MaximSDK–Secure Boot Tool User Guide

UG7236; Rev 0; 7/20

Abstract

This security user guide contains detailed information about the usage of Secure Boot Tool applications for the Low-Power Maxim microcontroller ICs with Secure ROM Feature. The document must be used in conjunction with the **MaximSDK Installation and Maintenance User Guide.**

Table of Contents

List of Figures

List of Tables

Introduction

The MaximSDK-Secure Boot Tool (SBT) is a software tool set containing command-line applications such as device supporting, Secure Communication Protocol (SCP) session building, signing binary files, and managing key/program load with source codes that helps customers to develop their own boot/load tools for production if needed.

This document provides general information on Secure ROM including secure boot and secure load into the Maxim Low-Power Microcontroller ICs family, a quick example for customers in the development stage, and an explanation of command-line applications in this package and usage examples of applications.

“Maxim Test Key” is used as the Customer Root Key (CRK) throughout this user guide and MAX32520 is selected as a sample IC for a clear example.

Note that the screenshots can differ according to the MaximSDK and SBT versions, CRKs, and selected IC, but the steps are the same.

Supported Chips

The MAX32520 and MAX32652 are supported by this documentation.

General Information on Secure ROM

Maxim Low-Power Microcontrollers with Secure ROM features come with a secure boot and a secure loader which are stored in the chip's internal ROM. Secure boot cannot be circumvented, and it checks the signature of the application to be run before running the application code in internal flash. Secure Loader also checks the signature of the downloaded data before using them.

The ROM application uses the Customer Root Key (CRK) for controlling digital signatures on boot and load. Therefore, customers should load the key in CRK space in One Time Programmable (OTP) area before using ICs and running their program.

In the development phase, customers can use the 'Maxim Test Key', which is used for all key options and applications in this user guide. Moreover, ICs in Maxim EV Kits already come preloaded with the "Maxim Test Key" to start evaluating and developing quickly. Customers must use their own keys in the production phase for security purposes.

This section summarizes select concepts of Secure ROM and SCP. For more information on ROM and communication, refer to the *Secure ROM User Guide* and *SCP Protocol Specification* documents of the related IC.

Secure ROM

The main objective for secure ROM code is to guarantee the chain of trust, from reset to the customer's first application.

At reset, the chip automatically checks the integrity of this ROM code and jumps to the beginning of this ROM to start executing its code. This ROM code can securely:

- Program the embedded flash
- Program the embedded OTP memory
- Start applications from embedded flash
- Load and run test programs using a flexible mechanism and applets loaded in internal RAM

Secure Update/Bootloader

The ROM code offers embedded flash and OTP secure updates. This secure update protocol can also program the internal OTP, used for memory and security configuration. Thanks to a very powerful mechanism of applet loading, this secure update can also load and run small programs in internal RAM. The links available for the secure download can be serial port (UART), SPI, I2C, and/or USB link according to parts.

Secure Boot

Depending on the OTP configuration, the ROM code loads, authenticates, and runs the second level application, i.e. the first customer application to be run after the ROM code. The digital signature verification guarantees that no illicit application can be run from the secure SoCs.

Keys Management

SCP keys are:

- Maxim Root Key (MRK):
    - Owned by Maxim
    - Private part managed securely (generation, storage, and use under dual control with HSM) at Maxim Secure Microcontrollers Business Unit used for customers' public keys certification
    - Public part stored within the ROM code
    - Used for CRK authentication and download
    - Used by ROM code for digital signature verification
- Customer Root Key (CRK):
    - Owned by the customer
    - Private part managed securely (generation, storage, and use under dual control with HSM) at customer premises, used for secure downloads
    - Public part stored within the OTP
    - Used by the secure loader in the ROM for secure data/firmware downloads in the terminal using digital signature verification
    - Stored with its MRK signature

Table 1. Keys Table

Key Name	Purpose	Owner	Generation	Location
MRK	CRK certification	Maxim	Any Secure Process	Private in HSM, public in OTP
CRK	SCP packets authentication	Customer	Any Secure Process	Private in HSM, public in OTP

Secure Boot Tool

The Secure Boot Tool consists of several different useful applications that can be used to sign binaries, create SCP sessions, and send packets over different communication interfaces with secure ROM application in Maxim LP uC parts.

Applications

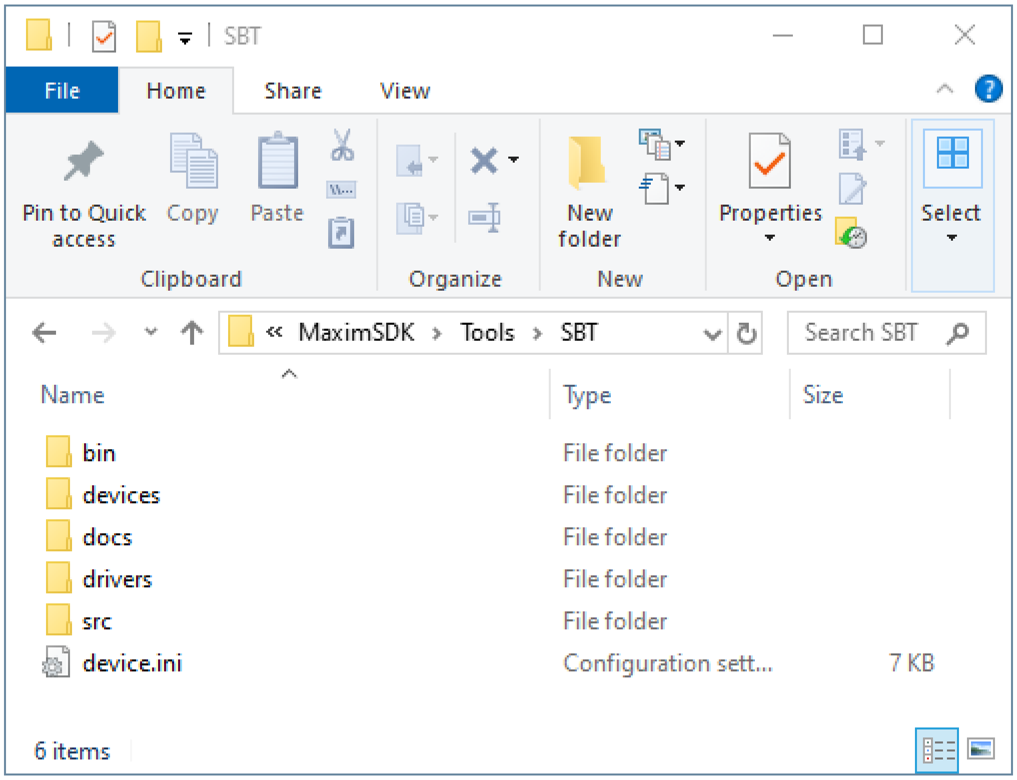
Command-line applications included in the Secure Boot Tool are listed as follows:

- SBT Device Parameters Applications: set/get/list_sbt_device commands
- Signing Binary Application: sign_app and ca_sign_build command
- Building SCP Session Application: build_scp_session command
- Send SCP Session Application: send_scp command

The ca_sign_build command is the same as the sign_app command, however, it is recommended to use sign_app in the user's environment. Details on how to use them are explained in the following chapters.

Folder Structure

After installing the MaximSDK, a new folder named SBT is created at the following path on the user's Windows® PC: C:\MaximSDK\Tools\SBT.



Name	Type	Size
bin	File folder	
devices	File folder	
docs	File folder	
drivers	File folder	
src	File folder	
device.ini	Configuration sett...	7 KB

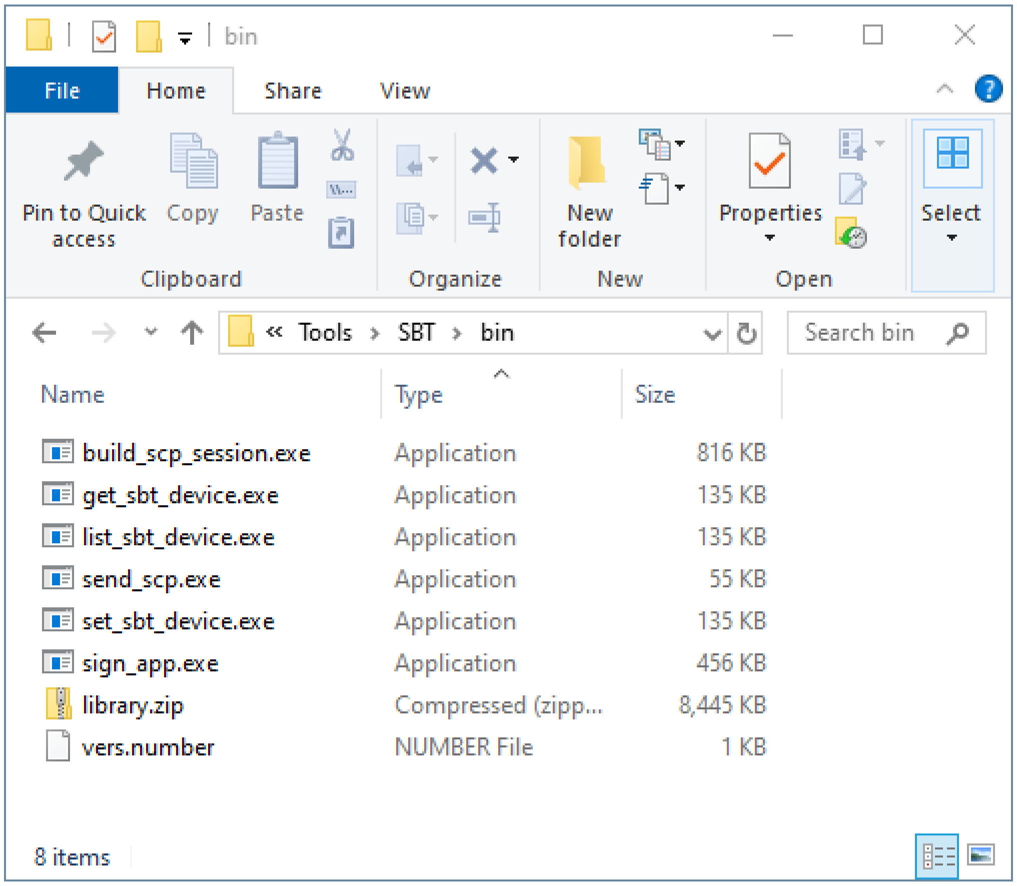
6 items

Figure 1. SBT Folder Content

Windows is a registered trademark and registered service mark of Microsoft Corporation

This folder structure contains the following subfolders:

- ***bin:*** all application executable files

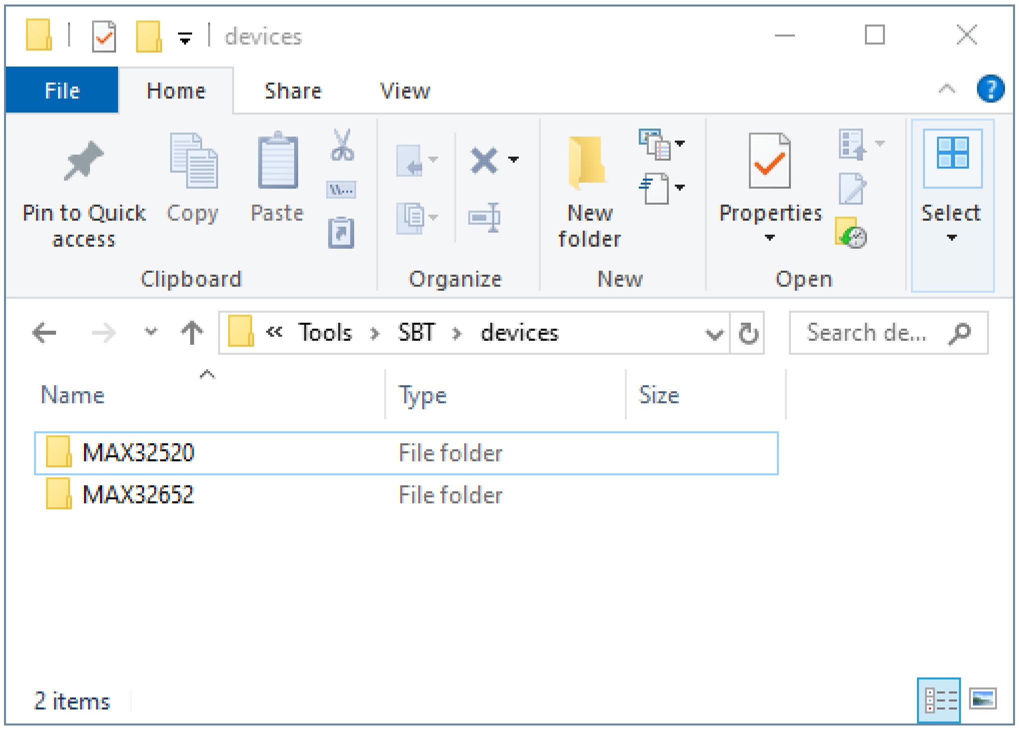


Name	Type	Size
build_scp_session.exe	Application	816 KB
get_sbt_device.exe	Application	135 KB
list_sbt_device.exe	Application	135 KB
send_scp.exe	Application	55 KB
set_sbt_device.exe	Application	135 KB
sign_app.exe	Application	456 KB
library.zip	Compressed (zipp...	8,445 KB
vers.number	NUMBER File	1 KB

Figure 2. bin Folder Content

- ***devices:*** supported parts folder and part specific files. The user can find out which parts are supported in their SBT version. devices/MAX32xxx folders have the following items:
    - keys: maximtestcrk key file for the development phase, all parts delivered in the EV kit are preloaded with this key
    - scp_packets: Ready to use SCP Session files such as writemaximcrk, dump_otp, and set_binary_location applet
    - scripts: Preloaded scripts for generating SCP session packets like write_sla, dump_otp, and timeout scripts files

For more details on the dump_otp usage case, refer to the *Dump OTP Applet User Guide.*



Name	Type	Size
MAX32520	File folder	
MAX32652	File folder	

Figure 3. devices Folder Content

- ***docs:*** Secure Boot Tool and related documentation provided by Maxim
- ***src:*** Source codes of sign_app, build_scp_session, and send_scp. The user can use these source codes as a reference when developing their programming tools and test applications in the production/manufacturing stage

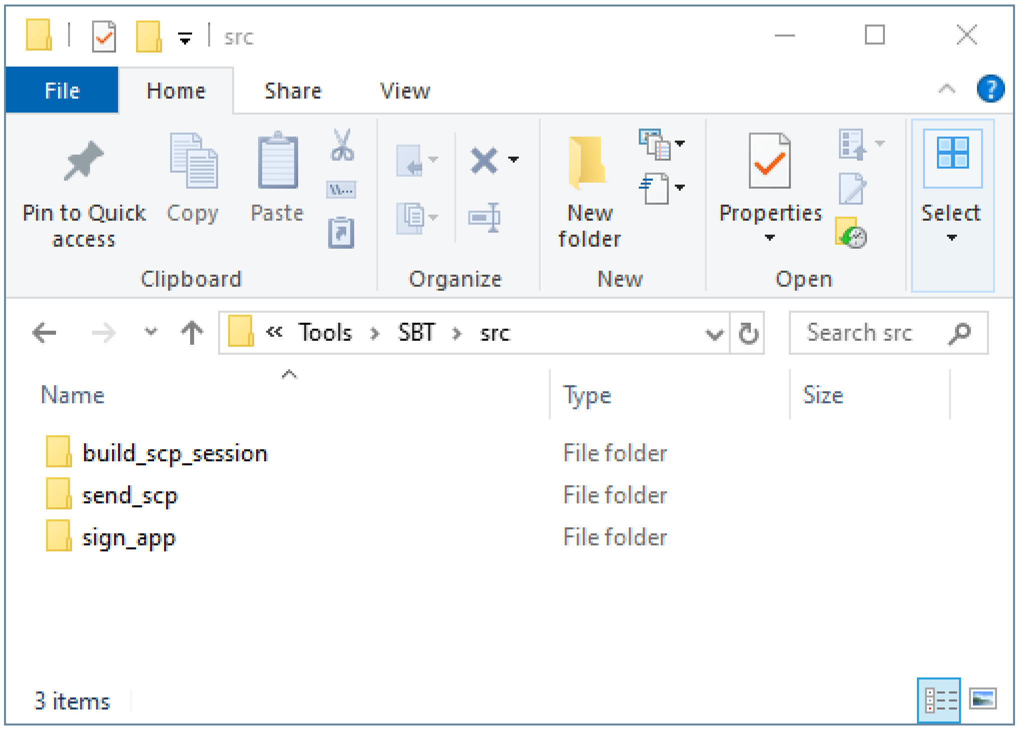
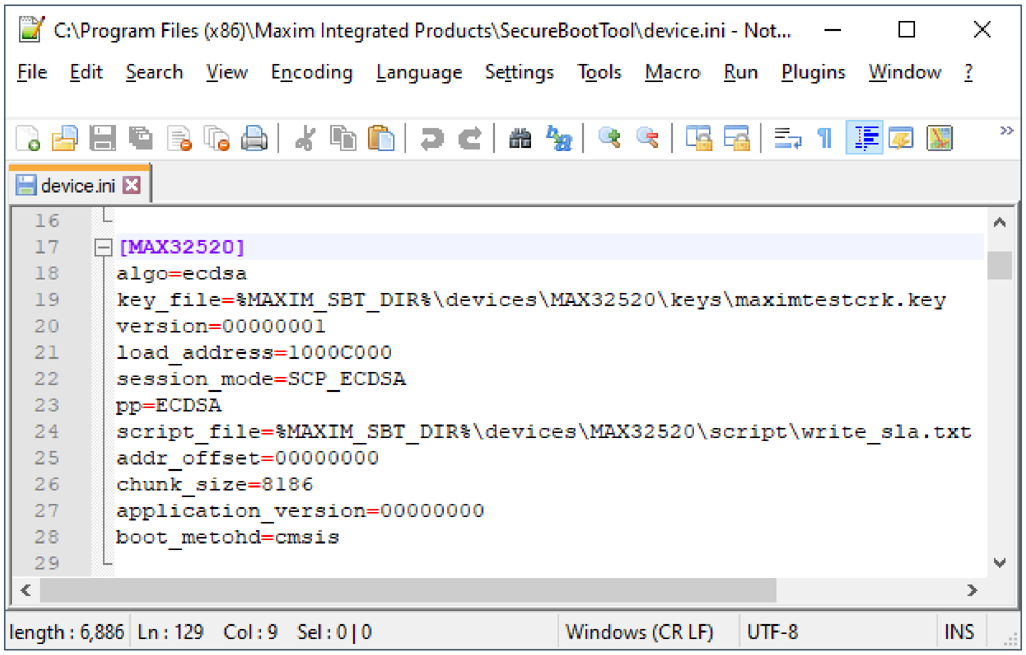


