



Research & Advanced Engineering EESE

Feature – Software Signing Specification

Version 1.4

UNCONTROLLED COPY IF PRINTED

Version Date: Nov 29, 2018

Contact: Bradley Smith
Bradley.W.Smith@ford.com

FORD CONFIDENTIAL



Revision History

Date	Version	Author	Notes
Sep. 25, 2015	0.1	Acaushi1	Initial DRAFT
Oct 14, 2015	0.2	Acaushi1	Updated SW signing assumptions
Oct 22, 2015	0.3	Xye7	Updated Section 3
April 11, 2016	0.4	Acaushi1	Clarified requirements per AR release
Dec 19, 2016	1.0	Acaushi1	Clarified requirements and released the first official version.
Feb 7, 2017	1.1	Acaushi1	Updated VS diagram per Vector's Request (REQ-SC-004) Updated SHA256
Sep 21, 2017	1.2	Acaushi1	Updated verbiage to be consistent throughout the document REQ-SC-003 : Updated with definitions of VBF fields: added default value of VS version number; added default value of signature field for submitted files REQ-SC-004 : Modified requirement to reject files with signature field populated in VBF file REQ-TE-004 : Updated routine information REQ-TE-005 : Updated DID information REQ-TE-002 : Updated to include option of using two keys in PBL with a secure switch to development mode.
April 17, 2018	1.3	Acaushi1	Updated Acronyms Section 2.3: Deleted foot notes from assumption notes Section 3.5: Added illustration on section heading and deleted the foot note EQ-KM-005: Moved "built to print" HW from definition to requirement text as rule #4. REQ-SC-001: Moved note to requirement text under item (a) REQ-SC-003: Changed VS version value from 0x0001 to 0x0000; Added clarification for VS and Signature to be stored as big-endian. Changed "data block" to data segment to align with Vector FNOS documentation; Updated Verification structure table. REQ-TE-002: Updated requirement text and Option1 and added persistent DTC when the ECU uses development key for signature verification. REQ-TE-003: Updated requirement for possibility of allowing host micro to verify secondary micro's code when they are isolated (no external connections) REQ-FF-001: Updated to include a valid format of .vbu file and .vbf file. Appendix: Added Token high level design guide; added valid headers and Verification Structure for both unsigned and signed files.
Nov 2, 2018	1.4	Bsmit582	Specified DTC Code for module in development mode Added and Specified DTC Code for module running development software but module may/may not be in development mode. Cleaned up formatting.



Table of Contents

Revision History	2
1 Overview	4
1.1 Background and Scope	4
1.2 Acronyms	4
2 System Architecture	5
2.1 Assumptions	5
2.2 System Architecture	5
2.3 High Level Flow (Production SW signing diagram)	6
2.4 High Level Process Flow	7
2.5 Signing Method Illustration	8
3 Assumptions and Requirements	9
3.1 SW signing feature overview	9
3.2 Requirements	10
3.2.1 Key Management	10
3.2.2 Signature Creation	12
3.2.3 Target ECU	15
3.2.4 Crypto Requirements	18
3.2.5 File Format	18
4 Appendix:	20
4.1 Token Implementation Design Guide	20
4.2 Valid unsigned file (header + VS)	21
4.3 Valid signed file (header + VS + Signature)	22
4.4 Reference Documents	23



1 Overview

1.1 Background and Scope

Ford's vision is to be able to update all ECU's containing software securely by authenticating software at a trust center. This document describes signing process, system use cases, architecture and requirements from SW originator to destination ECU.

A digital signature provides authenticity and integrity of data. A digital signature algorithm contains both a signature generation process and a signature verification process. The sender (**Ford**) uses the signature generation process to generate a signature on data and the receiver uses the signature verification process to verify a signature on data to ensure the data is authentic. In this instance, software has been signed by Ford and ECU's will verify to ensure that the SW has not been modified.

Scope for these specification and applicability is for embedded ECU's updated through CAN using standard mechanisms (e.g. .VBF) or extensions of those for FOTA. Large connected ECU (SYNC, TCU) may use more sophisticated SW signing mechanisms. SW signing is required for all SW updates.

1.2 Acronyms

ECU	Electronic Control Unit
SDLC	Smart Data Link Connector
IVS	In-Vehicle Software
IVSS	In-Vehicle Security Services
LIN	Local Interconnect Network
SW	Software
FCSD	Ford Customer Service Division
FOTA/OTA	Flash Over – the - Air
FNOS	Ford Network Operating System
GIVIS	Global in Vehicle Information SW
RSA	Rivest Shamir Adleman
HSM	Hardware Security Module
SHA	Secure Hash Algorithm
PBL	Primary Boot Loader or Ford Flash Loader
PN	Part Number
SBL	Secondary Boot Loader
IPC	Inter Processor Communication
CAN	Controller Area Network
VBF	Versatile Binary Format
VO	Vehicle Operations

