Tagthenticate Applet User Manual

Version 1.0

Table of Content

Legal Notice

Document Format and Convention

This document will use the following typographical formatting and conventions for denoting different types of information.

Component	**Meaning**
Code	This is a command line information that can be executed or part of a script or configuration document.
Italicized Bold	This is a segment of information that requires careful reading or attention.
[Square Bracket Code]	This is a command line, scripting or configuration option that one should carefully fill with the relevant values.

Supported Tagthenticate Component Versions

The supplied Tagthenticate components and versions utilized under the Microsoft Windows and Linux Operating System supported by this user documentation includes the following:

Component	**Version**	**Software**	**Hardware**
Tagthenticate Applet	1.0.0	✓	✓

The supported software and hardware versions listed above are the minimum version requirement which this document pertains to. Any version number with its major version number above the specified major version number requires consultation with the vendor whom one had purchased the Tagthenticate solution from or qualified Tagthenticate solution support personnel.

License of Use

The supported software and hardware versions listed above are the minimum version requirement which this document pertains to. Any version number with its major version number above the specified major version number requires consultation with the vendor whom one had purchased the Tagthenticate solution from or qualified Tagthenticate solution support personnel.

Introduction

The Tagthenticate Applet is a custom applet specifically designed for the Tagthenticate System's Blockchain Based Digital Attestation use case. The applet maybe used to store and attest to critical pieces of digital information and assets not more than 255 bytes in size.

One of the most important use case for the Tagthenticate Applet is to be used for digital attestation of physical or digital objects. The Tagthenticate Applet supports ECC-P256K1 private key algorithm that is used in Bitcoin, Ethereum and many other types of Blockchains. The applet also reserves 255 bytes of memory to store important descriptors and information of the object to be attested (i.e. smart contract address).

The private key maybe used to host an initial value for the Blockchain and to publish its smart contract in the beginning phases of the deployment of the applet with a subsequent attestation of the applet via its replay attack resistant Digital Signature Challenge-Response based digital attestation protocol.

The Tagthenticate Applet could also be paired to a centralized system for digital attestations of objects by becoming a node in an Enterprise PKI scheme like in an Enterprise IoT backed product line.

