JSS MAHAVIDYAPEETHA

JSS SCIENCE AND TECHNOLOGY UNIVERSITY

SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING

- Constituent College of JSS Science and Technology University
- Approved by A.I.C.T.E
- Governed by the Grant-in-Aid Rules of Government of Karnataka
- Identified as lead institution for World Bank Assistance under TEQIP Schem

Report on case study analysis

"*AUTOMOTIVE SOTA UPDATE*"

Carried out as a part of Event –IV for the elective course offered by **BOSCH BGSW**

Bosch Global Software Technologies
alt_future

20EC647–AUTOMOTIVE CYBER SECURITY

Submitted by:

Group No.	Sl. No	USN	Name	Dept	Block alloted	Event II
1	1	01JST20EC054	M V Sankalp Reddy	ECE	Tester Block	
	2	01JST20EC079	Rajaneesh R	ECE		
	3	01JST20EC084	Sanath Kumar K S	ECE		
	4	01JST20EC093	Skanda M Bharadwaj	ECE		
	5	01JST20CS174	Vinayak Nagapati Bhat	CSE		
2	1	01JST20EC013	Ashwin Vijayakumar Bhandari	ECE	PKI Block	
	2	01JST20EC038	Janhavi V B	ECE		
	3	01JST20EC041	K Ganapathi Sharma	ECE		
	4	01JST20EC091	Shubhaprada S	ECE		
	5	01JST20CS016	Akshay urs M	CSE		
3	1	01JST20EC014	B P Rashmi	ECE	ECU Block	
	2	01JST20EC042	K Preksha Bhat	ECE		
	3	01JST20EC067	Pavitra C M	ECE		
	4	01JST20EC078	Radhika H R	ECE		
	5	01JST20CS148	Sharadhi N N	CSE		

Under the guidance of

EXTERNAL MENTORS	INTERNAL MENTORS	COURSE INSTRUCTOR
Sathyamoorthy Ilangovan Karthileyan	Prof. Vinay Prasad M S	Prof. Supreetha M Assistant Professor Department of ECE JSS S&TU, SJCE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
JSS SCIENCE AND TECHONOLOGY UNIVERSITY
MYSURU-570006
2022-2023

JSS MAHAVIDYAPEETHA
JSS SCIENCE AND TECHNOLOGY UNIVERSITY
SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
Constituent College of JSS Science and Technology University
Approved by AICTE
Governed by the Grant Aid Rules of Government of Karnataka
Identified as institution for World Bank Assistance under TEQIP Scheme
DIAMOND JUBILEE YEAR : 1963-2023

AUTOMOTIVE CYBER SECURITY (EC647/CS653)

Report on "*SOTA Use Case Implementation*" as part of the Event-II and Event-IV for the course offered by BOSCH BGSW

Submitted by

Group No.	Sl. No	USN	Name	Dept	Block alloted	Marks
1	1	01JST20EC054	M V Sankalp Reddy	ECE	Tester Block	
	2	01JST20EC079	Rajaneesh R	ECE		
	3	01JST20EC084	Sanath Kumar K S	ECE		
	4	01JST20EC093	Skanda M Bharadwaj	ECE		
	5	01JST20CS174	Vinayak Nagapati Bhat	CSE		

Under the guidance of

SJCE Mentor

Prof. Supreetha M,
Assistant professor,
Department of ECE,
JSS S&TU, SJCE, Mysore

BOSCH Mentors

Sathyamoorthy Ilangovon
Karthikeyen

Department of Electronics and Communication Engineering,
JSS Science and Technology University,
Mysuru-570006

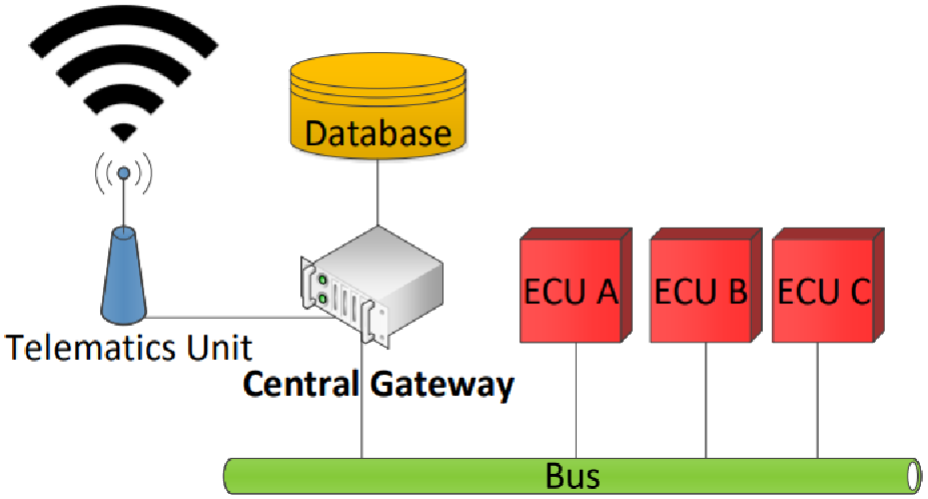
2023

1. INTRODUCTION:

The automotive industry is witnessing a rapid transformation with the increasing integration of advanced electronics and software-driven functionalities in vehicles. As a result, the ability to update and upgrade software remotely has become a critical requirement for manufacturers and service providers. Software Over-The-Air (SOTA) update technology enables them to deliver software updates, bug fixes, security patches, and new features to vehicles without the need for physical access or manual intervention.

SOTA update has emerged as a game-changing technology in the automotive domain, revolutionizing the way software is managed in vehicles post-production. It provides an efficient and cost-effective solution to address software-related issues, improve vehicle performance, enhance cybersecurity, and comply with regulatory standards. With SOTA, manufacturers can ensure that vehicles stay up to date, deliver an optimal driving experience, and adapt to evolving customer needs. The concept of Software Over-The-Air update in the automotive industry, examining its significance, benefits, and implementation considerations. We will delve into the key aspects of SOTA update, including the update process, security mechanisms, and the role of various stakeholders. Furthermore, we will discuss real-world use cases and examples of successful SOTA implementations in the automotive sector.