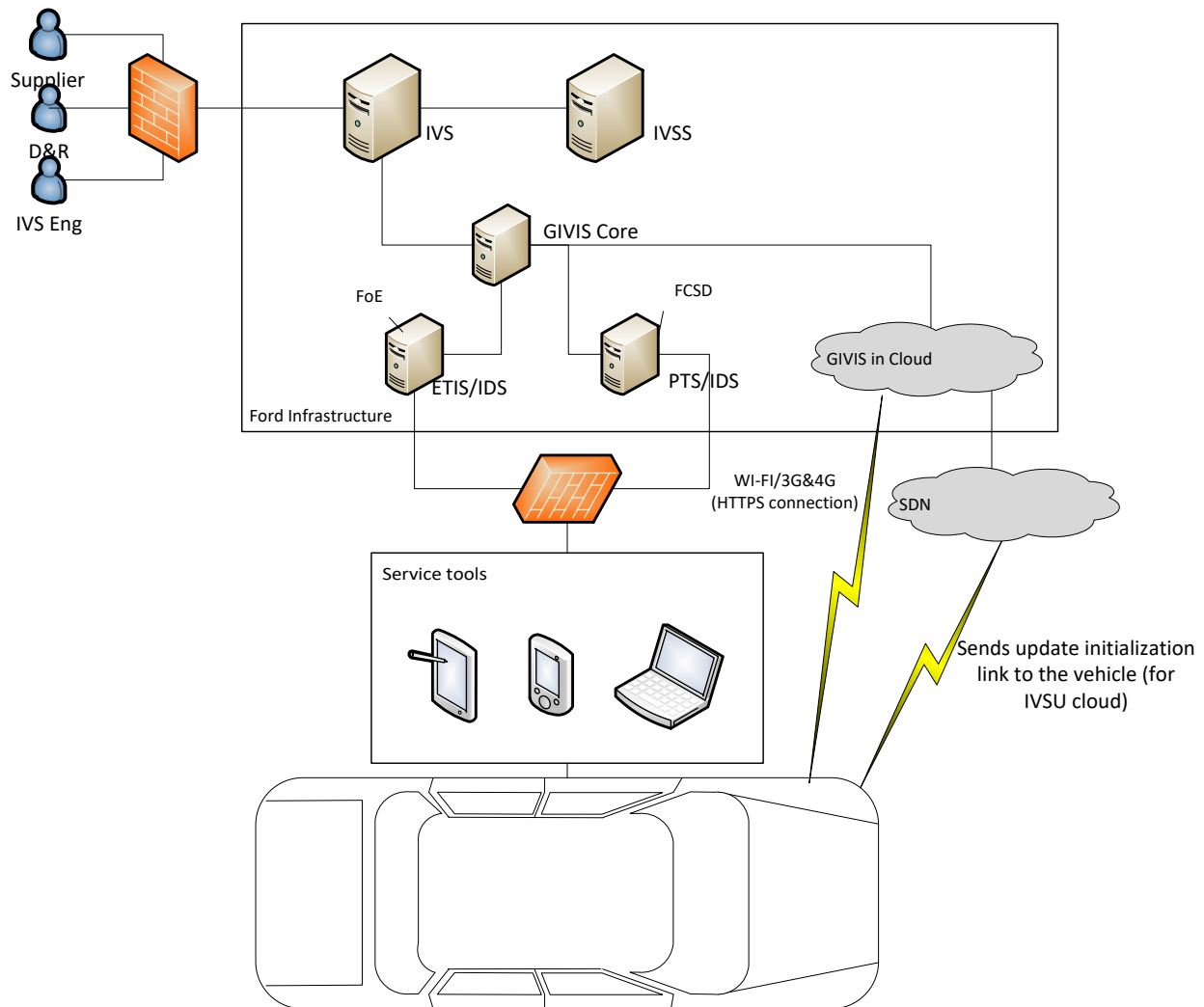
2 System Architecture

2.1 Assumptions

1. ECU's have adopted SW signing requirements
2. ECU's are using FNOS
3. SW update has been executed either via OBDII or OTA

2.2 System Architecture

High Level System Architecture





2.3 High Level Flow (Production SW signing diagram)

Step 1: Supplier request RSA keys from Ford

- Development key pair (private + public)(1)
- Production key (public only)(2)

Step 2: IVS passes the request to IVSS where the keys are generated

Step 3: IVSS send the applicable keys to assigned module owner/suppliers via IVS

Step 4: RSA keys are distributed to the supplier engineering and manufacturing

Step 5: SW files are uploaded to IVS to be signed (**released development and all production SW**)

Step 6: IVS passes the SW files to be signed to IVSS (internal)

Step 7: IVSS signs the SW files (internal)

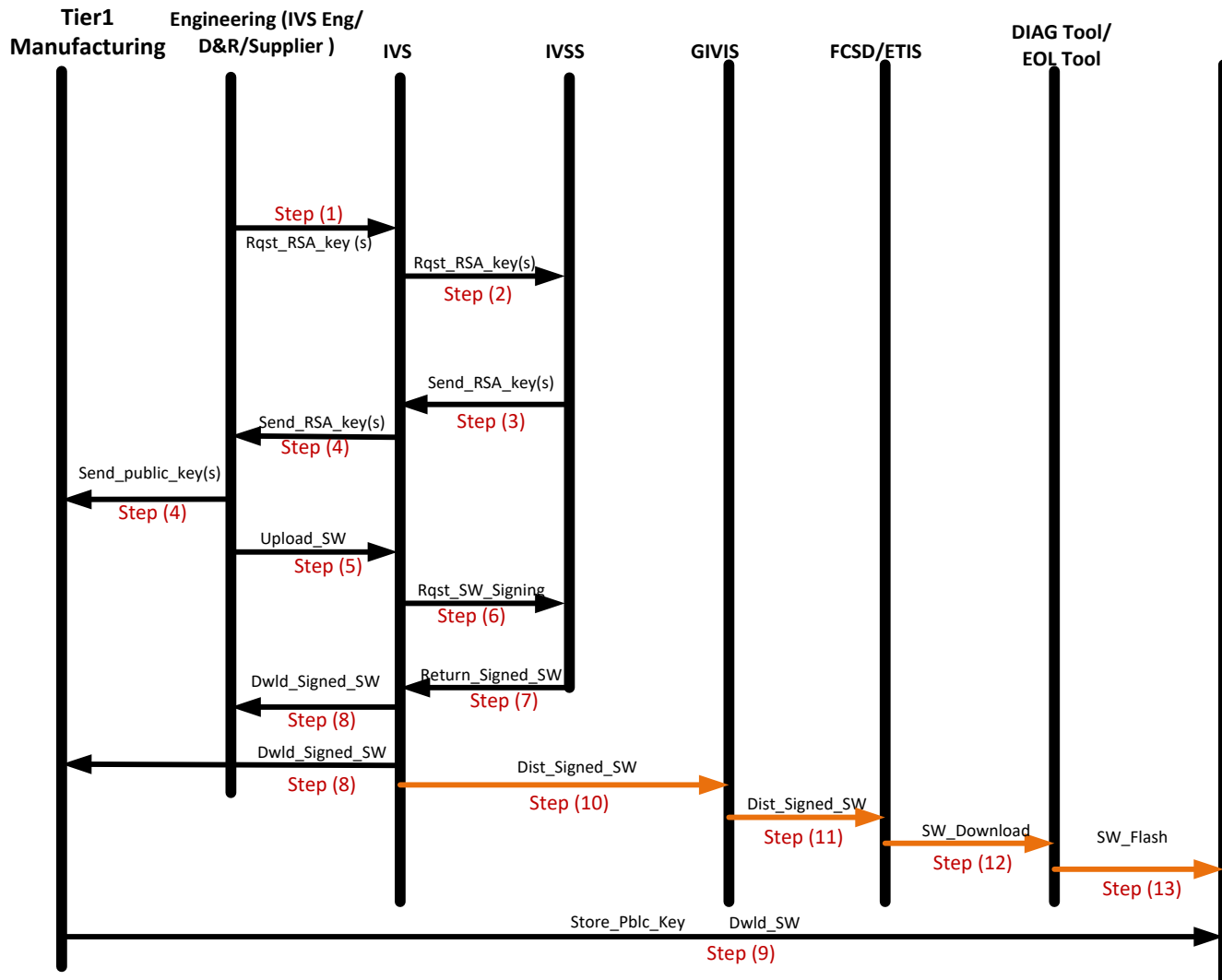
Step 8: SW files are distributed to supplier (Engineering or Manufacturing)

Step 9: Public key and SW is downloaded to the ECU

- Public key will be integrated with primary bootloader

Step 10 - Step 13: SW are distributed to either service and/or EOL.