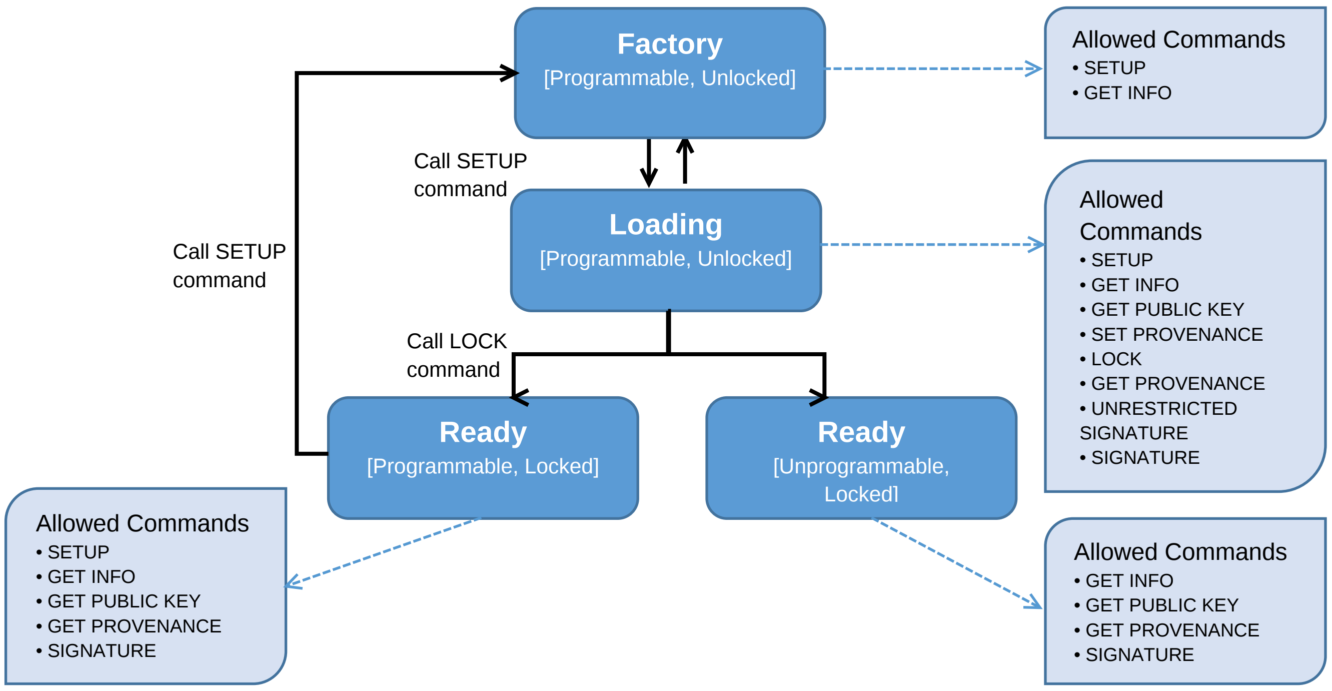
Tagthenticate Applet Lifecycle

The Tagthenticate Applet uses the following lifecycle as described below. The Tagthenticate Applet goes through three stages – Factory, Loading and Ready. The applet also has two important configuration flags, the programmable flag which tells the applet is it allows factory resetting of the applet and the locked flag which indicates where the applet is locked and ready for use.

1. The Tagthenticate Applet is delivered in Factory state which is programmable and unlocked. The applet would have no keypair and content loaded in the applet and calling the SETUP command would automatically generate an ECC-P256K1 keypair in the applet using the onboard random keypair generator. Once a random keypair is successfully generated in the applet, the applet will be in Loading state which is still programmable and unlocked.
2. Digital Content (i.e. digital assets and certificates) can now be loaded into the Tagthenticate applet.
3. The Challenge-Response based Digital Siganture Challenge Response APDU command (SIGNATURE command) is accessible as long as there is digital content loaded and not in a Factory state. The applet need not be in a locked or unprogrammable configuration state to be able to access the SIGNATURE command.
4. While in Loading state, blind signatures using the applet's ECC private key maybe issued via the UNRESTRICTED SIGNATURE command.
5. The Tagthenticate Applet maybe locked after digital content have been loaded into the applet and no more changes are required. The LOCK command must be issued to the Tagthenticate applet while the applet is in a Loading state and at the same time the applet's programmable configuration flag maybe set to the unprogrammable configuration flag to prevent factory resetting of the applet. Once the state is locked, no further changes including attempting to

make the applet unprogrammable would be allowed. This will advance the applet from the Loading to Ready state.

6. Factory resetting of the applet maybe done by calling the SETUP command if the programmable flag is not set as unprogrammable during the Factory, Loading or Ready state. The SETUP command would factory reset the applet causing the erasure of all applet content and private key. The lifecycle would go back to Step 1.



Tagthenticate Applet AID

The AID for Tagthenticate Applet in hexadecimal is “54545441473031303000”.

APDU Commands for the Tagthenticate Applet

APDU Command Name	CLA	INS	P1	P2	Content	Description
GET INFO	0x88	0x00	0x00	0x00	N/A	Returns the current applet state information. Please read the chapter Applet State Binary Indicators.
SETUP	0x88	0x10	0x00	0x00	N/A	Destroys all keys and content of the previous setup and resets the applet’s locked state to unlock state if the applet is in a programmable state.

GET PUBLIC KEY	0x88	0x12	0x00	0x00	N/A	Returns a X || Y public key bytes of the current ECC public key if a public key is available else return no data.
SET PROVENANCE	0x88	0x14	0x00	0x00	Binary content up to 255 bytes.	Uploads binary content not more than 255 bytes to the applet's content storage memory if the applet is in an unlocked and programmable state. Every time SET PROVENANCE is called, the previous content will be wiped and the newly uploaded content will replace the old content.
LOCK	0x88	0x16	0x00 / 0x01	0x00	N/A	Locks the applet if the applet is in a programmable and unlocked state. Setting the P1 to 0x01 will permanently make the applet unprogrammable forever.
GET PROVENANCE	0x88	0x18	0x00	0x00	N/A	Retrieves up to 255 bytes of applet content data.
UNRESTRICTED SIGNATURE	0x88	0x1A	0x00	0x00	Binary data or hash up to 255 bytes.	Digitally signs any data or hash up to 255 bytes blindly when the applet is in an unlocked state. Returns a R || S formatted digital signature of 64 bytes.
SIGNATURE	0x88	0x1C	0x00	0x00	Binary challenge data up to 255 bytes.	Performs a digital signature based challenge response by accepting a challenge to be signed as part of the SIGNATURE CHALLENGE RESPONSE PACKET response digital signature of 64 bytes. Please read chapter on Signature Challenge Response Format.

