Application Note

STZAN0110EN0100

Revision 1.00

March 2019

Using the Secret Wrapping Tool (SWT) with the Microchip SAM L11

Introduction

This application note describes the process of creating an encrypted data file that can be consumed by the Secure Thingz Secure Deploy™ platform. The process makes use of the Secure Thingz Secret Wrapping Tool (SWT) which is available from Secure Thingz or any secure programming service provider that supports the Secure Thingz Secure Deploy™ architecture. The document assumes that the reader is familiar with standard security terminology and cryptographic functions.

Target Devices

This application note refers to the following Secure Microcontrollers :

- Microchip SAM L11

Software Requirement

- Secret Wrapping Tool (SWT) PC application
    - SWT is provided on request from both Secure Thingz and service providers of the Secure Deploy™ secure programming architecture.

- Operating Systems supported
    - Windows 10 or later only

1. Overview

The Secure Deploy™ (SD) manufacturing system, developed by Secure Thingz, provides a way for OEM's to generate and manage their secure content (key pairs, signature keys and certificates) and securely deploy this content to programming facilities to be programmed into microcontroller devices on their behalf. The OEM's secure content is protected at the programming facility itself while also controlling the production quantity.

In order to comply with Secure Thingz "zero-trust" philosophy, the OEM is provided with a high security PC based application (SWT) that allows them to encrypt their secure content. This encrypted file can then be transported over an unsecure medium directly to the secure programming facility. Part of the encryption process is to use the cryptographic certificate of the Hardware Security Module (HSM) integrated into the secure programming facility. This ensures that ONLY the targeted HSM can decrypt the OEMs secure content prior to programming the target microcontrollers.

Figure 1 shows a diagrammatic representation of the Secure Deploy™ architecture. Both the SWT and the Embedded Trust security development tool (also available from Secure Thingz) create encrypted data files that can be consumed by Secure Deploy™ and used to securely program microcontroller devices.

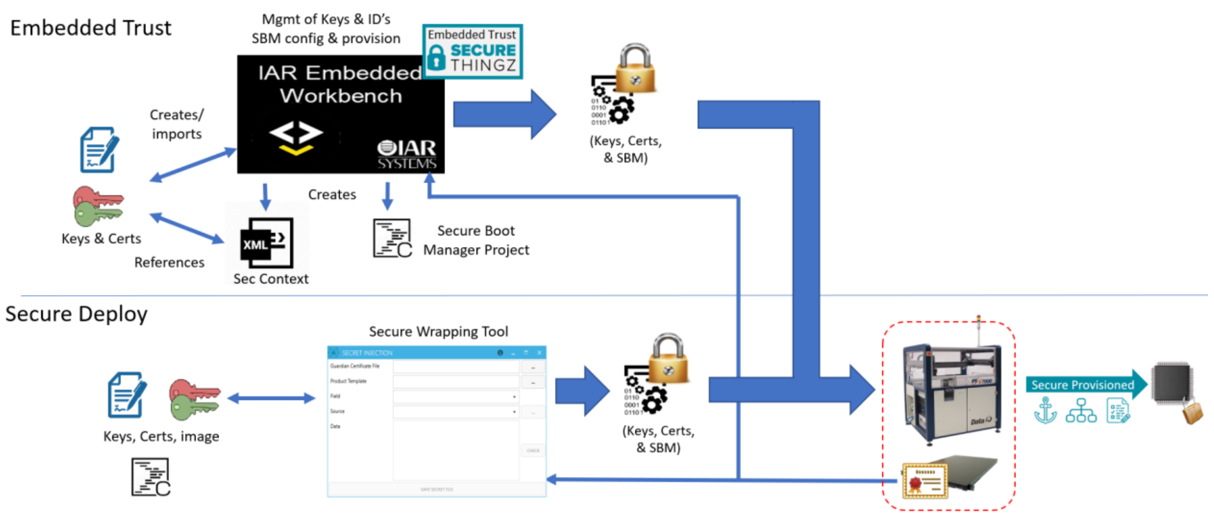


Figure 1: Diagrammatic representation of the Secure Deploy™ architecture

2. Secure Deploy™ Production Package

If an OEM wishes to take advantage of the Secure Deploy™ secure programming system, they should contact an authorised service provider. Details of Secure Deploy™ authorised service providers can be found on the Secure Thingz website (search for programming services). Once the OEM has confirmed the part number of the target device, the service provider will deliver a **Secure Deploy™ Production Package** to the OEM customer, which will contain :

