Secure Boot Tools User Guide

UG7637; Rev 0; 6/22

Abstract

The Secure Boot Tools (SBT) is a tool set that provides signed image, generates packages which is sent to the Secure Boot Loaders (SBL). This user guide contains detailed information about the usage of SBT applications for the Maxim microcontroller ICs with secure ROM feature.

Table of Contents

List of Figures

List of Tables

Introduction

All the MAX32xxx secure microcontrollers come with a secure boot and loader stored in the chip internal ROM. The secure boot cannot be circumvented. It checks the signature of the application to run before running the application code in internal flash. The secure boot loader also checks the signature of the downloaded data before using it.

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

The customer may use the Maxim Test Key in the development phase. All the key options and applications in this user guide use the Maxim Test Key. Moreover, ICs in the Maxim EV kits already come with the loaded Maxim Test Key to start evaluating and developing quickly. The customers must use their own keys in the production phase to satisfy the PCI-PTS standards. For more information about secure ROM, see the section *Appendix A: Secure ROM*.

The SBT is a software package that includes all binaries and utilities to sign application, generate Secure Communication Protocol (SCP) packages and send them to the SBL.

The SBT includes:

- GUI(Graphical User Interface).
- SCP sender (scp_sender.exe).
- SCP package builder (build_scp_session.exe).
- Image signer (sign_app.exe).
- Prebuilt SCP packages (dump_otp, writemaximcrk, …).
- Reference SCP script files.
- Source codes.

The main path that a user needs to know:

- The SBT documents: *"<YourInstallationPath> \ docs",*
- The SBT source code: *"<YourInstallationPath> \ src",*
- The SBT devices package: *"<YourInstallationPath> \ devices",*
- Prebuild scp packages: *"<YourInstallationPath> \ devices \ PartNumber \ scp_packets",*

Default installation path is:

"C:\Program Files (x86)\Maxim Integrated Products\SecureBootTool"

The purpose of this document is to explain the SBT components over GUI.

SBT GUI

The SBT GUI provides a simple interface to use SBT binaries over the GUI. The GUI provides the services mentioned below:

- Sign application.
- Generate SCP Packages.
- Send it to SBL.

The SBT installer links GUI on the startup menu. To launch the ADI Secure Boot Tools, in the windows search box, type ADI Secure Boot Tools as shown in **Figure 1**.

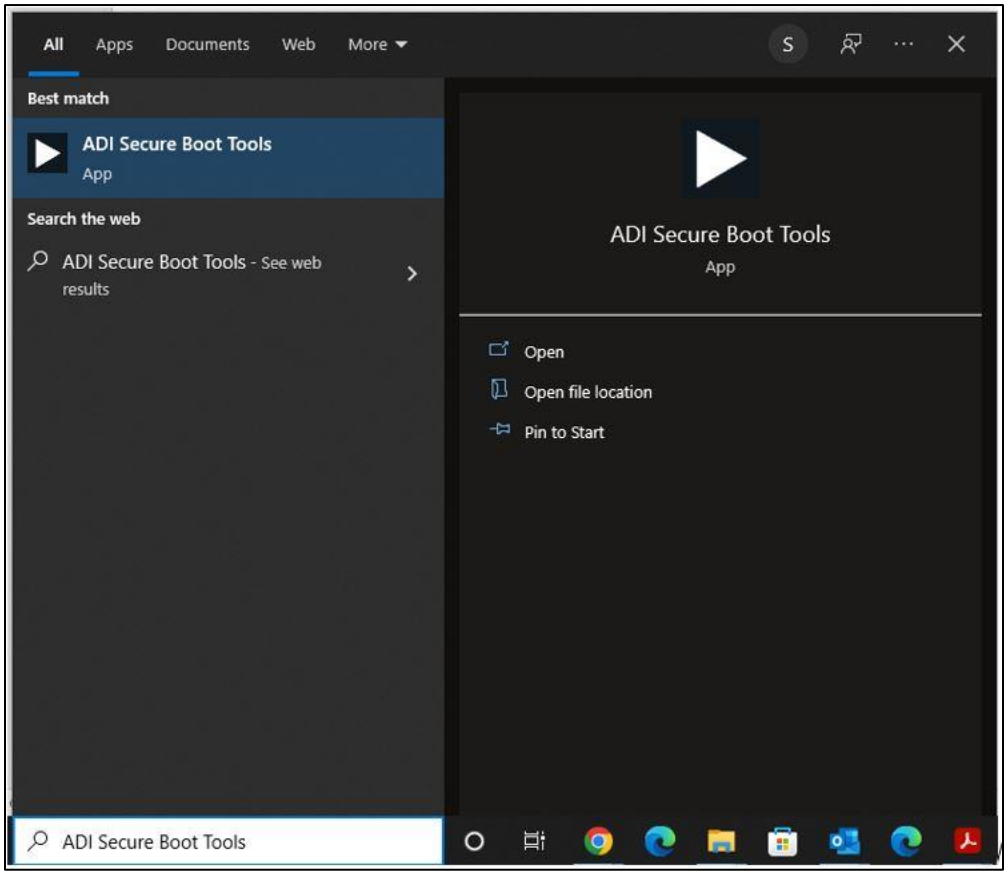

Figure 1. GUI on startup.

The GUI has a dedicated tab for each SBT function. On the next pages, these tabs are explained in detail.

SCP Sender

The SCP Sender is used to communicate with SBL, it sends and receives SCP frames to SBL. The “scp_sender.exe” binary handles this communication which is triggered either through the GUI or command line. The user should select the items given below to trigger the operation through GUI:

- Device part number.
- Communication interface (UART, USB …).
- Comport.
- SCP Package file (.zip or .list file).

To restart your board, click the **Start** button as shown in **Figure 2**. The output appears in the “Output Terminal” screen.

Note: Dependent on the IC, press the stimulus pin to the device to go to the boot loader mode during bootup.

For more information, check the device documentation.

Additionally, if the device boots over the USB, the related comport appears only while booting. During bootup, the SBL initializes the USB interface as Communication Device Class (CDC) and on the PC side it appears as a comport. At the end of the bootup, it got terminated and the comport disappears, so that while using the USB interface during bootup, check the comport number (by checking the Comport drop-drown menu on the GUI).

To see more information during communication, enable the “Verbose” mode on the GUI.

Figure 2 shows an example of the MAX32520.

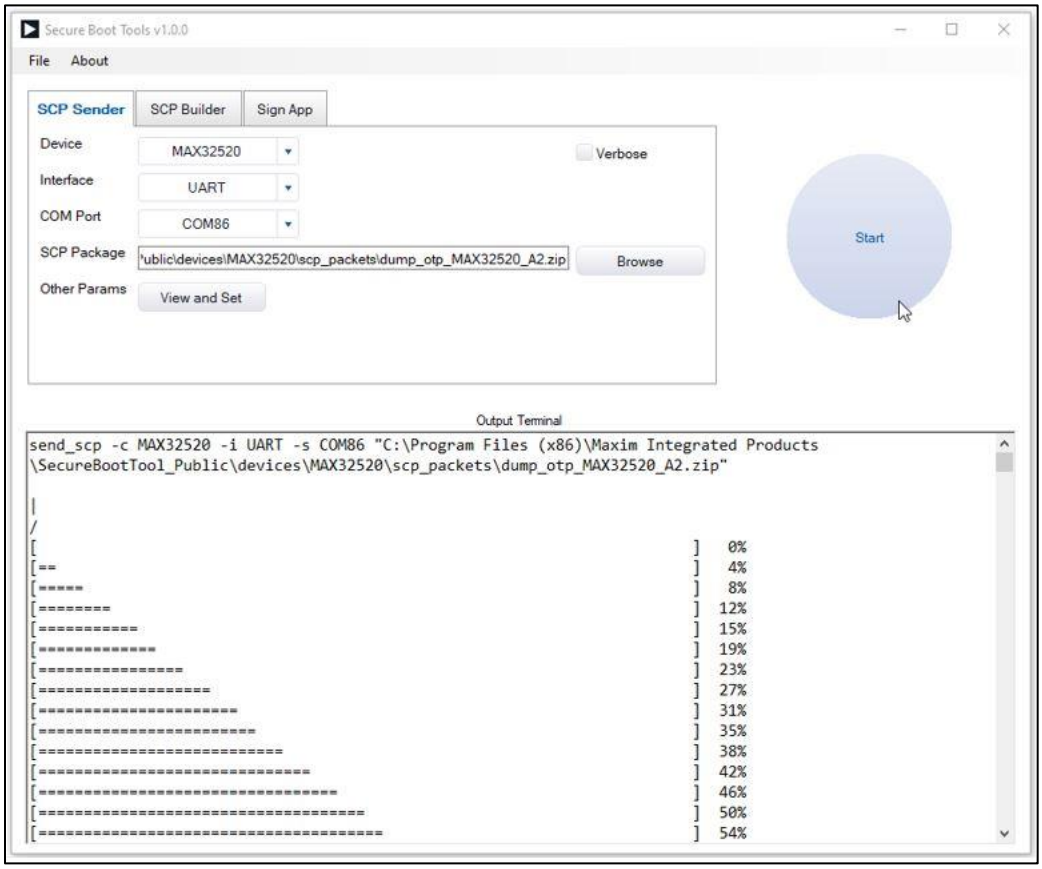

Figure 2. SCP sender GUI screen.

