AN12324

LPC55Sxx 将 PUF 和 Hash Crypt 用于 AES 编码

版本 0 — 2019 年 2 月

应用笔记

目录

1 介绍.....1
2 物理上不可克隆功能（PUF）.....1
2.1 PUF 功能.....1
2.2 PUF 密钥的唯一性原理.....1
2.3 PUF 用法.....2
2.4 PUF 注册（Enroll）.....2
2.5 PUF 启动（Start）.....3
2.6 PUF 设置密钥（SetKey）.....4
2.7 PUF 生成密钥（GenerateKey）.....5
2.8 PUF 获得密钥（GetKey）.....6
2.9 PUF 归零（Zeroize）.....7
2.10 PUF DisableEnroll.....7
2.11 PUF DisableSetKey.....7
3 密钥管理.....7
3.1 受保护的闪存区域（PFR）中的密钥存储.....7
3.2 ISP KeyProvision 命令.....9
3.3 通过 blhost PC 应用程序配置密钥存储区.....9
3.4 通过 blhost-elftosb-gui 工具进行密钥配置.....10
3.5 密钥管理 IAP 命令.....12
4 PUF 测试软件的使用.....13
4.1 设置 PUF 示例代码.....13
4.2 应用.....14
4.3 注册 PUF.....15
4.4 启动并将 AC 加载到 PUF 的命令.....16
4.5 生成密钥代码命令.....17
4.6 Getkey 命令.....19
4.7 使用密钥加密数据块.....21
4.8 杂项菜单.....23

1 介绍

本应用笔记介绍了如何使用根密钥安全地生成、存储和恢复用户密钥。根密钥用作密钥加密密钥（KEK），以保护其他的用户密钥。根密钥是由物理不可克隆功能（PUF）外设生成的。PUF 技术会生成设备唯一的 256 位 KEK，这是设备的数字指纹。然后可以将加密的密钥（密钥代码）存储在 MCU 中。本文档讨论了密钥代码存储在受保护的闪存区域（PFR）的闪存中以及通过 ISP 和 IAP 命令向系统设置密钥的问题，还介绍了通信工具（blhost 和 elftosb）。

2 物理上不可克隆功能（PUF）

2.1 PUF 功能

- PUF 外设生成一个 256 位密钥。 密钥是一种数字指纹，它是设备唯一的，不可克隆的，并用作密钥加密密钥（KEK）。
- PUF 可以生成 64 位至 4096 位密钥。
- PUF 将 64 位至 4096 位的密钥加密为密钥代码（Key Code）。
- PUF 将密钥代码解密回密钥。
- 可以通过内部硬件总线将密钥发送到哈希加密引擎（或 PRINCE）。
- CPU 可以通过寄存器和 AHB 总线读取密钥。
- PUF 提供了其他防止黑客入侵的功能，例如阻止功能（注册、代码输出、密钥锁）或增强的测信道攻击（掩码（keymask））。

2.2 PUF 密钥的唯一性原理

PUF 生成的 KEK 的唯一性基于制造芯片时（长度、宽度、厚度）晶体管自然发生的变化。每次 SRAM 模块上电时，单元将变为 1 或 0。启动值创建一个随机且可重复的模式，这对于每个芯片都是唯一的。它称为 SRAM 启动数据（SD）。

SRAM SD 与激活码（AC）一起被转换为数字指纹，用于构建安全子系统基础的密钥。数字指纹用作根密钥（或 KEK）。

图 1. 基于 SRAM 单元阵列的 PUF 原理

PUF 模块嵌入了纠错功能，因此密钥重建失败的可能性小于 10^{-9}（在最坏的情况下）。

2.3 PUF 用法

任何系统的安全性都取决于密钥的存储安全性。 如果将密钥放在无法访问的闪存中，黑客可能会将芯片开盖并读取内存。 保险丝（Fuse）也是如此，开盖后即可读取。

加密密钥时，必须将密钥存储在某处。如果黑客读出了整个闪存，则他们可以克隆您的设备，因为存储的密钥在整个生产过程中都是相同的。如果黑客成功破解了您的密钥，则该密钥对所有设备均有效。而使用 PUF 可以解决这些问题。每个 MCU 都有自己独特的数字指纹，该数字指纹没有存储在芯片上，并且在设备未通电时也无法读取。以下各节说明如何使用 PUF 生成、存储和恢复密钥。

PUF 控制器功能包括以下命令：归零（Zeroize），注册（Enroll），启动（Start），设置密钥（ SetKey），生成密钥（GenerateKey）和获得密钥（GetKey）。 这些命令用于控制 PUF 密钥管理。

2.4 PUF 注册（Enroll）

开始使用 PUF 前，您需要进行注册。在注册期间，将 SRAM 启动数据导出为数字指纹，并生成相应的激活码（Activation Code）。激活码通过 CODEOUTPUT 寄存器读出。

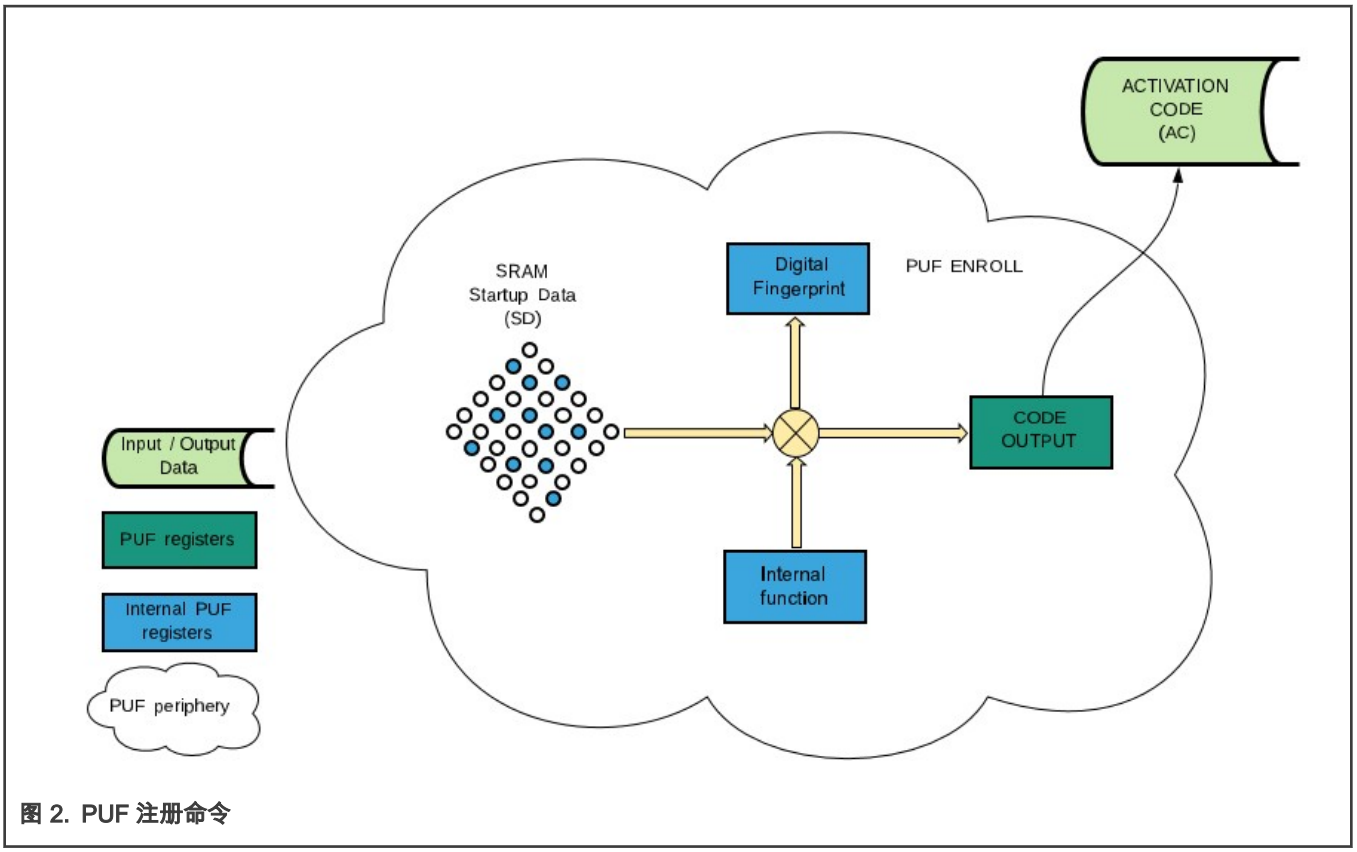


图 2. PUF 注册命令

每次执行 PUF 注册时，都会生成一个新的不同的激活码（和数字指纹）。密钥码可以被加密密钥时使用的数字指纹解密回密钥。这就是为什么必须存储 AC 以便以后解密密钥的原因。AC 是一个 1192 字节的数据块，必须将其存储到非易失性存储器中。

成功执行 PUF 注册命令后，SetKey 命令将变为可用。必须对 PUF 重新下电上电以使能密钥（GetKey）命令。密钥和激活代码也可以使用 ISP 命令生成（它们使用 ROM 代码在内部调用 PUF 函数）并存储到闪存中。

2.5 PUF 启动（Start）

执行 PUF 注册命令并存储 AC 后，即可使用 PUF。必须对 PUF 重新通电并通过 PUF 启动命令将其启动。激活码通过 CODEINPUT 寄存器加载到 PUF 中，PUF 引擎将其与 SRAM 启动数据结合在一起形成数字指纹。