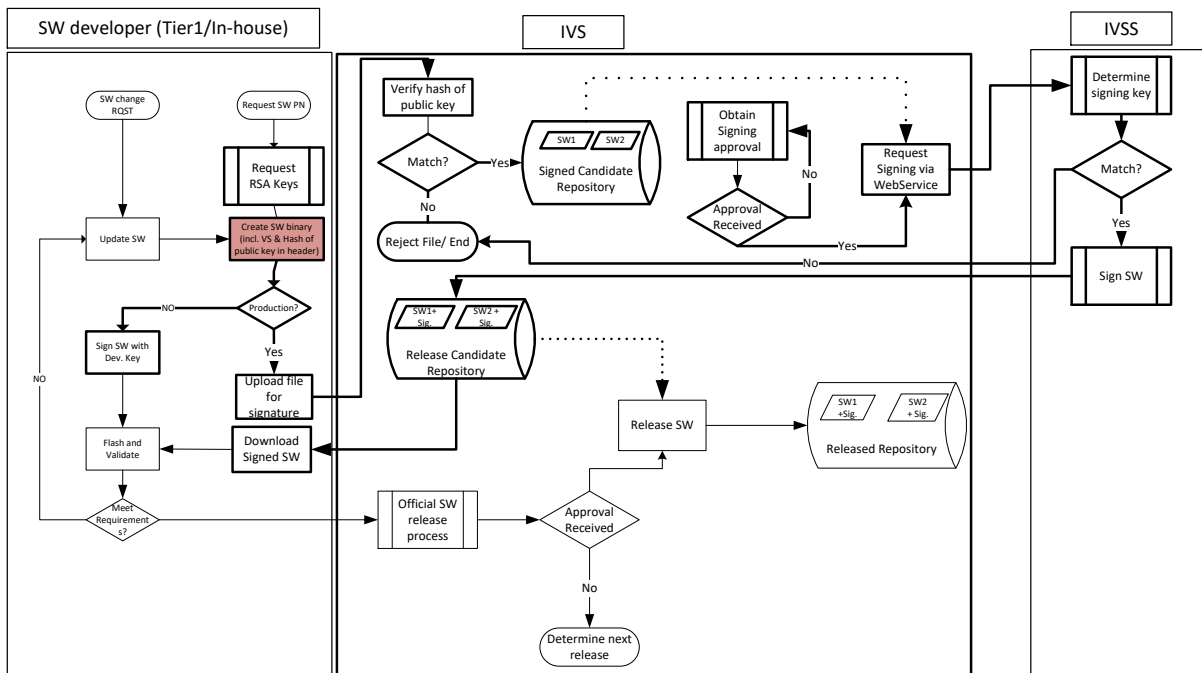
Process Step Diagram



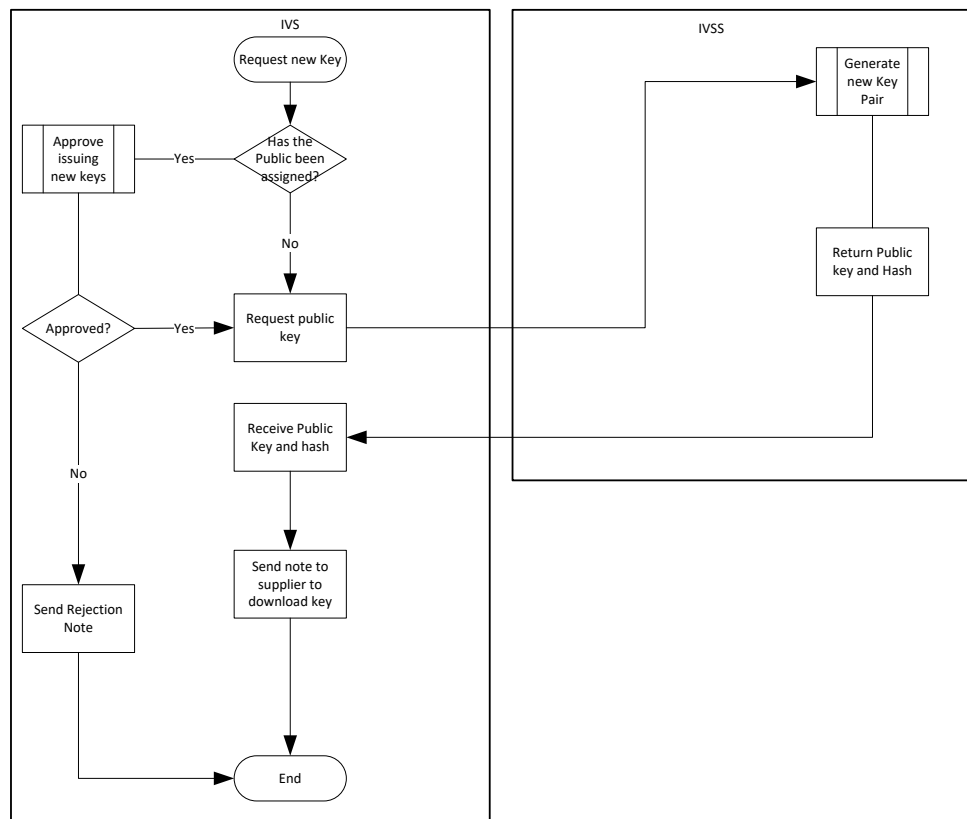


2.4 High Level Process Flow

Software Signing Process (Production)

