LPC54S0xxSICTUG

LPC54S0xx Secure Image Creator Tool User's Guide

Rev. 0 — 26 October 2021 User Guide

Contents

1 Introduction

The LPC54S0xx is a family of Arm® Cortex®-M4 based microcontrollers for embedded applications featuring a rich peripheral set with very low power consumption and enhanced debug features.

The LPC54S0xx family includes 360 KB of on-chip SRAM, a quad SPI Flash Interface (SPIFI) for expanding program memory, one high-speed and one full-speed USB host and device controller, Ethernet AVB, LCD controller, Smart Card Interfaces, SD/MMC, CAN FD, an External Memory Controller (EMC), a DMIC subsystem with PDM microphone interface and I2S, five general-purpose timers, SCTimer/PWM, RTC/alarm timer, Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), ten flexible serial communication peripherals (USART, SPI, I2S, I2C interface), Secure Hash Algorithm (SHA), AES-256 engine, Physical Unclonable Function (PUF), 12-bit 5.0 Msamples/sec ADC, and a temperature sensor.

The following sections explain the use of the LPC54S0xx secure image creator tool to generate various secure boot images supported by LPC54S0xx.

2 LPC54S0xx secure image creator tool

The LPC54S0xx secure image creator tool is a command line tool to create LPC54S0xx secure images.

The “lpc54xxx_imgcr” tool has multiple subcommands to perform the different operations related to LPC54xxx secure image creation. This includes generating keys, certificates, signing images, and encrypting the binary images bootable by LPC54S0xx boot ROM.

Following are the subcommands supported by the image creator tool.

- **genrsakey**: Generates private-public key pair for Root of Trust (RoT) or Image key
- **genaeskey**: Generates AES key for image encryption
- **gencert**: Generates an image key certificate
- **genkeystore**: Generates key store image with provided key codes
- **encimage**: Creates encrypted image using AES-128 key provided
- **showotp**: Shows OTP values to be programmed for given RoT key and AES key
- **signimage**: Signs an image file using provided image key and certificate

Use the `-h` or `--help` arguments after the subcommand to get information about that subcommand.

3 Generation of RSA keys

There are two key pairs used for LPC54S0xx image creation.

1. Root of Trust (RoT) key (public and private keys)

2. Image Key (public and private keys)

The RoT private key is used by the customer to sign the image key certificates, potentially many of them. The SHA-256 hash of the RoT public key is stored in the OTP memory.

Each image can be signed using individual Image Key pair, which is authenticated by the image key certificate signed by the RoT private key.

The **genrsakey** subcommand is used to generate both RoT key pair and Image Key pair.

Command Format:

```
lpc54xxx_imgcr.exe genrsakey [-h] [--exponent <exponent>] [--password <password>] <keyfilename>.pem
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `--exponent` is an optional argument used to provide the public exponent for RSA. `<exponent>` is a 32-bit integer value. The default value is set to 3 for faster boot in LPC54S0xx. The other recommended value is 65537.
- `--password` is an optional argument used to provide a password with which the output key file is encrypted. If the argument is not provided, the output key file is not encrypted.
- `<keyfilename>.pem` is a positional argument to provide the output file path for the PEM file.

Example:

```
lpc54xxx_imgcr.exe genrsakey --password foo rotk.pem

lpc54xxx_imgcr.exe genrsakey image_key.pem
```

This generates a 2048-bit RoT key pair and 2048-bit Image Key pair. The RoT key PEM file is encrypted with the given password but the Image Key PEM file is not encrypted.

4 Generation of AES keys

For encrypted boot, a 128-bit or a 256-bit AES symmetric key is used. LPC54S0xx image creation tool provides a mechanism to generate the AES keys using the **genaeskey** subcommand.

Command Format:

```
lpc54xxx_imgcr.exe genaeskey [-h] [--keysize <keysize>] [--password <password>] <aeskeyfilename>.bin
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `--keysize` is an optional argument used to provide the AES key size. The default `<keysize>` value is set to 128.

When on-chip OTP memory bank is used to store AES keys, use 128-bit keys only. For example, `--keysize 128`.

When PUF Key Store is used to store AES keys (in form of key codes), use 256-bit keys only. For example, `--keysize 256`.

- `--password` is an optional argument used to provide a password with which the output key file is encrypted. Save the password securely as it is required when generating the encrypted images. If the argument is not provided, the output key file is not encrypted.
- `<aeskeyfilename>.bin` is a positional argument to provide the output file path for the AES key.

Example:

```
lpc54xxx_imgcr.exe genaeskey aes128_key.bin

lpc54xxx_imgcr.exe genaeskey --keysize 256 --password test123 aes256_key.bin
```

5 Creation of image key certificate

The **gencert** subcommand generates an image key certificate based on image key certificate format (see LPC540xx/LPC54S0xx User Manual for details) signed by the Root of Trust (RoT) private key.