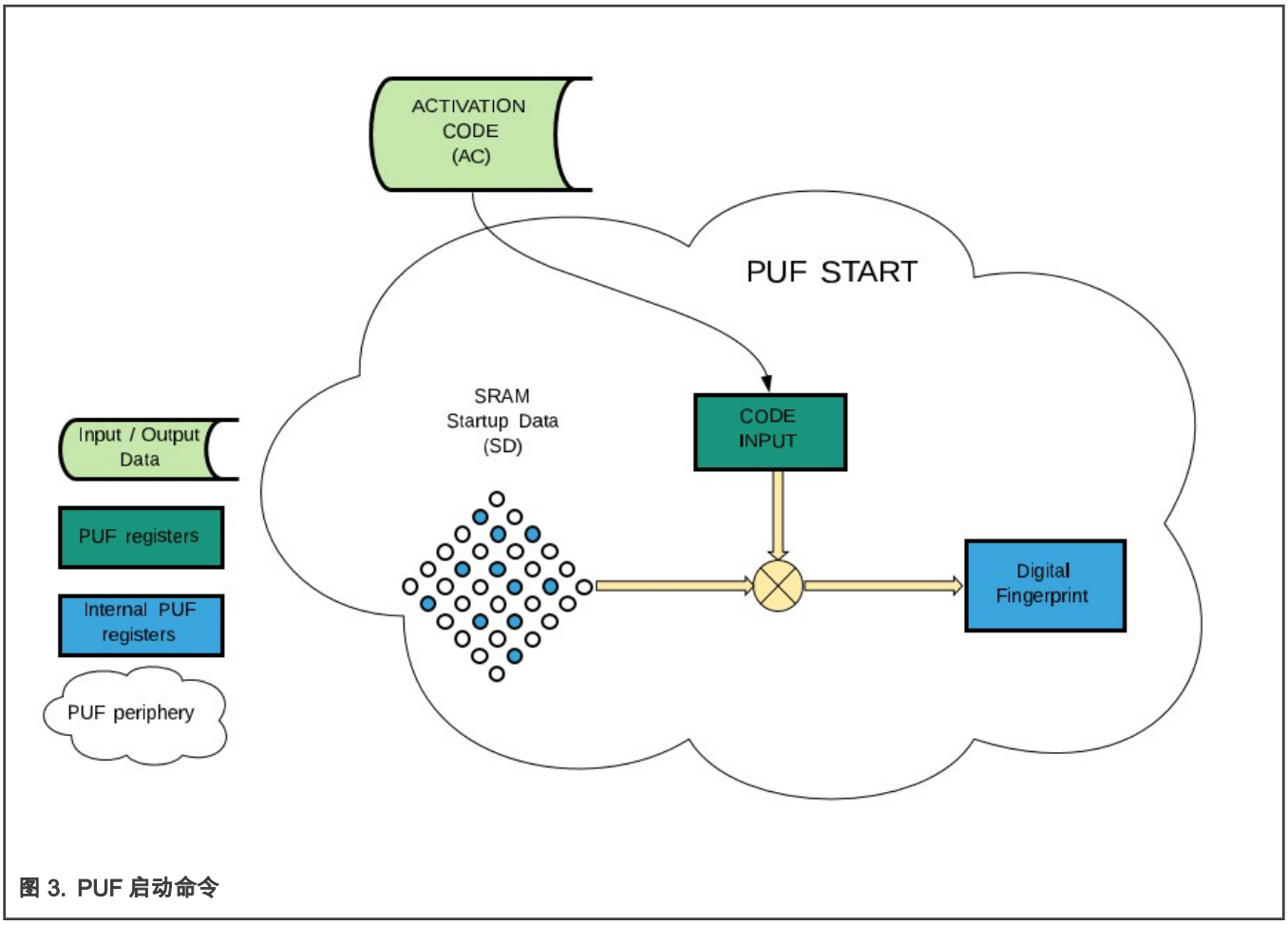


图 3. PUF 启动命令

数字指纹是一个 256 位密钥，用作根密钥或用于加密/解密用户密钥的 KEK。成功执行 Start 命令后，SetKey 和 GetKey 命令将变为可用。数字指纹在 PUF 中是有效的，直到下一个电源周期。

2.6 PUF 设置密钥（SetKey）

设置密钥（SetKey）命令用于创建用户密钥和相应的密钥代码。 用户密钥和数字指纹被组合以生成密钥代码。 输入参数是用户密钥、密钥大小和密钥索引(index)。

- 用户密钥通过 KEYIN 寄存器发送到 PUF，并且必须具有正确的长度，具体取决于密钥的大小。
- 密钥大小通过 KEYSIZE 寄存器发送到 PUF。 它必须具有介于 0 和 63 之间的数字。它指示 PUF 外设生成介于 64 和 4096 位之间的对应密钥大小（0 生成 4096 位密钥；请参阅用户手册）。
- 密钥索引是从 0 到 15 的数字，它被添加到生成的密钥代码的标头(Header)中。 它通过 KEYIDX 寄存器发送到 PUF。 密钥索引在获得密钥（GetKey）命令期间被恢复，并可以在 KEYOUTIDX 寄存器中访问。

一种特殊情况是使用密钥索引 0 来指示获得密钥（GetKey）命令通过内部安全总线或 AHB 将解密的密钥发送到外围设备之一（AES 或 PRINCEx）。密钥代码是加密密钥的安全描述，必须通过 CODEOUTPUT 寄存器将其读出并安全地存储到闪存中。

还有一种机制可以阻止具有给定索引的密钥出现在 APB 寄存器接口上以及被 CPU 读取。该设置通过 IDXBLK 和 IDXBLK_DP 寄存器完成。

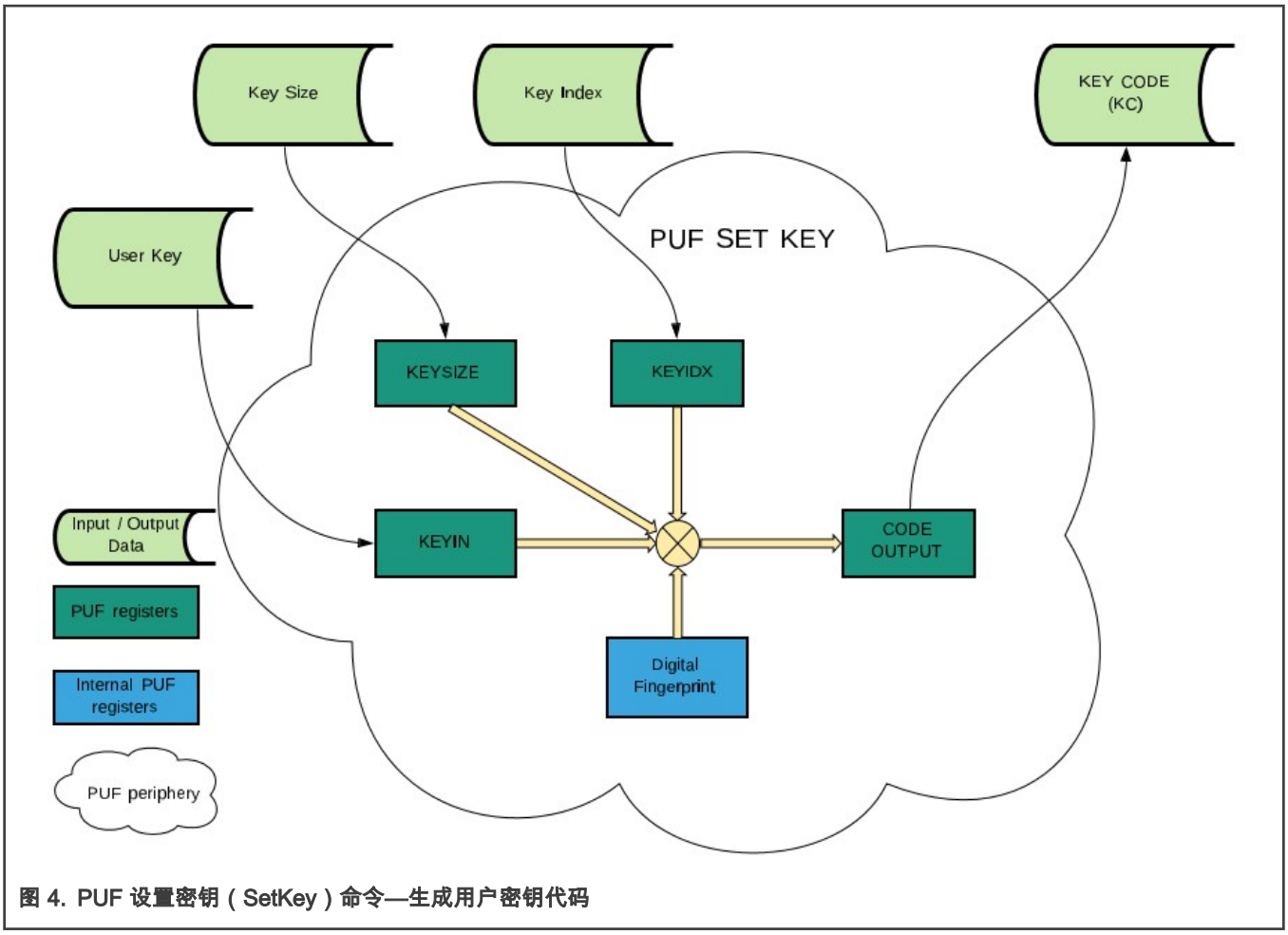


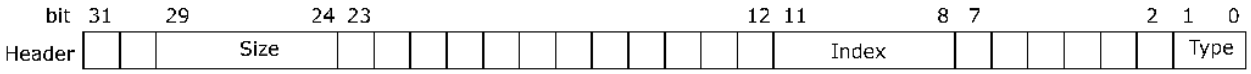
图 4. PUF 设置密钥（SetKey）命令—生成用户密钥代码

生成的密钥代码是一个数据块，开头是 32 位密钥头。

32 位密钥头具有以下功能：

- 类型：
  - 0：生成的密钥。
  - 1：用户密钥。
- 索引：从 0 到 15。0 是一种特殊情况，用于在内部将密钥发送到 AES 或 PRINCE 引擎。它也可以用作用户标签。
- 大小：从 0 到 63，范围从 64 位到 4096 位（0 表示 4096 位）。

2.7 PUF 生成密钥（GenerateKey）



生成密钥（GenerateKey）和设置密钥（SetKey）命令的区别在于，与用户定义和提供的密钥相反，生成的密钥是 PUF 外设生成的随机数。 必须将唯一需要的参数（密钥大小）输入 KEYSIZE 寄存器，并将密钥索引输入 KEYIDX 寄存器。 密钥代码从 CODEOUTPUT 寄存器中读取输出，并且必须将其存储以用于以后的密钥重建。

