, options = 0x0
Inject command 'key-provisioning'
Successful generic response to command 'key-provisioning'
  - took 0.009 seconds
Response status = 0 (0x0) Success.

(2018-11-16 13:55:23) >> \blhost\win\blhost.exe -V -p COM32,57600 -- key-provisioning set_user_key 3 ".\temp\tempS
bkek.bin"
Ping responded in 1 attempt(s)
Fra                      = 0x50010300, options = 0x0
Inje                     sioning'
                         )) bytes to the target.
Succ                     mmand 'get-property(max-packet-size)'
Succ                     mmand 'key-provisioning'

Succ                     se to command 'key-provisioning'
  -
Resp                     Success.

                         st\win\blhost.exe -V -p COM32,57600 -- key-provisioning write_key_nonvolatile 0
Ping                     pt(s)
                         = 0x50010300, options = 0x0
                         sioning'
Succ                     se to command 'key-provisioning'
  -
Response status = 0 (0x0) Success.
```

Figure 8. Key provisioning through the elftosb-gui tool

The elftosb tool can also set the security options of the PUF, disable the Enroll command, and disable the SetKey command.


Select target device: LPC55xx

Device configuration — Connection: [✓ UART] USB; Port: COM32; Baud rate: 57600; BusPal: ☐ BusPal

Security:
- Secure Boot: Boot plain images
- Watch Dog: Disabled
- ☐ Lock device on boot fail
- ☐ Disable sensitive ISP commands
- ☑ Disable PUF enrollment
- ☑ Disable PUF key code generation
- TZ-M mode: From image header
- ☐ Disable DICE calculation
- ☐ Customer factory area (with keys) in DICE calcuation
- ☐ NXP area in DICE calculation
- ☐ Accept RSA4096 keys only
- RKTH:
- ☐ !!Seal security configuration!!

Process output (☑ Verbose):

```
(2018-11-19 16:51:42) >> .\blhost\win\blhost.exe -V -p COM32,57600 -- read-memory 0x9E400 512 ".\temp\secConfOrig.
bin"
Ping responded in 1 attempt(s)
Framing protocol version = 0x50010300, options = 0x0
Inject command 'read-memory'
Successful response to command 'read-memory'
(1/1)100% Completed!
Successful generic response to command 'read-memory'
  - took 0.130 seconds
Response status = 0 (0x0) Success.
Response word 1 = 512 (0x200)
Read 512 of 512 bytes.

(2018-11-19 16:51:42) >> .\blhost\win\blhost.exe -V -p COM32,57600 -- write-memory 0x9E400 ".\temp\secConfMod.bin"
Ping responded in 1 attempt(s)
Framing protocol version = 0x50010300, options = 0x0
Inject command 'write-memory'
Preparing to send 512 (0x200) bytes to the target.
Successful response to command 'get-property(max-packet-size)'
Successful generic response to command 'write-memory'
(1/1)100% Completed!
Successful generic response to command 'write-memory'
  - took 0.352 seconds
Response status = 0 (0x0) Success.
Wrote 512 of 512 bytes.
```

Figure 9. Various security settings supported by the elftosb tool

Both PC applications (elftosb-gui and blhost) are in the MCU SDK at this path: <SDK_ver\middleware\mcu-boot\bin\Tools>.

3.5 Key management IAP commands

You can program the key codes into the key storage area of the PFR in the MCU by the user application. There are several FRR IAP ROM commands dedicated to read the key codes and the activation code and to store a block of data into the key storage area.

To use the key codes stored in the PFR key storage section, start the PUF with the activation code that was used to create the key codes.

- The `ffr_keystore_get_ac` returns the AC from the key storage:

  ```
  status_t ffr_keystore_get_ac (flash_config_t *config, uint8_t* pActivationCode)
  ```

`pActivationCode` is a pointer to a buffer long enough to hold a 1192-byte activation code.

The AC must be passed to the PUF using the PUF Start command.

- `ffr_keystore_get_kc` is used to load the key code from the PFR:

  ```
  status_t ffr_keystore_get_kc (flash_config_t *config, uint8_t* pKeyCode, ffr_key_type_t keyIndex);
  ```

  `pKeyCode` is the buffer into which the key code is copied.

The key types are specified in this table:

Table 1. Key types

Key type	Value
SB KEK	0
User KEK	1
UDS	2
PRINCE0	3
PRINCE1	4
PRINCE2	5

There are also commands to read the PFR non-volatile memory, such as `ffr_keystore_write`, `ffr_infield_page_writem`, and `ffr_get_customer_infield_data`.

For more details, see the user manual.

4 Usage of the PUF testing software

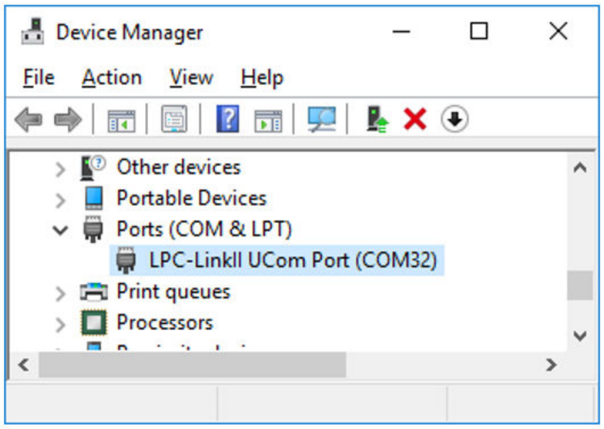
The sw_an12324.zip software package comes together with this application note. This software enables you to perform basic PUF commands (Enroll, Init, Start, Stop, SetKey, GetKey) and security-enhancing commands (DisableEnroll, DisableSetKey). The software also enables you to get the key code and provide it to the AES module through a standard memory bus or a secret bus between the PUF and the hash crypt engine. You can also encrypt/decrypt a 16-byte data block with the key provided. The example code is based on the SDK and uses the SDK drivers.

The software is controlled via a serial line terminal and uses the on-board built-in virtual serial COM port (LPC-LinkII UCom Port). Use any ASCII-based serial COM port terminal application, such as Teraterm or Putty. The terminal must have the local echo enabled.

4.1 Setting up the PUF example code

Open the application note software package, compile it, and run it on the LPCXpresso5500 board. Connect the LPCXpresso5500 board to the PC via connector P9. The LPCLinkII CMSIS DAP and the virtual serial COM port LPC-Link UCOM appear in the Windows® OS Device Manager.

Use the terminal to open a correct COM port and set the parameters to 115200 bps, bits, no parity, no hardware control. After a reset, an initial menu appears in the console.



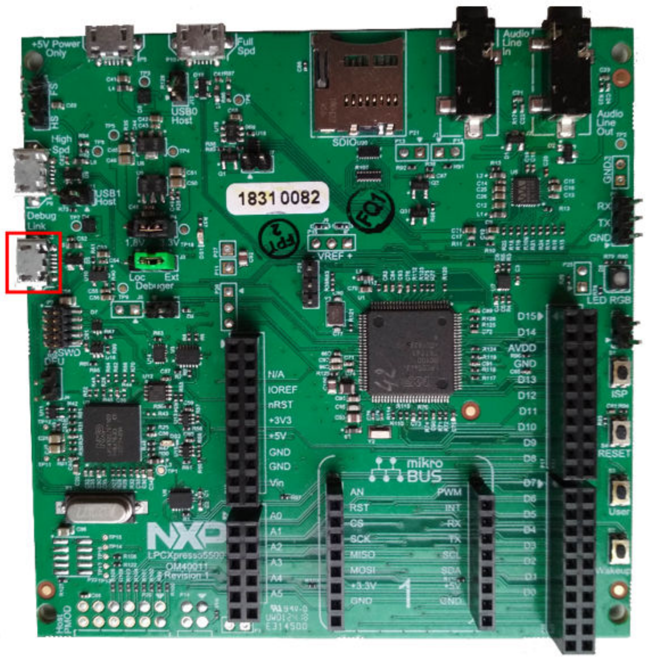


Figure 10. Virtual serial COM port and board

4.2 Application

When the application is running, a menu is printed out in the terminal window. On the screen, the PUF status and the PUF allowed operation are always shown.

The initial status shown in the console is out of reset, so the PUF periphery is not started and no commands are allowed. However, the PUF may be started from the boot code, depending on the CMPA key storage area and the "Disable PUF enrollment" and "Disable PUF key code generation" boot options.

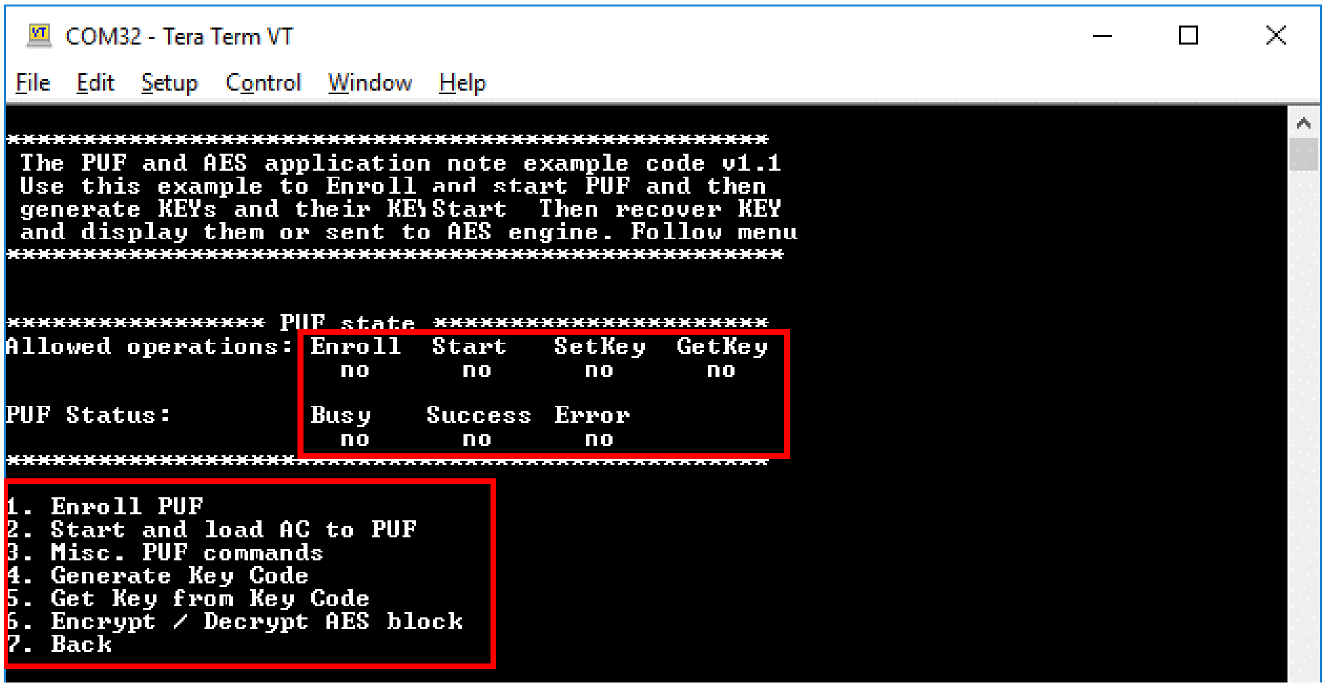
The PUF periphery must be initialized before it is run. During the initialization, the SRAM memory is re-powered and prepared to be read.

The PUF status means the following:

- Busy—an operation is in progress.
- Success—the last operation was successful.
- Error—the PUF is in the error state and no operations can be performed.

At the start of the PUF-allowed operation, all four status bits mean the following:

- Enroll—the Enroll operation is allowed.
- Start—the Start operation is allowed.
- SetKey—the SetKey operations are allowed.
- GetKey—the GetKey operation is allowed.