2. Complete Picture of SOTA Update Procedure:

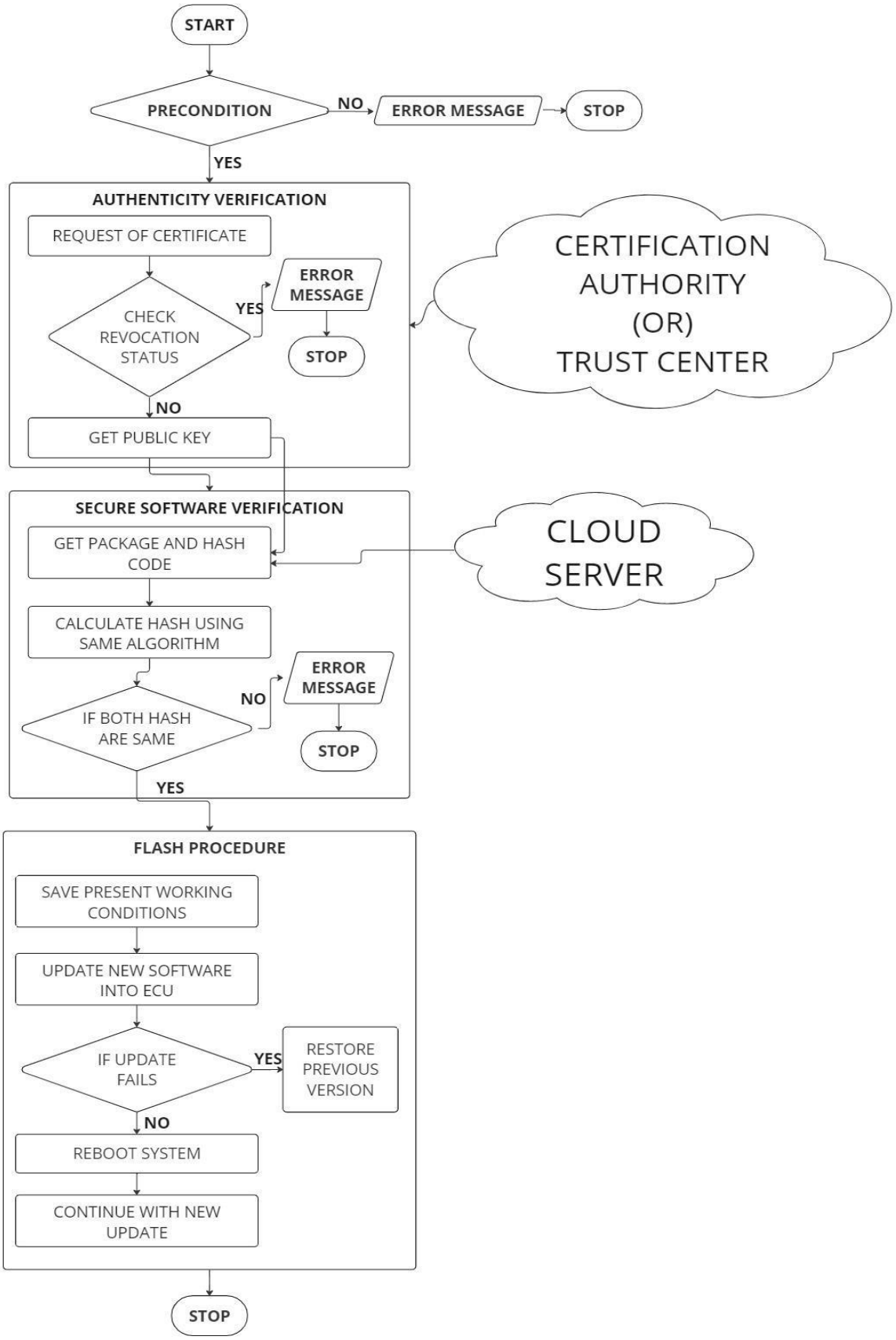


The Telematics Control Unit (TCU): The Telematics Control Unit (TCU) in automobiles handles various functions related to connectivity, communication, and data processing. The TCU provides wireless connectivity, enabling the vehicle to connect to the internet, mobile networks, and other devices. This allows for features like infotainment systems, **over-the-air software updates.**

Gateway ECU: The specific capabilities and functionalities of the Central Gateway ECU role is to act as a central control unit that manages the secure communication, distribution, and installation of OTA updates throughout the vehicle's electronic system. The Central Gateway ECU determines the routing of data and signals within the vehicle's network. It filters and forwards relevant update-related data to the appropriate ECUs, ensuring that only the necessary components receive the update files.

Target ECU: In the context of software over-the-air (SOTA) updates, the Target Electronic Control Unit (ECU) refers to the specific electronic control unit or module in a vehicle that is the intended recipient of the software update. It is the component that will receive, install, and apply the updated software.

3. SOTA FLOWCHART:



4. PRECONDITION FOR SOTA:

- The vehicle must have a reliable and stable network connection to receive the update. This can be achieved through cellular networks, Wi-Fi, or a combination of both.
- The vehicle should have a sufficiently charged battery or be connected to a power source during the update process. This ensures that there is enough power to support the update installation without the risk of the vehicle shutting down during the process.
- The vehicle's software and communication systems must have robust security measures in place to protect against unauthorized access, tampering, or interception during the update process. This includes encryption, digital signatures, authentication protocols, and secure communication channels.

5. STEPS INVOLVED IN SOTA:

Authenticity Verification: The external device provides its digital identity and certificate to the ECU as part of the secure communication establishment. The ECU verifies the authenticity of the device by validating its digital certificate, which is issued by a trusted certification authority (CA). The CA's public key is pre-installed in the ECU's trusted root store.

Secure Software Verification: The ECU and the external device exchange the necessary information to perform secure software verification. This includes the software package, cryptographic hashes (e.g., SHA-256) of the software components, and digital signatures.

Digital Signatures: The software package contains the software binaries and associated metadata. The external device signs the software package using its private key, creating a digital signature. The ECU holds the corresponding public key to verify the digital signature.

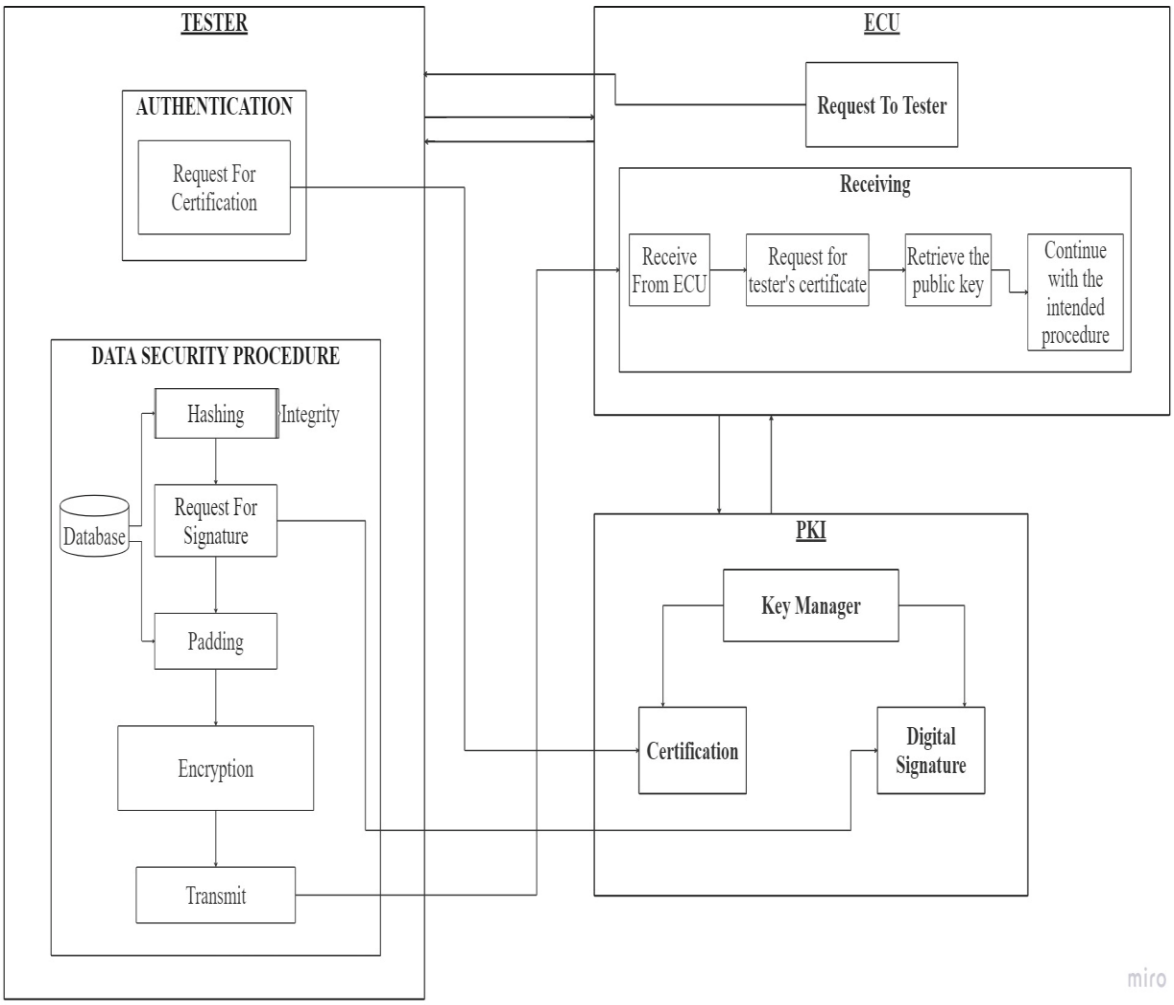
Hash Verification: The ECU calculates the cryptographic hash of the received software package independently using the same algorithm (e.g., SHA-256). It compares the calculated hash with the one provided by the external device to ensure the integrity of the software.