Applet Binary State Indicators

The Applet Binary State Indicator is a binary data of 11 bytes. These 11 bytes can be parsed using the following table.

State Indicator	Offset	Length	Description
Version	0	2	0x0100
Programmable State	2	1	0x01 – Programmable 0x00 – Unprogrammable
Locked State	3	1	0x01 – Locked 0x00 – Unlocked
Private Key Length	4	2	0x0020 – 32 bytes of private key data 0x0000 – Private key is not initialized or available
Public Key Length	6	2	0x0040 – 64 bytes of public key data 0x0000 – Private key is not initialized or available
Content Data Length	8	2	0xNNNN – NNNN bytes of content data 0x0000 – Content data is unavailable
ATR Set State	10	1	0x01 – ATR Set OK 0x00 – ATR Not Set Note: This is not a critical function and most likely not going to be set.

Signature Challenge Response Format

The Tagthenticate Applet creates a binary To Be Signed Challenge Packet before using SHA-256 to hash the To Be Signed Challenge Packet to derive a To Be Signed Hash which is signed by the applet’s private key if the private key is initialized and available for use.

The following table shows the format of the To Be Signed Challenge Packet in sequence.

Packet Component	Offset	Length	Content	Description
Magic	0	4	Magic header	Magic header - 0x41544643. In ASCII, it is “ATFC”.
Version	4	2	Version	Version header - 0x0100.
Challenge	6	ChalLen	Challenge issued by enquiring party in the APDU.	Challenge issued by enquiring party in the APDU.
Content Data	6 + ChalLen	ContentLen	Content data stored in the applet.	Content data stored in the applet.

The digital signature returned is in a 64 bytes R || S format derived from the SHA-256 hashing of the To Be Signed Challenge Packet components in sequence above.

Status Word Codes

Below are the following Status Word Codes one might encounter while working with the Tagthenticate Applet.

Status Word	**Calling APDU Commands**	**Descriptions**
0x9000	• GET INFO • SETUP • GET PUBLIC KEY • SET PROVENANCE • LOCK • GET PROVENANCE • UNRESTRICTED SIGNATURE • SIGNATURE • SELECT APPLET	Processed command successfully. Response data might be bundled with it.
0x6FXX	• GET INFO • SETUP • GET PUBLIC KEY • SET PROVENANCE • LOCK • GET PROVENANCE • UNRESTRICTED SIGNATURE • SIGNATURE • SELECT APPLET	Generic error. SW error like 0x6F00 to 0x6F0F maybe present as debugging SW codes.
0x6999	• SELECT APPLET	Applet selection failed. Initialization of applet might have failed while executing during initial applet selection.
0x6E00	• GET INFO • SETUP • GET PUBLIC KEY • SET PROVENANCE • LOCK • GET PROVENANCE • UNRESTRICTED SIGNATURE • SIGNATURE • SELECT APPLET	Incorrect CLA is used. Please use CLA == 0x88.
0x6985	• SETUP • UNRESTRICTED SIGNATURE • SIGNATURE	When the SETUP command is called and the applet is unprogrammable, this SW will be thrown. When the UNRESTRICTED SIGNATURE command is called and the applet does not have a valid and initialized private key, this SW will be thrown.

		When the SIGNATURE command is called and the applet does not have a valid and initialized private key or the content data is not set, this SW will be thrown.
0x6986	• SET PROVENANCE • UNRESTRICTED SIGNATURE	When the SET PROVENANCE command is called and the applet is locked, this SW will be thrown. When the UNRESTRICTED SIGNATURE command is called and the applet is locked, this SW will be thrown.
0x6D00	ALL OTHER INSTRUCTIONS	When an invalid **INS**truction byte is used, this SW will be thrown.
0x6FAF	• UNRESTRICTED SIGNATURE • SIGNATURE	When UNRESTRICTED SIGNATURE or SIGNATURE command call encounters an error attempting to generate a valid digital signature, this SW will be thrown.