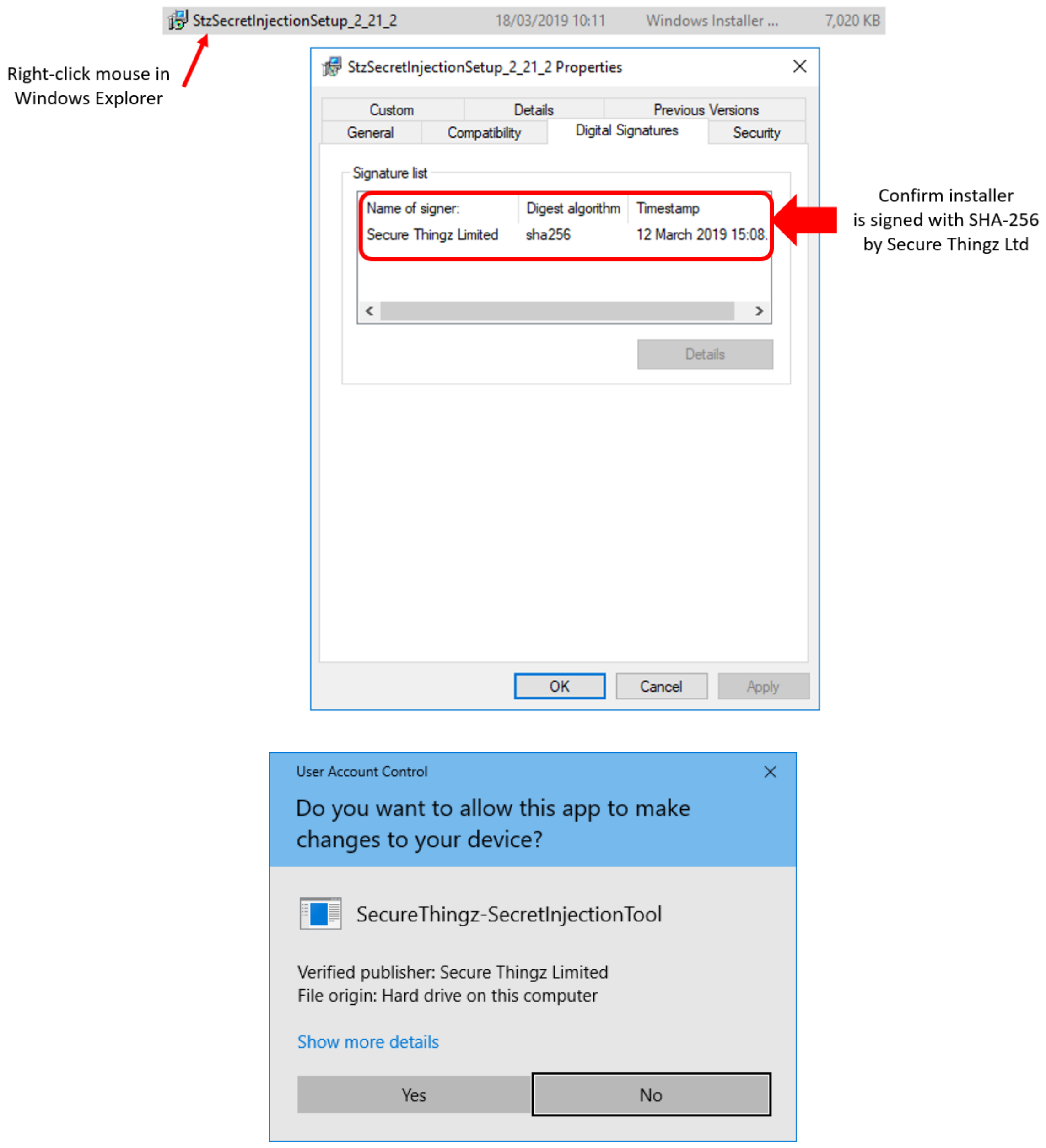
a) A digitally signed PC application (.msi file extension)
b) A product template (.stt file extension)
c) A Guardian (Hardware Security Module) certificate (.stz file extension)
d) This application note (.pdf file extension)

The sections below describe the components of the **Secure Deploy™ Production Package** in more detail.

2.1 PC Application

The Secret Wrapping Tool is a PC application that is provided to the Original Equipment Manufacturer (OEM) customer by the service provider of the Secure Deploy™ secure programming system as part of the Secure Deploy™ Production Package.