Digital Signature Verification: The ECU verifies the digital signature using the public key of the external device. If the signature is valid, it indicates that the software package has not been tampered with and was indeed signed by the trusted external device.

Re-flashing Procedure: Once the authenticity and secure software verification are successful, the ECU authorizes the re-flashing procedure. The ECU applies the new software to update its firmware or configuration parameters based on the validated software package.

6. DESIGN ANALYSIS OF TESTER BLOCK:

DESCRIPTION: Steps that a TESTER performs in an Automotive SOTA update.



TESTER: Testers in automotive SOTA updates play a crucial role in ensuring the quality, reliability, and safety of the software being deployed to vehicles, ultimately contributing to a positive user experience, and maintaining the integrity of the automotive ecosystem. Testers also employ secure communication channels, such as encrypted protocols like Transport Layer Security (TLS), to establish a secure connection between the cloud server and the vehicle. This ensures that the SOTA

update is transmitted in a protected manner, preventing unauthorized access or tampering.

Secure Communication Channels: Secure communication channels in SOTA updates is achieved through cryptographic protocols that provide secure communication channels between the cloud server and vehicles. It involves encrypting the data transmission to protect it from unauthorized access, ensuring data integrity through digital signatures, and implementing mutual authentication to verify the identities of both the server and the vehicle, establishing a secure and trusted connection.

7. TESTER BLOCK DIAGRAM IN DETAIL:

AUTHENTICATION: In the SOTA update process, tester typically requests a certificate from a Public Key Infrastructure (PKI) to obtain a trusted certificate for the authentication of the software. The tester submits a certificate enrolment request to the PKI, along with other necessary information, such as the entity name, domain, and contact details.

HASHING: The tester performs a process called as Hashing on the software data that is to be uploaded or flashed into the ECU of the automobile. Since HASHING is a one-way process, the hash digest cannot be converted back into the data. This is sent to the PKI. The PKI then performs an operation called as DIGITAL SIGNATURE by encrypting the hash digest with it's own private key.

There are several types of hashing algorithms commonly used. They are:

1. MD5 (Message Digest Algorithm 5)
2. SHA-1 (Secure Hash Algorithm 1)
3. SHA-256 (Secure Hash Algorithm 256-bit)
4. SHA-3 (Secure Hash Algorithm 3)

SHA is widely used and considered to be secure for most applications.

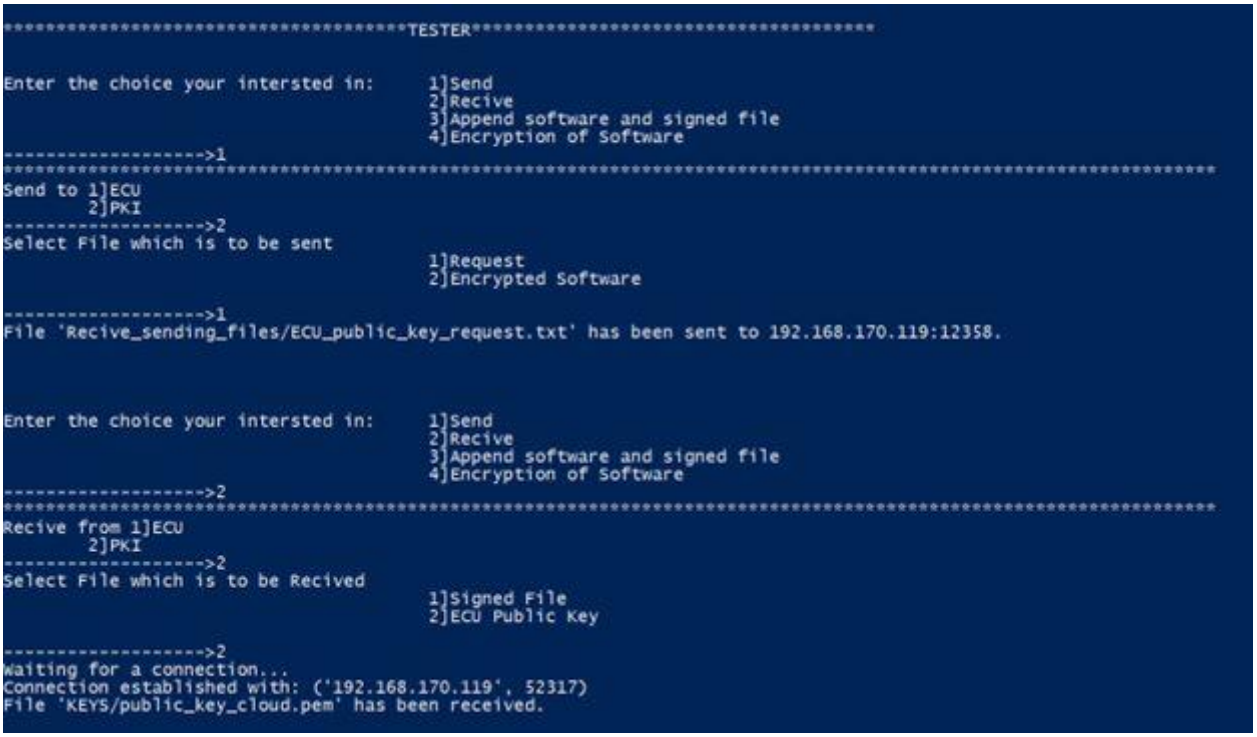
PADDING: The sotware data that is to be flashed or updated into the ECU is then padded with the signature done by the PKI.

ENCRYPTION: After padding the software data and the signature from PKI, we encrypt it. We preferably use symmetric encryption techniques so that the ECU can decrypt it with the already existing key. AES is a symmetric encryption technique which can be employed. It supports key sizes of 128, 192, and 256 bits and is considered secure and efficient.

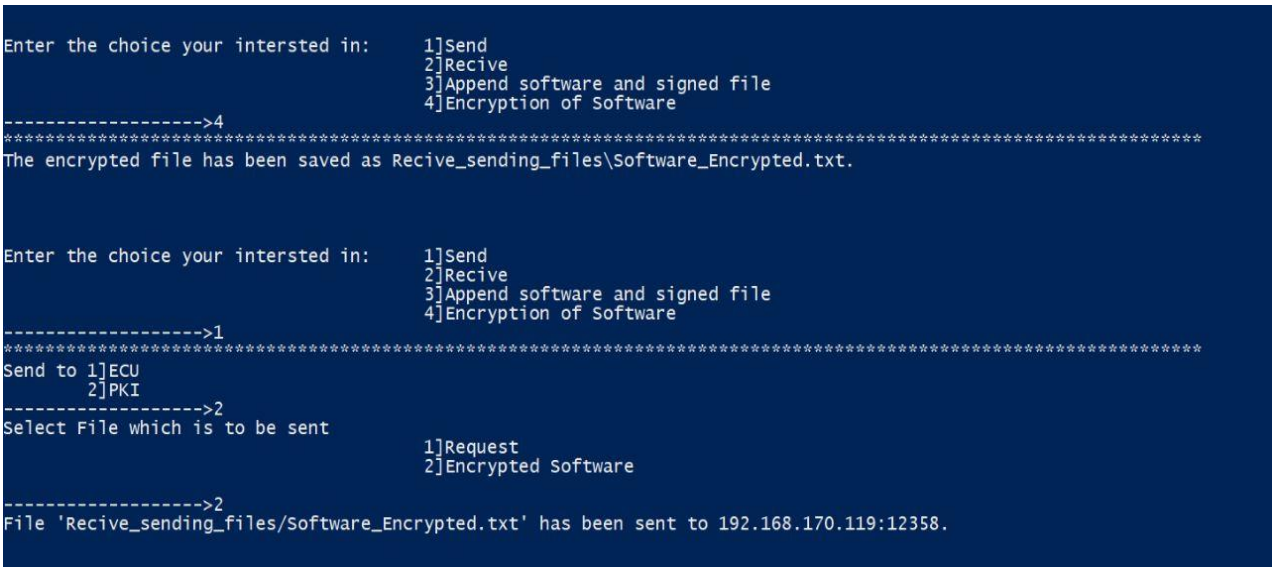
IMPLEMENTATION

8. RESULTS:

1. Requesting the public key from PKI to encrypt the file



2. Performing encryption and sending it to PKI for digital signature



3. Sending the software package to the ECU

JSS MAHAVIDYAPEETHA
JSS SCIENCE AND TECHNOLOGY UNIVERSITY
SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
- Constituent College of JSS Science and Technology University
- Approved by AICTE
- Governed by the Grant Aid Rules of Government of Karnataka
- Identified as institution for World Bank Assistance under TEQIP Scheme