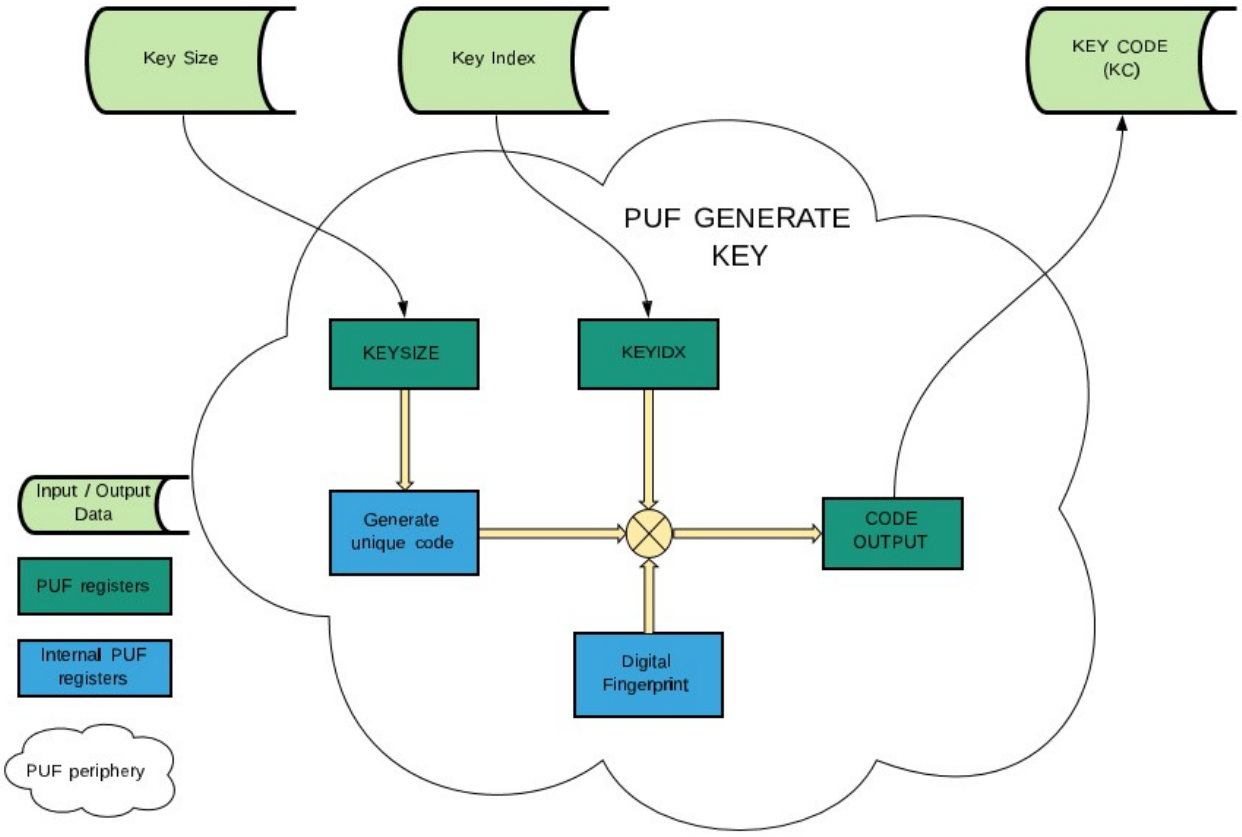


图 5. PUF 生成密钥（GenerateKey）命令—生成密钥代码

2.8 PUF 获得密钥（GetKey）

获得密钥（GetKey）命令用于从密钥代码中获取密钥。在使用相应的激活代码启动 PUF 之后，将密钥代码输入到 CODEINPUT 寄存器中（在注册命令之后，GetKey 命令不可用）。

将密钥代码与数字指纹结合后，即可恢复密钥。 根据密钥代码头中密钥索引的值，密钥被发送到内部秘密硬件总线（密钥索引= 0）或 CODEOUTPUT 寄存器（密钥索引 > 0）。用户代码必须从该寄存器中读出密钥，以及从 KEYIDX 寄存器中读出密钥索引。

内部硬件总线可以将密钥提供给硬件加密单元 AES，PRINCE1，PRINCE2 或 PRINCE3（取决于 KEYENABLE 寄存器的设置）。

KEYMASK [0..3]寄存器必须装入一个随机数，用于隐藏存储在密钥保持寄存器中的密钥值。必须将一个随机值加载到该寄存器中。这样可以避免测信道分析。

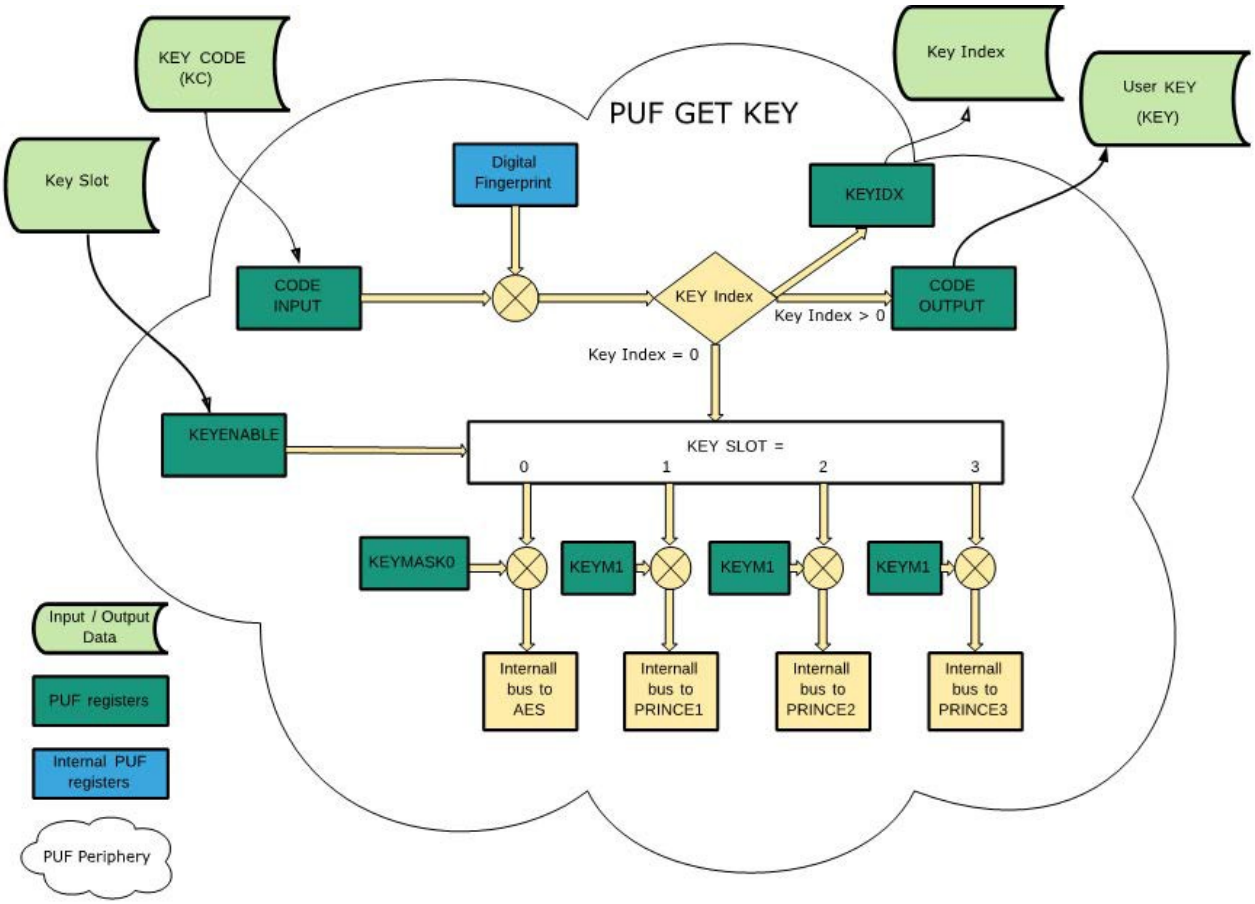


图 6. PUF 获得密钥（GetKey）命令从密钥代码中获取密钥

2.9 PUF 归零（Zeroize）

调用归零（Zeroize）命令时，所有内部参数、关键参数和安全性参数都将被擦除，PUF 控制器将进入错误状态。在设备重新供电之前无法执行任何新操作。

2.10 PUF DisableEnroll

调用此命令会导致 PUF 外设阻止注册操作。

2.11 PUF DisableSetKey

此命令阻止输出密钥代码数据。 如果该位设置为 1，则当密钥代码索引等于 15 时，PUF 外设会阻止密钥代码输出（密钥代码输出数据= 0）。如果该位设置为 0，则不会阻止密钥代码输出（即使密钥输出索引= 15）。

3 密钥管理

3.1 受保护的闪存区域（PFR）中的密钥存储

物理上不可克隆功能（PUF）说明可以将密钥代码安全地存储到 MCU 闪存（或任何其他外部存储器）中，因为 MCU 在物理上无法访问主密钥。存储密钥的位置在 MCU 的受保护闪存区域（PFR）中。

PFR 区域受到保护。它具有三个区域：

- CFPA（客户现场可编程区域）具有单调计数器，密钥撤销和 PRINCE IV 代码。
- CMPA（客户生产车间可编程区域）保存启动配置，安全策略，用户定义的数据和 PUF 密钥存储。

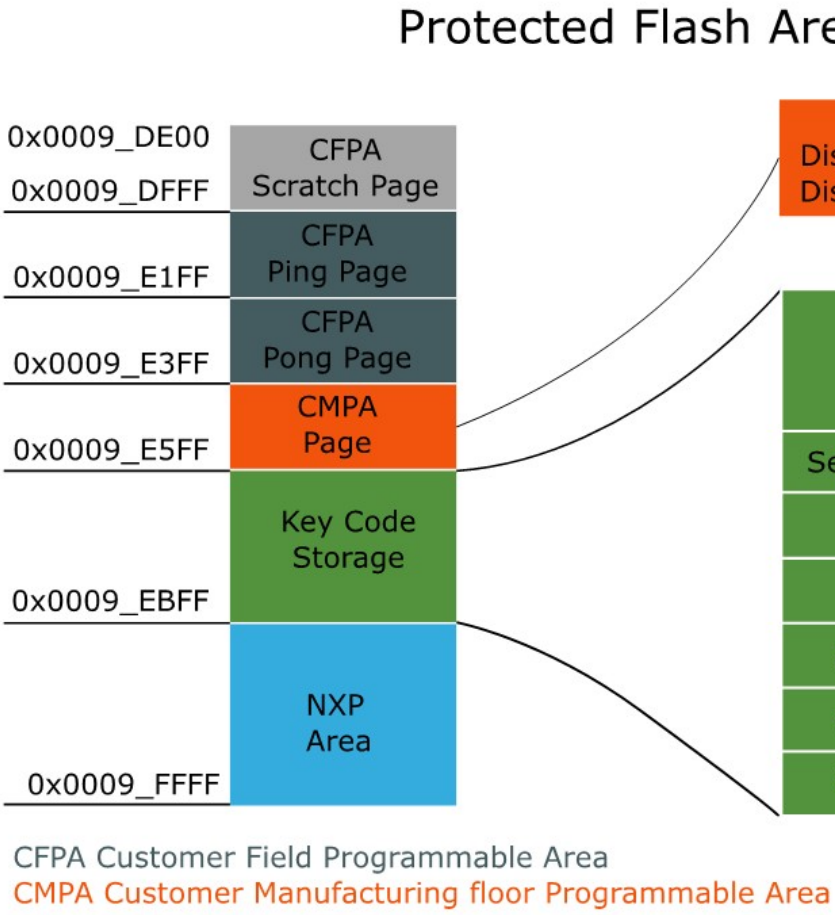


图 7. PFR 中的密钥，PUF 和启动选项

注意

密钥存储主要由 ROM 启动代码使用和管理。

安全启动使用安全启动 KEK 代码。 PRINCE 的闪存加密将 PRINCE1 至 PRINCE3 密钥代码用于其受保护区域。 找到正确的密钥代码后，它们会在 ROM 启动过程中应用于 PRINCE。ROM 启动过程将检查密钥代码存储区域中现有代码的有效性。 如果 AC（或 KC）损坏，则自动将停止并永远阻塞 MCU 启动。