If an error occurs, check the section *Appendix B: SBT System Parameters*.

The command line tools are also capable of executing the same operation. It is also valid for SCP builder and Sign App as well. To do this, select and copy (by right-click + copy) the command on the output terminal, open Windows PowerShell. Then paste and execute it as shown in **Figure 3** and **Figure 4**.

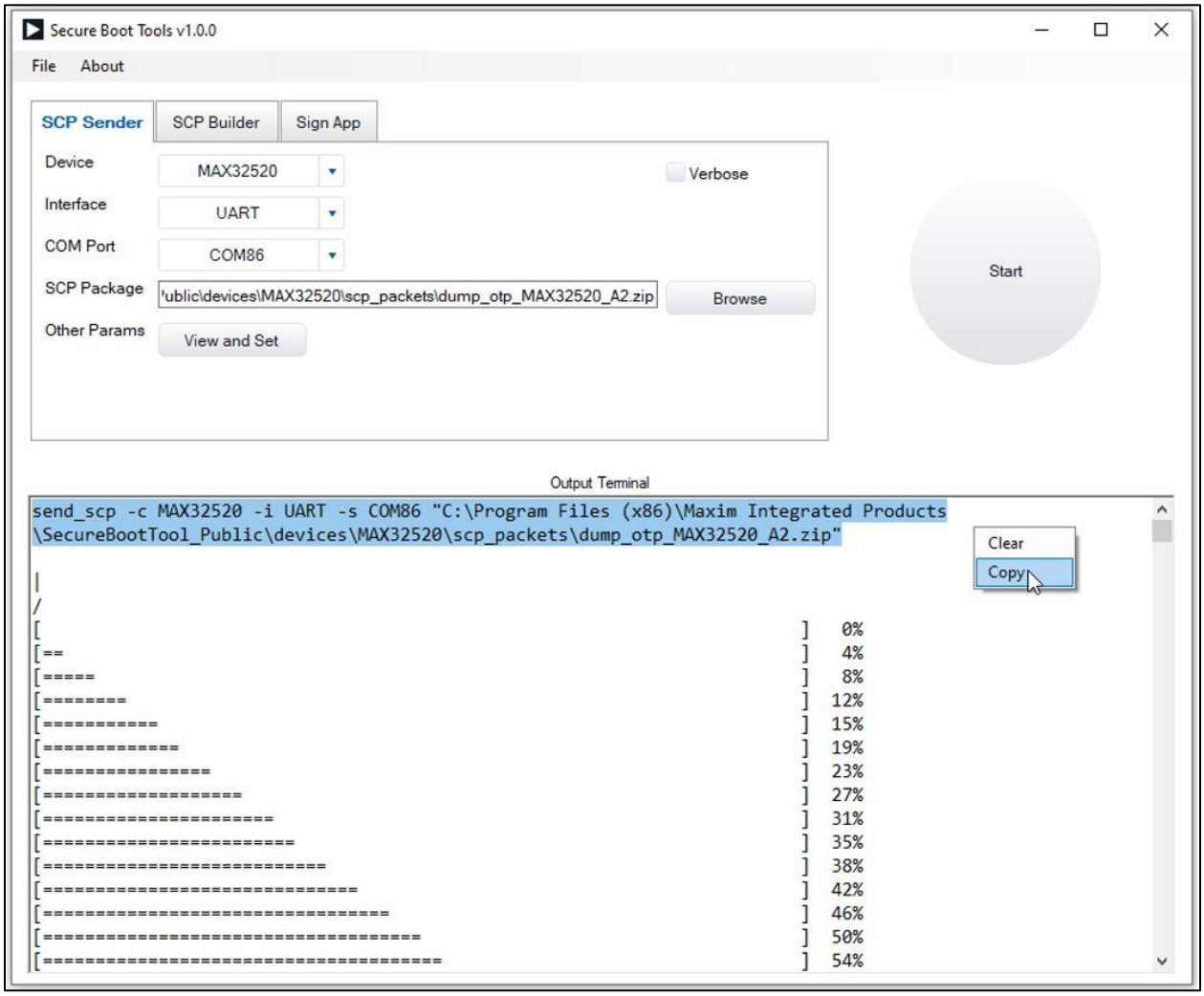

Figure 3. SCP sender command selection.

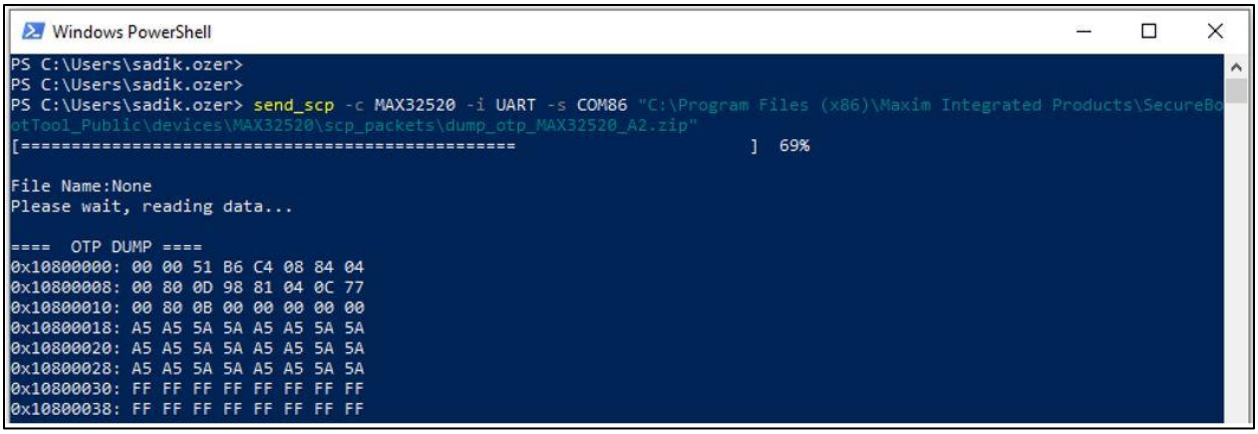

Figure 4. Execution SCP sender command on console.

The SCP sender has some advanced parameter (see **Figure 5**) that most users do not require access. To access it, click the **View and Set** button. An explanation of advance parameters is shown below:

- Timeout (s): Max timeout to get response on device side.
- Erase Timeout (s): Timeout for erase memory command.
- First Retry Count: Max number of retry to get response for the first packet.
- Retry Count: Number of retry for each packet.
- Packet Delay (ms): Delay between each packet.
- Auto Reset: To reset target over RTS before SCP session.
- Dump File Name: File name to store output of dump packet.

Also, the widgets have a tooltip that appears when the mouse hovers over the widget for a while.