DIAMOND JUBILEE YEAR : 1963-2023

AUTOMOTIVE CYBER SECURITY (EC647/CS653)

Report on "Public Key Infrastructure in *SOTA* " as part of the Event-II and Event-IV for the course offered by BOSCH BGSW

Submitted by

Group No.	Sl. No	USN	Name	Dept	Block alloted	Marks
2	1	01JST20CS016	AKSHAY URS M	CSE	PKI Block	
	2	01JST20EC013	ASHWIN VIJAYAKUMAR BHANDARI	ECE		
	3	01JST20EC038	JANHAVI V B	ECE		
	4	01JST20EC041	K GANAPATHI SHARMA	ECE		
	5	01JST20EC091	SHUBHAPRADA S	ECE		

Under the guidance of

SJCE Mentor:

Prof. Supreetha M,

Assistant professor,

Department of ECE,

BOSCH Mentors:

Sathyamoorthy Ilangovon

Karthikeyen

Department of Electronics and Communication Engineering,

JSS Science and Technology University,

Mysuru-570006

2023

Task 1

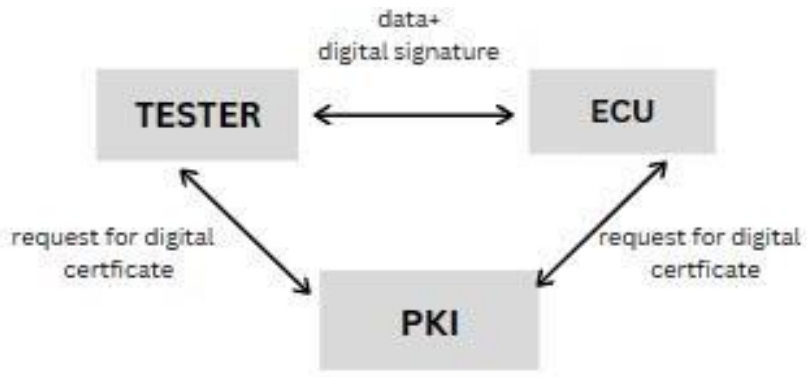
Requirement Analysis

DESCRIPTION: The Big Picture

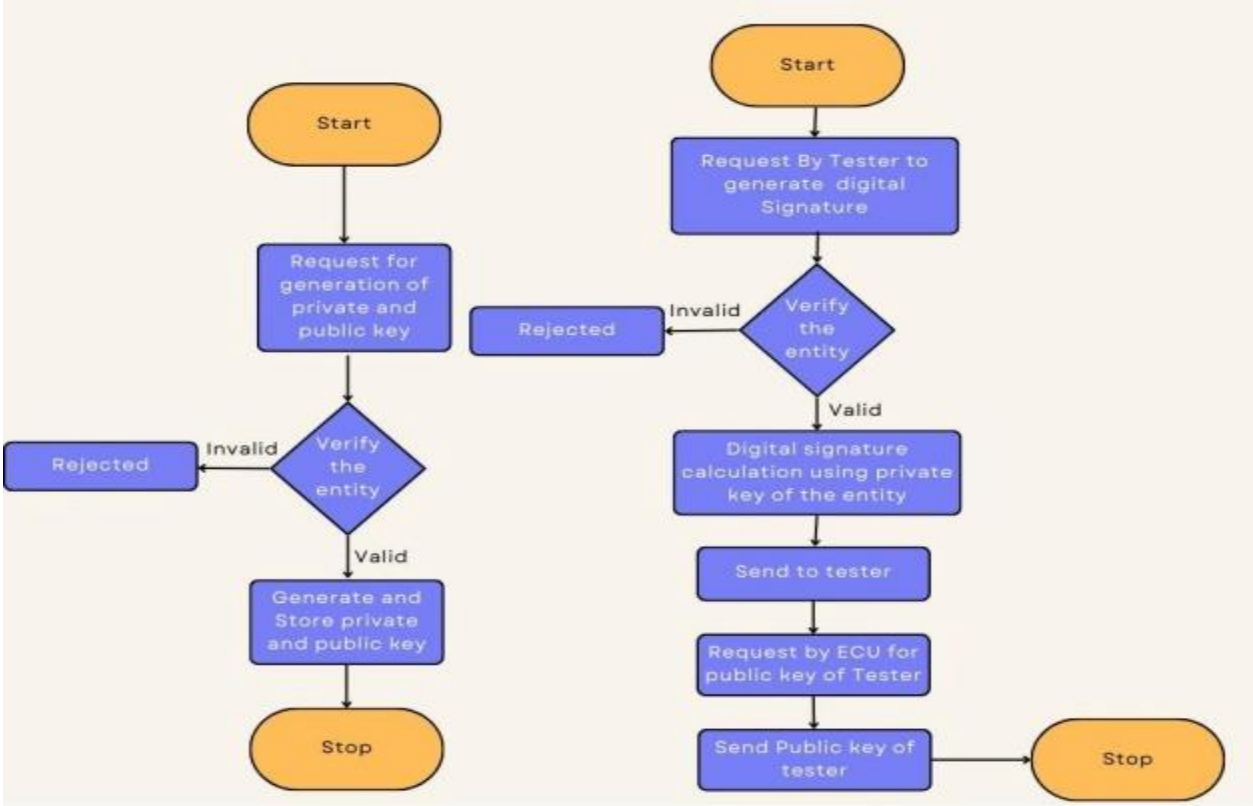
In this situation, the tester sends a request to the ECU (Electronic Control Unit). The ECU responds by sending a challenge, which is essentially a random number, back to the tester.

The tester then sends this challenge to the PKI (Public Key Infrastructure), requesting a signature. The PKI generates the signature using the tester's private key and the challenge. The resulting signature is sent back to the tester. Next, the tester sends this signature to the ECU. The ECU needs to verify the authenticity of the signature using the tester's public key. In a typical setup, the tester's public key would be stored in a database on the PKI side. The ECU would request the tester's public key from the PKI. The PKI, using its root private key, signs the tester's public key and sends it to the ECU. The ECU, having the root public key, verifies the authenticity of the tester's public key. Once the tester's public key is confirmed to be genuine, the ECU uses this public key to verify the signature provided by the tester.

However, in this specific scenario, the implementation does not follow the usual process due to the absence of necessary tools and infrastructure. Instead, the tester's public key is hardcoded in the ECU, and the ECU uses this hardcoded key for signature verification, bypassing the standard PKI-based authentication process.



WORK FLOW



FUNCTIONS OF PKI

- Public and private key generation
.
- Generation of signature

PRECONDITION

- The PKI server is up and running.
- The tester has to register itself and get its public and private key.
- The Public key has to be made available to the ADAS ECU.
- A proper communication has to be established between the tester and the PKI.

FREQUENCY OF ACTIVITY

- Public and private key generation.
- PKI should generate signature for each and every challenge provided by each tester.