该区域也受到保护，以防数据损坏或修改。当被 ROM 锁定时，所有区域均受到擦除/写入命令的保护。每个区域都有一个 SHA256 哈希摘要字段，该字段在部署生命周期状态中用于交叉检查区域的完整性。

图 7 显示了 PFR 存储器中密钥代码的位置。激活码和五个密钥代码都有足够的位置—安全启动密钥代码，用户密钥代码（可用于 AES）和三个 PRINCE 密钥代码。设备唯一密钥（UDS）也有一个安全的位置。每个密钥代码位置的最大大小限制为 52 个字节，这使得最大密钥大小为 256 位。这看似很低，但是内部加密引擎接受的最大密钥大小为 256 位。AES 允许的最大密钥大小为 128/196/256 位。PRINCE 允许的最大密钥大小为 128 位。

从 PUF 外设使用的角度来看，重要部分也位于 CMPA 页面中。有“禁用 PUF 注册”和“禁用 PUF 密钥代码生成”选项。这些选项在引导过程中被复制到 PUF 外设。请记住，只有在重新启动电源后才能更改这些设置。如果在引导过程中加载了选项，则可能永远不会注册 PUF，或者在用户代码中可能永远不会生成新的密钥代码。但是，您可以从 PFR 中读取现有的激活码和密钥代码并使用它们。

可以通过 ROM API（ISP 和 IAP）更新 CFPA 区域。ISP 功能支持密钥提供命令，以生成密钥代码并将其存储到 PFR 中。另一个选择是使用 flash IAP 命令从用户代码管理 PFR 中的密钥存储区域。

以下各节介绍专用于密钥存储区的 IAP 和 ISP 命令。

3.2 ISP KeyProvision 命令

“系统内编程（ISP）”是一系列功能，支持通过串行接口（UART，I2C，SPI，USB HID）进行 image 烧写。有一个与安全性相关的命令（KeyProvision）可安装预共享密钥，生成随机密钥并将其密钥代码(Key Code)保存到受保护的闪存区域（PFR）密钥存储区域中。

KeyProvision 命令有三个可能的参数：密钥操作，密钥类型和密钥大小。

- 密钥操作：需要指定 KeyProvision 命令的行为，可能是以下值：Enroll，SetUserKey，SetIntrinsicKey，WriteNonVolatile，ReadNonVolatile，WriteKeyStore，ReadKeyStore。
- 密钥类型：此参数定义生成的密钥代码将用于什么目的。

表 1. 密钥类型

密钥类型	取值
UserKEK	11
SBKEK	3
PRINCE0	7
PRINCE1	8
PRINCE2	9
UDS	12

- 密钥大小：密钥大小参数以字节为单位定义。

3.3 通过 blhost PC 应用程序配置密钥存储区

blhost 命令行应用程序是 SDK 软件包的一部分。blhost 应用程序在主机计算机上使用，以向运行 MCU 引导加载程序的 MCU 发出命令。blhost 应用程序与 MCU 引导加载程序结合使用，无需编程工具即可将固件应用程序编程到 MCU 设备上。 blhost 应用程序还支持以下示例中描述的几个与密钥配置相关的命令：

- 注册 PUF

  该命令调用 ROM 代码，用于配置、启动和注册 PUF 外设：

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

- 生成 UDS

  该命令调用使用 PUF 生成内部 32 字节密钥及其密钥代码的 ROM 代码，并将其存储到 CMPA 密钥代码存储[UDS 密钥代码]位置（RAM 版本）：

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

- 生成用户安全引导加载程序代码 SBKEK

  该命令调用 ROM 代码，该 ROM 代码根据 tempSbkek.bin 文件中存储的密钥生成密钥代码，并将其存储到 CMPA 密钥代码存储[安全启动 KEK]位置（RAM 版本），此处的 tempSbkek.bin 是带有 SB KEK 的纯文本文件：

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

- 生成用户密钥代码 User KEK

该命令设置用户密钥代码并将其存储到 CMPA 密钥代码存储[用户密钥代码 User KEK]位置（RAM 版本），其中 tempSbkek.bin 是带有 User KEK 的纯文本文件：

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

- 生成 PRINCE0-的内部密钥代码

  该命令为 PRINCE0 生成 128 位内部密钥代码，并将其存储到 CMPA 密钥代码存储[PRINCE0 密钥代码]（RAM 版本）：

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

- 生成 PRINCE1 的内部密钥代码

  该命令为 PRINCE1 生成 128 位内部密钥代码，并将其存储到 CMPA 密钥代码存储[PRINCE1 密钥代码]（RAM 版本）：

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

- 为 PRINCE2 生成 16 字节的内部密钥代码

  该命令为 PRINCE2 生成 128 位内部密钥代码，并将其存储到 CMPA 密钥代码存储[PRINCE2 密钥代码]（RAM 版本）：

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

- 该命令用于使用先前操作中创建的密钥代码的 RAM 版本（包括在注册操作期间创建的激活代码）更新到 PFR 中的密钥存储区域：

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

3.4 通过 blhost-elftosb-gui 工具进行密钥配置

elftosb-gui 是基于窗口的 PC 工具，该工具基于命令行 blhost 应用程序构建，可帮助生成镜像并添加所需的所有设置和加密需求。该工具可用于告诉 PUF 注册并存储激活码，生成密钥，然后将其存储到 PFR。 目前，该工具不支持用户 KEK 代码的生成。