Command Format:

```
lpc54xxx_imgcr.exe gencert [-h] -r <rotkeyfilename>.pem -k <imagekeyfilename>.pem \

[-u <userkeyfilename>.pem] [--rid <cert_id>] [-p] <ikcertificatefilename>.bin
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `-r` or `--rotk` is a required argument to provide the path to the key file containing the RoT private key in PEM format.
- `-k` or `--imgk` is a required argument to provide the path to the key file containing the Image private key in PEM format.
- `-u` or `--usrk` is an optional argument to provide the path to the key file containing the User private key in PEM format. If this argument is provided, version 3 of image key certificate is generated.
- `--rid` is an optional argument to provide the 8-bit certificate revocation identifier. `<cert_id>` value should match the 8-bit Revoke ID (REVOKE_ID) field in OTP bank 3 word 0. This parameter is used to revoke signed images which are released. Only 9 certificate revocation identifiers are possible; 0x0, 0x1, 0x3, 0x7, 0xF, 0x1F, 0x3F, 0x7F, and 0xFF.
- `-p` or `--preformat` is an optional argument that generates pre-formatted signature using Montgomery reduction for the image key certificate. This causes RSA signature verification to be faster during boot on LPC54S0xx.
- `<ikcertificatefilename>.bin` is a positional argument to provide the output file path for the image key certificate file.

Example:

```
lpc54xxx_imgcr.exe gencert -r rotk.pem -k image_key.pem --rid 0x3 image_key_cert.bin
```

The above example generates an image key certificate file with certificate revocation identifier set to 0x3.

6 Creation of signed images

The **signimage** subcommand generates a RSA-2048 signed image. The input application image is signed using the provided image private key.

Command Format:

```
lpc54xxx_imgcr.exe signimage [-h] -c <ikcertificatefilename>.bin \

-k <imagekeyfilename>.pem -i <inputimagefilename>.bin [-p] \

<signedimagefilename>.bin
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `-c` or `--cert` is a required argument to provide the path to the image key certificate file.
- `-k` or `--imgk` is a required argument to provide the path to the key file containing the image private key in PEM format.
- `-i` or `--image` is a required argument to provide the path to the binary input image that must be signed.
- `-p` or `--preformat` is an optional argument that generates pre-formatted signature using Montgomery reduction. This causes RSA signature verification to be faster during boot on LPC54S0xx.
- `<signedimagefilename>.bin` is a positional argument to provide the output file path for the signed image.

Example:

```
lpc54xxx_imgcr.exe signimage -c image_key_cert.bin -k image_key.pem -i app_image.bin signed_image.bin
```

The above example generates a signed image “signed_image.bin” from the input binary image “app_image.bin”, image key certificate “image_key_cert.bin” and signed using the image private key in “image_key.pem”.

7 Creation of encrypted images

The **encimage** subcommand generates an encrypted image. The input application image is encrypted using the provided AES key and AES-GCM encryption. LPC54S0xx ROM accepts AES-GCM encrypted images.

Command Format:

```
lpc54xxx_imgcr.exe encimage [-h] -e <aeskeyfilename>.bin \
-i <inputimagefilename>.bin <encryptedimagefilename>.bin
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `-e` or `--enck` is a required argument to provide the path to the AES key file.
- `-i` or `--image` is a required argument to provide the path to the binary input image that must be encrypted.
- `<encryptedimagefilename>.bin` is a positional argument to provide the output file path for the encrypted image.

Example:

```
lpc54xxx_imgcr.exe encimage -e aes_key.bin -i app_image.bin \
encrypted_image.bin
```

The above example generates an encrypted image “encrypted_image.bin” from input image “app_image.bin” using AES key in “aes_key.bin” file.

8 Creation of enhanced secure images

An enhanced secure image is both encrypted and signed (first encrypted and then signed). To generate an enhanced secure image, use the **encimage** and the **signimage** subcommands in sequence.

Command Format:

```
lpc54xxx_imgcr.exe encimage -e <aeskeyfilename>.bin \
-i <inputimagefilename>.bin <encryptedimagefilename>.bin
lpc54xxx_imgcr.exe signimage -c <ikcertificatefilename>.bin \
-k <imagekeyfilename>.pem -i <encryptedimagefilename>.bin [-p] \
<enhancedimagefilename>.bin
```

Example:

```
lpc54xxx_imgcr.exe encimage -e aes_key.bin -i app_image.bin encrypted_image.bin lpc54xxx_imgcr.exe \
signimage -c image_key_cert.bin -k image_key.pem -i encrypted_image.bin enhanced_image.bin
```

9 Signed-encrypted and encrypted-signed images

An encrypted-signed image (signed first and then encrypted) is generated using the **signimage** and the **encimage** subcommands in sequence.

Example:

```
lpc54xxx_imgcr.exe signimage -c image_key_cert.bin -k image_key.pem -i app_image.bin signed_image.bin
lpc54xxx_imgcr.exe encimage -e aes_key.bin -i signed_image.bin signed_encryptedimage.bin
```

A signed-encrypted image (encrypted first and then signed) is generated using the **encimage** and the **signimage** subcommands in sequence.

Example:

```
lpc54xxx_imgcr.exe encimage -e aes_key.bin -i app_image.bin encrypted_image.bin

lpc54xxx_imgcr.exe signimage -c image_key_cert.bin -k image_key.pem \

-i encrypted_image.bin encrypted_signedimage.bin
```

10 Generation of key store

The **genkeystore** subcommand generates a key store image with the provided key codes and key store header parameters.

Command Format:

```
lpc54xxx_imgcr.exe genkeystore [-h] -a <activationcodefile>.bin \

-k <imagekeycodefilename>.bin -u <udskeycodefilename>.bin \

-f <fwupdatekeycodefilename>.bin [-e <aeskeyfilename>.bin] \

-i <inputimagefilename>.bin [--ks_offset <keystoreoffset>] \

[--ks_file <ksdatafilename>.bin] [--img_offset <image_offset>] \

[--puf_delay <puf_delay>] <keystoreimagefilename>.bin
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `-a or --activation` is a required argument to provide the path to the file containing activation code generated during PUF enrollment.
- `-k or --imgkeycode` is a required argument to provide the path to the file containing key code for AES-256 image key.
- `-u or --udskeycode` is a required argument to provide the path to the file containing key code for Unique Device Secret (UDS) key needed for DICE.
- `-f or --fwupdkeycode` is a required argument to provide the path to the file containing key code for firmware update key.
- `-e or --enck` is an optional parameter to provide the path to password protected AES key file. If input image provided in `-i` is not an encrypted image, then it will be encrypted.
- `-i or --image` is a required parameter to provide the path to the input binary image.
- `--ks_offset` is an optional argument to provide the key store offset relative to key store header.
- `--ks_file` is an optional argument to provide the path to an output file containing the key store data (only key codes). This is used with the `--ks_offset` parameter.
- `--img_offset` is an optional argument to provide image offset relative to key store header.
- `--puf_delay` is an optional argument to provide delay for PUF SRAM initialization. Delay is in microseconds. The default value is 240000 microseconds.
- `<keystoreimagefilename>.bin` is a positional argument to provide the output file path for the key store image.

11 Generation of random bytes

The **genrandom** subcommand generates a file containing the specified number of random bytes.

Command Format:

```
lpc54xxx_imgcr.exe genrandom [-h] [-c <bytecount>] <randombytesfilename>.bin
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `-c` or `--count` is an optional argument to provide the number of random bytes. The default count value is 52 bytes (256 bits).
- `<randombytesfilename>.bin` is a positional argument to provide the output file path of the generated random bytes.

Example:

```
lpc54xxx_imgcr.exe genrandom random_bytes.bin
```

The above example generates the file random_bytes.bin with 52 random bytes.

```
lpc54xxx_imgcr.exe genrandom -c 100 random_bytes.bin
```

The above example generates the file random_bytes.bin with 100 random bytes.

12 Obtain DICE ID

The **getdiceid** subcommand is used to calculate DICE ID based on the Unique Device Secret (UDS) and the image provided.

Command Format:

```
lpc54xxx_imgcr.exe getdiceid [-h] -u <udsfilename>.bin -i <inputimagefilename>.bin
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `-u or --udskey` is a required argument to provide the path to the file containing UDS in plain binary form.
- `-i or --image` is a required parameter to provide the path to the input binary image.

13 Show OTP values

The **showotp** subcommand is used to show the OTP values to be programmed for given RoT key and AES key.

Command Format:

```
lpc54xxx_imgcr.exe showotp [-h] -k <rotkeyfilename>.pem \
[-e <aeskeyfilename>.bin] [--usbvid <usb_vid>] \
[--usbpid <usb_pid>] [--usbpwr]
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `-k or --rotk` is a required argument to provide the path to the file containing RoT private key in PEM format.
- `-e or --enckey` is an optional argument to provide the path to the file containing AES encryption key.
- `--usbvid` is an optional argument to provide USB-IF vendor ID that LPC54S0xx ROM uses for DFU boot.
- `--usbpid` is an optional argument to provide USB-IF product ID that LPC54S0xx ROM uses for DFU boot. Bit0 indicates if device is bus(1) powered or self(0) powered.
- `--usbpwr` is an optional argument to provide the option to enumerate as USB bus powered device during DFU boot.

14 Generation of CRC

The **setcrc** subcommand is used to set CRC32 value in the image header. This subcommand is used to create unsecure plain load boot or XIP images with CRC checking.

Command Format:

```
lpc54xxx_imgcr.exe setcrc [-h] -i <inputimagefilename>.bin \
```

```
[-x <imagelength>] <signedimagefilename>.bin
```

Where:

- `-h` is an optional argument to display help menu for the command.
- `-i or --image` is a required parameter to provide the path to the input binary image.
- `-x or --xip` is an optional argument to provide the length of the image to be considered for calculating CRC of the XIP image.
- `<signedimagefilename>.bin` is a positional argument to provide the path to the output signed image file.

15 Revision history

The following table lists the substantive changes done to this document since the initial release.

Table 1. Revision history

Revision number	Date	Substantive changes
0	26 October 2021	Initial release

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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX,EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2021. **All rights reserved.**

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

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

Date of release: 26 October 2021

Document identifier: LPC54S0xxSICTUG

arm