On receipt of the PC application installer (*StzSecretInjectionSetup_<version>.msi*), the OEM customer should install it on a personal computer that is held within a secure facility and only accessible by the appropriate security personnel. Prior to installation, it is recommended that the user confirm that the application installer has been correctly signed. Figure 2 shows how the user should confirm that the Secret Wrapping Tool application received has been legitimately signed by Secure Thingz Ltd (Please note : version numbers higher than that shown are also valid), first by checking the signature of the installer and secondly by checking the installer confirmation dialog.



StzSecretInjectionSetup_2_21_2 18/03/2019 10:11 Windows Installer ... 7,020 KB
Right-click mouse in Windows Explorer
StzSecretInjectionSetup_2_21_2 Properties
Custom
Details
Previous Versions
General
Compatibility
Digital Signatures
Security
Signature list
Details
OK
Cancel
Apply
Confirm installer is signed with SHA-256 by Secure Thingz Ltd
User Account Control
Show more details
Yes
No

Name of signer:	Digest algorithm	Timestamp
Secure Thingz Limited	sha256	12 March 2019 15:08:

Do you want to allow this app to make changes to your device?

SecureThingz-SecretInjectionTool

Verified publisher: Secure Thingz Limited
File origin: Hard drive on this computer

Figure 2: How to check for legitimate SWT

2.2 Product Template

The Product Template file is a text file with extension ".stt" (for **S**ecure **T**hingz **T**emplate) that contains a PEM-like[1] header and footer line enclosing information in JSON format. The Product Template provides the scripts and instructions for an OEM (that is using the Securer Thingz Secure Deploy architecture) to provision its IoT devices using the Guardian HSM. The OEM must, in addition, provide the Field Data, that is, the certificates and keys, separately, which make up the Product Definition. The input of addition data is carried out during the configuration of the SWT prior to wrapping (see Figure 4). A Product Definition contains a reference to a Product Template, (the reference being the 'Template Meta Data Hash').
Figure 3 shows the structure of a Product Template. There are two "blocks", one for the Script and one for the Template Meta Data. The Script block is encrypted for transfer whereas the Meta Dat Block is not.

Each "block" has:

- Header line "-----BEGIN STZ METADATA-----" or "-----BEGIN STZ SCRIPT-----"
- Plain text information that is not integrity protected (optional)
- Hash Identity, which is integrity protected by the signature
- Text information section, in JSON format, base64 encoded, also integrity protected
- Data content portion, base64 encoded, integrity protected
- Signature information section, JSON/base64 encoded
- Footer line "-----END STZ METADATA-----" or "-----END STZ SCRIPT-----"

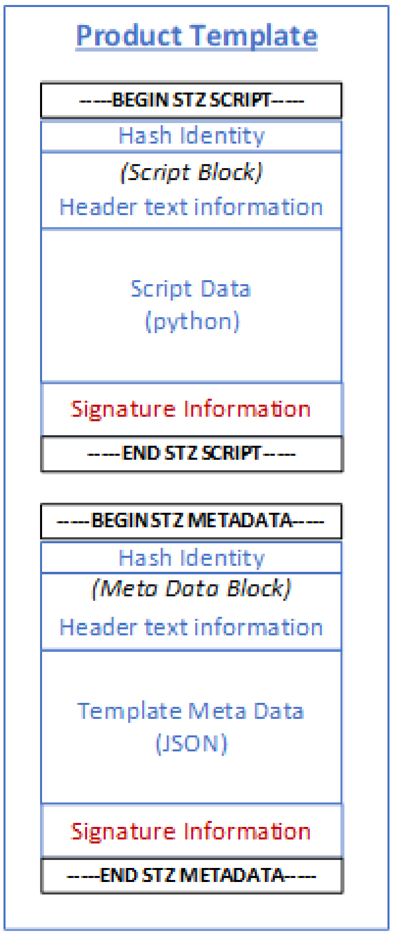


Figure 3: Product Template overview

[1] Note : The use of a PEM-like header/footer separates Script from Template Meta Data and means the JSON parser can be applied to each block one at a time: the JSON parser hence will require less memory.