INTERFACE AMONG STAKEHOLDERS

- Communication takes place between Tester and ECU.
- Tester requests PKI for the signature by sending the challenge.
- A secure communication channel is established between the tester and PKI
- using a networksecurity protocol.
- PKI returns the generated signatures to the tester.

POSTCONDITION

- The tester sends the signature generated by PKI to ECU and the authenticity ofthe tester is verified by decrypting the signature.

ABNORMAL CONDITION

- Malware attack
- Denial of service
- Replay attacks
- Key recovery and regeneration in case of loss
- Authentication of PKI

HARDWARE AND SOFTWARE REQUIREMENTS:

- PC
- Python3 IDE
- RSA python library

Task 2

Design Analysis

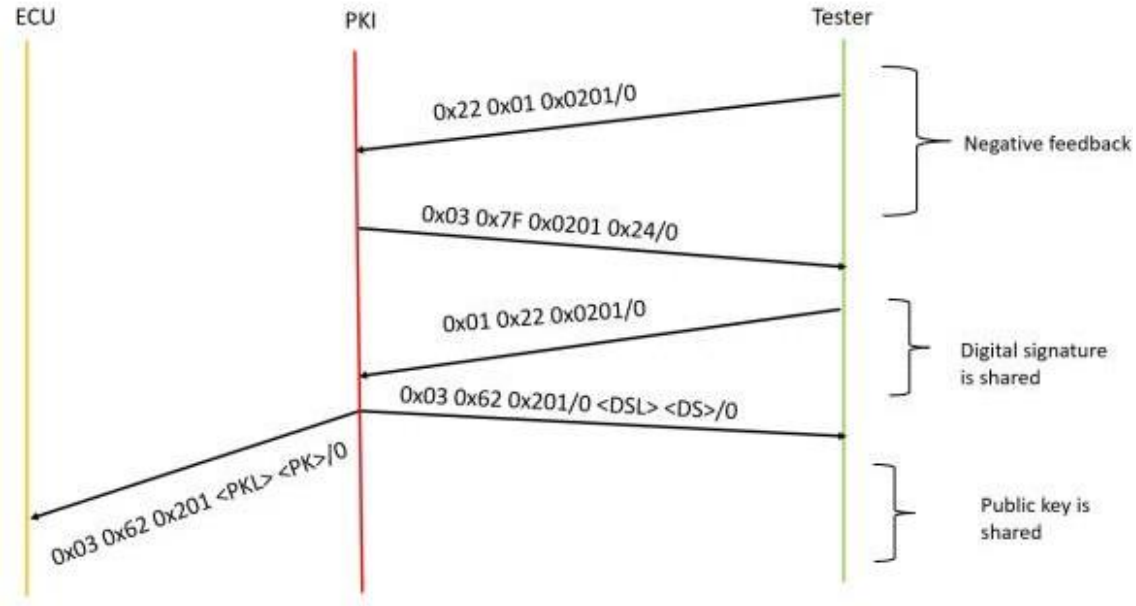
Key Generation

1. RSA Algorithm: The RSA algorithm is used for key generation in this code snippet.
2. Key Generation Process: The code generates a new set of RSA keys with a key size of 2048 bits using the rsa.newkeys() function.
3. Private Key and Public Key: The generated keys are assigned to the private_key and public_key variables respectively.
4. File Suffix Input: The user is prompted to enter a file suffix to differentiate the key files.
5. Private Key File: The private key is saved as a PKCS#1 formatted PEM file with the filename "private_key_{suffix}.pem". The private_key.save_pkcs1() method is used to obtain the PKCS#1 representation, which is then written to the file.
6. Public Key File: The public key is saved as a PKCS#1 formatted PEM file with the filename "public_key_{suffix}.pem". The public_key.save_pkcs1() method is used to obtain the PKCS#1 representation, which is then written to the file.

Digital Signature Generation

1. The code snippet demonstrates the generation of a digital signature using the cryptography library.
2. Imports: The required modules are imported from cryptography.hazmat.primitives and cryptography.hazmat.primitives.asymmetric for cryptographic operations.
3. Function Definition: The function calculate_signature is defined to generate a digital signature.
4. Function Arguments: The function takes three arguments: file_path (path of the file to be signed), private_key_path (path to the private key file), and signature_path (path to save the generated signature).
5. Private Key Loading: The private key is loaded from the specified private key file using the serialization.load_pem_private_key() method.
6. File Data Reading: The content of the file to be signed is read from the file specified by file_path.
7. Signature Generation: The digital signature is generated using the private key, file data, and the PSS padding scheme with SHA256 hashing.
8. Signature Saving: The generated signature is saved to the specified file path using the write() method.
9. Confirmation Message: A confirmation message is printed, indicating that the signature has been saved to the specified path.

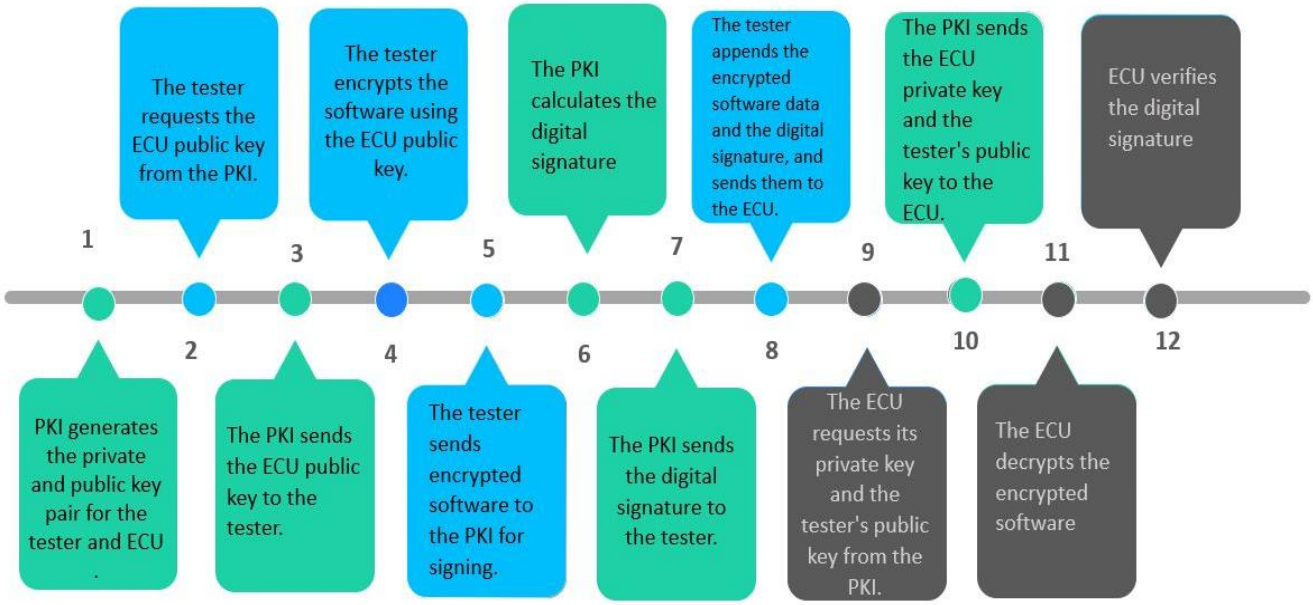
Communication



- The command 0x220x010x0201/0 has a sequence error, resulting in the PKI responding with a negative acknowledgement 0x030x7F0x02010x24/0.
- The tester then requests the digital signature in the correct format, specifically 0x01 0x22 0x0201/0. Subsequently, the PKI generates the digital signature and sends it to the ECU.
- The digital signature is represented by 0x03 0x62 0x201/0 <DSL> <DS>/0, where DSL denotes the length of the digital signature and DS represents the actual digital signature.
- Once the digital signature is transmitted, the PKI sends the public key to the ECU for signature verification.
- The public key format is 0x03 0x62 0x201 <PKL> <PK>/0, with PKL indicating the length of the public key and PK representing the public key data