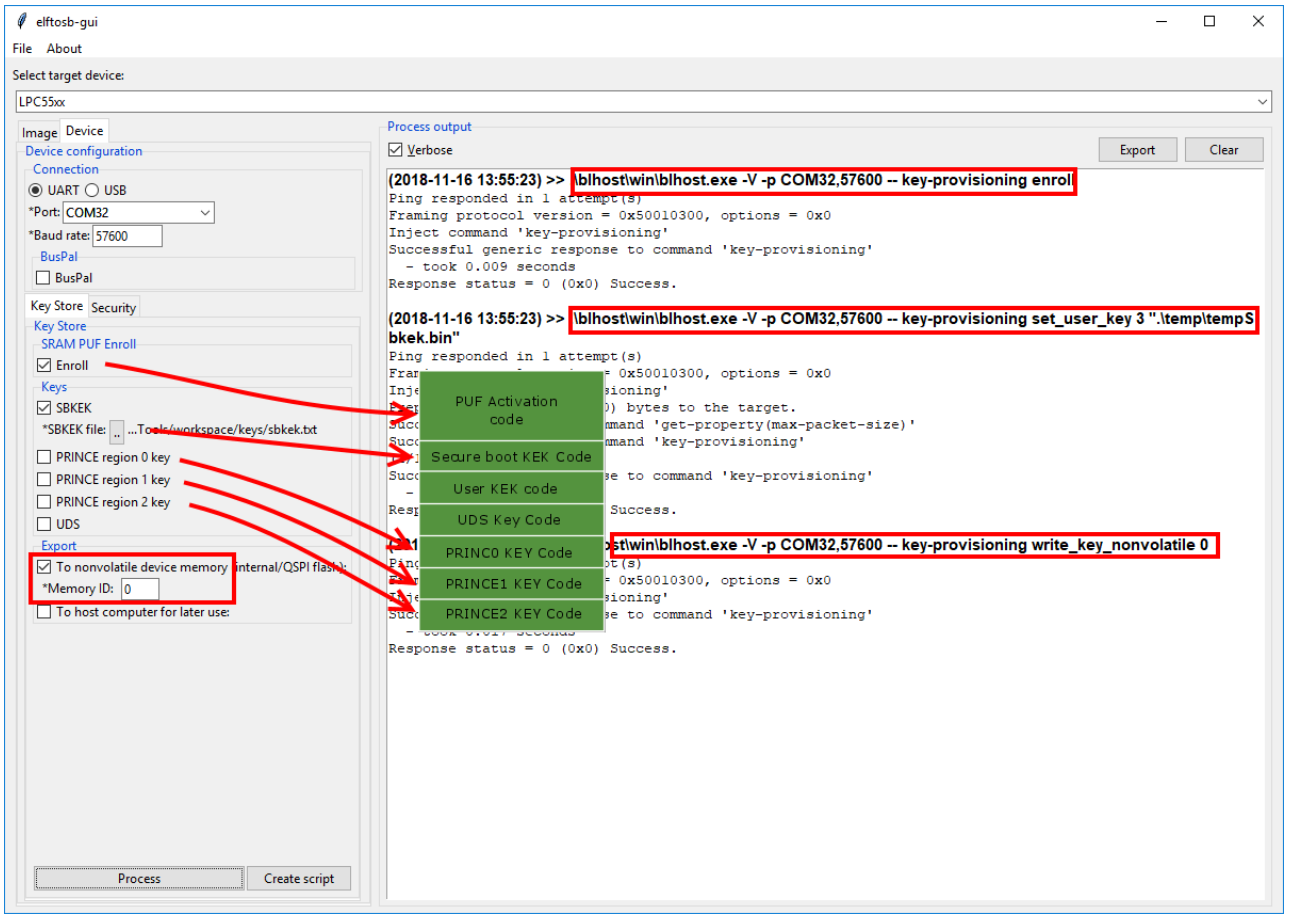


图 8. 通过 elftosb-gui 工具进行密钥配置

elftosb 工具还可以设置 PUF 的安全性选项，禁用“注册”命令和“ SetKey”命令。

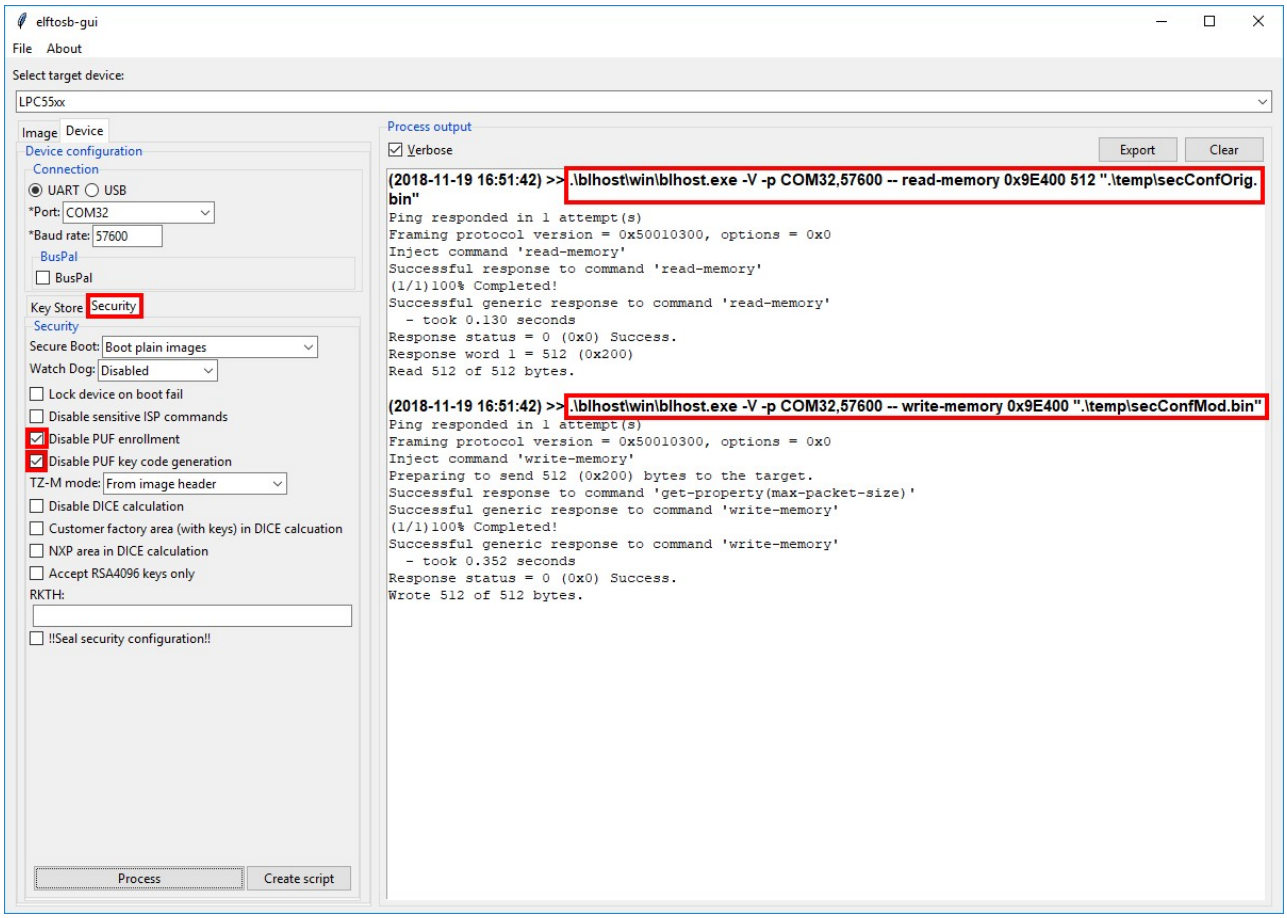


图 9. elftosb 工具支持的各种安全设置

两个 PC 应用程序（elftosb-gui 和 blhost）都在以下路径中的 MCU SDK 中：<SDK_ver\middleware\mcu-boot\bin\Tools>。

3.5 密钥管理 IAP 命令

您可以通过用户应用程序将密钥代码编程到 MCU 中 PFR 的密钥存储区域中。有几个 FFR IAP ROM 命令专用于读取密钥代码和激活代码，并将数据块存储到密钥存储区域中。

要使用存储在 PFR 密钥存储部分中的密钥代码，请使用用于创建密钥代码的激活代码来启动 PUF。

- ffr_keystore_get_ac 从密钥存储区返回 AC：

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

`pActivationCode` 是指向缓冲区的指针，该缓冲区的长度足以容纳 1192 字节的激活码。

必须使用 PUF 启动命令将 AC 传递到 PUF。

- `ffr_keystore_get_kc` 用于从 PFR 加载密钥代码：

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

`pKeyCode` 是用于复制密钥代码的缓冲区。

表 2 中指定了密钥类型：

表 2. 密钥类型

密钥类型	取值
SB KEK	0
User KEK	1
UDS	2
PRINCE0	3
PRINCE1	4
PRINCE2	5

还有一些命令可读/写 PFR 非易失性存储器，例如 `ffr_keystore_write`, `ffr_infield_page_write` 和 `ffr_get_customer_infield_data`。

有关更多详细信息，请参见用户手册。

4 PUF 测试软件的使用

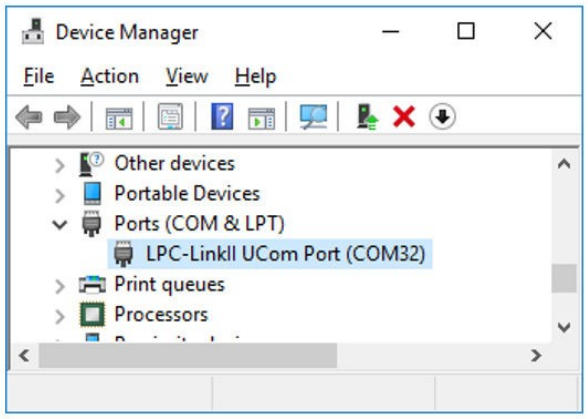
sw_an12324.zip 软件包随附于本应用笔记。该软件可以执行基本的 PUF 命令（注册，初始化，启动，停止，SetKey，GetKey）和增强安全性的命令（DisableEnroll，DisableSetKey）。该软件还能够获取密钥代码，并通过标准内存总线或 PUF 与哈希加密引擎之间的秘密总线将其提供给 AES 模块。也可以使用提供的密钥对 16 字节的数据块进行加密/解密。该示例代码基于 SDK，并使用 SDK 驱动程序。

该软件通过串行线路终端进行控制，并使用板载内置虚拟串行 COM 端口（LPC-LinkII UCom 端口）。使用任何基于 ASCII 的串行 COM 端口终端应用程序，例如 Teraterm 或 Putty。终端必须启用本地回显。

4.1 设置 PUF 示例代码

打开应用说明软件包，对其进行编译，然后在 LPCXpresso5500 板上运行它。通过连接器 P9 将 LPCXpresso5500 板连接到计算机。LPCLinkII CMSIS DAP 和虚拟串行 COM 端口 LPC-Link UCOM 出现在 Windows OS 设备管理器中。

使用终端打开正确的 COM 端口，并将参数设置为 115200 bps 位，无奇偶校验，无硬件控制。复位后，控制台中会出现一个初始菜单。



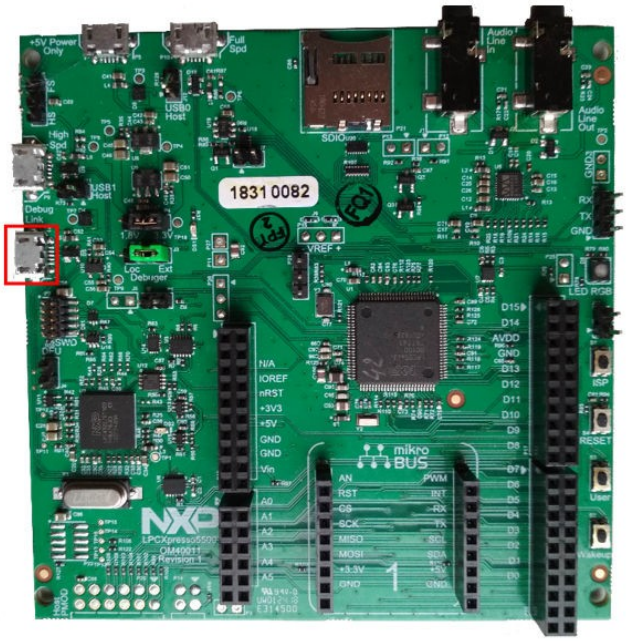


图 10. 虚拟串行 COM 端口和电路板

4.2 应用

当应用程序运行时，在终端窗口中会打印出一个菜单。在屏幕上，始终显示 PUF 状态和 PUF 允许的操作。

控制台中显示的初始状态未重置，因此 PUF 外设未启动并且不允许命令。但是，PUF 可以从 ROM 启动代码启动，具体取决于 CMPA 密钥存储区以及“禁用 PUF 注册”和“禁用 PUF 密钥代码生成”启动选项。