2.3 Guardian Certificate

The Guardian certificate is a Secure Thingz proprietary certificate format that is exported from the secure programming system at the factory facility. This certificate is important as it includes the Public key of the Guardian. The key is used to store the private data in such a way that it is itself encrypted using an algorithm based on a one-time ephemeral key. Further, the wrapped secrets can only be recovered by the Guardian for which the secret is intended. This method of wrapping secrets and the algorithm used is a well-established practice in cryptography. Only the Hardware Security Module that is integrated into the Guardian, that has provided the certificate, can decrypt the OEM customers secret information. This ensures compliance with Secure Thingz zero-trust philosophy.

3 Secrets

The Secure Deploy™ Production Package provides the components required by the SWT to wrap the customers secrets. The secrets themselves are provided by the customer. The type and number of secrets that can be provided is device and use case dependant. Descriptions of the type and number of secrets is defined within the Product Template (see section 2.2) which is only made available to the user via the SWT Graphical User Interface (GUI).

4 Secret Wrapping Tool GUI

Once correctly installed and verified, the Secret Wrapping Tool application can be run by executing the SecretInjectionApp.exe file. Figure 3 shows the contents of the installation folder for the SWT.

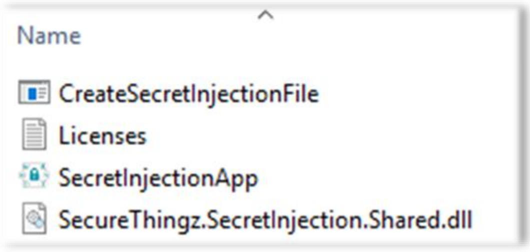

Name
CreateSecretInjectionFile
Licenses
SecretInjectionApp
SecureThingz.SecretInjection.Shared.dll

Figure 3: SWT install folder contents

Once the application is running the dialog box shown in Figure 4 will appear.

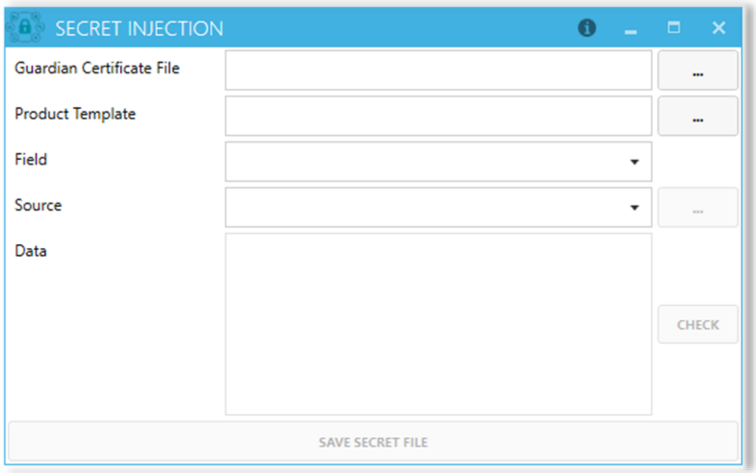
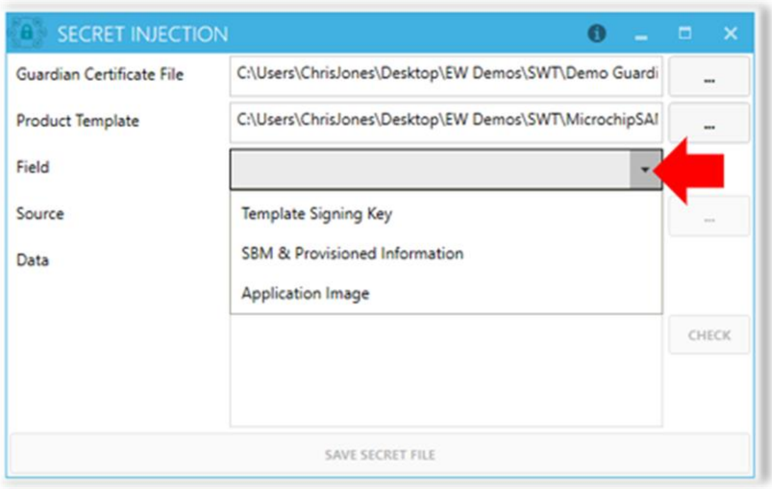


Figure 4: SWT opening dialog box

To create a secret file, the user will need to first select a valid Guardian Certificate file for the target guardian using the ellipses (…) button adjacent to the Guardian Certificate File field, this will display a standard file open dialog from which the Guardian Certificate file (.stz) is selected. The user must select the Guardian Certificate file provided as part of the Secure Deploy™ Production Package (see section 2.3).