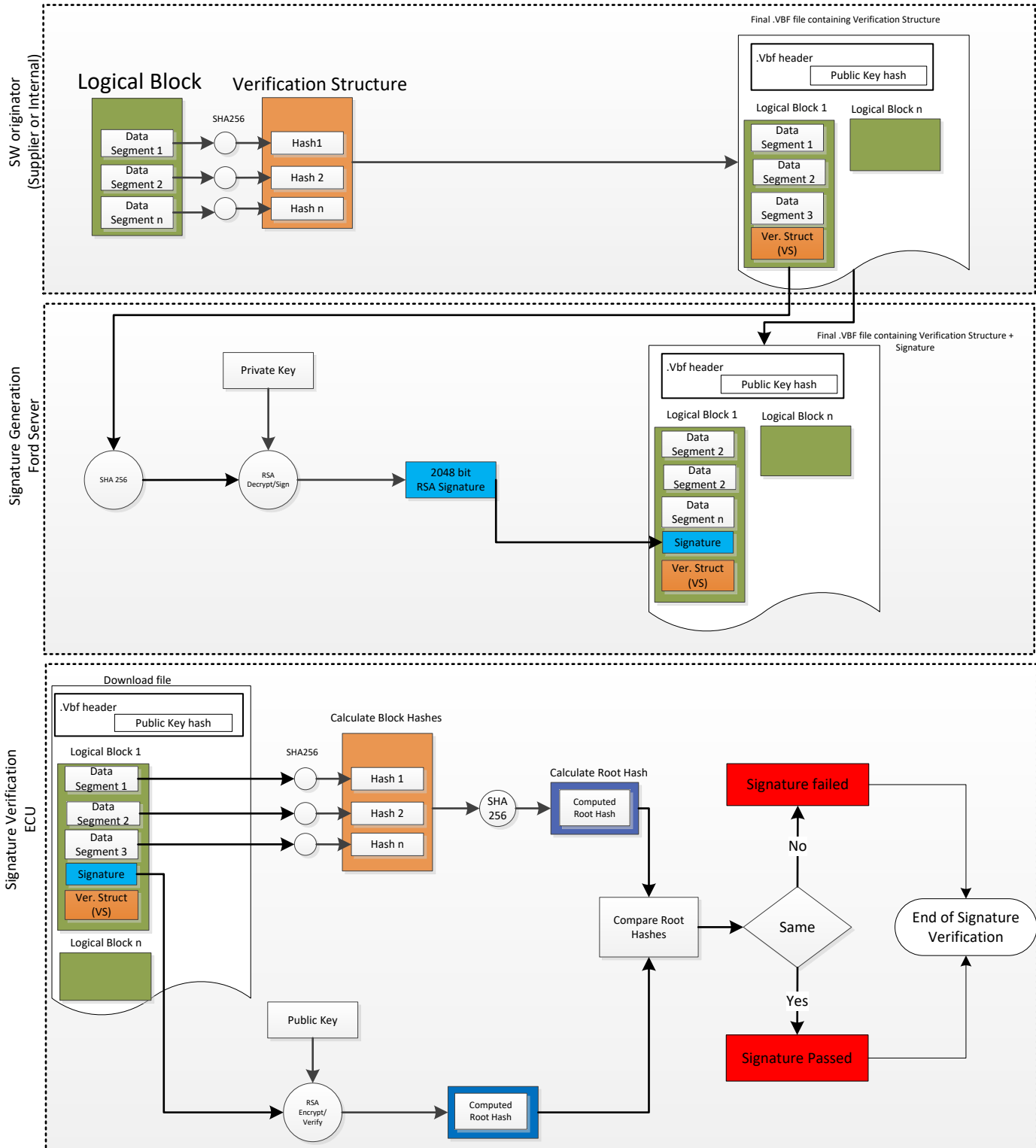


Public Key Request





2.5 Signing Method Illustration





3 Assumptions and Requirements

3.1 SW signing feature overview

1. All SW binaries (e.g., secondary bootloaders, applications, calibrations, cal-configs, etc.) flashed to an ECU shall be signed (includes CAN or OTA SW update).
2. Any other applications allowing update to the primary boot-loader will be signed.
3. Each module will use only one public key (either production key or development key) to verify the signature of multiple software files on that particular module.
4. Production and development keys will be generated at Ford and will be delivered to Tier1 suppliers via IVS interface
5. Ford will be the only entity to sign production binaries including calibration and cal-configurations files
6. Production binary files will be submitted to Ford to be signed before official release
7. Suppliers will be allowed to use development key pairs to sign their own software for development purposes
8. Hashing (SHA256) and crypto libraries (RSA) will be included in the Primary Boot Loader (PBL)
9. Supplier (Tier1) will compile public key to the primary boot-loader (PBL) and flash it to the module
10. Public key can be updated/changed via PBL update.
11. IVS will update current interface to include the following:
 - a. Request Development key pair (private and public key)
 - b. Request Production public key
 - c. Submit SW for signature before it is released
 - d. Govern SW signature approval
 - e. Check for public key hash in addition to current checks
 - f. Provide means for signed SW to be downloaded
12. Production SW will be signed automatically by Ford once the SW owner (Ford D&R/Supervisor) approves it
13. There is no impact to current process at VO or Service
14. The public key for verification shall not change except by manual process when the Ford secret key is compromised.

Note: Multiple methods of signature verification and or public key storage exist based on microcontroller capabilities. These alternatives may be allowed to be implemented once approved by Ford In-Vehicle Cyber Security.

Note: PBL, SBL are defined in Ford Software Download Specification (00.06.15.002).



3.2 Requirements Summary

Summary of requirements and roles and responsibilities:

Req List	SW Author /Tier1	PBL Supplier	Ford Backend	
			IVS	IVSS
KM-001				x
KM002	x			x
KM-003	x			x
KM-005			x	
SC-001	x			
SC-002				x
SC-003	x			x
SC-004				x
TE-001		x		
TE-002	x			
TE-003	x			
TE-004	x	x		
TE-005	x	x		
CR-001	x*	x		x
FF-001	x			

x - indicates that the requirement applies

* - Applies if crypto Libraries are not bought from PBL supplier

3.2.1 Key Management

REQ-KM-001

Title: Production key generation

Requirement Text: Production RSA keys shall be generated by Ford HSM (IT) and private key shall be stored securely in backend infrastructure (e.g. HSM)

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi, Xin Ye

Goal: Ensure that RSA keys are unique and meet the latest cryptographic standards. Ensure that RSA key generation does not produce common prime number between any two keys (including all product keys and development keys). The server HSM shall prevent/mitigate readability of the generated private keys.

Rationale: RSA keys are unique and all the security relies on the secrecy of the private key. RSA is based on hardness of factorization of products of large prime numbers. If two RSA key generations produce one common prime, the greatest common divisor of the two products gives out the factorization.

Verification Type: HSM security certification review. Must be FIPS 140-2 certified or later.

REQ-KM-002

Title: Development Keys

Requirement Text: Development RSA keys shall be generated by Ford HSM (IT) and both private and public key will be distributed to the suppliers. Development keys shall be used **ONLY** for signing development SW (Pre-TT).



Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal:

Rationale: Development and production SW will be signed with different keys for the following reasons:

- 1) Increased security for production SW
- 2) Reduce potential development delays from signing process
- 3) Verify signing process at early stages of ECU development

Verification Type: Design Review, IT security policy

REQ-KM-003

Title: Production key update

Requirement Text: Under special circumstances such as compromise of private key(s), FMC reserves the right to issue new RSA keys to an ECU. Any production key replacement shall result in PN change (\$F111) and shall be reviewed and approved by security team. Any PBL update application shall be reviewed by security team before submission for signing.

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Invalidate keys that are compromised by issuing new keys

Rationale: If private keys are compromised or SW is mis-signed to include more than one production key, FMC can issue new key pairs for future production modules and update vehicles build via service tools.

Verification Type: ECU DV, IT infrastructure test

REQ-KM-004

Title: Key pair back-up at FMC infrastructure (HSM)

Requirement Text: FMC shall ensure that all the keys are backed up properly and recovered in case of data loss. Signing keys shall be available to support SW update lifecycle for a particular ECU.

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Backup key pair database for future SW updates.

Rationale: Key pairs must be backed up securely to protect against any data loss. In addition, all the keys must be available to support SW updates per the life of the vehicle.

Verification Type: IT infrastructure verification

REQ-KM-005

Title: Key assignment in ECU's

Requirement Text: FMC shall issue different signing keys based on the following rules:

- 1) Each supplier of the same module shall have a different key
- 2) Each module family shall have a different key
- 3) Major HW changes where SW isn't backwards compatible



- 4) Under special circumstances, such as build to print HW where the same SW files are flashed in multiple HW PNs the same key pair can be shared to reduce overall SW complexity

Definition:

1: Supplier A and supplier B produce the same part for a one or multiple vehicle programs. In this case each supplier must have different productions signing keys.

2: Supplier A can produce two or more different modules in the same vehicle program(s). In this case each module must be assigned different key for SW

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Avoid use of the same production key pair for multiple modules or suppliers

Rationale: The exposure should be minimized in case of a security breach such if the private key is leaked.

Verification Type: IVS testing, IT backend capability review

3.2.2 Signature Creation

REQ-SC-001

Title: Signing Methods

Requirement Text: All applicable SW files shall be signed:

- a) Development SW files (pre-TT) shall be signed using development keys
Ford SMEs may propose alternative methods to meet this requirement and minimize the impact on the development process. These proposals shall be reviewed and be accepted by In-Vehicle cyber security group
- b) Production (TT and beyond) software such as application binary, SBL, calibration and configuration files shall be signed solely by FMC using valid production certificates (private keys)

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Ensure that only valid and authenticated software can be flashed in the vehicle

Rationale: RSA2048 public/private key is an approved algorithm which is considered to be secure by NIST until 2030.

Verification Type: Design Review, Component and Vehicle DV

REQ-SC-002

Title: Signing production SW with private key

Requirement Text: FORD secure backend (IVSS) shall sign all production applicable software using **Private Key** assigned to the module.

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

**Goal: Ensure that the signature is created by using valid key pair**

Rationale: IVSS must ensure that each applicable SW file is signed with the correct private key that was assigned previously to the module. To ensure that, each of the VBF file will include the hash (SHA-256) of the public key that will be used by IVSS to ensure that the correct private key is used.

