**Cybersecurity**

Asymmetric Encryption

In 5DH there are 3 kind of protection involved

1. JTAG protected by password
2. Diagnostics Hardening using SaSrv

Public key Private key encryption: Public key encryption

1. Digital signing for the application: Private key encryption

Here for the encryption there are 2 key pairs

1)Engineering Key pair: Used by delta/internal purposes of testing

2)Serial Key Pair: Only the public component of it is available with us. The private part is with the customer

IF DID F062 SaSrv\_Virginflag

If flag=00; Dummy Security

If flag=FF; Actual security

If Distance Totalizer >500: Flag sets to FF Always, even if you force to change the flag with 2E service

**HW Compatibility**

In **Info.txt** file of application

**FBL\_apdi.c** of FBL you can find the Hw compatibility bit.

To get the memory location: check into helper file.py

0xA0020051 TO 0xA0020057

It gets incremented according to the SW

As of now

Serial: 07 in FBL 23\_19\_00(with serial public key)

Engineering: 06 in FBL 23\_19\_01(with engineering public key)

While flashing in Vflash If error

1. DTC failed: Due to HW incompatibility
2. Security Denied

In Dummy Security it will not check for security while flashing

In Engineering Release FBL: 23\_19\_01

0xA001C106 TO 0xA001C205 (which we can get from the **map file** by looking for the variable in **bin**)

Public Key: CBEF------0C5F (256 BYTE)

Private Key: In S:\70\_Tools\30\_CANoe\Nodes

In Serial FBL 23-19\_00

Public key: BC11-----D7E7 (256 Byte)

For FBL we can find the Public key in SecMpar.c file

D:\SVN\STEP1M\_auto\_renault\DES\_SWEET400\_FBL\_Base\_22\_26\_00\10\_FBL\_LV\40\_Appl\GenData

In APPl we can find it in SaSrv SWC

FBL will be writing the Public key into OTP section (0xA001C106 TO 0xA001C205) which cannot be over written by the application

* During the init phase after the ECU start up
  + The random number generator is initialized
  + The public key is read from the memory and stored in the CSM key for asymmetric encryption
  + The same public key is read from the memory and stored in the CSM key for signature verification
* When the seed is requested by the user
  + The conditions regarding the Virgin Flags are checked
  + The plain seed is calculated which consist of a random number, a global counter for the security access attempts, information about the mode and security level and the serial number of the device.
  + And then the plain seed is encrypted with the public key with RSA-OAEP 2048
* The user decrypts the seed and generates the signature of the seed and sends this back as the “key”.
* The “key” is verified by the ECU
  + The verification function sets a flag, so that the seed generation function generates a new seed, when it is called again.
  + The global counter is increased and stored in NvM.
  + The signature of the generated seed is verified with the received signature (“key”)
  + If successful the security level is set.

An RSA key with 2048bit key length is used. The RSA-OAEP primitive is used for the encryption and RSA-PSS is used for the signature verification. SHA2 256 bit hashing is used for all RSA operations. The cryptographic library provided by vector with the Vector Autosar stack is used. No known vulnerabilities have been found for this library.

In the security access implementation in the application the random number generator is seeded every time freshly when a Security Access Seed request is done. But the seed generationfor the PRNG is not sourced by anunpredictable source (see Finding 2).

In the flashbootloader the random number is generated with a cyclic HW counter value. Although the entropy of the random number may not be very high, given that the PRNG is based on the previous value and the current HW counter value, catalogue attacks seem difficult, at least for the security access implementation in the flashbootloader.

The security access implementation in application and flashbootloader does not accept multiple responses after a failed attempt without executing the “get seed” request again. Exhaustive brute force attacks can not be done. Furthermore, a time delay for further attempts is activated, if 10 failed attempts are done.

Finding\_2: The seed is not set yet and still has default values when used as a seed value for the the random number generator in the security access function in the application. Moreover there is no unpredictable source used for seeding the PRNG. The PRNG will most probably provide predictable outcome. Furthermore both DRBG\_CMAC and DRBG\_Hash seem to be disabled in the configuration. It is not clear, which PRNG algorithm is used in the application.

Finding\_3: In the application, the security access function is configured to read the public key from a locally defined constant array. The public key value is set as part of the application code to the constant array. The public key is not read from the protected OTP section.