```
***************************************************
 The PUF and AES application note example code v1.1
 Use this example to Enroll and start PUF and then
 generate KEYs and their KE\Start  Then recover KEY
 and display them or sent to AES engine. Follow menu
****************************************************


***************** PUF state **********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      no      no

PUF Status:         Busy    Success Error
                      no      no      no
***************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
```

Figure 11. Application main screen

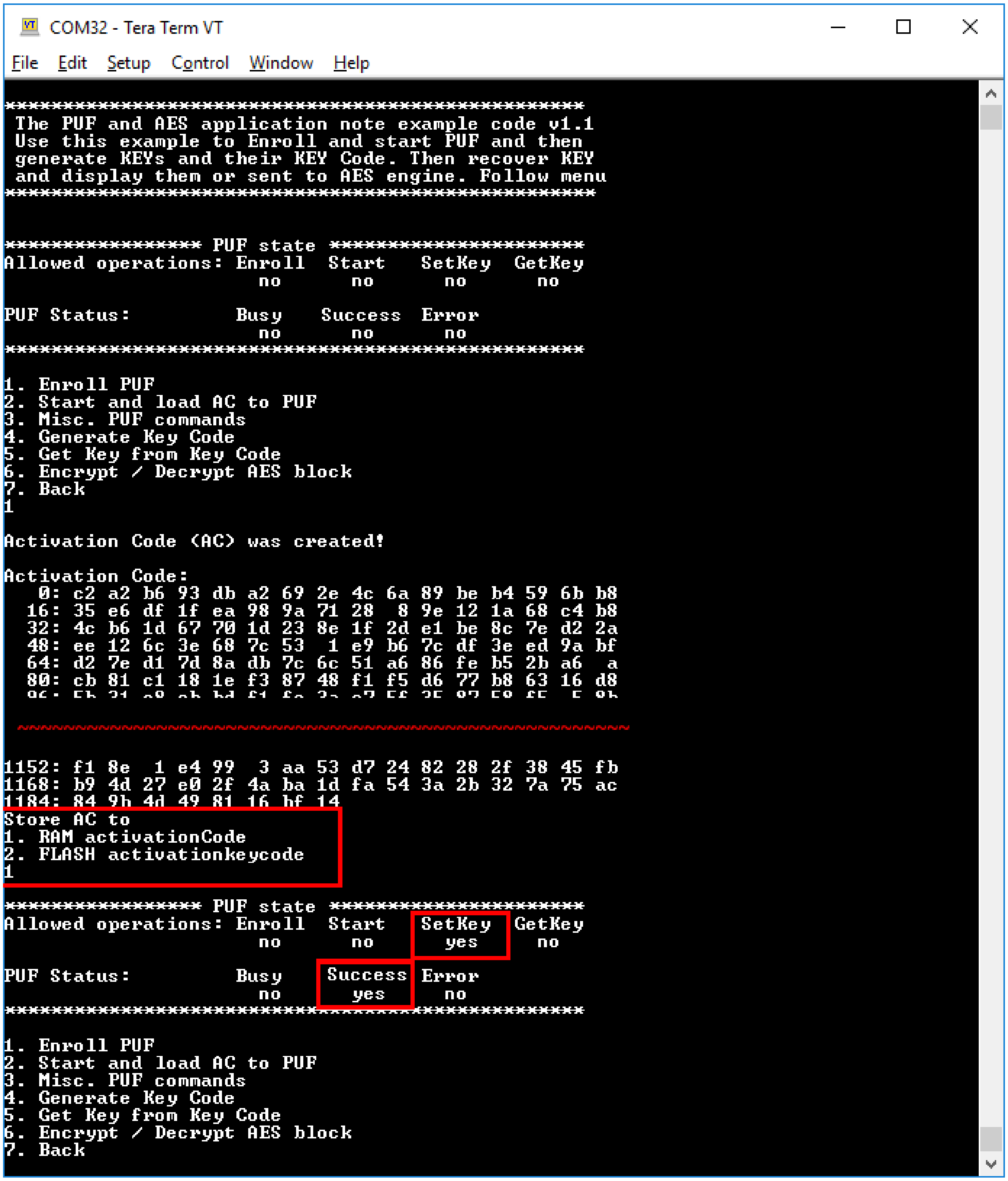
4.3 Enroll PUF

The first step to start using the PUF is to enroll. During the enrolling, an activation code is created. The enroll operation firstly powers up the PUF periphery, initializes it (Start), and then calls the Enroll command. The activation code is created and can be stored to the RAM or flash memory for further use.

NOTE

After a successful Enroll command, only the SetKey command is allowed.

To call the Enroll command again, the PUF periphery must be power-cycled and initialized again. To enable the GetKey function, the PUF periphery must be power-cycled and started again. The AC can be stored into the RAM or flash memory for further usage.



COM32 - Tera Term VT

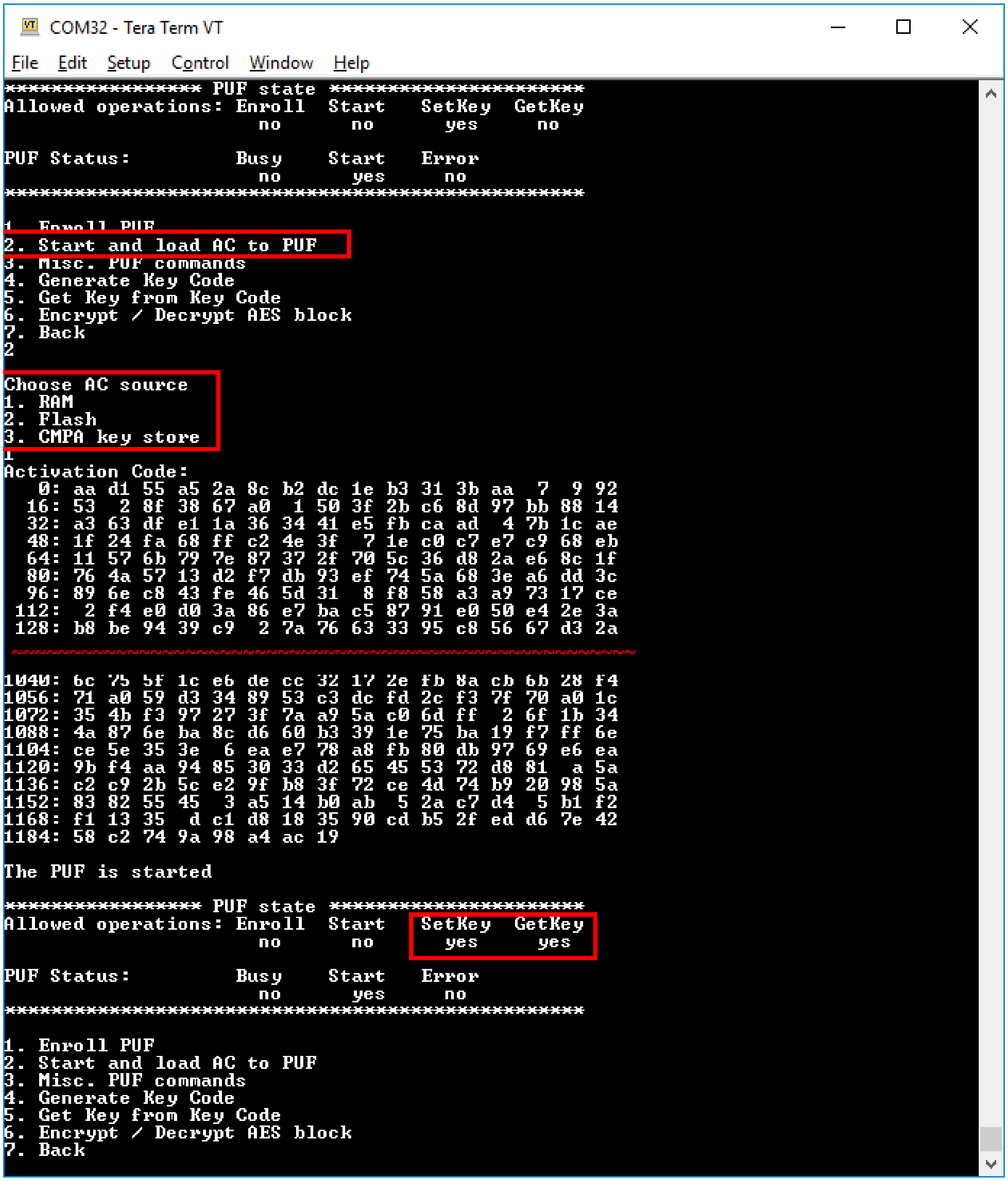
File Edit Setup Control Window Help