TASK 3
IMPLEMENTATION

FLOW OF EXECUTION



Results:

1. Output of Key Generation of tester and ECU

```
Enter the choice your intersted in:     1]Send
                                        2]Recive
                                        3]generate Keys
                                        4]Decrypt Software to sign
                                        5]sign the Software
------------------->3
**************************************************************************************************************************************************************

ECU and Tester Key pairs are Generated
```

2. Generated Key pairs and file structure

3. Output of Received Encrypted file and signature calculation.

```
File 'PKI_recived/Software_Encrypted.txt' has been received.



Enter the choice your intersted in:     1]Send
                                        2]Recive
                                        3]generate Keys
                                        4]Decrypt Software to sign
                                        5]sign the Software
------------------>4
**********************************************************************************************************************************

The decrypted file has been saved as PKI_recived\Software_Encrypted.txt.decrypted
.



Enter the choice your intersted in:     1]Send
                                        2]Recive
                                        3]generate Keys
                                        4]Decrypt Software to sign
                                        5]sign the Software
------------------>5
**********************************************************************************************************************************
Signature saved to PKI_recived\signature.txt
```

4. Output of sending ECU private key and Tester Public key

```
Enter the choice your intersted in:     1]Send
                                        2]Recive
                                        3]generate Keys
                                        4]Decrypt Software to sign
                                        5]sign the Software
------------------>1
**********************************************************************************************************************************
Send to 1]ECU
        2]Tester
------------------>1
Send to 1]ECU Private key
        2]Tester public key
------------------>1

File 'KEYS\private_key_car.pem' has been sent to 192.168.170.119:12358
.



Enter the choice your intersted in:     1]Send
                                        2]Recive
                                        3]generate Keys
                                        4]Decrypt Software to sign
                                        5]sign the Software
------------------>1
**********************************************************************************************************************************
Send to 1]ECU
        2]Tester
------------------>1
Send to 1]ECU Private key
        2]Tester public key
------------------>2

File 'KEYS/public_key_cloud.pem' has been sent to 192.168.170.119:12358
```

JSS MAHAVIDYAPEETHA
JSS SCIENCE AND TECHNOLOGY UNIVERSITY
SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
- Constituent College of JSS Science and Technology University
- Approved by A.I.C.T.E
- Governed by the Grant-in-Aid Rules of Government of Karnataka
- Identified as lead institution for World Bank Assistance under TEQIP Scheme

Report on case study analysis

"*Electronic Control Unit in SOTA*"

Carried out as a part of Event –II for the elective course offered by **BOSCH BGSW**

Bosch Global Software Technologies
alt_future

20EC647–AUTOMOTIVE CYBER SECURITY

Submitted by:

Sl. No.	USN	NAME	DEPT	EVENT - II
1.	01JST20EC014	B P Rashmi	ECE	
2.	01JST20EC042	K Preksha Bhat	ECE	
3.	01JST20EC067	Pavitra C M	ECE	
4.	01JST20EC078	Radhika H R	ECE	
5	01JST20CS148	Sharadhi N N	CSE	

Under the guidance of

EXTERNAL MENTORS	INTERNAL MENTORS	COURSE INSTRUCTOR
Satyamoorthy Ilangovan Karthikeyan	Prof. Vinay Prasad M S Assistant Professor Department of ECE JSS S&TU, SJCE	Prof. Supreetha M Assistant Professor Department of ECE JSS S&TU, SJCE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
JSS SCIENCE AND TECHONOLOGY UNIVERSITY
MYSURU-570006

2022-2023

Task 1
Requirement Analysis

1. INTRODUCTION:

Software over the air (SOTA) refers to the process of wirelessly updating software on a device or a fleet of devices. It allows manufacturers and developers to deliver updates, patches, and new features to their products without requiring physical access or manual intervention.

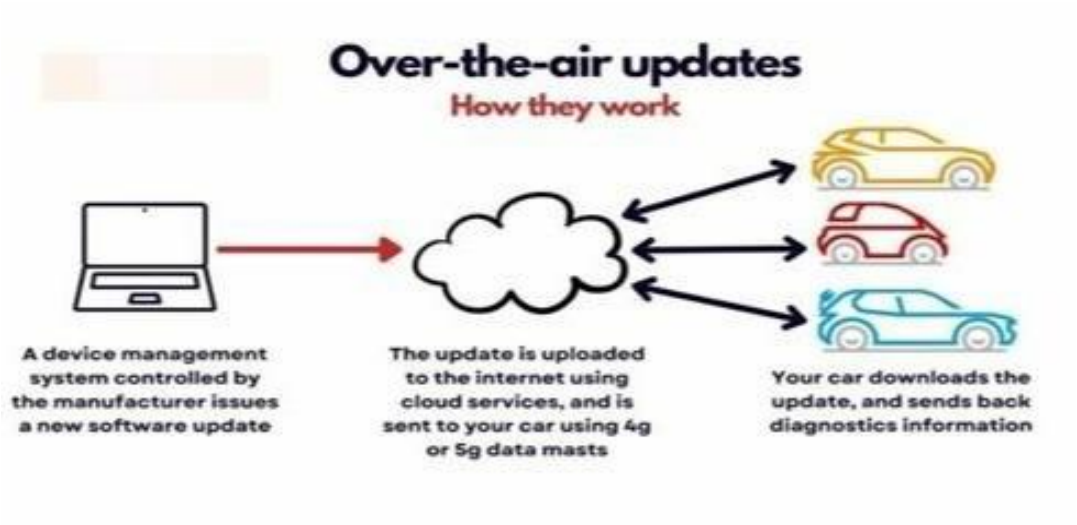


Fig.1:SOTA update flow diagram.

Automotive Cloud Server: This component resides in the cloud and acts as a central hub for managing SOTA updates. It stores and distributes software updates to vehicles.

SOTA Management Server: It is responsible for coordinating and managing the SOTA process. It receives software updates from the Automotive Cloud Server and communicates with the Telematics Control Unit (TCU) in the vehicle.

Vehicle Backend Infrastructure: It provides the necessary infrastructure for the vehicle's backend systems to communicate with the SOTA Management Server. It includes services such as authentication, data routing, and security.

Telematics Control Unit (TCU): The TCU serves as a gateway between the vehicle and the external servers. It receives software updates from the SOTA Management Server and facilitates the distribution to the appropriate Electronic Control Units (ECUs) within the vehicle.

Vehicle ECUs: These are the individual Electronic Control Units within the vehicle responsible for controlling various systems.