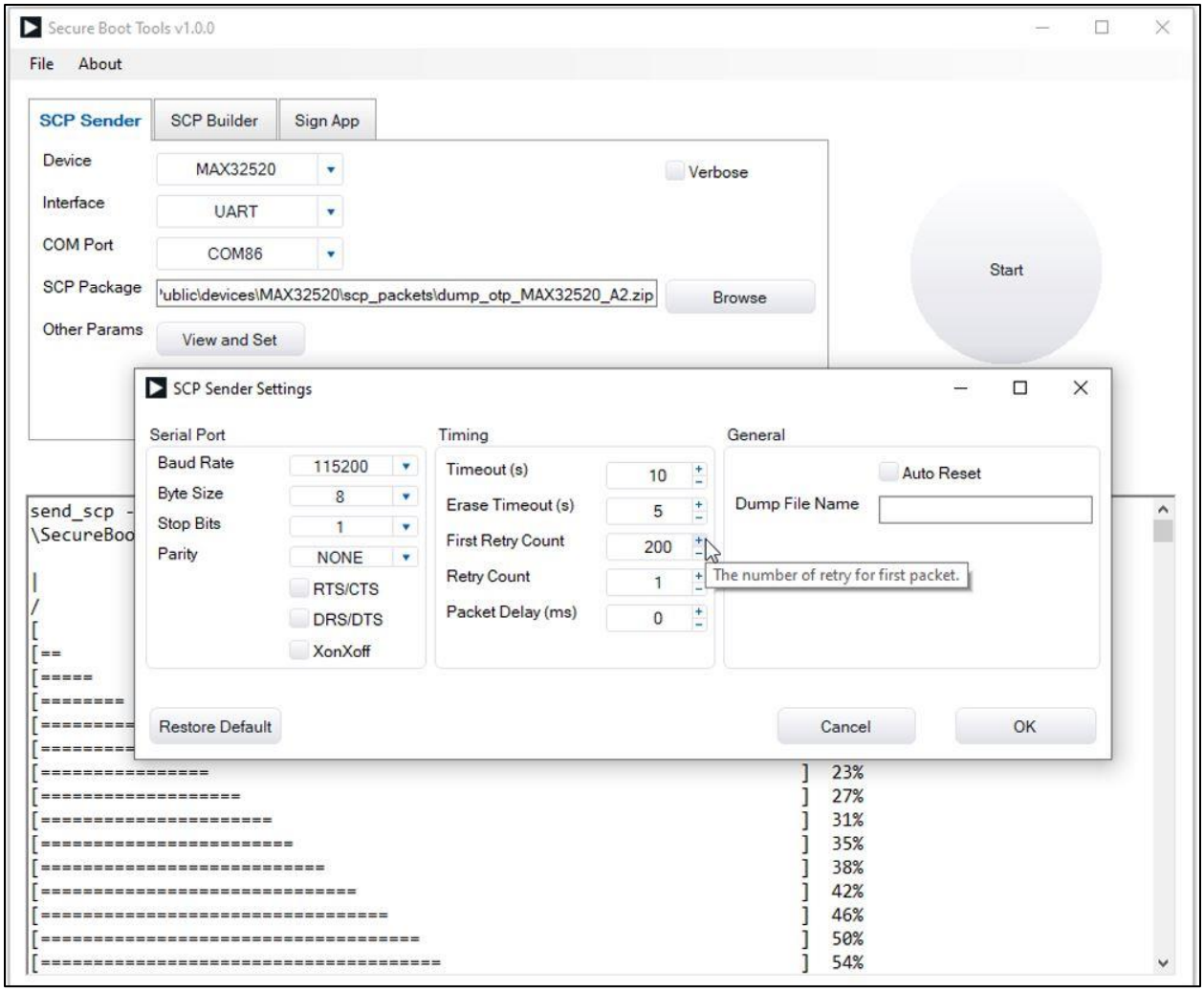

Figure 5. SCP sender advance parameters.

SCP Builder

The SCP Builder is used to generate the SCP package from scp script or image file (.sbin). Depending on the operation, it generates the SCP frames and sign them with CRK (Customer Root Key). The “build_scp_session.exe” binary handles this operation. This binary is triggered either through GUI or command line. To trigger it through GUI, select the following:

- Device part number.
- Key File (Leave it empty to use Maxim Test Key).
- Signed Binary File (*.sbin).
- Output location (where the SCP output will be put).

Then click the **Start** button as shown in **Figure 6**. The output appears on “Output Terminal” screen.

To see more information during communication, enable the “Verbose” mode on the GUI.

Figure 6 shows an example of the MAX32520.

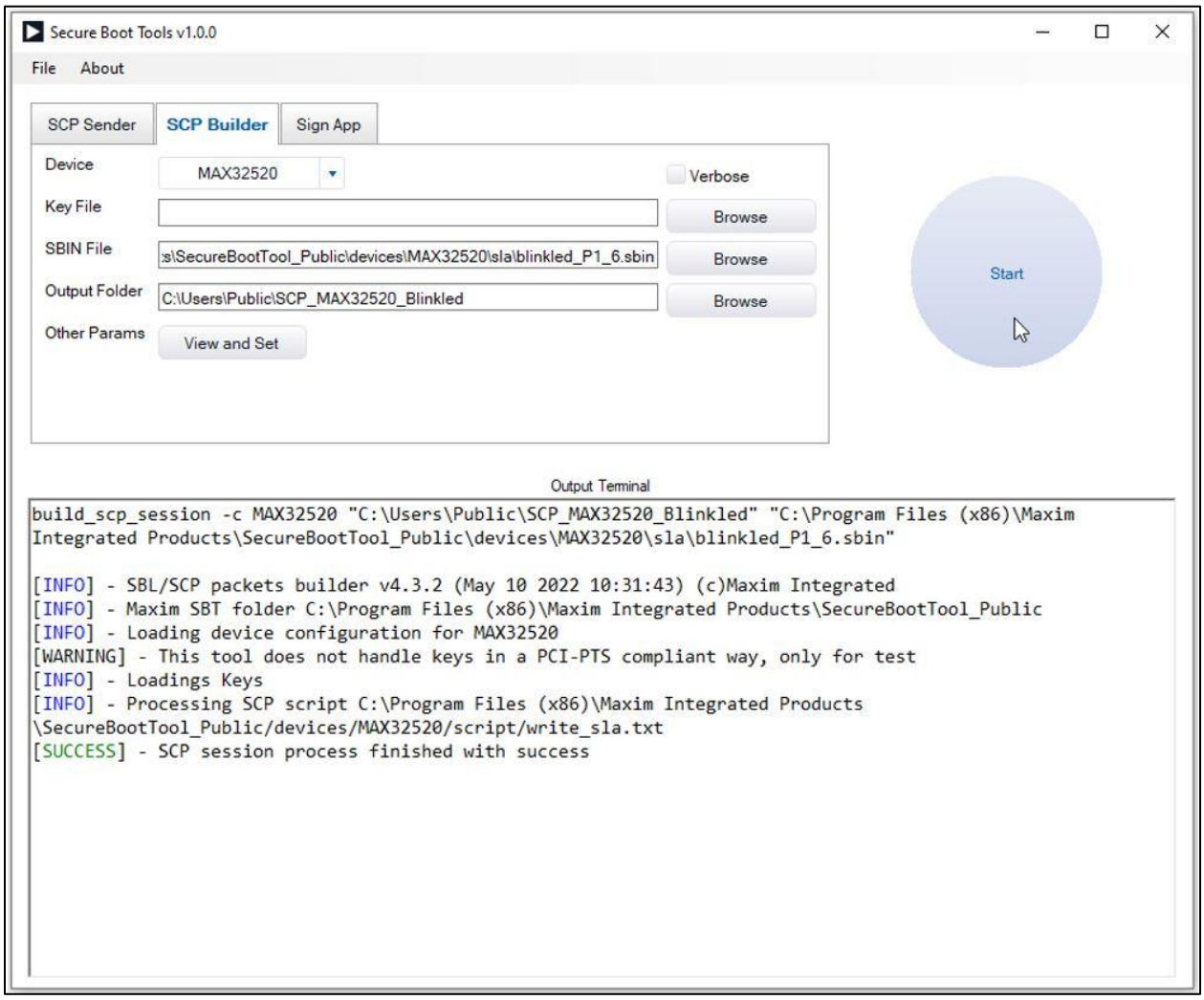

Figure 6. SCP builder GUI screen.

The SCP builder has some advanced parameters, click the **View and Set** button to change them as shown in **Figure 7**. An explanation of the parameters is shown below:

- Script File: Text file that contains SCP command to perform. For more information, see the section *Appendix C: SCP Script Commands*.
- Session mode: SCP mode to be used for the communication with SBL.
- PP mode: SCP Protection profile mode to be used for the communication with SBL.
- Output File Prefix: Prefix that adds on the SCP file generated.

- USN: 13 bytes Unique Serial Number of the device that needs to personalize the SCP session for (i.e., kill-chip command).