```
**************************************************
 The PUF and AES application note example code v1.1
 Use this example to Enroll and start PUF and then
 generate KEYs and their KEY Code. Then recover KEY
 and display them or sent to AES engine. Follow menu
***************************************************

***************** PUF state **********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      no      no

PUF Status:         Busy   Success  Error
                      no      no      no
**************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
1

Activation Code (AC) was created!

Activation Code:
   0: c2 a2 b6 93 db a2 69 2e 4c 6a 89 be b4 59 6b b8
  16: 35 e6 df 1f ea 98 9a 71 28  8 9e 12 1a 68 c4 b8
  32: 4c b6 1d 67 70 1d 23 8e 1f 2d e1 be 8c 7e d2 2a
  48: ee 12 6c 3e 68 7c 53  1 e9 b6 7c df 3e ed 9a bf
  64: d2 7e d1 7d 8a db 7c 6c 51 a6 86 fe b5 2b a6  a
  80: cb 81 c1 18 1e f3 87 48 f1 f5 d6 77 b8 63 16 d8
  [illegible]

1152: f1 8e  1 e4 99  3 aa 53 d7 24 82 28 2f 38 45 fb
1168: b9 4d 27 e0 2f 4a ba 1d fa 54 3a 2b 32 7a 75 ac
1184: 84 9b 4d 49 81 16 bf 14
Store AC to
1. RAM activationCode
2. FLASH activationkeycode
1

***************** PUF state **********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      yes     no

PUF Status:         Busy   Success  Error
                      no      yes     no
**************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
```

Figure 12. Enroll operation

4.4 Start and load AC to PUF command

The Start and Load AC to PUF command starts the PUF and loads one of the stored activation codes. The activation code must match the key code generated by the same AC. Otherwise, the GetKey command fails. The activation code can be loaded from the RAM or flash memory (stored in the previous step) or from the CMPA position (created and stored by blhost).



```
***************** PUF state **********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      yes     no

PUF Status:         Busy    Start   Error
                      no      yes     no
**************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
2

Choose AC source
1. RAM
2. Flash
3. CMPA key store
1
Activation Code:
   0: aa d1 55 a5 2a 8c b2 dc 1e b3 31 3b aa  7  9 92
  16: 53  2 8f 38 67 a0  1 50 3f 2b c6 8d 97 bb 88 14
  32: a3 63 df e1 1a 36 34 41 e5 fb ca ad  4 7b 1c ae
  48: 1f 24 fa 68 ff c2 4e 3f  7 1e c0 c7 e7 c9 68 eb
  64: 11 57 6b 79 7e 87 37 2f 70 5c 36 d8 2a e6 8c 1f
  80: 76 4a 57 13 d2 f7 db 93 ef 74 5a 68 3e a6 dd 3c
  96: 89 6e c8 43 fe 46 5d 31  8 f8 58 a3 a9 73 17 ce
 112:  2 f4 e0 d0 3a 86 e7 ba c5 87 91 e0 50 e4 2e 3a
 128: b8 be 94 39 c9  2 7a 76 63 33 95 c8 56 67 d3 2a

1040: 6c 75 5f 1c e6 de cc 32 17 2e fb 8a cb 6b 28 f4
1056: 71 a0 59 d3 34 89 53 c3 dc fd 2c f3 7f 70 a0 1c
1072: 35 4b f3 97 27 3f 7a a9 5a c0 6d ff  2 6f 1b 34
1088: 4a 87 6e ba 8c d6 60 b3 39 1e 75 ba 19 f7 ff 6e
1104: ce 5e 35 3e  6 ea e7 78 a8 fb 80 db 97 69 e6 ea
1120: 9b f4 aa 94 85 30 33 d2 65 45 53 72 d8 81  a 5a
1136: c2 c9 2b 5c e2 9f b8 3f 72 ce 4d 74 b9 20 98 5a
1152: 83 82 55 45  3 a5 14 b0 ab  5 2a c7 d4  5 b1 f2
1168: f1 13 35  d c1 d8 18 35 90 cd b5 2f ed d6 7e 42
1184: 58 c2 74 9a 98 a4 ac 19