Figure 4. src Folder Content

device.ini File

Application defaults for each part are stored in this file. If the user does not specify the options and arguments of the command-line application, these default definitions are used during the application run-time. Parameter definitions are as follows:

- ***algo:*** Algorithm used to sign the file like ECDSA, RSA, etc.
- ***key_file:*** UCL format private key file path for SCP packet signing, keys\maximtestcrk.key is default
- ***version:*** Targeted Secure bootloader version, corresponding parts, and its revision
- ***load_address:*** Application binary loading address
- ***session_mode:*** SCP communication mode lie SCP_ECDSA, SCP_RSA, etc.
- ***pp:*** SCP protection profile to use: RSA_4096, RSA_2048, ECDSA
- ***script_file:*** Text file containing SCP operation to perform, script\write_sla.txt is default
- ***Addr_offset:*** Address offset when reading S19 or S20 files
- ***chunk_size:*** Maximum data size for one SCP packet (in bytes)


C:\Program Files (x86)\Maxim Integrated Products\SecureBootTool\device.ini - Not...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
device.ini
length : 6,886 Ln : 129 Col : 9 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

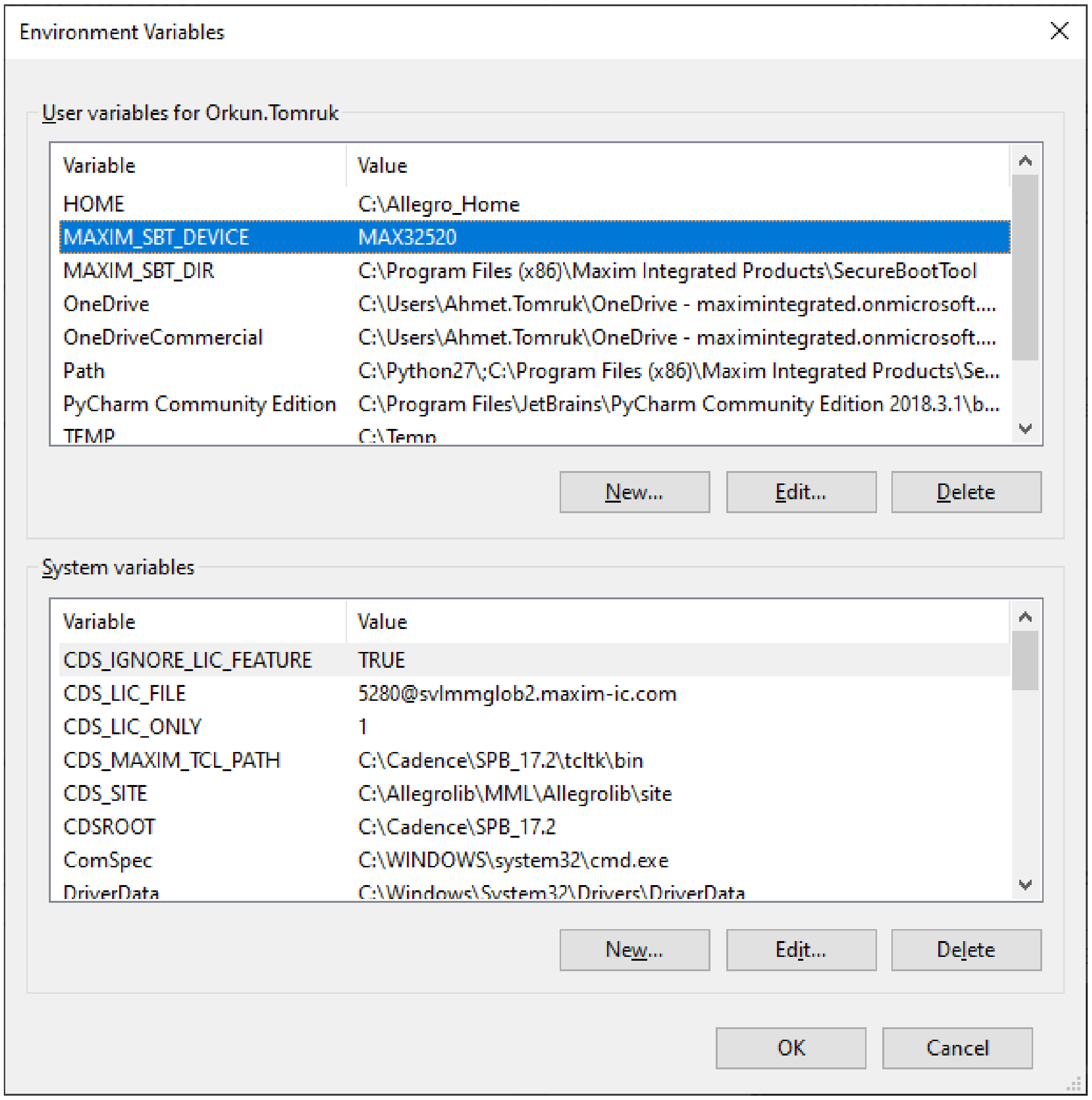
```
[MAX32520]
algo=ecdsa
key_file=%MAXIM_SBT_DIR%\devices\MAX32520\keys\maximtestcrk.key
version=00000001
load_address=1000C000
session_mode=SCP_ECDSA
pp=ECDSA
script_file=%MAXIM_SBT_DIR%\devices\MAX32520\script\write_sla.txt
addr_offset=00000000
chunk_size=8186
application_version=00000000
boot_metohd=cmsis
```