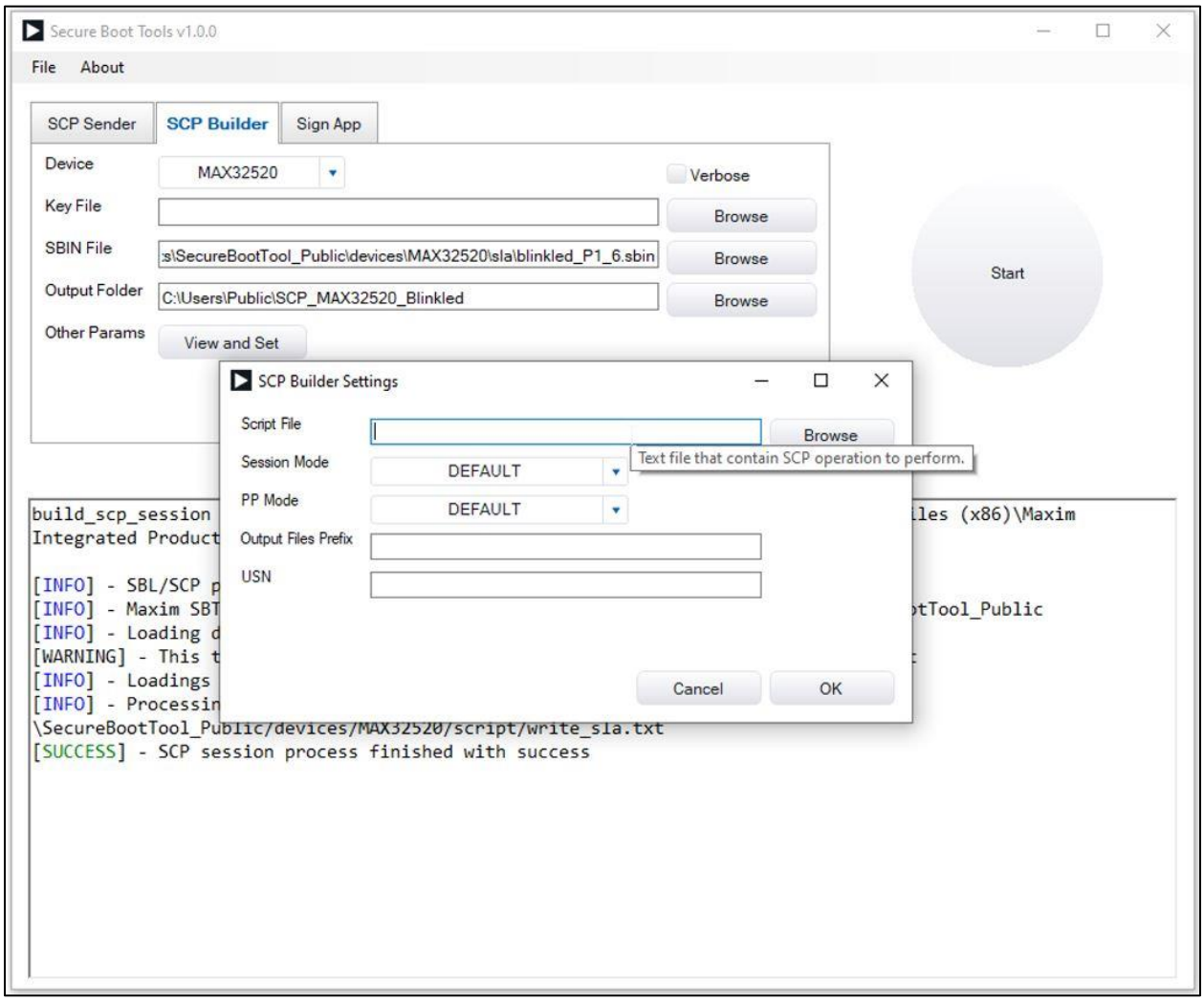

Figure 7. SCP builder advance parameters.

Sign App

The Sign App is used to sign an image file (.bin) and generate .sbin file (signed binary file). The "sign_app.exe" binary handles this operation. This binary is triggered either through the GUI or the command line. To trigger it through the GUI, select the following:

- Device part number.
- Key File (Leave it empty to use Maxim Test Key).
- Binary File (*.bin).
- Output location (where the *.sbin will be put).

Then click the **Start** button as shown in **Figure 8**. The output appears on "Output Terminal" screen.

To see more information during communication, enable the "Verbose" mode on the GUI.

Figure 8 shows an example of the MAX32520.

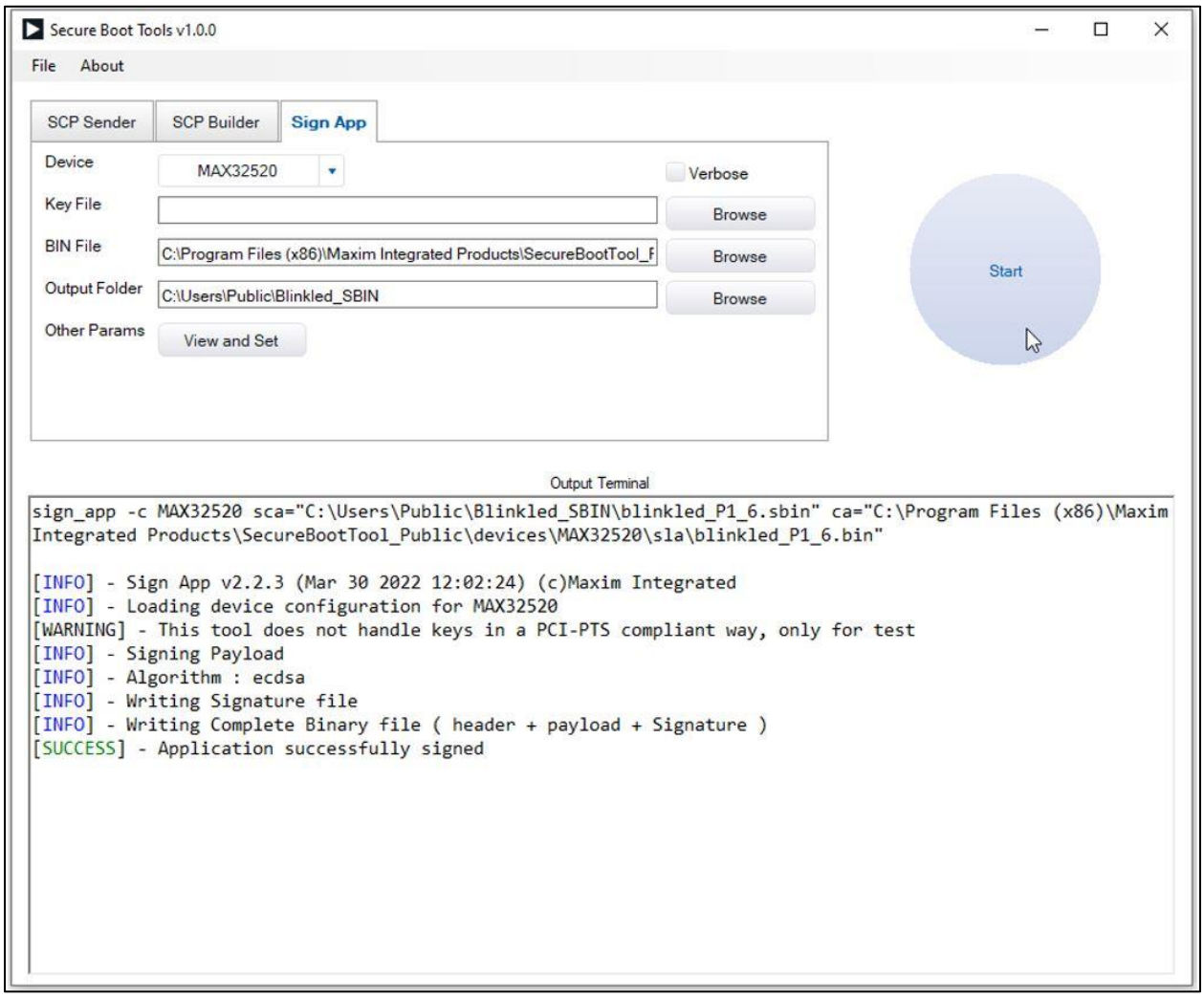

Figure 8. Sign App GUI screen.

The Sign App has some advanced parameter, click the **View and Set** button to change them as shown in **Figure 9**. The Sign App Settings screen has the following fields:

- Algo: Algorithm used to sign the file.
- Sign Only: If set to yes only .sig file generates, no .sbin file.
- App Header: Set it yes if binary (.bin) file does not have header section. “Yes” means generate header, below parameters requires to generate header if “Yes” selected.
- Boot Method:
  - CMSIS - The Jump address points to the value to the Stack pointer followed by the address of the reset handler. The boot loader sets up the Stack pointer and then jump to the "reset handler*.
  - DIRECT -The bootloader will directly jump to the Jump address and the application is responsible to setting up the stack.
- App Version: Version of the application, 4 bytes hexadecimal encoded (ex: 012AC567).
- Load Addr: Address of the location where the application copies before executed, 4 bytes hexadecimal encoded (ex: 012AC567).
- Jump Addr: Address of the instruction where the bootloader jumps, 4 bytes hexadecimal encoded (ex: 012AC567).
- ROM Version: Version of the targeted Bootloader. To select the correct one, 4 bytes hexadecimal encoded (ex: 012AC567), refer to the CHIP documentation.