The PUF is started

***************** PUF state **********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      yes     yes

PUF Status:         Busy    Start   Error
                      no      yes     no
**************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
```

Figure 13. Start and Load AC to PUF command

4.5 Generate Key Code command

The Generate Key Code command enables you to create a user key or an intrinsic key.

- User key—a key is entered to the PUF and the key code is generated out of this key.
- Intrinsic key—a key is randomly generated inside the PUF and you get only the key code. If the intrinsic code has index 0, the key itself is hidden forever and cannot be read by the CPU, because the PUF never sends it to the AHB bus.

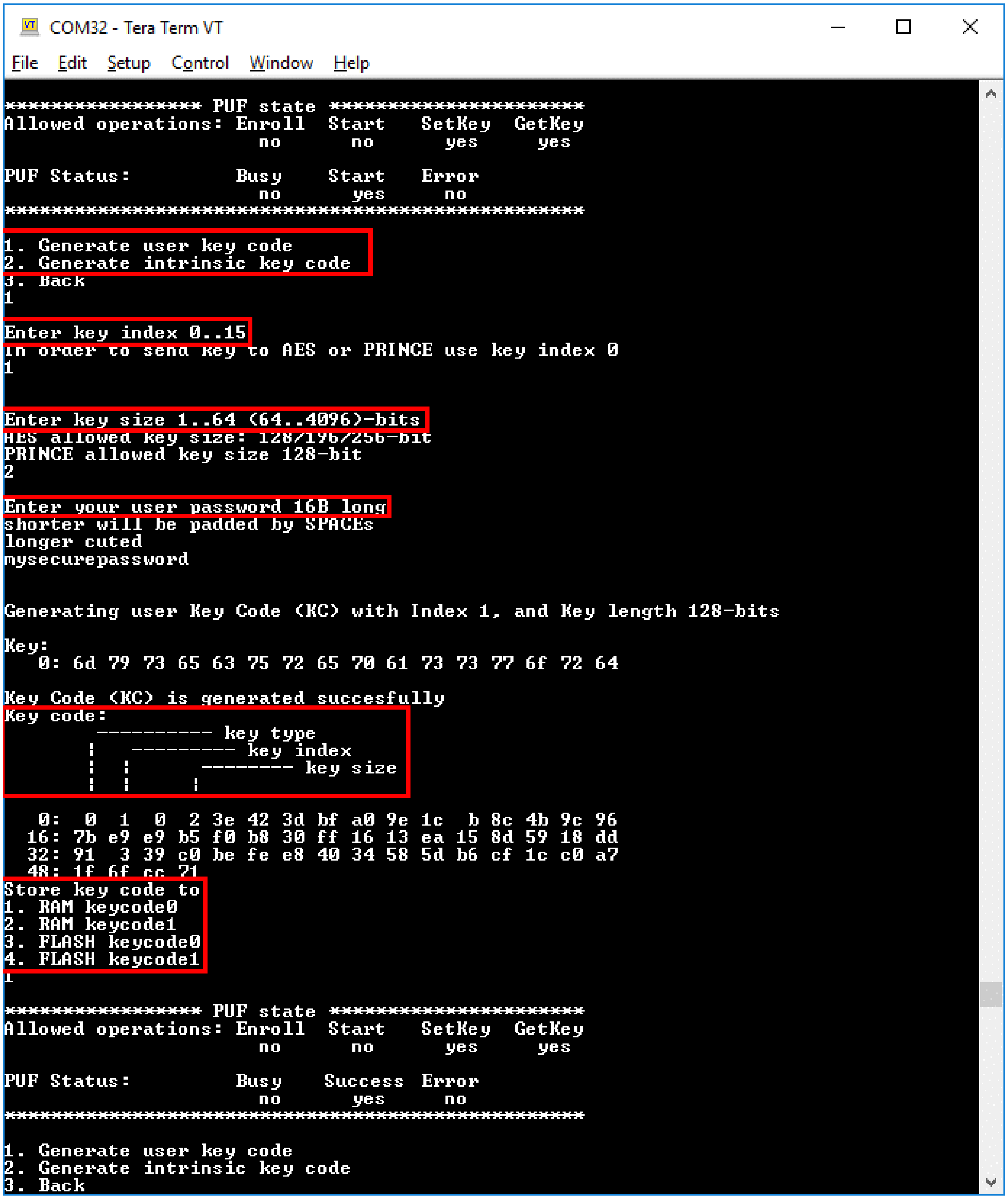
The next step is to set the key index in the interval from 0 to 15. A key index of 0 causes the key retrieved from the key code during the GetKey command to be sent to the internal hardware bus. A key index of 1 (and higher) sends the retrieved key to the AHB and it can be read by the CPU.

Enter the key size. The PUF supports key sizes from 64 to 4096 bits. However, the PRINCE engine supports only 128-bit keys and AES 128/196/256-bit keys.

The last step to generate the user key is to enter the user key size. The terminal supports only ASCII codes for simplicity reasons.

The entered parameters are sent to the PUF periphery and the key code is generated and printed out.

The generated key code can be stored in one of four available places, two key codes in the RAM, and two key codes in the flash memory for further use.



COM32 - Tera Term VT

File Edit Setup Control Window Help

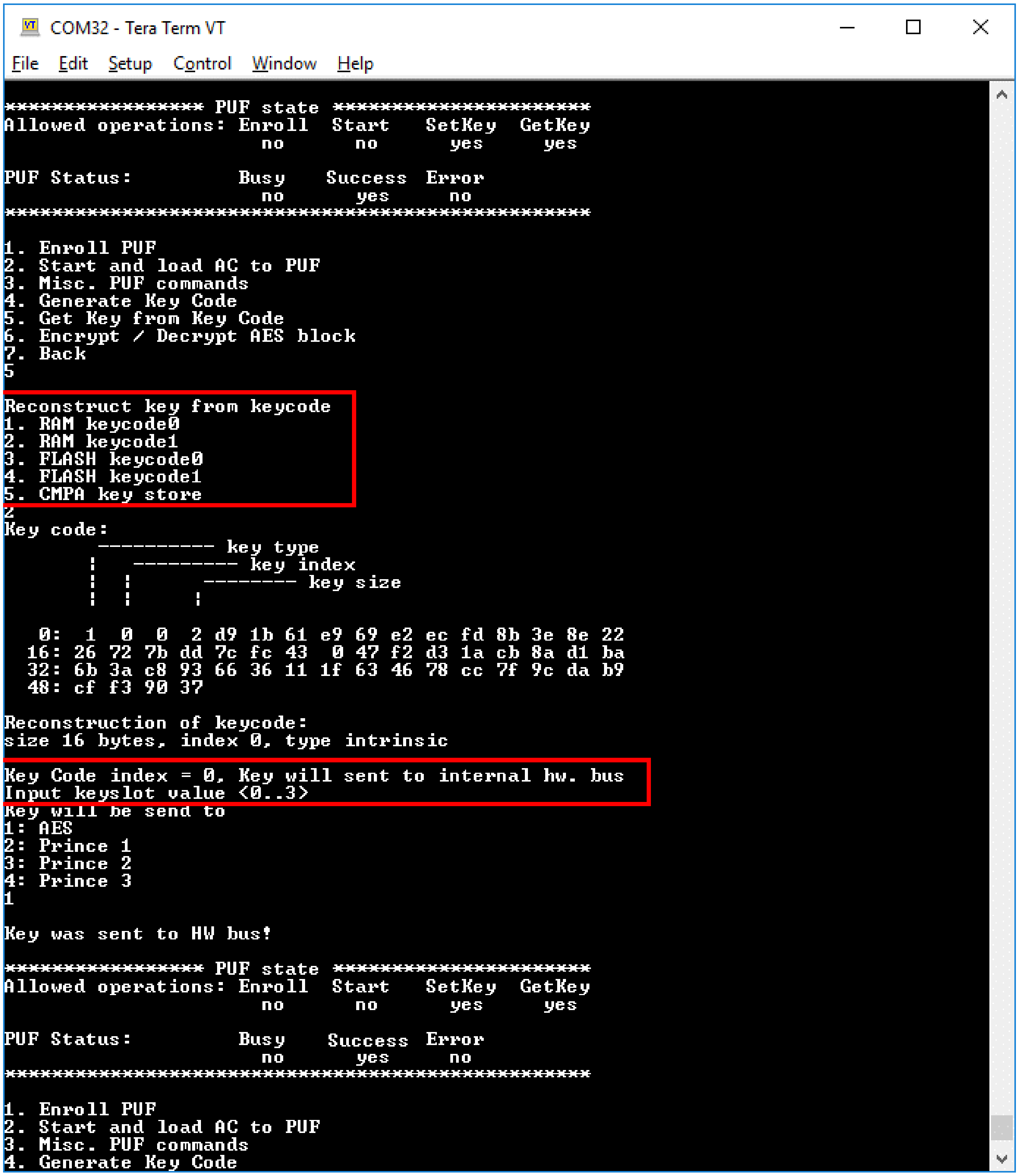
```
****************** PUF state ***********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no     no      yes     yes