Figure 5. device.ini File Content Example

Use Case: Quick Startup New Parts

This use case shows how to load the CRK in OTP, then sign and load the application over SCP with the secure bootloader. In this use case, MAX32520 is used as the sample part and CRK is “Maxim Test Key” installed with MaximSDK installer.

1. Check the Windows system environment variable to ensure that “MAXIM_SBT_DEVICE” is MAX32520.



Variable	Value
HOME	C:\Allegro_Home
MAXIM_SBT_DEVICE	MAX32520
MAXIM_SBT_DIR	C:\Program Files (x86)\Maxim Integrated Products\SecureBootTool
OneDrive	C:\Users\Ahmet.Tomruk\OneDrive - maximintegrated.onmicrosoft....
OneDriveCommercial	C:\Users\Ahmet.Tomruk\OneDrive - maximintegrated.onmicrosoft....
Path	C:\Python27\;C:\Program Files (x86)\Maxim Integrated Products\Se...
PyCharm Community Edition	C:\Program Files\JetBrains\PyCharm Community Edition 2018.3.1\b...
TEMP	C:\Temp

Variable	Value
CDS_IGNORE_LIC_FEATURE	TRUE
CDS_LIC_FILE	5280@svlmmglob2.maxim-ic.com
CDS_LIC_ONLY	1
CDS_MAXIM_TCL_PATH	C:\Cadence\SPB_17.2\tcltk\bin
CDS_SITE	C:\Allegrolib\MML\Allegrolib\site
CDSROOT	C:\Cadence\SPB_17.2
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData

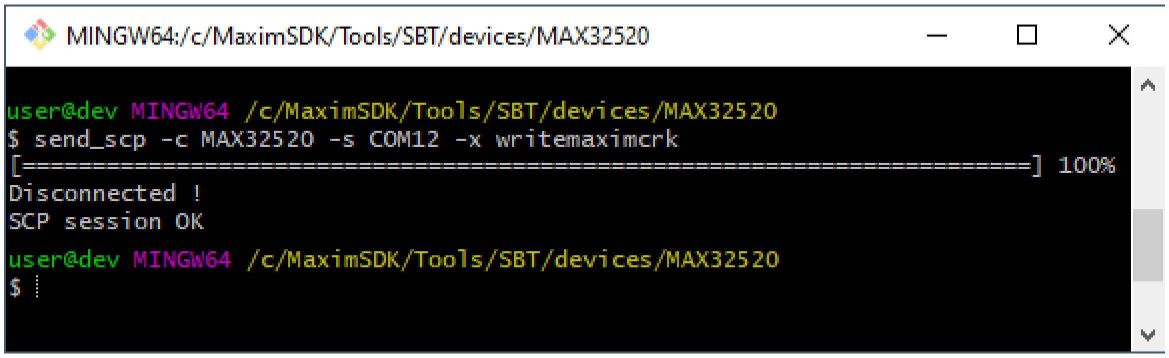
Figure 6. Environment Variable Window

2. Check and identify the COM port number of MAX32520. The user should change **COMxx** option in the following commands to communicate successfully.