Next, the user is required to select a product template metadata file using the ellipses (…) button next to the Product Template field. This file can either be a template metadata file (.stm file) or an algorithm package file containing a single template metadata file (.algpkg file). The user must select the Product Template file provided as part of the Secure Deploy™ Production Package (see section 2.2).

After selecting the product template, the field list will be populated with all fields for the product that support configuration using a secret file. The list is product dependant. Figure 5 is an example of the field list for a Microchip SAM L11 device:



Guardian Certificate File: C:\Users\ChrisJones\Desktop\EW Demos\SWT\Demo Guardi

Product Template: C:\Users\ChrisJones\Desktop\EW Demos\SWT\MicrochipSAI

Field

Source

Data

Template Signing Key

SBM & Provisioned Information

Application Image

Figure 5: Example fields for SAM L11

Once the field is selected, the Source drop down will be populated with one of three options below:

1. Generate
2. File
3. User Entry

Figure 6 is an example of the source list for a Microchip SAM L11 device:

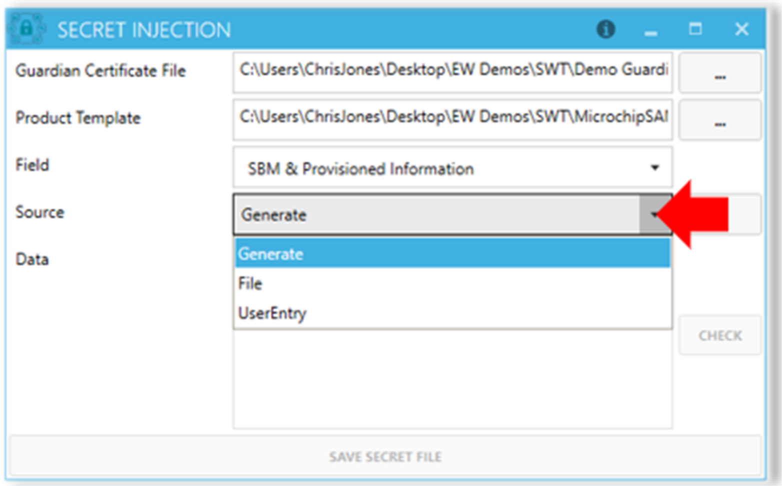

SECRET INJECTION
Guardian Certificate File
C:\Users\ChrisJones\Desktop\EW Demos\SWT\Demo Guardi
Product Template
C:\Users\ChrisJones\Desktop\EW Demos\SWT\MicrochipSAI
Field
SBM & Provisioned Information
Source
Generate
Data
Generate
File
UserEntry
CHECK
SAVE SECRET FILE

Figure 6: Example source for SAM L11

If “Generate” is selected, then clicking the ellipses (…) button next to the Source drop down will use internal cryptographic functions depending on the secret type to automatically generate a valid set of data for this field. The text “Secret Generated” will be displayed in the Data text box (see Figure 7), and assuming a valid Guardian Certificate file has been selected, then the Save Secret File button will be enabled.

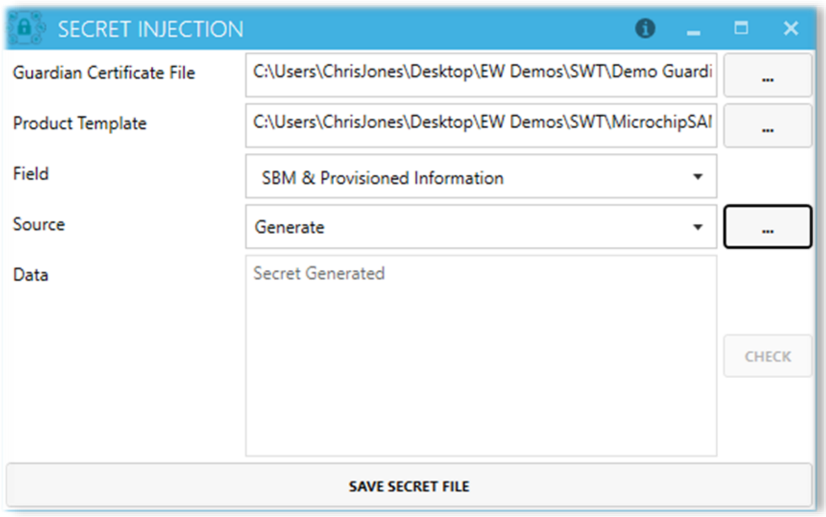

SECRET INJECTION
Guardian Certificate File
C:\Users\ChrisJones\Desktop\EW Demos\SWT\Demo Guardi
Product Template
C:\Users\ChrisJones\Desktop\EW Demos\SWT\MicrochipSAI
Field
SBM & Provisioned Information
Source
Generate
Data
Secret Generated
CHECK
SAVE SECRET FILE