PUF Status:          Busy    Start   Error
                      no      yes     no
****************************************************

1. Generate user key code
2. Generate intrinsic key code
3. Back
1

Enter key index 0..15
In order to send key to AES or PRINCE use key index 0
1


Enter key size 1..64 (64..4096)-bits
AES allowed key size: 128/196/256-bit
PRINCE allowed key size 128-bit
2

Enter your user password 16B long
shorter will be padded by SPACEs
longer cuted
mysecurepassword


Generating user Key Code (KC) with Index 1, and Key length 128-bits

Key:
   0: 6d 79 73 65 63 75 72 65 70 61 73 73 77 6f 72 64

Key Code (KC) is generated succesfully
Key code:
        ---------- key type
       |   --------- key index
       |  |      -------- key size
       |  |      |

   0:  0  1  0  2 3e 42 3d bf a0 9e 1c  b 8c 4b 9c 96
  16: 7b e9 e9 b5 f0 b8 30 ff 16 13 ea 15 8d 59 18 dd
  32: 91  3 39 c0 be fe e8 40 34 58 5d b6 cf 1c c0 a7
  48: 1f 6f cc 71
Store key code to
1. RAM keycode0
2. RAM keycode1
3. FLASH keycode0
4. FLASH keycode1
1

****************** PUF state ***********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no     no      yes     yes

PUF Status:          Busy   Success  Error
                      no      yes     no
****************************************************

1. Generate user key code
2. Generate intrinsic key code
3. Back
```

Figure 14. SetKey operation

4.6 GetKey command

Use the GetKey command to retrieve a key back from a key code. If a key code index is higher than 0, the key is reconstructed and printed out.

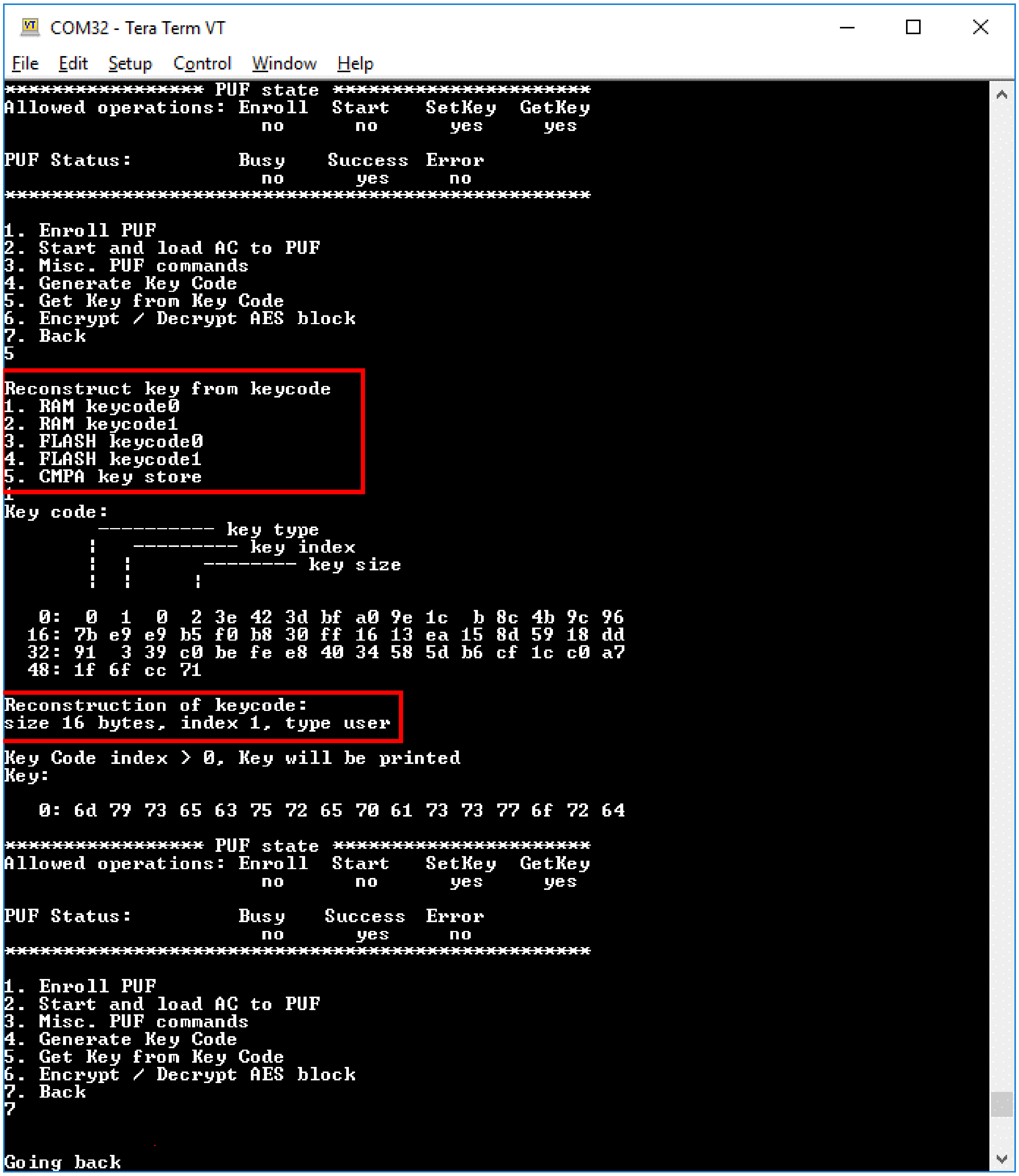
If a key code index equals 0, the key is sent to a secret internal hardware bus. This bus is connected to the AES and PRINCE engines. Enter a keyslot value of 1 to send the key to the AES engine. Values from 2 to 4 send the key to PRINCE1, PRINCE2, or PRINCE3. If the command is successful, the key is temporarily stored in the AES engine and can be used to encrypt/decrypt a 16-byte block of data.