PUF 外设必须在运行之前初始化。在初始化期间，SRAM 存储器已被重新供电并可以被读取。

PUF 状态表示以下含义：

- 忙（busy）：操作正在进行中。
- 成功（success）：上一次操作成功。
- 错误（error）：PUF 处于错误状态，无法执行任何操作。

在允许 PUF 操作的开始，四个状态位表示以下含义：

- 注册：允许注册操作。
- 启动：允许启动操作。
- SetKey：允许使用 SetKey 操作。
- GetKey：允许 GetKey 操作。

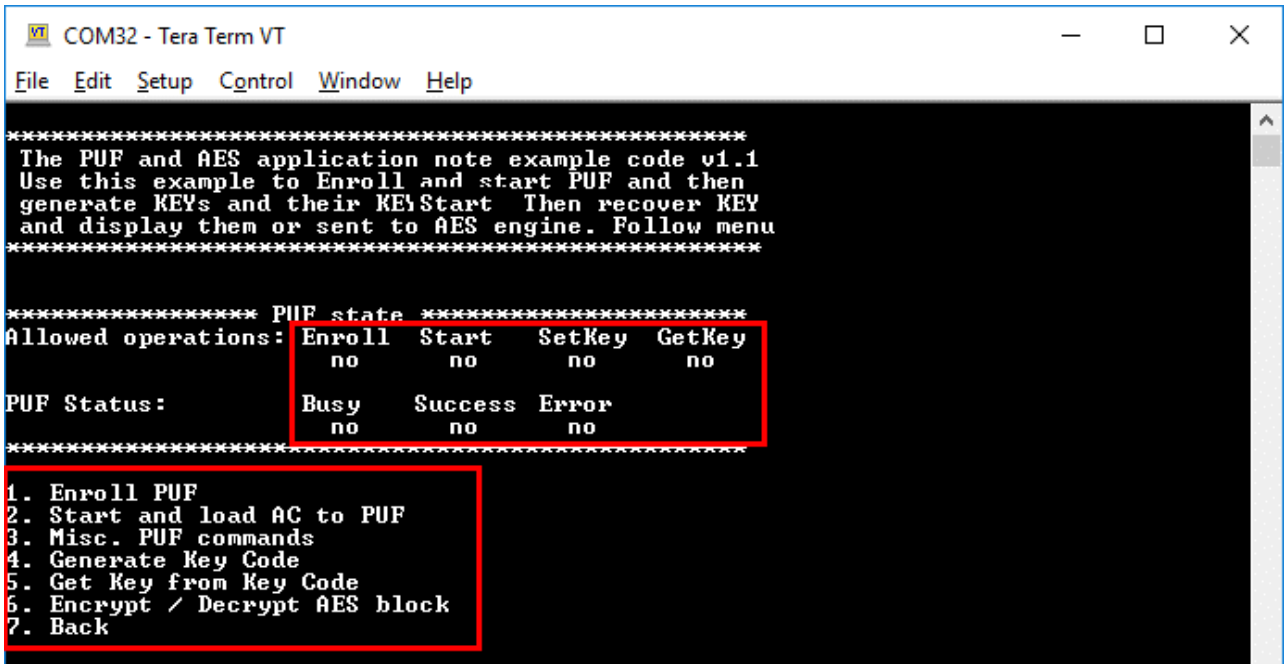


图 11. 应用程序主屏幕

4.3 注册 PUF

开始使用 PUF 的第一步是注册。在注册过程中将创建一个激活码。注册操作首先给 PUF 外设加电，对其进行初始化（启动），然后调用注册命令。在此过程中，激活码（Activation Code）被创建，可以将其存储到 RAM 或闪存中以备将来使用。

注意

成功执行 Enroll 命令后，仅允许使用 SetKey 命令。

成功执行 Enroll 命令后， 必须对 PUF 外设重新通电并重新初始化，才能再次调用 Enroll 命令。要启用 GetKey 功能，必须对 PUF 外设重新加电并重新启动。 可以将 AC 存储到 RAM 或闪存中以供进一步使用。

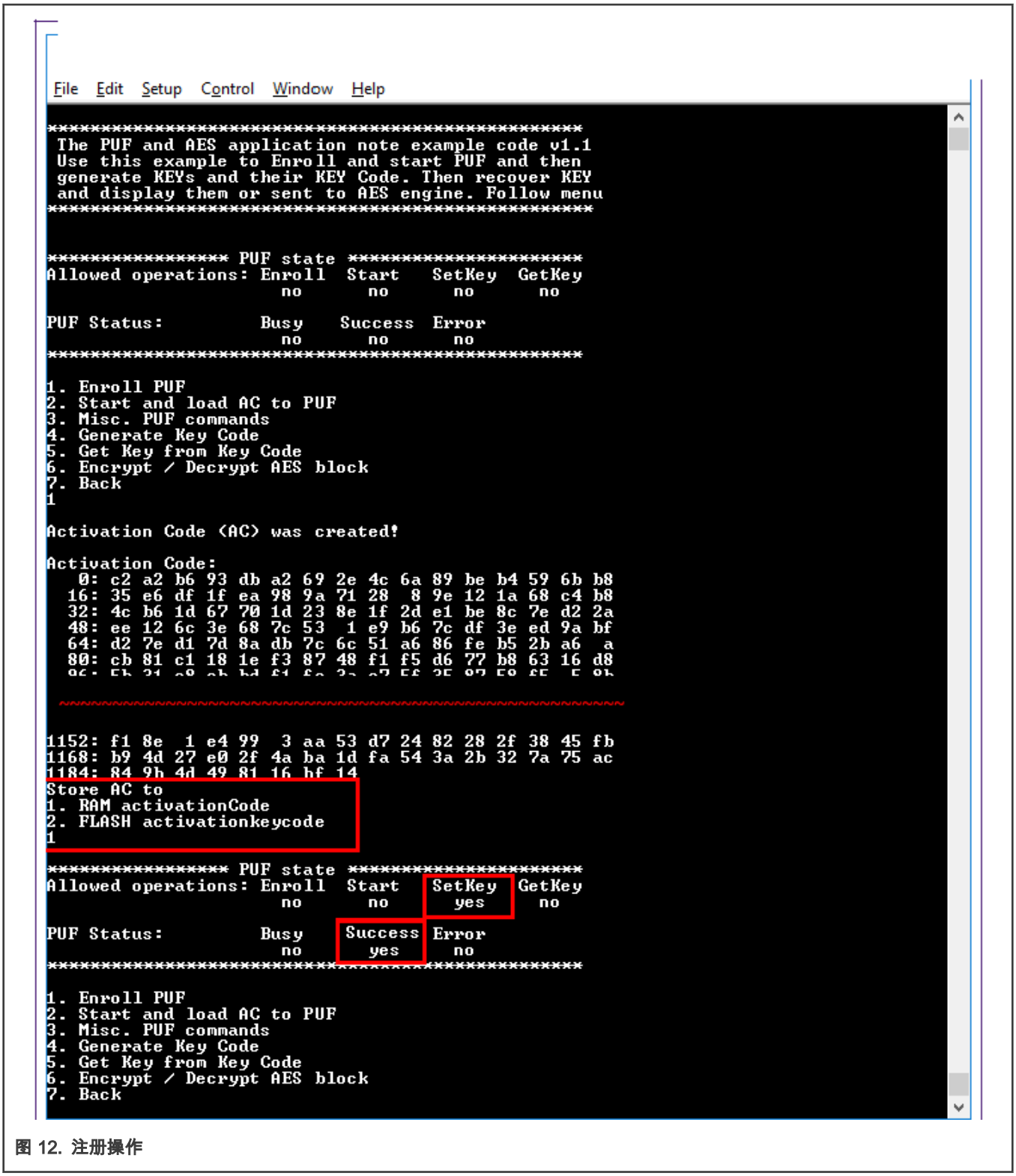


图 12. 注册操作

4.4 启动并将 AC 加载到 PUF 的命令

“启动并将 AC 加载到 PUF”命令启动 PUF 并加载一个已存储的激活码。激活码必须与同一 AC 生成的密钥代码匹配。否则 GetKey 命令将失败。可以从 RAM 或闪存（在上文的步骤中已存储）或 CMPA 位置（由 blhost 创建和存储）中加载激活代码（Activation Code, AC）。

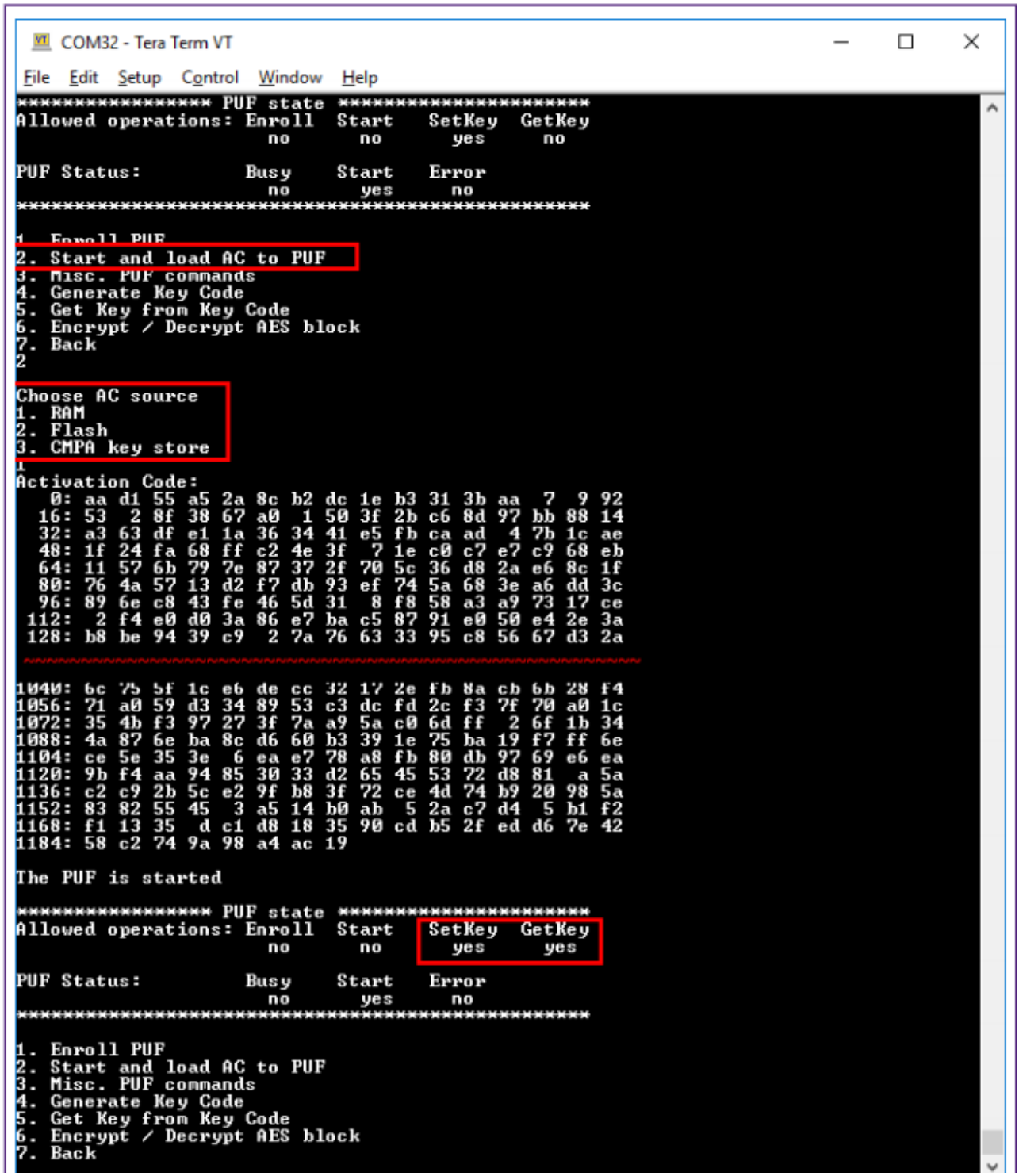


图 13. 启动并将 AC 加载到 PUF 命令

4.5 生成密钥代码命令

生成密钥代码命令使您可以创建用户密钥或内部密钥。

- 用户密钥：将密钥输入到 PUF，并从该密钥中生成密钥代码。

- 内部密钥：密钥是在 PUF 内部随机生成的，您只能获得密钥代码。如果内部代码的索引为 0，则密钥本身将被永久隐藏，并且无法被中央处理器读取，因为 PUF 永远不会将其发送到 AHB 总线。