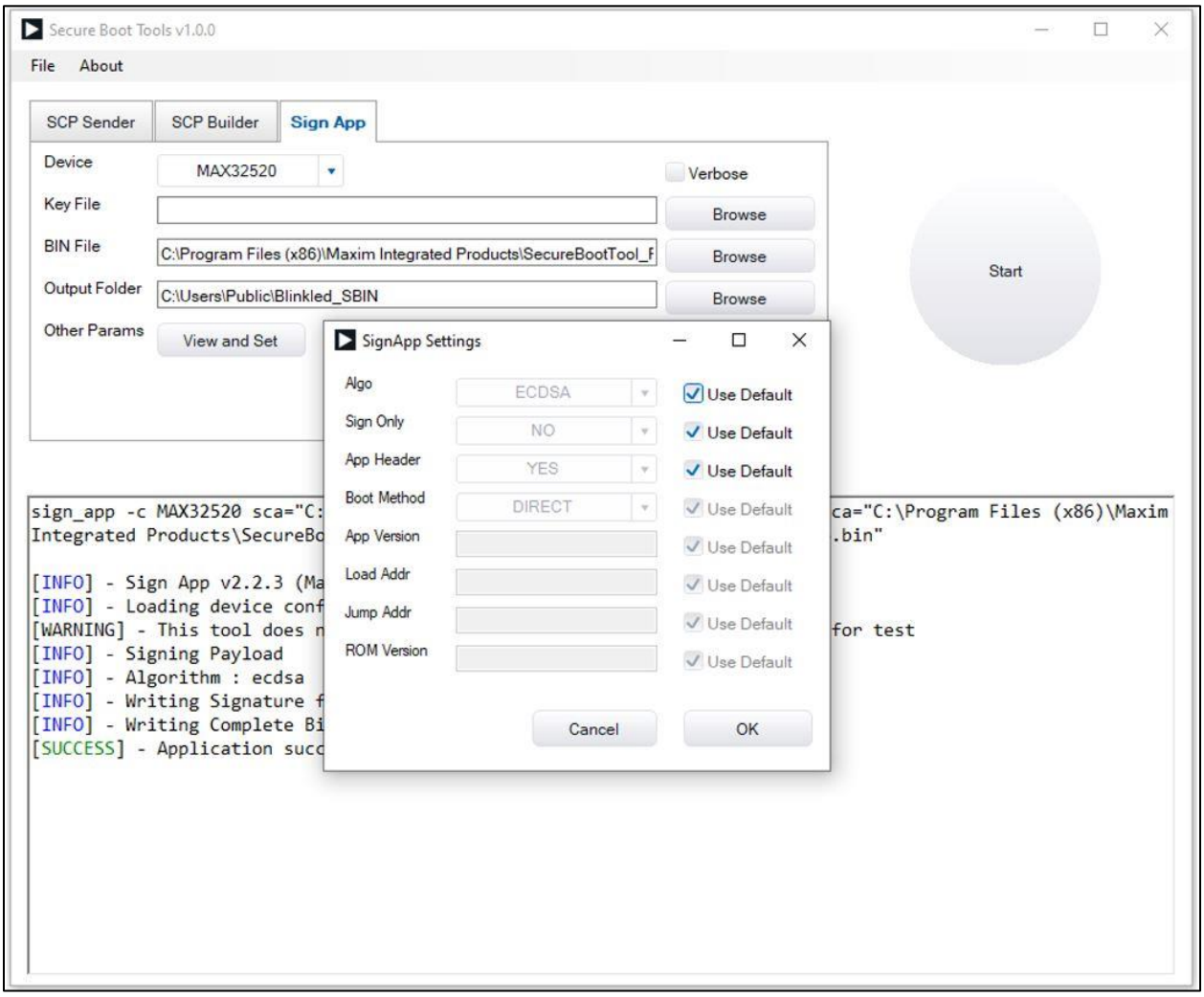

Figure 9. Sign App GUI screen.

Appendix A: Secure ROM

The MAX32xxx family Secure SoCs embed a ROM on the die. The main objective of this ROM code is to guarantee the chain of trust, from reset to the first application of the customer.

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

- Manage the chip life cycle.
- 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 the internal RAM.

The Life Cycles of Maxim Secure ICs

The life cycle of the chip has five phases:

- Phase 0:
  - Conception: Software development and hardware design.
  - Wafer: Wafer manufacturing.
- Phase 1:
  - EWS: Electric Wafer Sort (Flash Test Mode and Test Mode are enabled).
  - Testing: Final testing and execution of the test applets loaded through the GPIOs.
- Phase 3: OTP locked: The Maxim OTP is locked, and device is tested. The TM is locked.
- Phase 4: In-the-field: The chip starts and runs customer applications.
- Phase 5: End-of-life: The chip does not start.

The SCP is available in phase 3 (for the transition to phase 4 only) and in phase 4 for customer needs. The ICs with the EV kits are already in phase 4, i.e., the Maxim Test Key is loaded in the CRK field of the OTP in production.

Secure Update/Bootloader

The ROM code provides embedded flash and OTP secure update. This secure update protocol also programs the internal OTP, used for memories and security configuration. This secure update loads and runs small programs in the internal RAM, thanks to a very powerful mechanism of applet loading. The links for the secure download are the serial port and USB link.

Secure Boot

The ROM code, depending on the OTP configuration, loads, authenticates, and runs the second-level application, i.e., when the first customer application to run after the ROM code has ended.

The digital signature verification guarantees that no unauthorized application can be run from the MAX325xx secure SoCs.

Keys Management

The SCP keys are:

- Maxim Root Key (MRK):
  - Owned by Maxim Integrated.
  - Private part managed securely (generation, storage, and use under dual control with HSM) at the Maxim Secure Micro-controllers BU; used to certify customers' public keys.
  - Public part stored within the ROM code.
  - Used for CRK authentication and download in phase 3.
  - Used by the ROM code to verify digital signature.
- 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 (phase 4) secure downloads.
  - Public part stored within the OTP.
  - Used by the secure loader in the ROM for secure data/firmware downloads on the terminal using digital signature verification.
  - Stored with its MRK signature.

Appendix B: SBT System Parameters

The SBT installation adds some parameters to the environment variables to make the binaries available from the command line by default. The lists of parameters added are given below:

- Adds MAXIM_SBT_DIR and MAXIM_SBT_DEVICE in system variable (as shown in **Figure 10**).
- Adds SBT binaries location in PATH variable (as shown in **Figure 11**).

The MAXIM_SBT_DIR is used by the SBT binaries to get the root location of SBT files.

The MAXIM_SBT_DEVICE is used as a default device to be able to call SBT binaries (sign_app.exe, build_scp.exe, send_scp.exe) without specifying the device name.

Note: If there are difficulties in using SBT, then make sure these parameters are defined on the system.

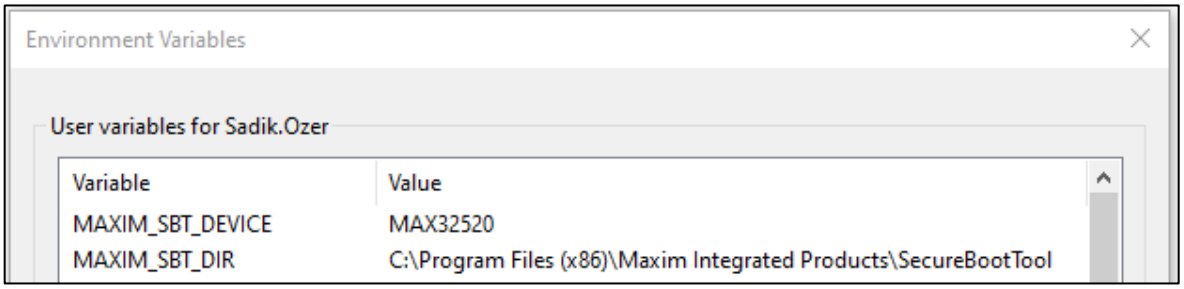

Figure 10. Required SBT system variables.

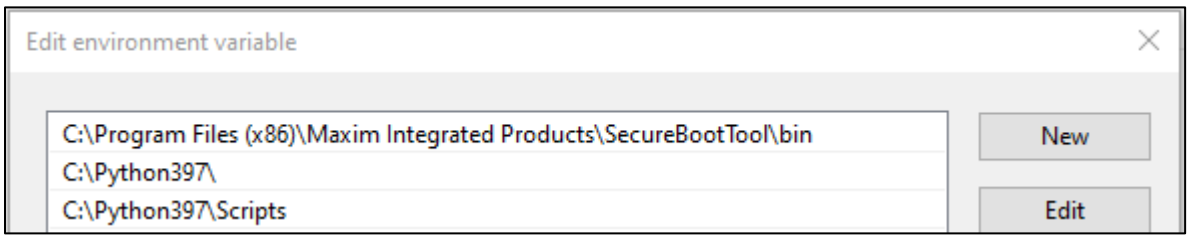

Figure 11. SBT binaries path.

Appendix C: SCP Script Commands

The SCP builder generates scp frames from script command. The script puts in a text file with related parameters. Then this text file should be provided on SCP Build. An example of such a process is shown in **Figure 12**.