```
****************** PUF state ***********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      yes     yes

PUF Status:         Busy    Success Error
                      no      yes     no
****************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
5

Reconstruct key from keycode
1. RAM keycode0
2. RAM keycode1
3. FLASH keycode0
4. FLASH keycode1
5. CMPA key store
2
Key code:
        ----------- key type
       |   ---------- key index
       |  |      -------- key size
       |  |     |

   0:  1  0  0  2 d9 1b 61 e9 69 e2 ec fd 8b 3e 8e 22
  16: 26 72 7b dd 7c fc 43  0 47 f2 d3 1a cb 8a d1 ba
  32: 6b 3a c8 93 66 36 11 1f 63 46 78 cc 7f 9c da b9
  48: cf f3 90 37

Reconstruction of keycode:
size 16 bytes, index 0, type intrinsic

Key Code index = 0, Key will sent to internal hw. bus
Input keyslot value <0..3>
Key will be send to
1: AES
2: Prince 1
3: Prince 2
4: Prince 3
1

Key was sent to HW bus!

****************** PUF state ***********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      yes     yes

PUF Status:         Busy    Success Error
                      no      yes     no
****************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
```

Figure 15. GetKey operation—internal hardware bus

For key codes with a key index higher than 0, the key is printed to the console.



COM32 - Tera Term VT

File Edit Setup Control Window Help

```
****************** PUF state ***********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      yes     yes

PUF Status:         Busy    Success  Error
                      no      yes     no
****************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
5

Reconstruct key from keycode
1. RAM keycode0
2. RAM keycode1
3. FLASH keycode0
4. FLASH keycode1
5. CMPA key store
1
Key code:
        ---------- key type
       |   ---------- key index
       |  |      --------- key size
       |  |      |

   0:  0  1  0  2 3e 42 3d bf a0 9e 1c  b 8c 4b 9c 96
  16: 7b e9 e9 b5 f0 b8 30 ff 16 13 ea 15 8d 59 18 dd
  32: 91  3 39 c0 be fe e8 40 34 58 5d b6 cf 1c c0 a7
  48: 1f 6f cc 71

Reconstruction of keycode:
size 16 bytes, index 1, type user

Key Code index > 0, Key will be printed
Key:

   0: 6d 79 73 65 63 75 72 65 70 61 73 73 77 6f 72 64

****************** PUF state ***********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      yes     yes

PUF Status:         Busy    Success  Error
                      no      yes     no
****************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
7