Verification Type: Design Review, Component and Vehicle DV

REQ-SC-003

Title: Verification structure

Requirement Text: SW author (e.g. Tier1/Ford SW) shall calculate hash values for each data segment and include them in the verification structure (VS) block (HexView or any compatible VBF creator tool can be used). VS shall be part of VBF file and Ford IVSS shall extract it to create the signature. SW author shall ensure the verification structure (s) includes **ALL** data segments of the binary file planned to be flashed into the module. VS components version, length, data segment's address and length are in Big-Endian format.

Address Offset	Size [byte]	Record
Prolog (verification structure starts at the address which is previously configure in the PBL)		
-0x100	256	RSA signature
0x00	2	Verification Structure version
0x02	2	Data segment count: the number of blocks which are described in the block info section
Data segment info section (i is the data segment index, each data segment record has as size of 10 bytes)		
0x04 + (i * 40)	4	Data Segment address: start address of the download block which shall be checked
0x08 + (i * 40)	4	Data Segment size: Size of the download block which shall be checked
0x0C + (i * 40)	32	Data Segment Hash: 32 byte SHA256 hash value of the download block which should be checked

Definition:

- 1) Logical block:** Block is a software unit that consists of a predefined erasable and programmable contiguous memory area that contains a single verification structure and a single signature. All data segments within a given verification structure are always contained within the same logical block as the verification structure itself
- 2) Data segment:** A physical address/length pair that represents a contiguous memory area that is included in the signature check
- 3) Verification structure (VS):** a structure required for each logical block that contains Verification Structure Version, number of data segments, starting address of each data segment, length of each data segment and the hash of the data within each data segment.
- 4) Verification structure version:** The current version of Verification Structure as specified by Ford. The verification structure version value required by ECUs supporting this specification is **0x0000**
- 5) Root Hash:** The hash calculated over a single verification structure used to generate the digital signature



Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Ensure that Hash table includes ALL SW components

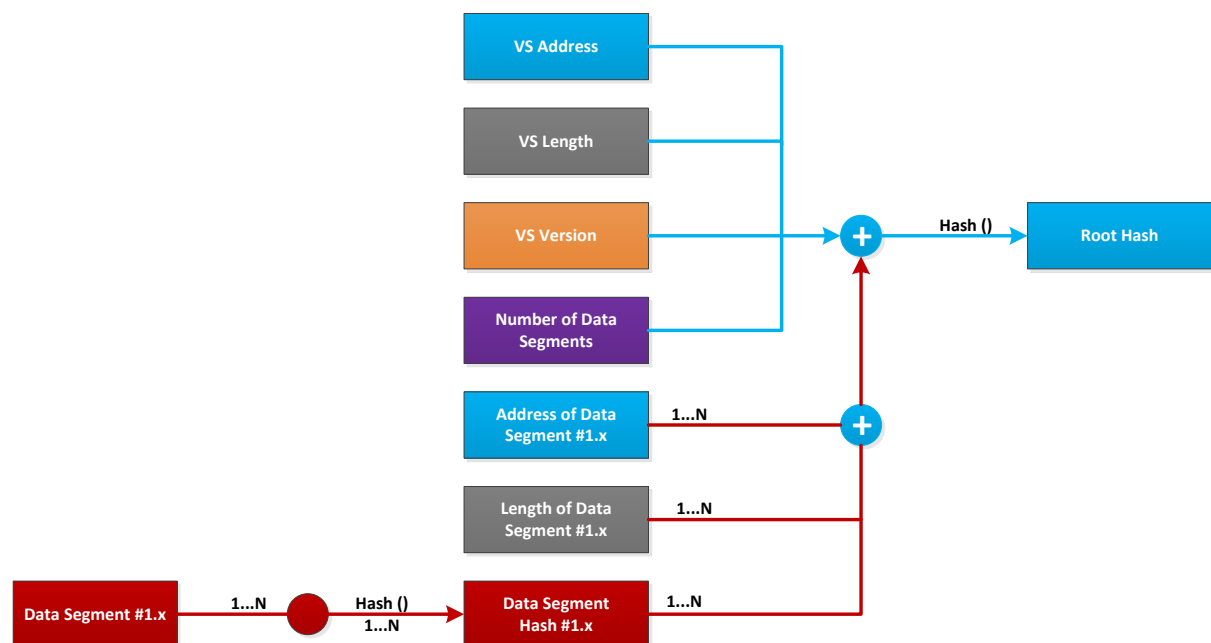
Rationale: It is important to ensure that the hash value of the SW includes the entire binary file that will be flashed in the module. Failure to do so, will result in signature failure or in a security flaw where part of SW can be modified after the signing process.

Verification Type: Design Review, Component and Vehicle DV

REQ-SC-004

Title: Verification of root hash upon submission

Requirement Text: Verification Structure shall be verified by Ford IVSS before root hash and signature is created. Ford IVSS shall recalculate all the “data segment hashes” and compare with the values included in the verification structure. Ford shall reject any file signing request with miscalculated hash values or files containing a signature.



Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Ensure the integrity of the SW received by FMC from the vendor before signature creation and software being released to the vehicle

Rationale: It is important to ensure that the hash value of the SW includes the entire binary file that will be flashed in the module. Ford IVSS will calculate the hash of the received file using the same algorithm/tool as the SW author before signing it.

Verification Type: Design Review, Component and Vehicle DV

**3.2.3 Target ECU****REQ-TE-001**

Title: Cryptographic libraries

Requirement Text: All cryptographic libraries shall be reviewed and approved by FMC security team.

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Ensure that any cryptographic libraries used meet security standards

Rationale: RSA and SHA 256 libraries will be created by FNOS supplier and will be included in the current security libraries distributed with FNOS. Alternatively, tier 1 suppliers may choose to implement Signature verification outside of PBL with approval of Ford Cyber Security team.

Verification Type: Design Review, Component and Vehicle DV

REQ-TE-002

Title: Public key storage

Requirement Text: Public key(s) shall be stored securely in the module and shall be protected against unauthorized modifications.

Option 1:

Single Key Solution: Primary/Ford bootloader shall contain only one key (development or production) which shall be integrated in the PBL code.

Public key shall NOT be allowed to be modified via diagnostics or internal ECU application(s). Any applications written to re-flash production primary bootloader via diagnostics shall be approved and signed by FMC.

Option 2:

Two Key Solution:

When primary bootloader contains both development and production key, the following requirements shall be met:

- 1) Public keys (asymmetric) to verify token shall be stored in a read-only area
- 2) Production key shall be the default key used to verify the signature.
- 3) Unique ESN (ECU Serial Number) Password/tokens shall be used to temporarily change default key to development key
- 4) Passwords/Signing keys shall be stored securely in either Ford or Supplier's Trust Center
- 5) Only authorized personnel shall be allowed to access passwords/token
- 6) Monitoring shall be in place in trust center to log following: Requested for (ESN), Requested By (user), Provided by (user), Date provided (date)
- 7) Persistent DTC(#tbd) shall be logged to indicate development key is default
- 8) Persistent DTC (#TBD) shall be logged if any software has been verified with a hash of the signed public key that does not match the hash of the production key stored in secure memory. The hash of the public key used to decrypt code is stored at the end of the file, as shown below.