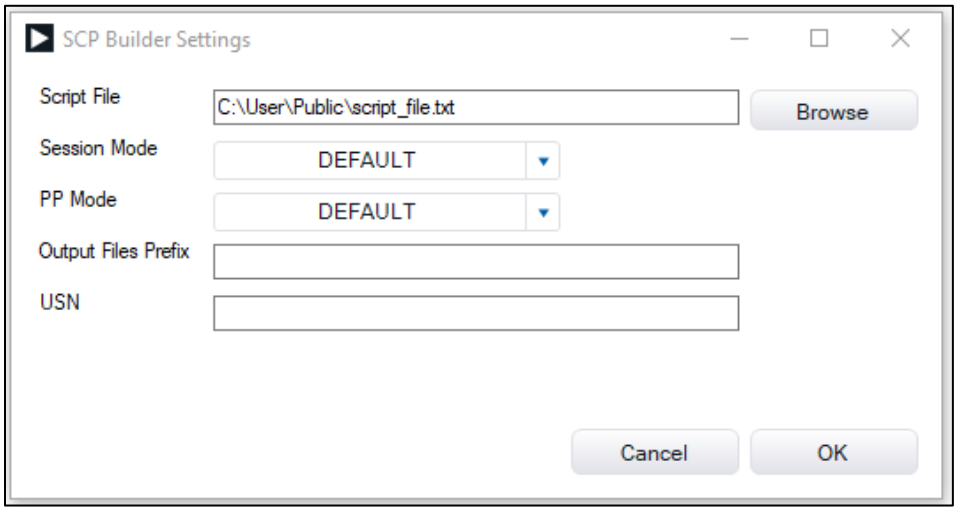

Figure 12. Using script file.

The following tables show the scp script commands and their details.

Table 1. Echo

Command	echo.
Detail	Check the communication with the SBL by sending an 'ECHO' SCP command.
Parameters	

Table 2. Write File

Command	write-file filename [address].
Detail	Send the binary data contains in the provided file to the SBL for writing using the 'WRITE DATA' SCP Command. It also erase the target memory are using the 'ERASE DATA' SCP Command.
Parameters	"filename": S19 or sbin file containing the data to be send for writing to the SBL (Secure Bootloader). "address": Start address to where writing data from sbin file or offset address to add to S19 addresses. Optional with S19 files but mandatory with sbin files.

Table 3. Write Only

Command	write-only filename [address].
Detail	This command send the binary data contains in the provided file to the SBL for writing using the 'WRITE DATA' SCP Command.
Parameters	*"filename":* S19 or sbin file containing the data to be send for writing to the SBL. *"address":* Start address to where writing data from sbin file or offset address to add to S19 addresses. Optional with S19 files but mandatory with sbin files.

Table 4. Verify File

Command	verify-file filename [address [dump]].
Detail	This command send the binary data contains in the provided file to the SBL for verification against the content of the memory writing using the 'COMPARE DATA' SCP Command.
Parameters	"filename": S19 or sbin file containing the data to be send for writing to the SBL. "address": Start address to where start data for verification from sbin file or offset adddress to add to S19 addresses. Optional with S19 files but mandatory with sbin files. "dump": yes/no Add a dummy dump packet for the SCP sender.

Table 5. Write CRK

Command	write-crk filename.
Detail	This command send 'WRITE-CRK' SCP Command. It send the CRK with its signature by the MRK.
Parameters	"filename": File containing the CRK sign by the MRK.

Table 6. Rewrite CRK

Command	rewrite-crk old_crk_filename new_crk_filename.
Detail	This command send 'REWRITE-CRK' SCP Command. It send the *old* CRK and the *new* CRK with its signature by the MRK.
Parameters	"old_crk_filename": File containing the old CRK sign by the MRK. "new_crk_filename": File containing the new CRK sign by the MRK.

Table 7. Write OTP

Command	write-otp offset data.
Detail	This command write data inside the CHIP OTP using the 'WRITE-OTP'SCP Command.
Parameters	"offset": Address offset inside the OTP memory. "data": Data to write at the offset specified.

Table 8. Write Timeout

Command	write-timeout target value.
Detail	This command write the timeout configuration for the different SCP bus using the 'WRITE-TIMEOUT' SCP Command.
Parameters	"target" Bus for which the timeout will be written. Possible value are : 0 - for UART V - for VBUS U - for USB E - for Ethernet S - for SPI "Value" HEX and Little-endian encoded timeout in ms. Examples: To set UART timeout to 250 ms: write-timeout 0 FA00 To set USB timeout to 500 ms: write-timeout U F401 To set SPI timeout to 1000 ms: write-timeout S E803

Table 9. Write Parameter

Command	write-param target value.
Detail	This command write the parameter configuration for the different SCP bus using the 'WRITE-PARAM' SCP Command.
Parameters	"target" Bus for which the parameter will be written. Possible value are: 0 - for UART V - for VBUS U - for USB E - for Ethernet S - for SPI "Value" Value of the parameter.

Table 10. Write Stimulus

Command	write-stim target value.
Detail	This command write the stimulus configuration for the different SCP bus using the 'WRITE-STIM' SCP Command.
Parameters	"target" Bus for which the stimulus will be written. Possible value are : 0 - for UART V - for VBUS U - for USB E - for Ethernet S - for SPI "value" Value of the stimulus.

Table 11. Write Deactivation

Command	write-deact target.
Detail	This command deactivate the different SCP bus using the 'WRITE-DEACT' SCP Command.
Parameters	

Table 12. Kill Chip

Command	kill-chip.
Detail	This command send the 'KILL-CHIP' SCP command to SBL.
Parameters	-

Table 13. Kill Chip USN

Command	kill-chip2.
Detail	This command send the 'KILL-CHIP2' SCP command to SBL with the Chip Unique Serial Number (USN) provided with the corresponding option.
Parameters	

Table 14. Execute Code / Register Applet

Command	execute-code address.
Detail	This command send the 'EXECUTE-CODE' SCP command to SBL. This registers an applet if the address point to an applet header or launches an application if the address point to an application header.
Parameters	"address" Hex encoded address of the previously loaded SCP applet or Application. Example: execute-code 2000f8b0

Table 15. Write Minimum Application Version

Command	write-app-ver version.
Detail	This command send the 'WRITE_APP_VER' SCP command to SBL. This setup the minimum required version for the customer application.
Parameters	"version" Minimum version required for the application 0xMMmmrrrr (i.e., 0x01021240 for version 1.2.4672).

Appendix D: Dump OTP Content

The applet is a small program that is loaded into the internal RAM to extend the secure ROM capability. By the applet, a user can implement several kinds of needs (for example, tests, key generation, drivers). The SBT includes dump otp applet which is used to dump OTP content by applet mechanism. The SBT OTP dump package dumps device OTP content over UART interface.

Note: Other interface does not support it.

The prebuilt dump OTP scp package is here:

"<YourInstallationPath> \ devices \ YourDevice \ scp_packets \ dump_otp_MAXXX",

Depend on the device, send the related scp package to the device (see the section *SCP Sender: Dump OTP package only works over UART interface*) to dump OTP content.

The prebuilt dump OTP package is generated with Maxim Test Key. To generate it with the CRK, follow the steps mentioned below over **SCP Builder** tab as shown in **Figure 13**:

1. In the **Device** list, select the Device part number.
2. In the **Key File** box, type the key file or click the **Browse** button to select the key file.
3. In the **Output Folder** box, type the output location or click the **Browse** button to select the output location(path for the SCP output).
4. Click the **View and Set** button.

   The SCP Builder Settings screen appears.
5. In the **Script File** box, type the script file or click the **Browse** button and then select "dump_otp.txt" script file in the SBT folder.
6. Click **Ok** button.
7. Then click **Start** button.

   The SCP package generates with the key.
8. The output appears on "Output Terminal" screen, as shown in **Figure 14**.