下一步是在 0 到 15 的区间中设置密钥索引。密钥索引为 0 会使 GetKey 期间，将密钥代码中重建出的密钥发送到内部硬件总线 。密钥索引 1（或更高）会将恢复出的密钥发送到 AHB，CPU 可以读取它。

输入密钥大小。 PUF 支持从 64 到 4096 位的密钥大小。但是 PRINCE 引擎仅支持 128 位密钥，AES 引擎仅支持 128/196/256 位密钥。

生成用户密钥的最后一步是输入用户密钥大小。出于简化原因，终端仅支持 ASCII 码。输入的参数被发送到 PUF 外设，生成并打印出密钥代码。

生成的密钥代码可以存储在四个可用的位置中的某一位置，RAM 中可以存放两个密钥代码，闪存中可以存放两个密钥代码中，可以供后续使用。

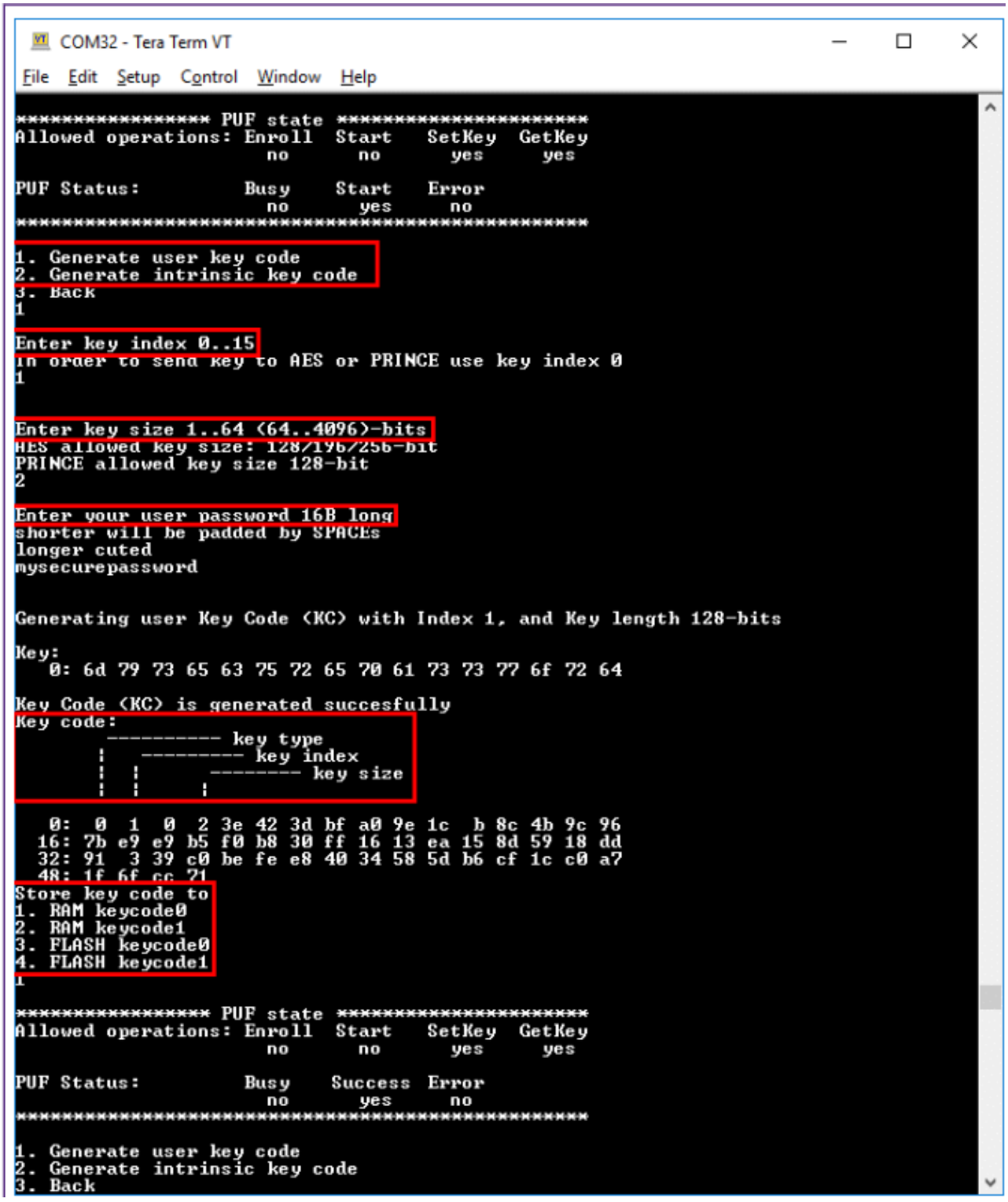


图 14. SetKey 操作

4.6 Getkey 命令

使用 GetKey 命令从密钥代码中恢复/重建出密钥。如果密钥代码索引大于 0，则将重建并打印该密钥。

如果密钥代码索引等于 0，则将密钥发送到秘密内部硬件总线。该总线连接到 AES 和 PRINCE 引擎。 输入密钥槽值 1，以将密钥发送到 AES 引擎。从 2 到 4 的值会将密钥发送到 PRINCE1，PRINCE2 或 PRINCE3。如果命令成功执行，则密钥将临时存储在 AES 引擎中，并可用于加密/解密 16 字节的数据块。