Note: Reference Appendix 4.1 for example of token implementation

Owning CPSC: 00.14.03



Requirement Author: Aldi Caushi

Goal: Ensure that public key is securely stored and not allowed to be modified

Rationale: Reading the public key from the module does not compromise the security. However, - unauthorized modification of the public key can vanish security protection from software signing process. Keys are burned/flushed in the ECU during manufacturing process and the ECU supplier must ensure that any future update of the primary boot loader via diagnostics is done by secure methods such as application signing. It is important, that any direct HW access such as JTAG has been disabled or locked with a password protection.

Verification Type: Design Review, Component DV

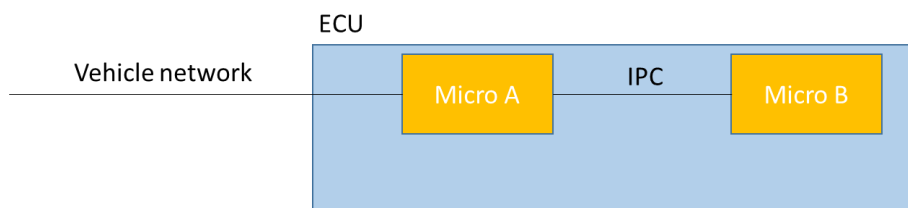
REQ-TE-003

Title: Multi processor modules and slave modules

Requirement Text: When an ECU has multiple microcontrollers, the target microcontroller shall be the only entity to verify its own SW signature independently of SW update method. In-vehicle cyber security team may consider alternate proposals upon detailed review of implementing adequate compensating controls.

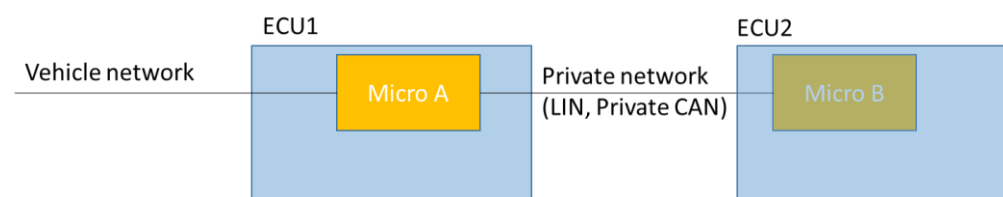
Example 1: Host micro (Micro A) may be allowed to verify SW signature of the secondary micro(s) (Micro B) when the all the following conditions are met

- 1) Micro B has limited resources (RAM, Flash) and can't implement SW signing
- 2) Micro B has not implemented FNOS
- 3) Micro B has no direct external connections (CAN, Serial, Wi-Fi, BlueTooth, RF etc)
- 4) Micro A implements satisfactory compensating controls (e.g. whitelisting of all data passed on vehicle networks)



Example 2: Host micro (Micro A) may be allowed to verify SW signature of the Micro B even if it located in a different ECU2 (e.g. sensors, antennas) when all the conditions below are met:

- 1) Micro B has limited resources (RAM, Flash) and can't implement SW signing
- 2) Micro B has not implemented FNOS
- 3) Micro B has no direct external connections (CAN, Serial, Wi-Fi, BlueTooth, RF etc)
- 4) Micro A implements satisfactory compensating controls (e.g. whitelisting of all data passed on vehicle networks)



Owning CPSC: 00.14.03



Requirement Author: Aldi Caushi

Goal: Ensure that each microcontroller verifies the signature for any SW update

Rationale: If an ECU has two or more processors that can be independently updated, verifying the signature by one processor is not enough to ensure that second processor will get authenticated code. When the secondary micro(s) are not connected to other networks (external or vehicle) the risk is slightly reduced if the main micro verifies the code and effective compensating controls are in place.

Verification Type: Design Review, Component and Vehicle DV

REQ-TE-004

Title: Troubleshooting via diagnostics

Requirement Text: ECU shall report the hash of public key (SHA256) and provide it via diagnostic requests through DID \$D03F (In-Use Application Signing Public Key Hash). DID \$D03F shall be supported in both bootloader (PBL) and application.

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Enable engineers to compare the key information that is stored in the module and the value that is embedded in VBF files.

Rationale: Provides troubleshooting methods to determine the reason why the signature failed and allows tester to check applicability of a given file prior to initiating the download.

Verification Type: Design Review, Component and Vehicle DV

REQ-TE-005

Title: Signature failure feedback via diagnostics

Requirement Text: The ECU shall provide feedback to flashing tool (OTA or CAN) if the signature verification failed. ECU shall implement DID \$D028 to support SW signature troubleshooting.

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Distinguish Signature failure with current SW download failures

Rationale: Provide feedback to flashing tool if the signature failed.

Verification Type: Design Review, Component and Vehicle DV



3.2.4 Crypto Requirements

REQ-CR-001

Title: Cryptographic Standards

Requirement Text: Software signing shall use PKCS#1 v.2.2 (PSS padding) standard with RSA2048 for key generation, signature generation, and signature verification. SHA-2 with 256 long digest shall be used to create the Verification Structure and Root Hash (see REQ-SC-003 for definition)

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi, Xin Ye

Goal: Use industry reviewed security algorithms to sign software binaries

Rationale: Integrity check of the binary is achieved by signing the hash table and RSA2048 is an industry standard.

Verification Type: Design Review, Component and Vehicle DV

3.2.5 File Format

REQ-FF-001

Title: File format

Requirement Text:

All unsigned files that require signing shall contain the following upon submission to Ford (see Appendix 4.2 for example):

- 1) Have .vbu extension
- 2) "public_key_hash" field populated with correct hash value in VBF header
- 3) Address(s) of the location of VS(s) (populated in verification_structure_address of VBF header)
 - a. Note the verification_structure_address always points to the most significant byte of the verification structure version
- 4) Properly formatted verification structure (see **REQ-SC-003** in VBF data section (where VBF data block address equals verification structure address))
- 5) No signature(s) present in the file (VBF data section)
- 6) No "sw_signature" field present in the VBF header
- 7) **When file compression is required, all files shall be uploaded compressed using the Ford Data Compression and Encryption Specification (00.06.15.005).**

All signed files returned from Ford shall have the following (see Appendix 4.3 for example):

- 1) Have .vbf extension
- 2) "public_key_hash" field populated with correct hash value in VBF header
- 3) Address(s) of the location of VS(s) (populated in verification_structure_address of VBF header)

Note: The verification_structure_address always points to the most significant byte of the verification structure version
- 4) Properly formatted verification structure (see **REQ-SC-003** in VBF data section where VBF data block address equals verification structure address)
- 5) Signature(s) present in the file (data section)



- 6) "sw_signature" field present in the VBF header with valid signatures matching data section values
- 7) When file compression is required, all files will be returned compressed to the submitter