Going back
```

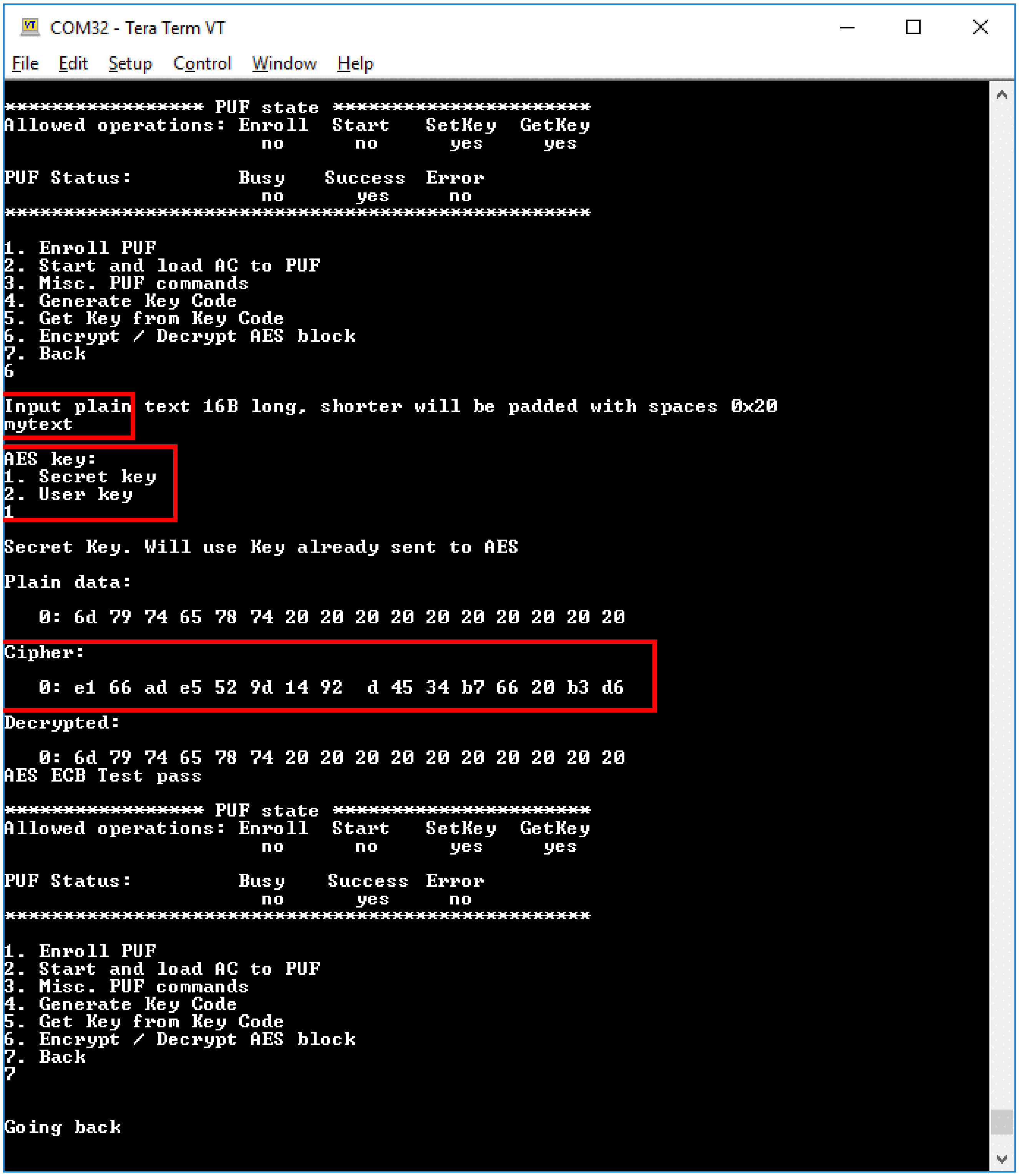
Figure 16. GetKey command—key is printed out

4.7 Using a key to encrypt a block of data

This command enables you to encrypt/decrypt a 16-byte block of data using a key reconstructed by the PUF.

You can choose between a secret key (sent to the AES through an internal secret hardware bus using the GetKey command) or a user key (generated by the PUF from one of the stored key codes).

Enter the plain text in ASCII. This block is then encrypted/decrypted using the provided key. AES uses the ECB mode.



```
****************** PUF state ***********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      yes     yes

PUF Status:         Busy    Success  Error
                      no      yes     no
****************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
6

Input plain text 16B long, shorter will be padded with spaces 0x20
mytext

AES key:
1. Secret key
2. User key
1

Secret Key. Will use Key already sent to AES

Plain data:

   0: 6d 79 74 65 78 74 20 20 20 20 20 20 20 20 20 20

Cipher:

   0: e1 66 ad e5 52 9d 14 92  d 45 34 b7 66 20 b3 d6

Decrypted:

   0: 6d 79 74 65 78 74 20 20 20 20 20 20 20 20 20 20
AES ECB Test pass

****************** PUF state ***********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no      yes     yes

PUF Status:         Busy    Success  Error
                      no      yes     no
****************************************************

1. Enroll PUF
2. Start and load AC to PUF
3. Misc. PUF commands
4. Generate Key Code
5. Get Key from Key Code
6. Encrypt / Decrypt AES block
7. Back
7


Going back
```

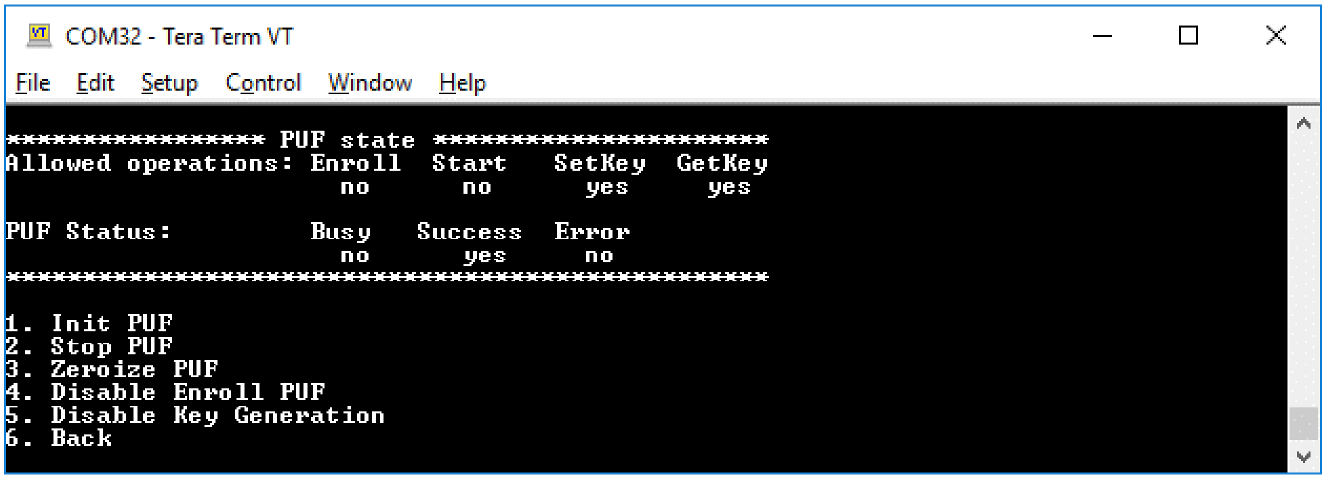
Figure 17. Encrypt/decrypt menu

4.8 Miscellaneous menu

The miscellaneous menu has these options:

- Init PUF.
- Stop PUF.
- Zeroize PUF.
- Disable Enroll PUF.
- Disable Key Generation.

The "Disable Enroll PUF" and "Disable Key Generation" settings can be cleared only by power-cycling the MCU.



```
****************** PUF state ***********************
Allowed operations: Enroll  Start   SetKey  GetKey
                      no      no     yes     yes

PUF Status:         Busy   Success  Error
                      no     yes     no
****************************************************

1. Init PUF
2. Stop PUF
3. Zeroize PUF
4. Disable Enroll PUF
5. Disable Key Generation
6. Back
```

Figure 18. Miscellaneous menu

Date of release: February 2019
Document identifier: AN12324