# Verification of Standard Diagnostic Services

Based on the requirements TECH\_2F12 and TECH\_3122 the diagnostic service configuration for “InputOuputControlByIdentifier” and “RoutineControl” has been assessed.

The InputOuputControlByIdentifier service is only accessible in the OEM security level and only in the extended or Delta session.

The Routine Control service is only accessible in the OEM security level and only in the extended or Delta session.

No safety conditions were found to restrict the access to these services.

Warning\_1: The OEM and supplier shall assess if a safety related condition for the restriction of the InputOuputControlByIdentifier and Routine Control services are needed.

The flashbootloader is a third party SW provided by Vector, with only configuration and adaptation from the supplier side. The flow of the SW flashing and authentication process within the flashbootlaoder has been assessed based on the requirements TECH\_1012, TECH\_1026 and TECH\_1027.

The PDX container includes the SW binary and the signature over the hash of the complete application binary. The SW only consists of 1 logical block and therefor no hash tables exist.

The verification process in the Flashbootloader calculates the hash of the flashed binary and verifies the provided signature with the public key stored in the protected area of the ECU.

## Cryptographic related controls

RSA-PSS with a key length of 2048 bit is used to sign/verify the SW binary. The hash is generated with SHA-2 256 bit algorithm.

The cryptographic library provided by the Vector Autosar Stack is used in the project. No known vulnerabilities could be identified for the used primitives.

The public key is read from the protected memory up of the flashbootloader and the public key corresponds to the public key provided by the OEM.

Requirements TECH\_1031 and TECH\_1035 are applicable and covered by the implementation. The requirements TECH\_1032, TECH\_1033 and TECH\_1036 are not applicable.

## Key Protection and Use

The key protection mechanisms have been assessed based on the requirement TECH\_1043. TECH\_1045 is not applicable, as no calibration data is available in the project.

The solution to protect the public keys used in the project is

* to store the public keys in a specific memory section
* and set this section to OTP (One-Time Programmable)

The TriCore processor provides means to set specific sections of the internal flash into OTP memory and deleting and re-writing is prohibited by the HW. The configuration f the OTP setting can also not be reversed once done. This is also assured by the processor architecture.

Only section 7 of the flash are is set to OTP by a dedicated supplier diagnostic service. This diagnostic service is called during the production process.

S7 refers to a 16kB block starting at address 0xA001C000. The following keys have all been linked to the S7 memory section by the adequate memory mapping.

* Public Key for OEM Security Access
* Public Key for SW Signature Verification
* Public Key for JTAG Password Encryption

Please refer to section 3.2 for possible finding of the violation of this protection mechanism.

# JTAG Locking

The JTAG interface is locked before the device is shipped to the OEM in the production site. The diagnostic service used for the locking has been assessed.

The service

* Generates a random 256 bit JTAG password
* And encrypts the password with RSA-OAEP 2048 bit algorithm

The public key used for the encryption is stored in the protected OTP memory area. The random number is created with the PRNG implemented in the flashbootloader.

Finding\_4: The function used for generating the seed for the PRNG in the Flashbootloader seems to be initialized with a default reset state, leading to the same seed at every start up of the device.

# Summary and Status of Findings

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Explanation | Status | Comment |
| Finding\_1 | The supplier write services are adequately protected with the Security Access mechanism. They are also only accessible in the supplier session. The password based protection is still used but is not relevant, as it is superseded by the Security Access mechanism. Yet the read services are not protected. The possible security vulnerabilities and information exposed by this configuration has not been assessed within the review. | Re-Assessed | Unless the supplier debug frame is not activated, no supplier debug messages can be read.To activate the Debug frames Security Access is required hence no vulnerabilities are present. |
| Finding\_2 | The seed is not set yet and still has default values when used as a seed value for the the random number generator in the security access function in the application. Moreover there is no unpredictable source used for seeding the PRNG. The PRNG will most probably provide predictable outcome. Furthermore both DRBG\_CMAC and DRBG\_Hash seem to be disabled in the configuration. It is not clear, which PRNG algorithm is used in the application. | Re-Assessed | Fips186 SHA1 algorithm is used for random number generator (using the vector stack). The secret seed is encrypted using the vector (SHA2) algorithm. |
| Finding\_3 | In the application, the security access function is configured to read the public key from a locally defined constant array. The public key value is set as part of the application code to the constant array. The public key is not read from the protected OTP section. | Re-Assessed | With the latest SW Release, the public key is now read from the OTP section. Now the public key is directly read from the FBL, there is no interference of public key from the application. |
| Finding\_4 | The function used for generating the seed for the PRNG in the Flashbootloader seems to be initialized with a default reset state, leading to the same seed at every start up of the device. | Re-Assessed | The assessed code was changed and fixed prior to the SW release “FBL\_23\_19\_11”. The release was tagged from the SBN revision 2011 and the code was fixed in revision 2001.  With the new code, the seed for the PRNG is created freshly before every use. The Seed creation uses an internal timer XORed with the serial number of the device to create a seed.   Note : It seems only the lower byte of the timer is used together with the serial number of the device. |
| Warning\_1 | Warning\_1: The OEM and supplier shall assess if a safety related condition for the restriction of the InputOuputControlByIdentifier and Routine Control services are needed. | Assessed | Service $2F – InputOuputControlByIdentifier service is not implemented in this project because it’s not needed (and is not asked by the requirements either).  Service $31 – RoutineControl service was analysed by Renault Cybersecurity department together with Vector (back in April, when Vector was asked to provide a bug-fix to solve the problem with the Security Access not being implemented at service level) and they decided that it should not be protected by the Security Access. The service is not supported in Default Session or in Extended Session. |