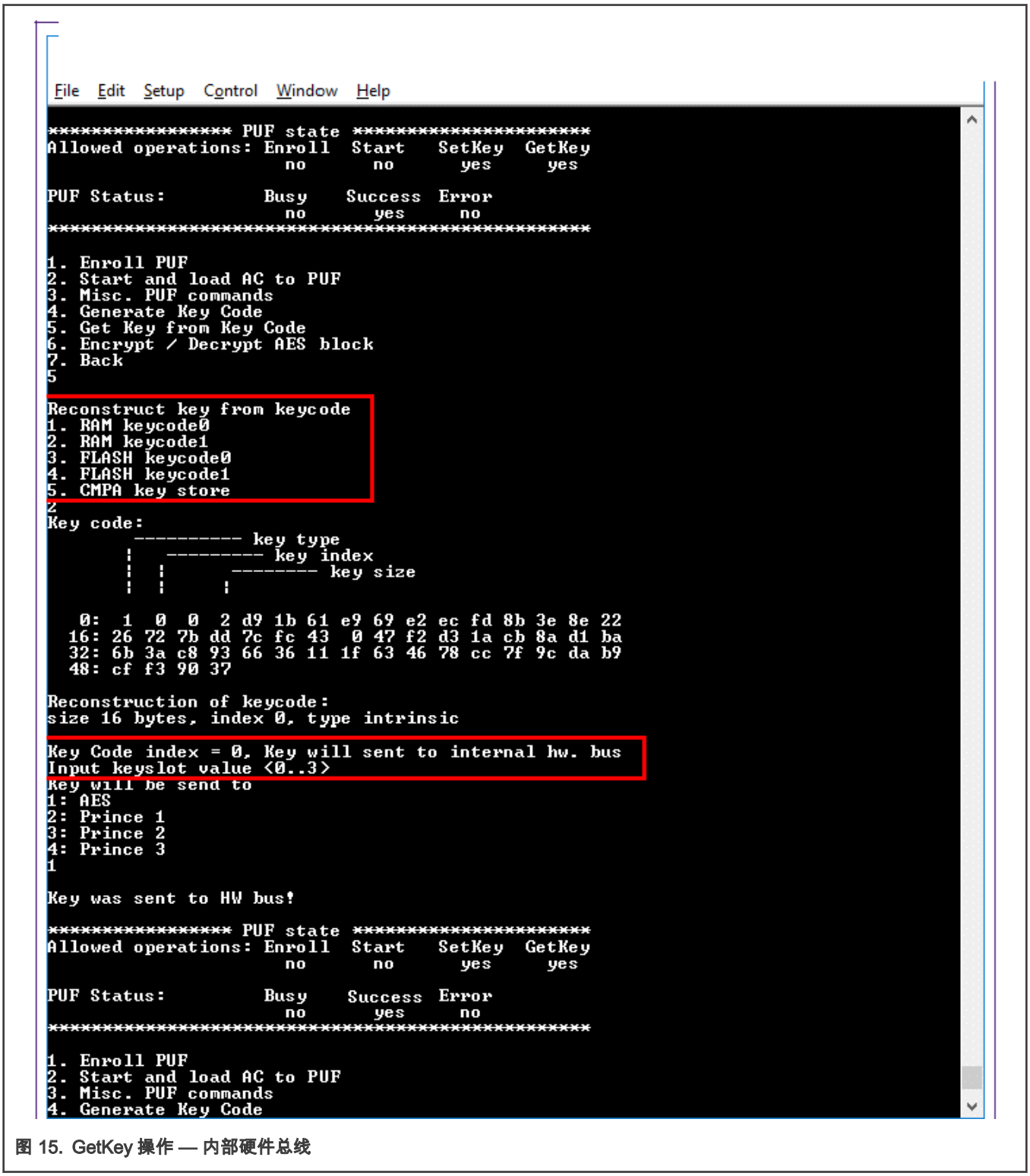


图 15. GetKey 操作 — 内部硬件总线

对于密钥索引大于 0 的密钥代码，该密钥将被打印到控制台。

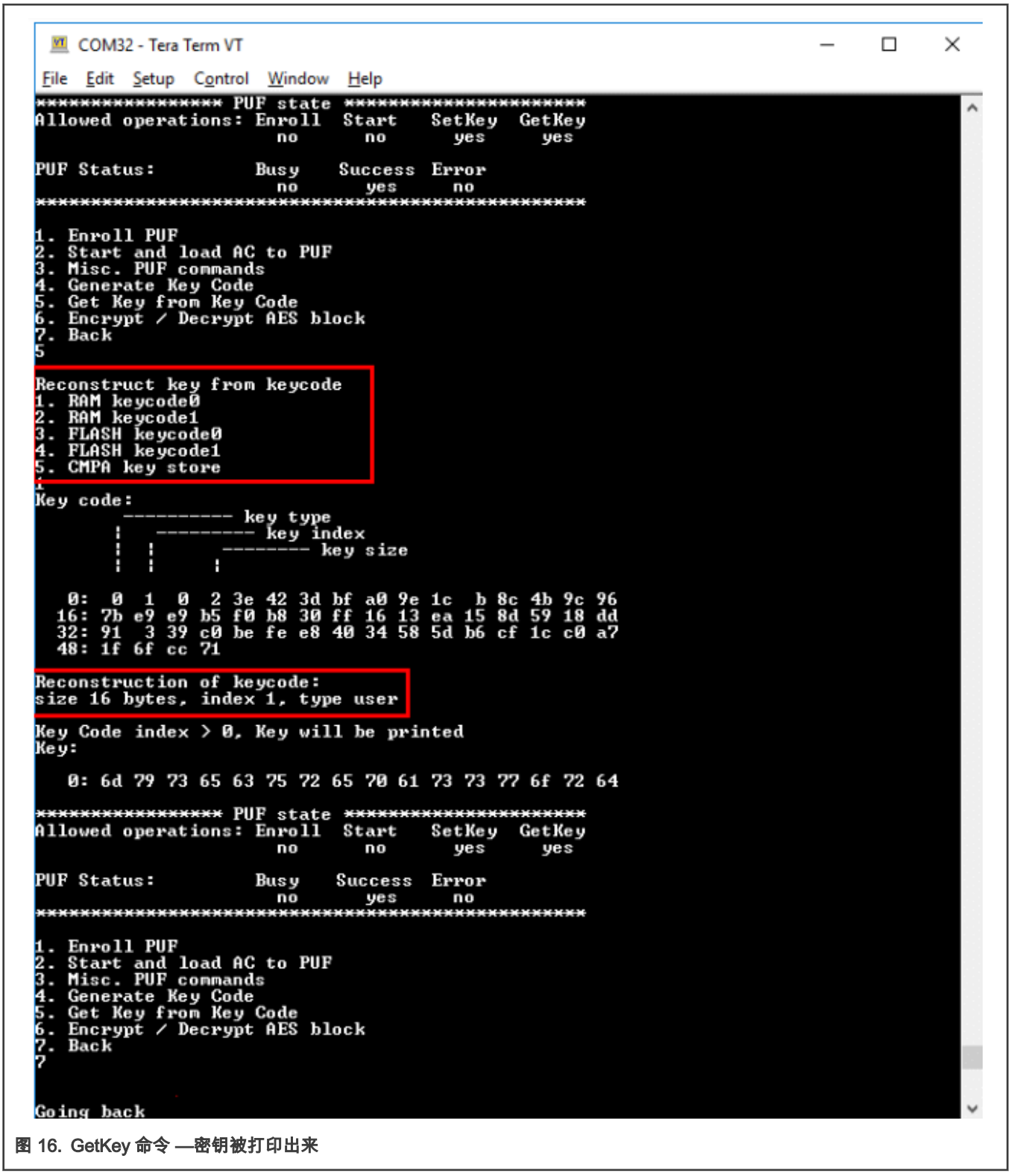


图 16. GetKey 命令 —密钥被打印出来

4.7 使用密钥加密数据块

通过此命令，您可以使用 PUF 重构的密钥对 16 字节的数据块进行加密/解密。

您可以在密钥（使用 GetKey 命令通过内部密钥硬件总线发送给 AES 的密钥）或用户密钥（由 PUF 从存储的一个密钥代码中生成）之间进行选择。

输入 ASCII 格式的纯文本， 然后使用提供的密钥对该数据块进行加密/解密。 AES 使用 ECB 模式。

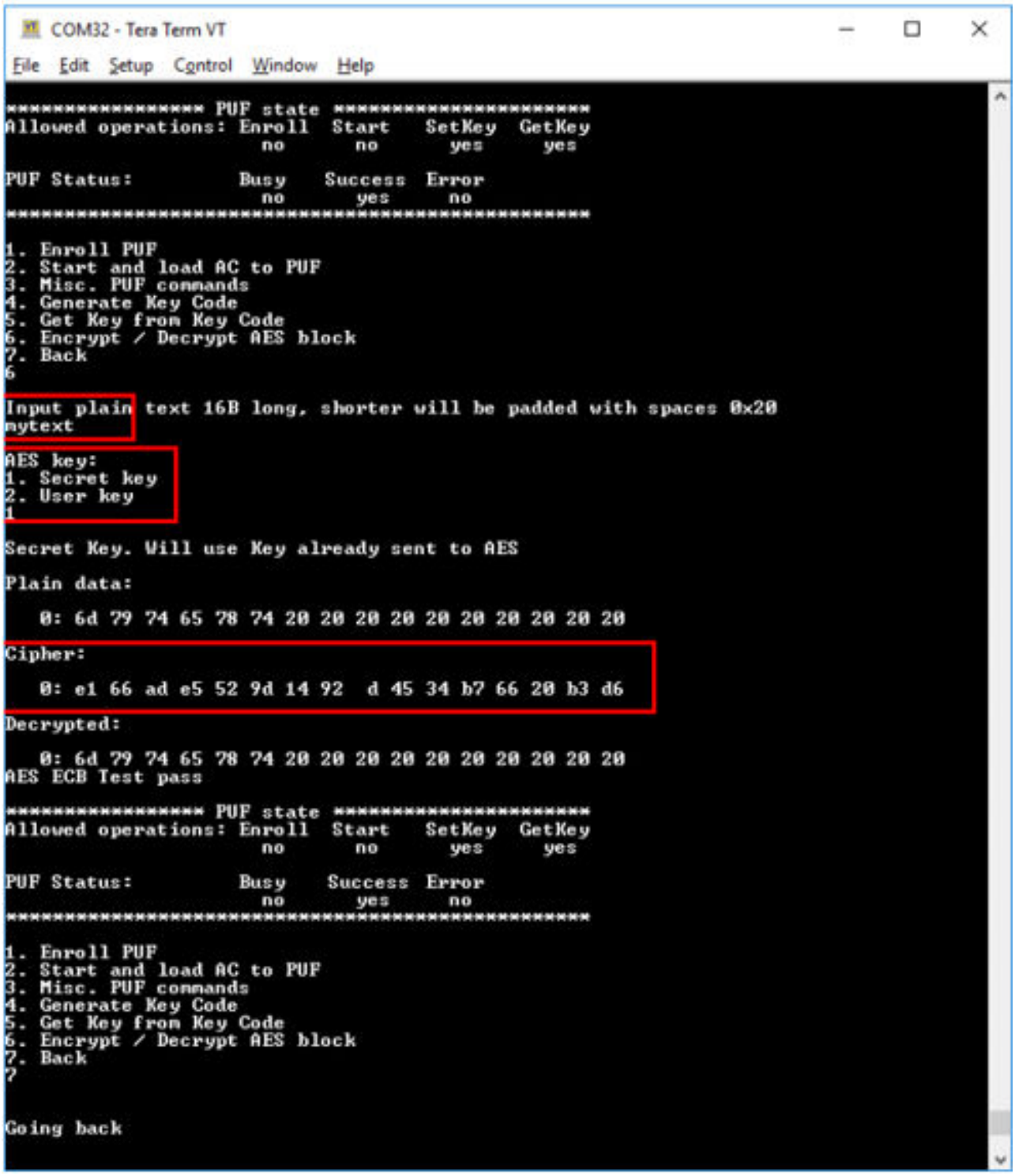


图 17. 加密/解密菜单

4.8 杂项菜单

杂项菜单具有以下选项：

- 初始化 PUF。
- 停止 PUF。
- 归零 PUF。
- 禁用注册 PUF。
- 禁用密钥生成。

只有重新启动 MCU 才能清除“禁用注册 PUF”和“禁用密钥生成”设置。

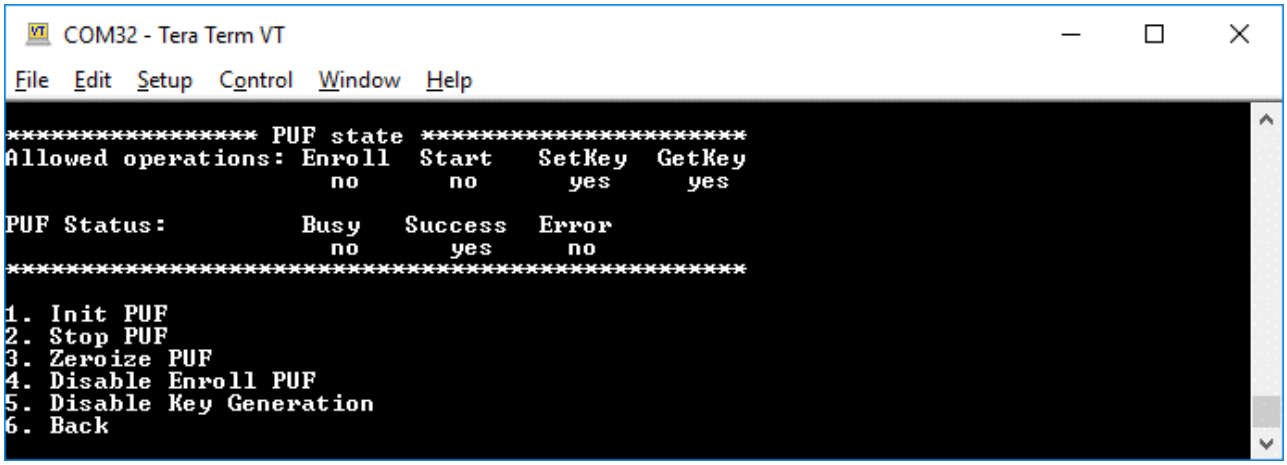


图 18. 杂项菜单

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2019 年 2 月

Document identifier: AN12324

arm