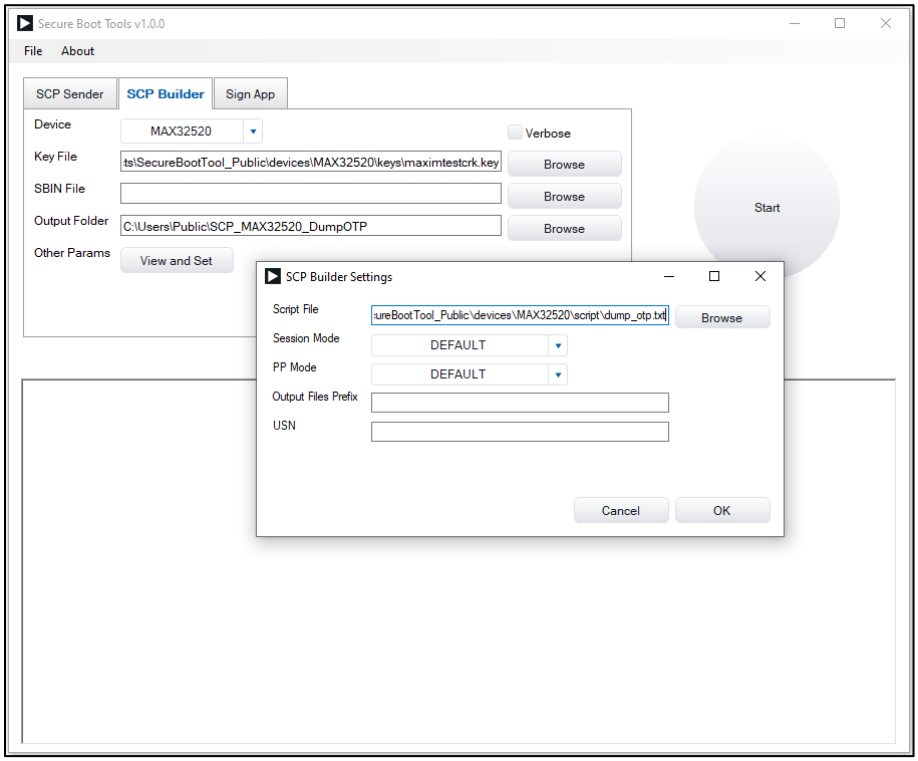

Figure 13. Dump OTP SCP package generation with CRK.

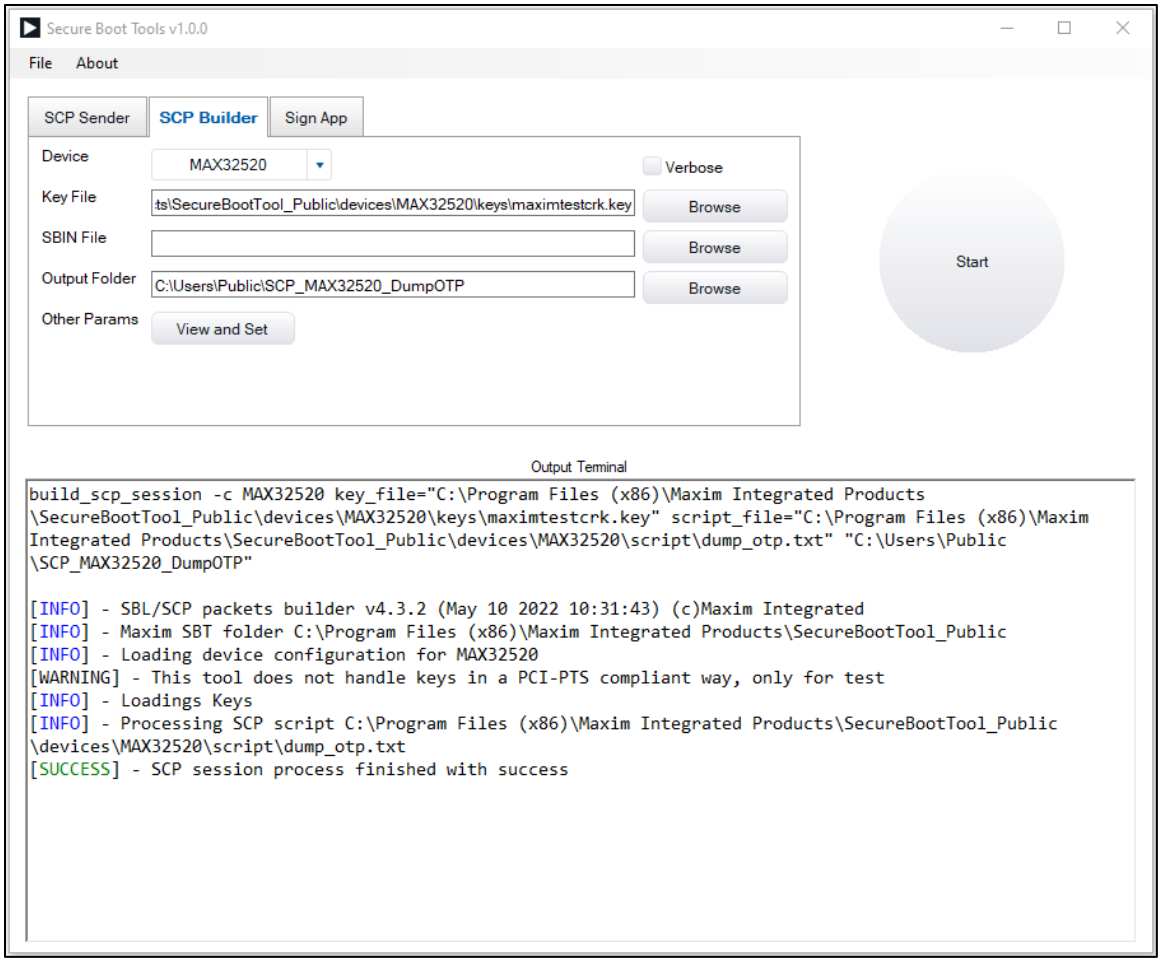

Figure 14. Dump OTP SCP package generation.

Then the generated scp package is sent to the device.

Appendix E: Known Issues

SCP Sender: Disconnection error over USB interface

As shown in **Figure 15** and **Figure 16**, ignore the error that occurs in the disconnection message while communicating over USB with SBL. This case occurs due to Bootloader closing the USB interface before being sure all messages have been flushed.

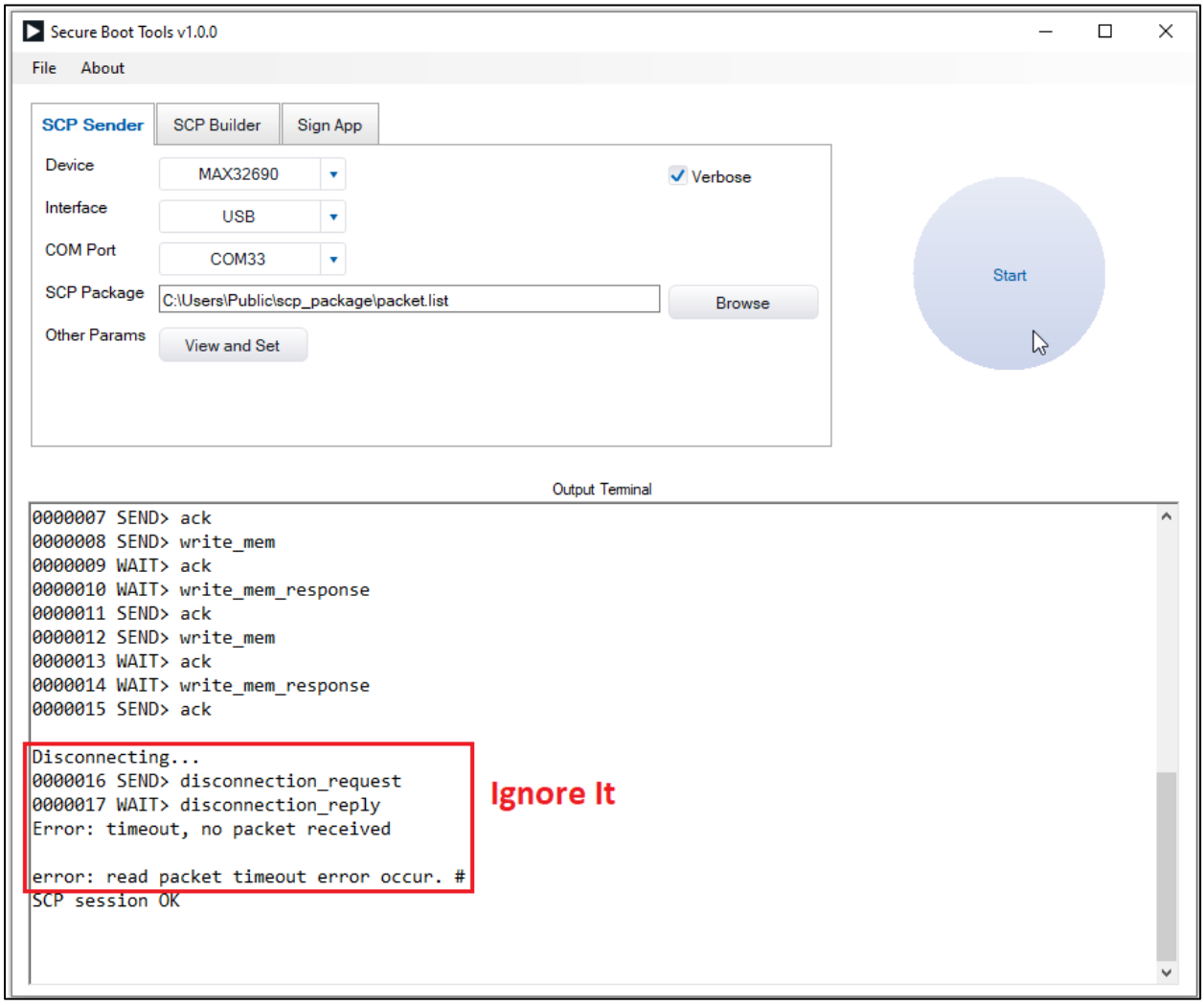

Figure 15. USB Disconnection error for USB interface, with Verbose mode.