# Appendix B: Relevant Requirements and Applicability

|  |  |  |  |
| --- | --- | --- | --- |
| Topic | Requirements to be verified |  | Applicability |
| *Signature application* | TECH\_1012 | The digital signature SHALL be created on a logical block basis (If application software and/or calibration, are divided into multiple logical blocks). | Y |
| TECH\_1026 | ECUs SHALL calculate the hash for each logical block of the updated application software and calibration. | Y |
| TECH\_1027 | ECUs SHALL verify signature through comparison of the expected hash (computed from signature) with the calculated hash over logical block data as per | Y |
| *Cryptographic-related controls* | TECH\_1031 | ECUs SHALL use RSA (2048 bits minimum) and SHA-2(SHA-256) for application software and calibration integrity verification. | Y |
| TECH\_1032 |  | N |
| TECH\_1033 |  | N |
| TECH\_1035 | As an alternative to [TECH\_1034] RSA-PSS SHALL be used as a robust implementation. | Y |
| TECH\_1036 |  | N |
| *Key protection and use* | TECH\_1043 | ECUs SHALL hold the public key for application software integrity verification (Public key 1) in the access-controlled area that can be read but cannot be rewritten by application/bootloader. (e.g.: Secure storage, One Time Programmable memory, HSM). | Y |
| TECH\_1045 |  | N |
| *InputOuputControlByIdentifier service access restrictions with vehicle state* | TECH\_2F12 | The ECU SHALL be able to stop inputOutputControlByIdentifier service at the timing when the state of the vehicle is changed to any other conditions than the safe condition. | Y |
| *RoutineControl service access restriction with vehicle state* | TECH\_3122 | The ECU SHALL stop RoutineControl service at the timing when the state of the vehicle is changed to any other conditions than the safe condition. | Y |
| *Protection / Removal of specific diagnostic services and resources* | TECH\_D101 | The supplier specific diagnostic services, diagnostic service parameters defined by suppliers and systemSupplierSpecificSession SHALL be deleted for production software. | N |
| TECH\_D111 | The supplier specific diagnostic services and diagnostic service parameters defined by suppliers SHALL only be allowed during systemSupplierSpecific Session or other session specified by ECU suppliers. | Y |
| TECH\_D121 | The supplier specific diagnostic services and diagnostic service parameters defined by suppliers SHALL be started after getting unlock status by OEM securityAccess 0x27 service (1st priority, Refer to chapter 10) or supplier specific SecurityAccess 0x27 service (2nd priority). | Y |
| *Specific SA implementation* | TECH\_D113 / TECH\_D122 | The contents of the supplier specific SecurityAccess 0x27 service specification, which include, but are not limited to the following, SHALL be reviewed by the security team. How to protect from dictionary attacks? - How to protect from brute force attacks? - How to protect from replay attacks? - How to protect from predictive analytics cyber- attacks? | Y |
| TECH\_D114 / TECH\_D123 | Only one SecurityAccess shall be selected in OEM SecurityAccess 0x27 service (1st priority, Refer to chapter 10) or supplier specific SecurityAccess 0x27service (2nd priority). | Y |
| *CAN communication management* | TECH\_D201 |  | N |