Owning CPSC: 00.14.03

Requirement Author: Aldi Caushi

Goal: Maintain a standard file formation

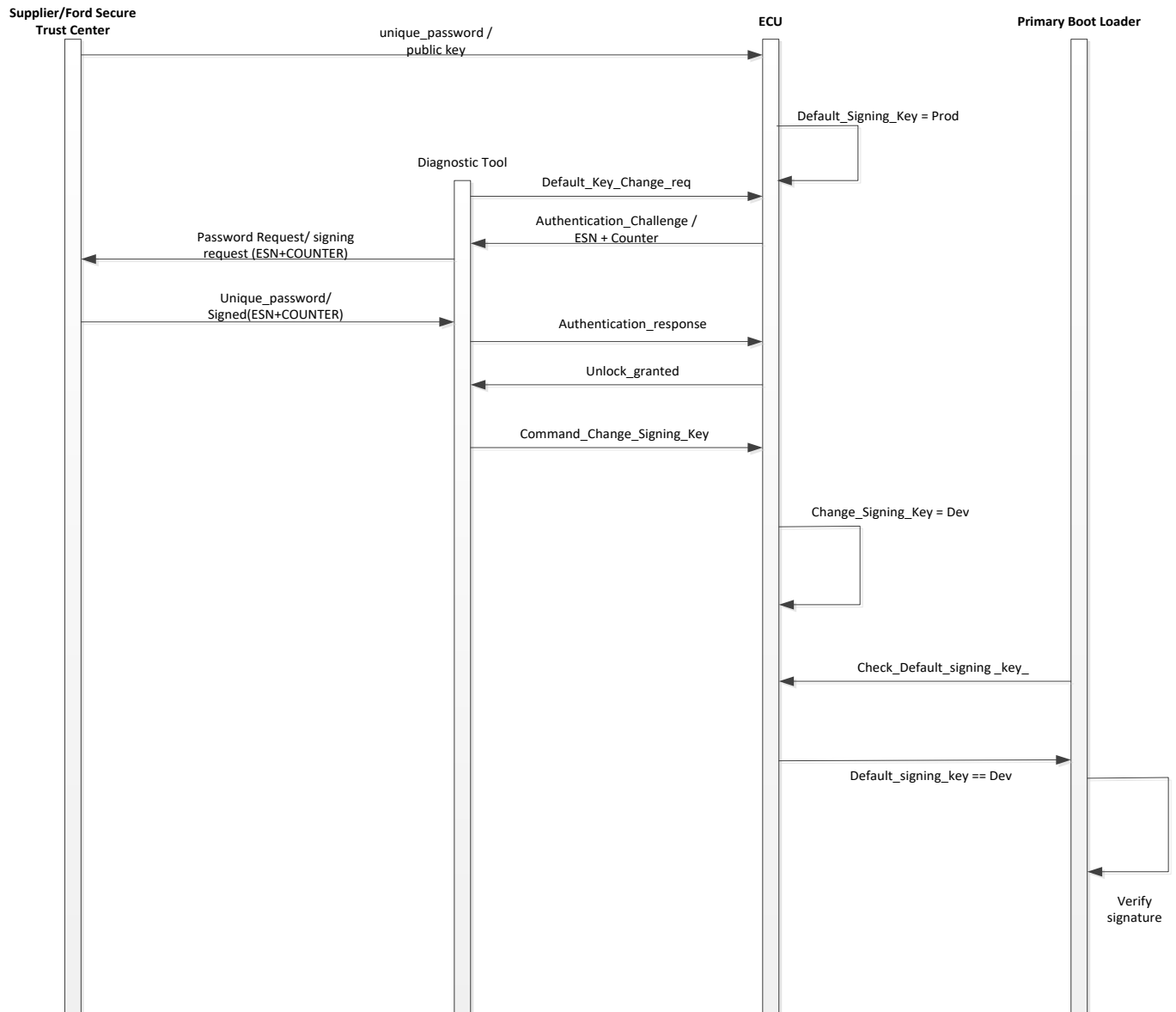
Rationale: SW signature will be created using automated process by FMC backend and a known format is necessary to complete the process.

Verification Type: Spec review, SW DV



4 Appendix:

4.1 Token Implementation Design Guide









4.2 Valid unsigned file (header + VS)

Example of valid unsigned file (.vbu)

```
vbf_version = 3.1;
header
{
    // Copyright Ford Motor Company.
    // Generated on Wed Mar 14 11:14:34 2018
    // Copyright Ford Motor Company.
    // Generated on Thu Mar 8 14:08:51 2018
    // P1864.h32 -> TEST-14C204-CAL.VBU
    description =
    {
        "PRISM: B_MDC1P006_TC29X_8M_IB",
        "Comment: FOR TEST ONLY. NOT FOR PRODUCTION.",
        "Released By: TBD",
        "File Name: TEST-14C204-CAL.VBU",
        "Vehicle Calibration: A",
        "CHIPID: 0x8040022C 0",
        "WERS: A",
        "Module PN: TEST-P1864-MOD",
        "Cal PN: TEST-14C204-CAL"
    };
    sw_part_number = "TEST-14C204-CAL";
    sw_part_type = EXE;
    ecu_address = 0x7E0;
    frame_format = CAN_STANDARD;
    erase =
    {
        {0x80080000,0x00180000},
        {0x80200000,0x00200000},
        {0x80400000,0x00200000},
        {0x80600000,0x00100000},
        {0x80700000,0x00100000}
    };
    verification_structure_address = { 0x801FFF00, 0x803FFF00, 0x805FFF00, 0x806FFF00, 0x807FFF00 };
    public_key_hash="921E08F889E7DD71AE5E3765C67E3B5C47E6B582EDF65EEF5038D640E0EBC490";
    file_checksum = 0xD7891D8D;
}
```

Example VS from VBF file for verification_structure_address 0x801FFF00.

801FFF00:	00 00 00 01	80 08 01 00	00 17 FD 00	5D 10 74 FB].t.
801FFF10:	C6 9C D3 88	23 1B AA 76	6F 29 8E D4	13 BE 31 EA#..vo)....1.
801FFF20:	46 8F 04 3E	62 AE A2 21	12 18 BD EA		F..>b..!....

	Verification Structure Version		Data segment address: start address of the data segment
	Data Segment Count		Data segment size: size of the data segment
	Data Segment Hash: 32 byte SHA256 Hash value of data segment		