3. Open a command line prompt like MinGW, Git Bash, or PowerShell on Windows.

4. Enter and run the following command to load the ‘maximtestcrk.key’ after resetting MAX32520.

   send_scp -c MAX32520 -s COM12 -x writemaximcrk



MINGW64:/c/MaximSDK/Tools/SBT/devices/MAX32520

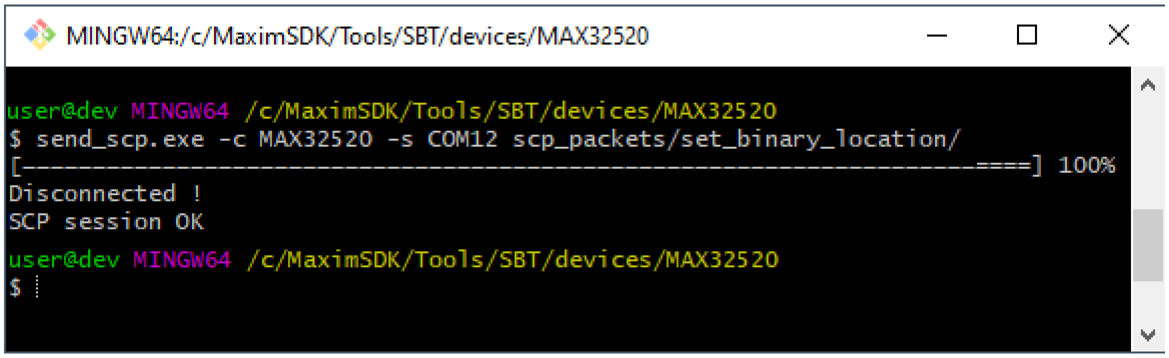
```
user@dev MINGW64 /c/MaximSDK/Tools/SBT/devices/MAX32520
$ send_scp -c MAX32520 -s COM12 -x writemaximcrk
[==========================================================================] 100%
Disconnected !
SCP session OK

user@dev MINGW64 /c/MaximSDK/Tools/SBT/devices/MAX32520
$ 
```

Figure 7. Load CRK to New Part with send_scp Application

5. Enter and run the following command to load the default SLA start address after resetting MAX32520.

   send_scp.exe -c MAX32520 -s COM12 scp_packets/set_binary_location/



MINGW64:/c/MaximSDK/Tools/SBT/devices/MAX32520

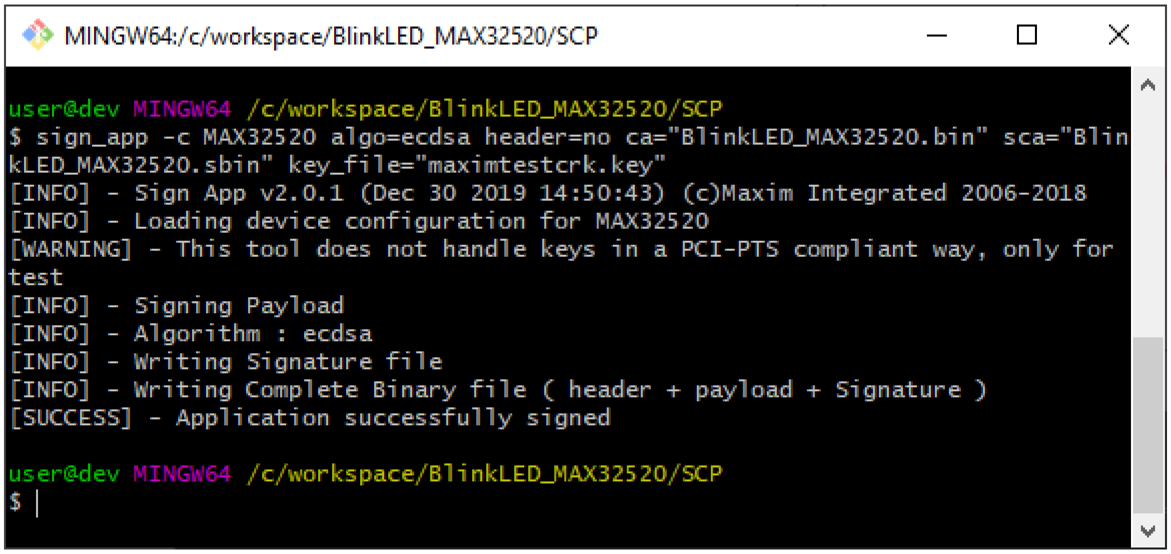
```
user@dev MINGW64 /c/MaximSDK/Tools/SBT/devices/MAX32520
$ send_scp.exe -c MAX32520 -s COM12 scp_packets/set_binary_location/
[----------------------------------------------------------------------====] 100%
Disconnected !
SCP session OK

user@dev MINGW64 /c/MaximSDK/Tools/SBT/devices/MAX32520
$ 
```

Figure 8. Load SLA Binary Start Location to New Part with send_scp Application

6. If application binary, **bin** file extension, is built in MaximSDK, signed binary with **sbin** file extension is generated automatically. If not, the user should sign their binary to get a runnable flash image. To sign the application, enter the following command.