Figure 7: Example of Generate source

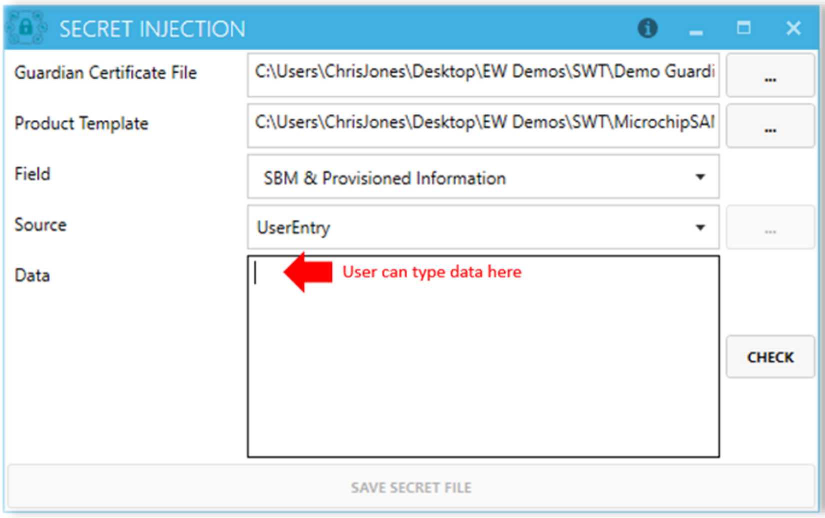

SECRET INJECTION
Guardian Certificate File
C:\Users\ChrisJones\Desktop\EW Demos\SWT\Demo Guardi
...
Product Template
C:\Users\ChrisJones\Desktop\EW Demos\SWT\MicrochipSAI
Field
SBM & Provisioned Information
Source
UserEntry
Data
User can type data here
CHECK
SAVE SECRET FILE

Figure 8: Example of UserEntry input

If "File" is selected, then clicking the ellipses (…) button next to the Source drop down will display a standard file open dialog from which the source file for the secret can be selected. Multiple formats are supported for this data, and these are shown in Figure 9. Assuming a file is selected that is in a valid format, and this file contains valid data for this field, then the text "Secret data loaded from file" will be displayed in the Data text box, and assuming a valid Guardian Certificate file has been selected, then the Save Secret File button will be enabled.
If an invalid file has been selected, or this file contains invalid data for this field type, then an error message will be displayed.

Format	Description
Raw Binary	File containing the correct number of bytes for the parameter, in the case of a key pair, it will consist of the private part first.
Raw Binary with Header Byte	File containing the correct number of bytes for the parameter as above, plus a leading 04 byte to be used as a header.
Asn1 encoded der	A der formatted file containing Asn1 encoded data containing the secret to be loaded into the parameter.
PEM encoded der	A pem formatted file containing an Asn1 encoded der file as above.
PKCS#12	An archive file format for storing many cryptography objects as a single file. It is commonly used to bundle a private key with its X.509 certificate or to bundle all the members of a chain of trust.

Figure 9: Support file formats

If "UserEntry" is selected, then the Data text box will become editable and will be pre-populated with some default data valid for the chosen field, the format of this data will depend on the field type chosen but commonly will be in a two character per byte hex format (see Figure 8).
Once the data is entered, clicking the Check button will validate the data is correct, and if it is and a valid Guardian Certificate file has been selected, then the Save Secret File button will be enabled.

Finally, clicking the Save Secret File button will display a standard file save dialog prompting for a location and filename for the wrapped secret file. Once saved, this file will then be ready to be delivered to the service provider for delivery to the Secure Deploy™ secure programming system.

4.1 SWT Command Line Tool

A command line tool is installed to the same directory as the Secret Injection Graphical User Interface and is called CreateSecretInjectionFile.exe (see Figure 3). This command line tool requires a set of arguments to correctly generate a secret file for injection into the Secure Deploy™ secure programming system. For further details of this tool, please contact Secure Thingz via their website.