2. SOTA: FLOW DIAGRAM

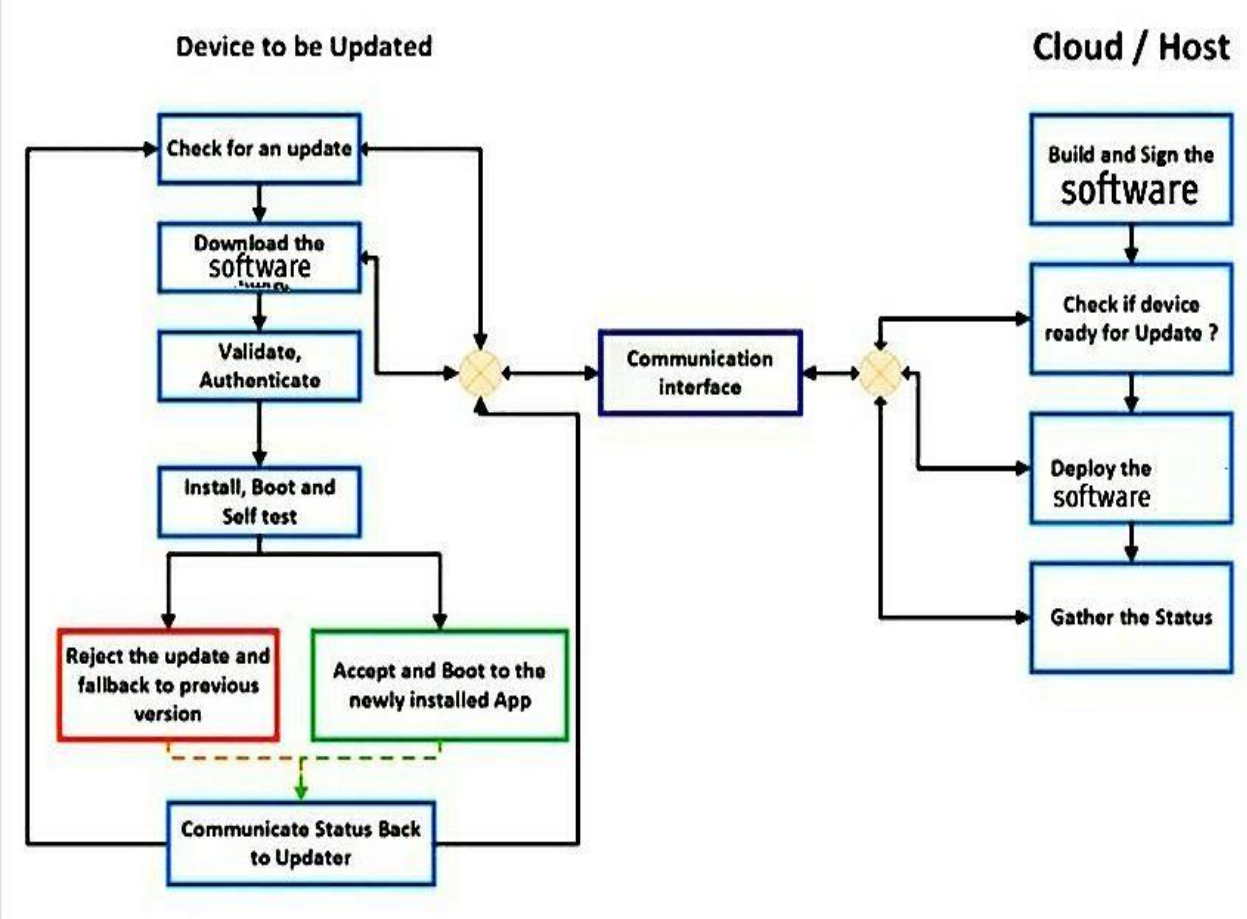


Fig.2:work flow of SOTA.

3. Holistic view

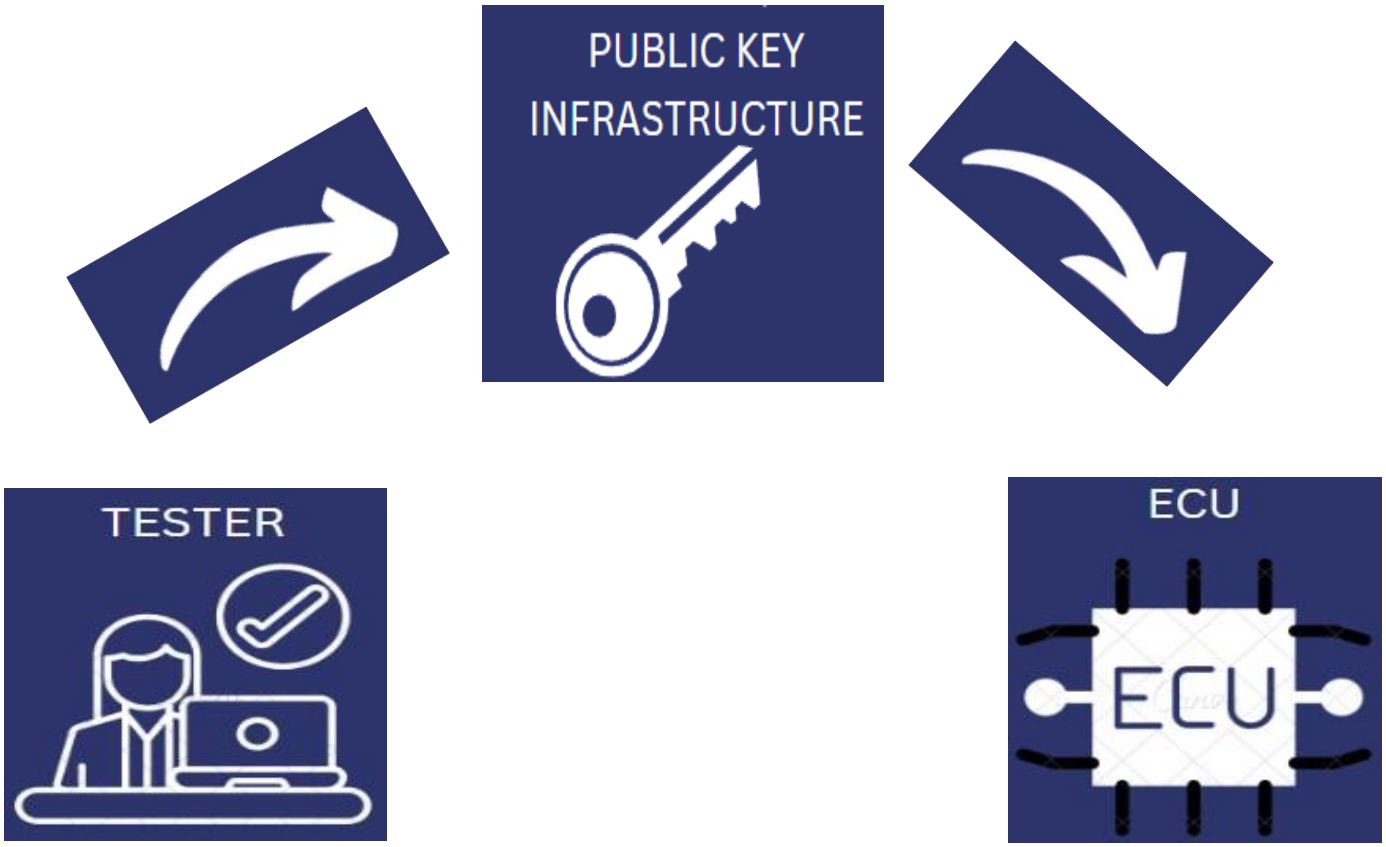


Fig.3:Interconnection among components of SOTA.

1. Testers in automotive SOTA updates play a crucial role in ensuring the quality, reliability, and safety of the software being deployed to vehicles, ultimately contributing to a positive user experience, and maintaining the integrity of the automotive ecosystem. Testers also employ secure communication channels, such as encrypted protocols like Transport Layer Security (TLS), to establish a secure connection between the cloud server and the vehicle. This ensures that the SOTA update is transmitted in a protected manner, preventing unauthorized access or tampering.
2. The tester then sends this challenge to the PKI (Public Key Infrastructure), requesting a signature. The PKI generates the signature using the tester's private key and the challenge. The resulting signature is sent back to the tester. Next, the tester sends this signature to the ECU. The ECU needs to verify the authenticity of the signature using the tester's public key.

In a typical setup, the tester's public key would be stored in a database on the PKI side.

3. The ECU would request the tester's public key from the PKI. The PKI, using its root private key, signs the tester's public key and sends it to the ECU. The ECU, having the root public key, verifies the authenticity of the tester's public key. Once the tester's public key is confirmed to be genuine, the ECU uses this public key to verify the signature provided by the tester.

Task 2

Design Analysis

4. Description: How the ECU is relevant in the context of SOTA updates

1. SOFTWARE UPDATES:

   The ECU contains software or firmware that controls the engine's operation. SOTA allows manufacturers to remotely update and modify this software without physically accessing the vehicle. The updates can optimize engine performance, improve fuel efficiency,address known issues, or introduce new functionalities to the ECU.