4.3 Valid signed file (header + VS + Signature)

Example of valid signed file (.vbf)

```
vbf_version = 3.1;
header
{
    // Copyright Ford Motor Company.
    // Generated on Wed Mar 14 11:14:34 2018
    // Copyright Ford Motor Company.
    // Generated on Thu Mar 8 14:08:51 2018
    // P1864.h32 -> TEST-14C204-CAL.VBU
    description =
    {
        "PRISM: B_MDC1P006_TC29X_8M_IB",
        "Comment: FOR TEST ONLY. NOT FOR PRODUCTION.",
        "Released By: TBD",
        "File Name: TEST-14C204-CAL.VBU",
        "Vehicle Calibration: A",
        "CHIPID: 0x8040022C 0",
        "WERS: A",
        "Module PN: TEST-P1864-MOD",
        "Cal PN: TEST-14C204-CAL"
    };
    sw_part_number = "TEST-14C204-CAL";
    sw_part_type = EXE;
    ecu_address = 0x7E0;
    frame_format = CAN_STANDARD;
    erase =
    {
        {0x80080000,0x00180000},
        {0x80200000,0x00200000},
        {0x80400000,0x00200000},
        {0x80600000,0x00100000},
        {0x80700000,0x00100000}
    };
    verification_structure_address = { 0x801FFF00, 0x803FFF00, 0x805FFF00, 0x806FFF00, 0x807FFF00 };

    sw_signature = {
"1F79D5E43C52976CF0E43B7D29DE7B88B7065A321CABE85097C37A61CB1C6F7AE394D5DC5AECFA95622755240D18B0A142645B
06E8D8C964ADFD233A16D5130F2E660DFB279950916E42315809AA5EBFA539A174369ABA283A9AFD441C58405C5DD1F244A93C0
ABF1A20858B562E5646FB808FE338F92D806781053D37D152186D0913D6AEDBF0692A5104D86832F1247B3A82E52E6C1A246F3A
545B0F7FB6BA3A61992ABC08F1D2BBC209E7EF0D687E32CBC6BB232908EB2E295715BCAF655AB95201D667EA07A3E92BD7C9053
90DE771B858975CBE4ACC2A93DBFB2F16CF124C0E6328BAA75A59EC319DF661E397DAF3E3BB44FBF51AA1C69477F083CB37D3",
"AEDDEF732A4D5E6D1300B3AD06008DFC4A30A87C87FFC28128969400521814EE9CE2FA814629CF6A98F69BFE6C780DD0A79286
AF0FE79C799EE101C19EC7BE0EECC1BAE751DFB86772F51A0194D8B7A2F1C910D37400AFC1403543E15B821E7B3608A6691D96F
7C7D92EA49BB69F12569CC5D653504BC8CD03800E50263C0FD3AC520C80FD259E656D7370214E261D09078A9D838117FFC06ECE
EC0C718E5CD047FB898506F18447CBAEF8EB67612018C4DB7472A5CE65042277F2D2040A17B85CB17E9881945CA16833929EFEA
D26BC4051940D8549C748E457200CEFA5997CB350A0F840CEA5E145F340D05B3CD412330717F4F37B19D200609B16295BCF83",
"B576EFFFFB740ECC38AF1573A7D225184C73DD6FBFA01AE26C1B1DB4797D81F19E19E7356C6E34CBFE5C81134BD15AE1AC617B6
59E0114273E5D31C773B1986B5AC7F7F68E9883D8412F07BE0B17FAE37920BE9D9F62F67E155EDA8CB427F80B3C3E673A5554AF
13AF3FD83B927F6609A9BD6B8D183FCBA8C5CE967D696E748394ECFF841852FC634A967024E16C40CAAC8EED332A7946038A20F
DF26C513000062FE8C797A35DDB699C563526425B62C5E639E47A7BFB0217E5377702D4F6BF9AA883AA4A087988A3163BC923D3
28ECC9A3401002B5774297D0A3B98C4259EDAFEE85916C3742F3FA37CE3B4E196926F43525EBCED41244902BE454E386C14AF",
"15E3A098E9FCC11F35D0510224E47EF8203A4C7AC46FAA38A39E098A09BF55E0C84EA3C89CAC3BE3C8C4B0E9A7426846159170
A6F7DCCCE5866F4D1E68BC48067118AC5D0844E1E97F14E1F2645F322055DE6EB1ADCB9D58524610DF071D2C64672060EF47B2
B9CF49C166D4ED9C7F3007AFDAAF60EC6C51A0F85D663BF89848A4C29FF496B4A9B129A2829ECDEFBE9D59A568417450A77F55D
2FA50231D10F2B3CD1EE7014916EA9A22349B5FAEDB1931071D7582C37C8B276E071D97045D65448B0C9352AA5D9AFFA65BBB94
2B512197CF54108DB355AD28F2EED2089228E0D14C297CBB7F6A676D794E3EC5E35B68CCCEEB1B8338A9A68EBCDC086E71FB1",
"609B06EB1E7FDE3236800B4B4EA484310ACDDE754E020F4A80712139AB57A485C724A714B46CC48FF455AF6BF3EC461196250C
16F089D5DBB0EC640752520FA275FDE64EC72C0BD49B61D905D24AC71FB14A9FBCD723999BD937E6C49044E9D9CECA6C84D832B
6A8A2F8AC31CEFC1349719ABAC45029981502463789FF8C038F827C7A21001F55652E3B2492BA60A020509E60088345F86866C4
5D3EF463B7132BAB0F71F26596A6B38E854CF931ADCCD2AC49780ADFE2931FC5ACB090B5CAFD8696268AD64E1E9CA2DA4D8ADF8
4E3926EACCF0C603F60A320374C55727EAE37603CE87B45B8D9C591EF209B3BF1ADB87039D9D84E31F43903E784E1CDA3DA92"
};
    public_key_hash="921E08F889E7DD71AE5E3765C67E3B5C47E6B582EDF65EEF5038D640E0EBC490";
    file_checksum = 0x6EDEB827E;
}
```



Example VS + Signature from VBF file for verification structure address 0x801FFF00.

801FFE00:	1F 79 D5 E4 3C 52 97 6C F0 E4 3B 7D 29 DE 7B 88	.y.<R.1...;)).{.
801FFE10:	B7 06 5A 32 1C AB E8 50 97 C3 7A 61 CB 1C 6F 7A	..Z2...P...za..oz
801FFE20:	E3 94 D5 DC 5A EC FA 95 62 27 55 24 0D 18 B0 A1Z...b'U\$....
801FFE30:	42 64 5B 06 E8 D8 C9 64 AD FD 23 3A 16 D5 13 0F	Bd[....d...#:... ..
801FFE40:	2E 66 0D FB 27 99 50 91 6E 42 31 58 09 AA 5E BF	.f...'.P.nB1X...^.
801FFE50:	A5 39 A1 74 36 9A BA 28 3A 9A FD 44 1C 58 40 5C	.9.t6...(:.D.X@\
801FFE60:	5D D1 F2 44 A9 3C 0A BF 1A 20 85 8B 56 2E 56 46]..D.<... ..V.VF
801FFE70:	FB 80 8F E3 38 F9 2D 80 67 81 05 3D 37 D1 52 188.-.g..=7.R.
801FFE80:	6D 09 13 D6 AE DB F0 69 2A 51 04 D8 68 32 F1 24	m.....i*Q..h2.\$
801FFE90:	7B 3A 82 E5 2E 6C 1A 24 6F 3A 54 5B 0F 7F B6 BA	{:...l.\$o:T[.[]..
801FFEA0:	3A 61 99 2A BC 08 F1 D2 BB C2 09 E7 EF 0D 68 7E	:a.*.....h~
801FFEB0:	32 CB C6 BB 23 29 08 EB 2E 29 57 15 BC AF 65 5A	2...#)...)W...eZ
801FFEC0:	B9 52 01 D6 67 EA 07 A3 E9 2B D7 C9 05 39 0D E7	.R..g....+...9..
801FFED0:	71 B8 58 97 5C BE 4A CC 2A 93 DB FB 2F 16 CF 12	q.X.\.J.*.../...
801FFEE0:	4C 0E 63 28 BA A7 5A 59 EC 31 9D F6 61 E3 97 DA	L.c(..ZY.1..a...
801FFF00:	F3 E3 BB 44 FB F5 1A A1 C6 94 77 F0 83 CB 37 D3	...D.....w...7.
801FFF00:	00 00 00 01 80 08 01 00 00 17 FD 00 5D 10 74 FBt.
801FFF10:	C6 9C D3 88 23 1B AA 76 6F 29 8E D4 13 BE 31 EA#.vo)....1.
801FFF20:	46 8F 04 3E 62 AE A2 21 12 18 BD EA	F..>b...!....

	Verification Structure Version		Data segment address: start address of the data segment
	Data Segment Count		Data segment size: size of the data segment
	Data Segment Hash: 32 byte SHA256 Hash value of data segment		Signature

4.4 Reference Documents

Reference #	Document Title
1	Software Download Specification Rev.006 or later
2	VBF file format specification ver.3.1 or later
3	Data Compression and Encryption Specification
4	Software Signing Users Guide v0.6 or Later