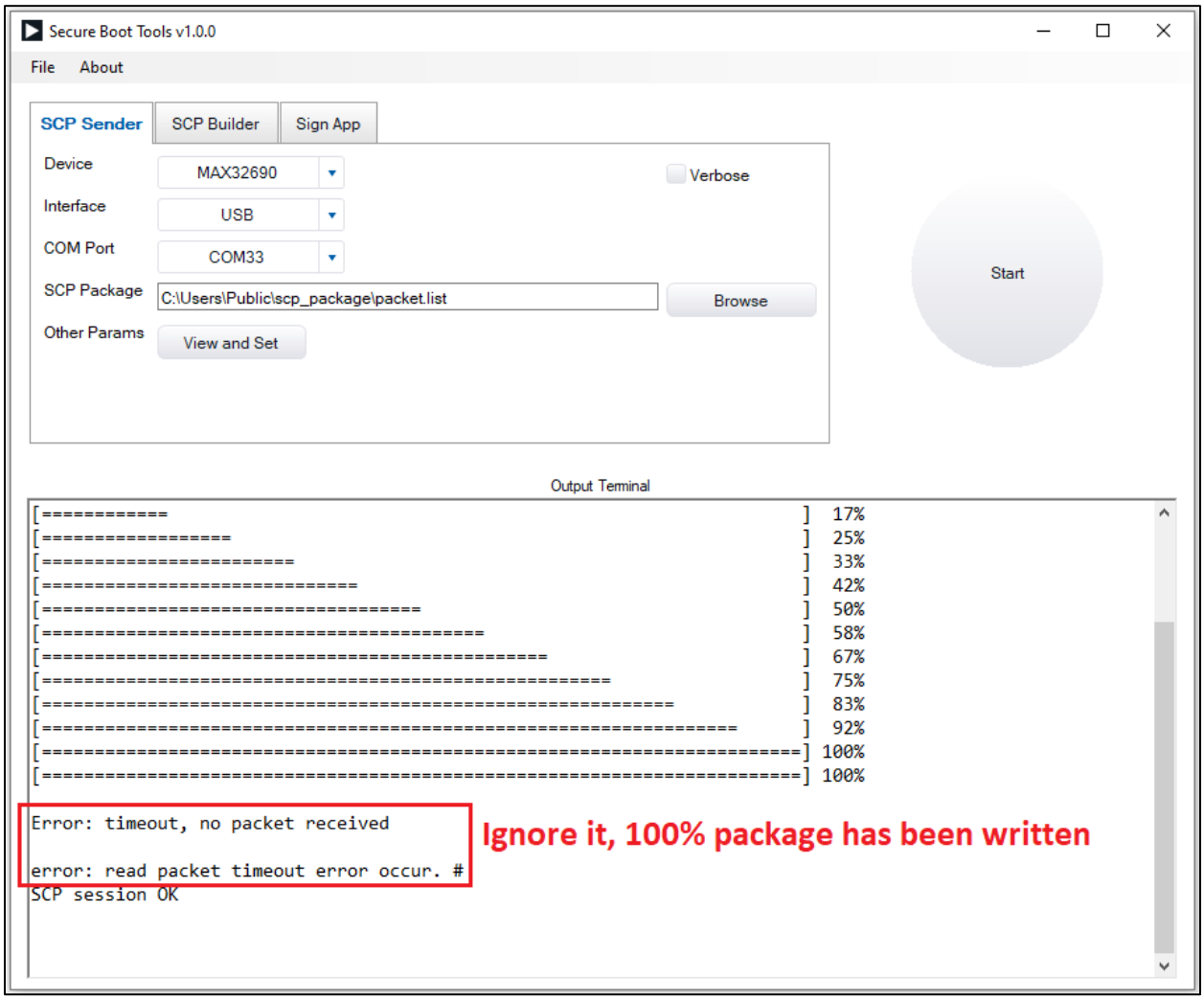

Figure 16. USB Disconnection error for USB interface.

SCP Sender: Dump OTP package only works over UART interface

As shown in **Figure 17**, an error occurs while trying to dump OTP content over another interface.

Note: This feature only works over the UART interface.

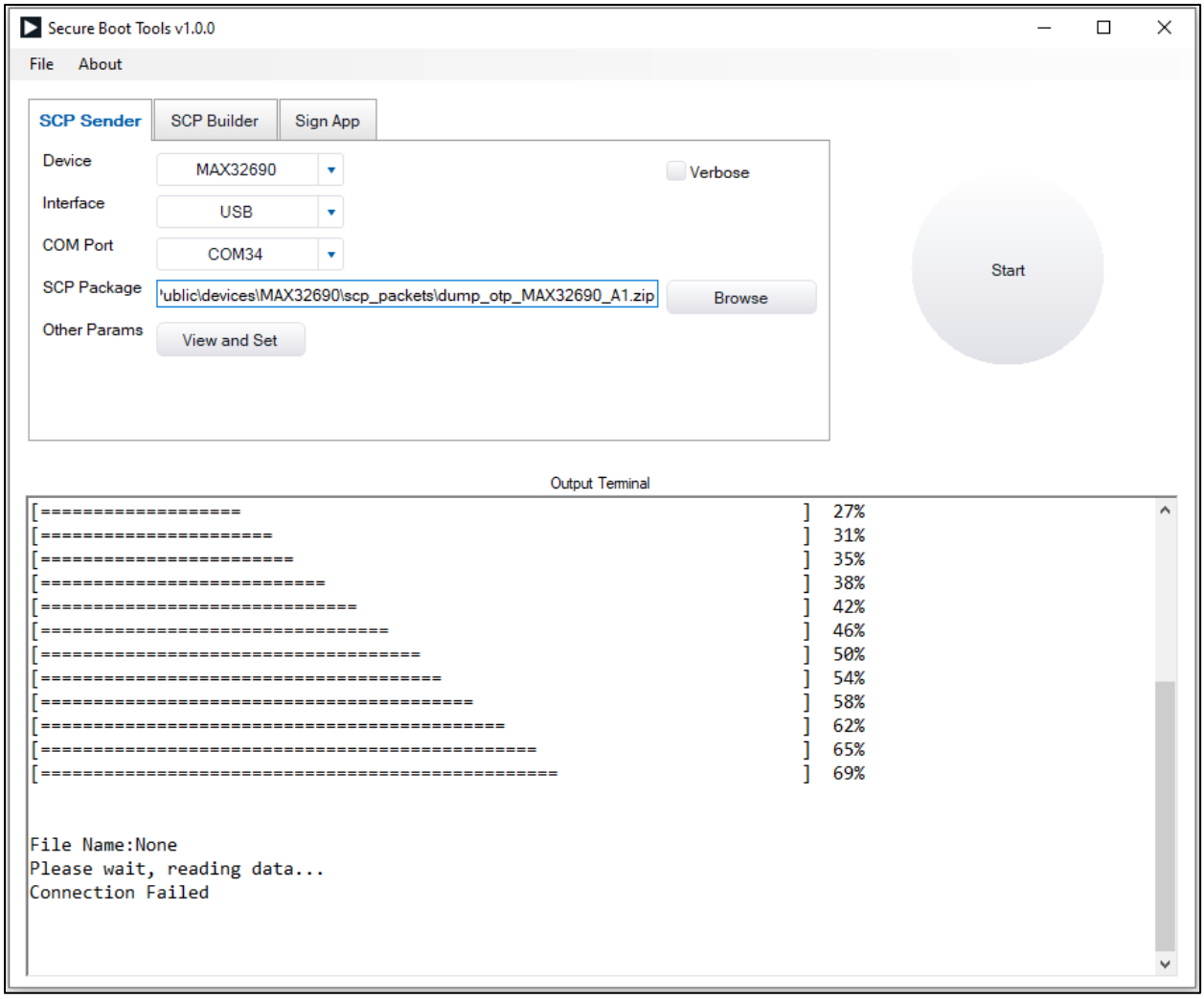

Figure 17. Dump OTP over USB interface.

Revision History

REV NUMBER	REV DATE	DESCRIPTION	PAGES CHANGED
0	06/22	Initial release	—