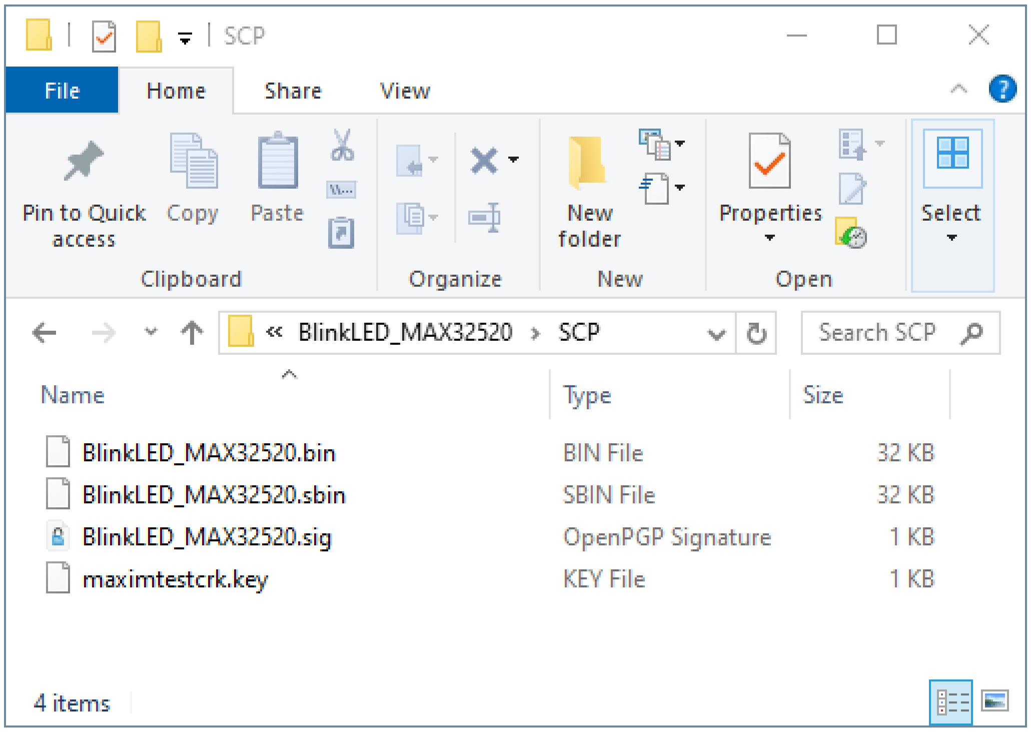
   sign_app -c MAX32520 algo=ecdsa header=no ca="BlinkLED_MAX32520.bin" sca="BlinkLED_MAX32520.sbin" key_file="maximtestcrk.key"



```
user@dev MINGW64 /c/workspace/BlinkLED_MAX32520/SCP
$ sign_app -c MAX32520 algo=ecdsa header=no ca="BlinkLED_MAX32520.bin" sca="Blin
kLED_MAX32520.sbin" key_file="maximtestcrk.key"
[INFO] - Sign App v2.0.1 (Dec 30 2019 14:50:43) (c)Maxim Integrated 2006-2018
[INFO] - Loading device configuration for MAX32520
[WARNING] - This tool does not handle keys in a PCI-PTS compliant way, only for
test
[INFO] - Signing Payload
[INFO] - Algorithm : ecdsa
[INFO] - Writing Signature file
[INFO] - Writing Complete Binary file ( header + payload + Signature )
[SUCCESS] - Application successfully signed

user@dev MINGW64 /c/workspace/BlinkLED_MAX32520/SCP
$ |
```

Figure 9. Signing Binary with sign_app Application



Name	Type	Size
BlinkLED_MAX32520.bin	BIN File	32 KB
BlinkLED_MAX32520.sbin	SBIN File	32 KB
BlinkLED_MAX32520.sig	OpenPGP Signature	1 KB
maximtestcrk.key	KEY File	1 KB

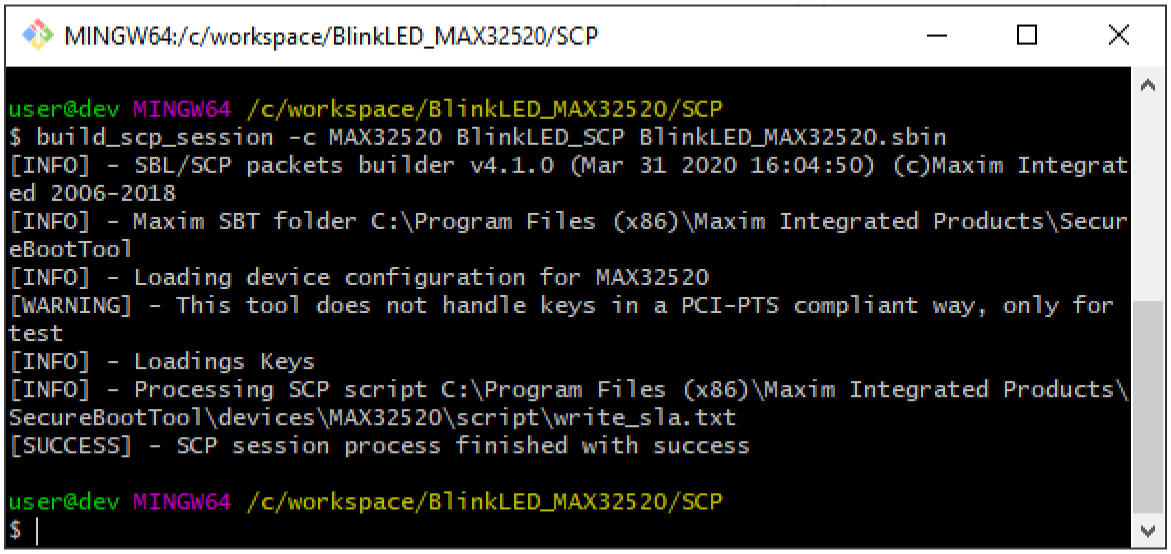
Figure 10. Folder Content After Running sign_app

- In this guide, '*maximtestcrk.key*' uses as CRK key, the user should change ***key_file*** parameter file with their CRK file to get **sbin** file signed with their key

- While MAX32520 key signing uses ECDSA algorithm, be sure to use the key signing algorithm according to IC. Generally Maxim Cortex-M cores use ECDSA or RSA algorithm. Moreover, the "device.ini" file in the 'C:\Program Files (x86)\Maxim Integrated Products\SecureBootTool' folder has the algo lines to specify the algorithm corresponding to IC part as shown in **Figure 5.**

7. To generate the SCP session packets to program the application, enter the following command. After a successful run, the **BlinkLED_SCP** folder is created in the main folder.

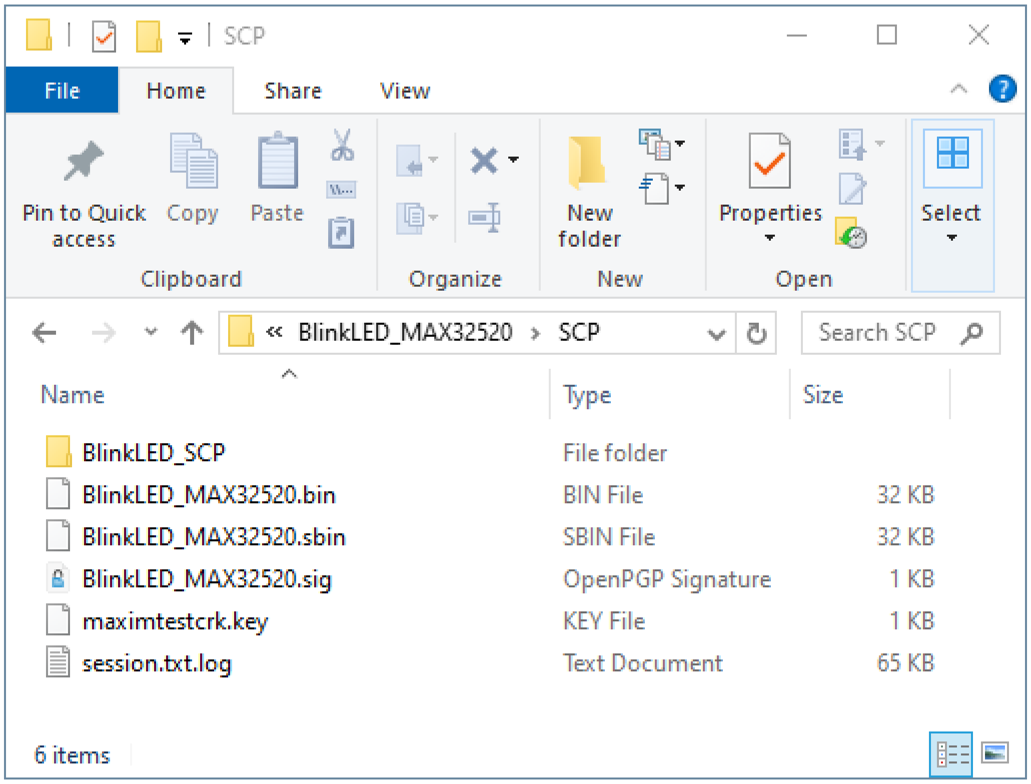
   build_scp_session -c MAX32520 BlinkLED_SCP BlinkLED_MAX32520.sbin



```
user@dev MINGW64 /c/workspace/BlinkLED_MAX32520/SCP
$ build_scp_session -c MAX32520 BlinkLED_SCP BlinkLED_MAX32520.sbin
[INFO] - SBL/SCP packets builder v4.1.0 (Mar 31 2020 16:04:50) (c)Maxim Integrat
ed 2006-2018
[INFO] - Maxim SBT folder C:\Program Files (x86)\Maxim Integrated Products\Secur
eBootTool
[INFO] - Loading device configuration for MAX32520
[WARNING] - This tool does not handle keys in a PCI-PTS compliant way, only for
test
[INFO] - Loadings Keys
[INFO] - Processing SCP script C:\Program Files (x86)\Maxim Integrated Products\
SecureBootTool\devices\MAX32520\script\write_sla.txt
[SUCCESS] - SCP session process finished with success

user@dev MINGW64 /c/workspace/BlinkLED_MAX32520/SCP
$ |
```

Figure 11. Generating SCP Packets with build_scp_session Application

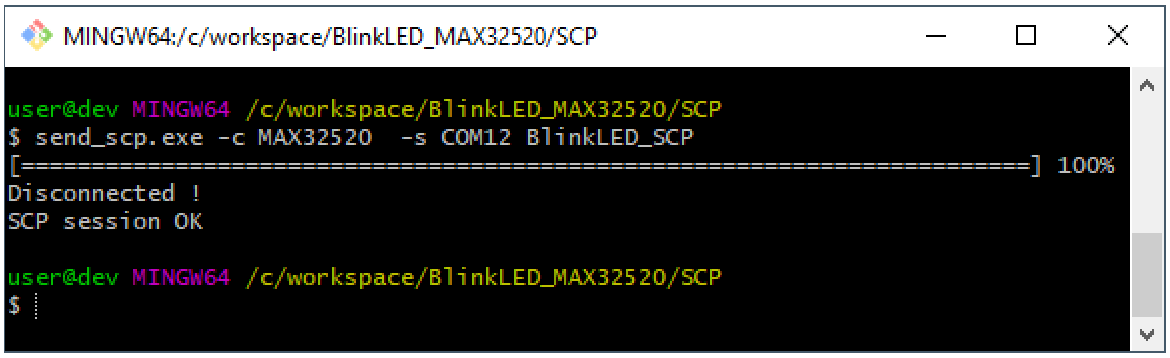


Name	Type	Size
BlinkLED_SCP	File folder	
BlinkLED_MAX32520.bin	BIN File	32 KB
BlinkLED_MAX32520.sbin	SBIN File	32 KB
BlinkLED_MAX32520.sig	OpenPGP Signature	1 KB
maximtestcrk.key	KEY File	1 KB
session.txt.log	Text Document	65 KB

Figure 12. Folder Content After Running build_scp_session

8. Execute the following command and reset the chip to start the Secure Bootloader. After loading successfully, the LED on the MAX32520 EV Kit blinks after reset.

   send_scp.exe -c MAX32520 -s COM12 BlinkLED_SCP



```
user@dev MINGW64 /c/workspace/BlinkLED_MAX32520/SCP
$ send_scp.exe -c MAX32520  -s COM12 BlinkLED_SCP
[============================================================================] 100%
Disconnected !
SCP session OK

user@dev MINGW64 /c/workspace/BlinkLED_MAX32520/SCP
$
```

Figure 13. Loading Application Binary over SCP Session Packets with send_scp Application

Revision History

REV NUMBER	REV DATE	DESCRIPTION	PAGES CHANGED
0	7/20	Initial release	—