2. PERFORMANCE ENHANCEMENT:

   SOTA updates can be used to enhance the performance of the ECU. Manufacturers may fine-tune the software parameters to improve power output, torque, throttle response, or overall drivability. These updates can help achieve better performance characteristics without the need for physical modifications or component changes.

3. EMISSION COMPLIANCE

   ECU software updates through SOTA can also address emissions-related issues. Manufacturers may release updates to ensure compliance with evolving emission standards or to rectify software bugs that could affect emission levels. These updates help maintain the vehicle's compliance with regulatory requirements.

4. SAFETY AND SECURITY

   SOTA updates for the ECU can also focus on enhancing safety and security features. Updates may include improvements to systems like traction control, stability control, or

anti-lock braking, as well as security patches to address potential vulnerabilities in the ECU's software.

5. Digital Signature Verification

Digital signature verification plays a crucial role in ensuring the integrity and authenticity of software updates during the Software over the Air (SOTA) process. It helps confirm that the received software update has not been tampered with and that it originates from a trusted source.

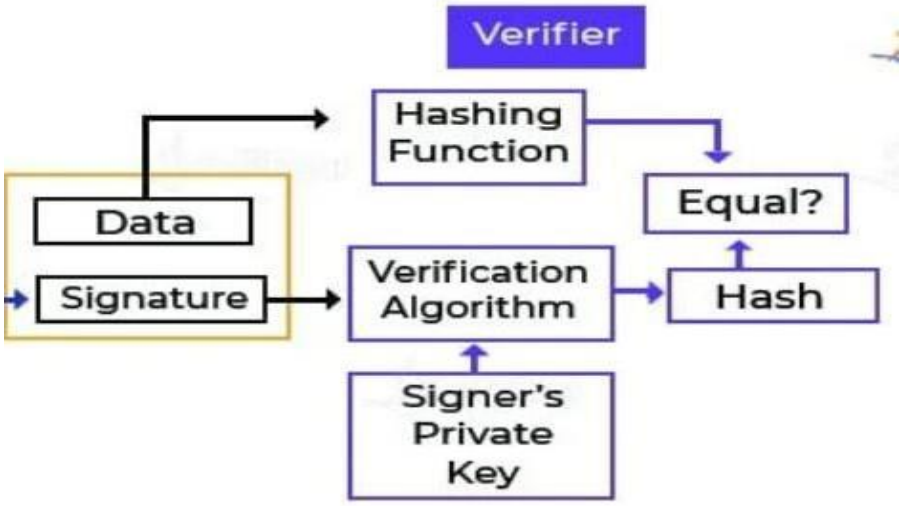


Fig.4:Flow diagram signatutre verification.

Here's how digital signature verification is typically implemented in SOTA:

1. Digital Signature Verification: Upon receiving the software update, the target device performs digital signature verification. It uses the provider's public key, which is already available or can be obtained from a trusted source, to verify the authenticity and integrity of the update.
2. Verification Process: The target device applies the same cryptographic algorithm used by the provider to generate a signature of the received software update. It then compares the generated signature with the attached digital signature. If they match, it signifies that the

update has not been modified during transit and that it was indeed signed by the trusted provider.

3. Trust in Public Key: To ensure the authenticity of the public key used for verification, it is crucial to establish trust in the key's origin. This can involve mechanisms such as certificate authorities, certificate chains, or trusted third parties that vouch for the authenticity of the public key.

6. ECU Flashing

ECU flashing is a process of modifying the software or firmware in a vehicle's Engine Control Unitech flashing, also known as ECU tuning or ECU remapping, refers to the process of modifying the software or firmware of an engine control unit (ECU) in a vehicle. The ECU is a crucial component that manages and controls various aspects of the engine's operation, including fuel injection, ignition timing, and other parameters.

ECU flashing is typically performed to optimize the performance, efficiency, or specific characteristics of the engine. It involves rewriting or modifying the software that controls the ECU's behavior, allowing for adjustments to parameters such as fuel mapping, boost pressure, rev limits, and more.

Some key aspects of ECU flashing:

1. Read Original ECU Data:

- Connect the vehicle to a computer or diagnostic tool that can communicate with the ECU.
- Extract the original software or firmware data from the ECU, creating a backup.

2. Modify ECU Data:

- Analyze and modify the extracted ECU data using specialized software.
- Adjust parameters such as fuel delivery, ignition timing, turbo boost, and throttle response to optimize performance.
- Some modifications may involve tweaking the air-fuel ratio, adjusting timing curves, or modifying limiters.
- Upload Modified ECU Data:

3. Prepare the modified ECU data for uploading back to the ECU.

- Connect the computer or diagnostic tool to the ECU.
- Transfer the modified ECU data to the ECU, replacing the original software or firmware.

4. Verify and Test:

- After uploading the modified data, verify that the changes were successfully written to the ECU.
- Perform diagnostic checks and tests to ensure the engine is functioning properly.
- Conduct road tests to evaluate the performance and drivability of the vehicle.

4. Fine-tuning and Iteration:

Fine-tune the modified ECU data based on the results of the initial tests.

Repeat the uploading, verification, and testing process if further modifications are desired.

Iteratively adjust the ECU data to achieve the desired performance and efficiency.

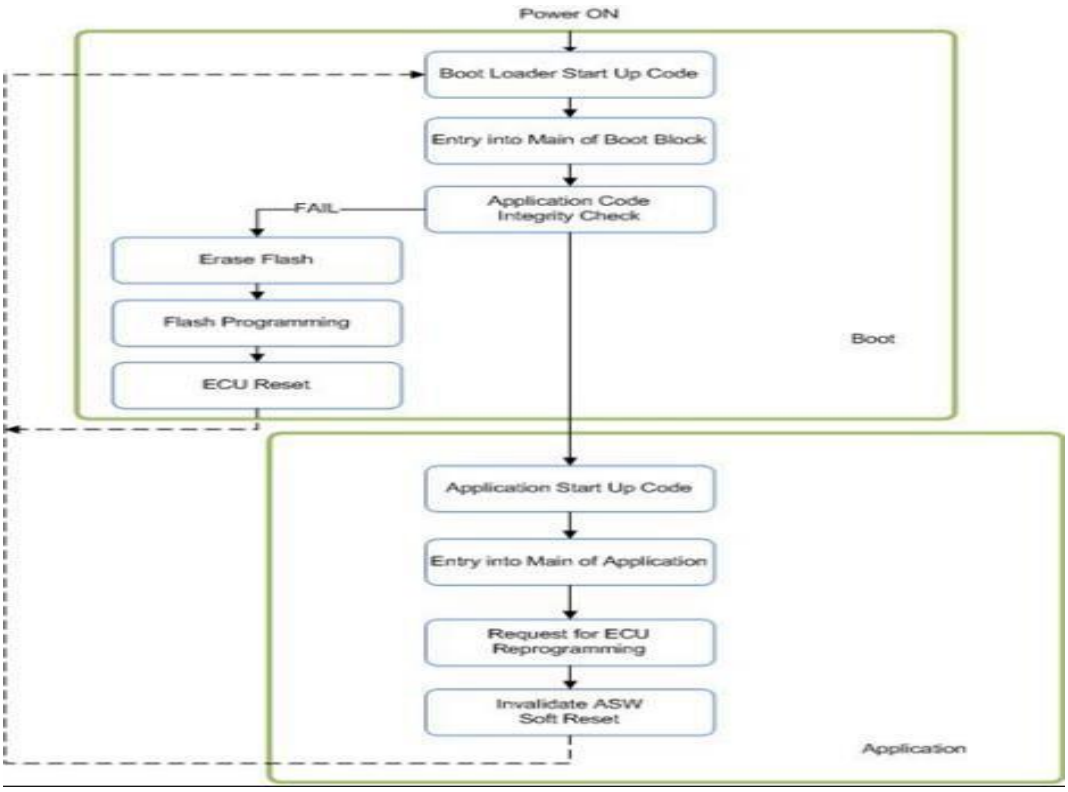
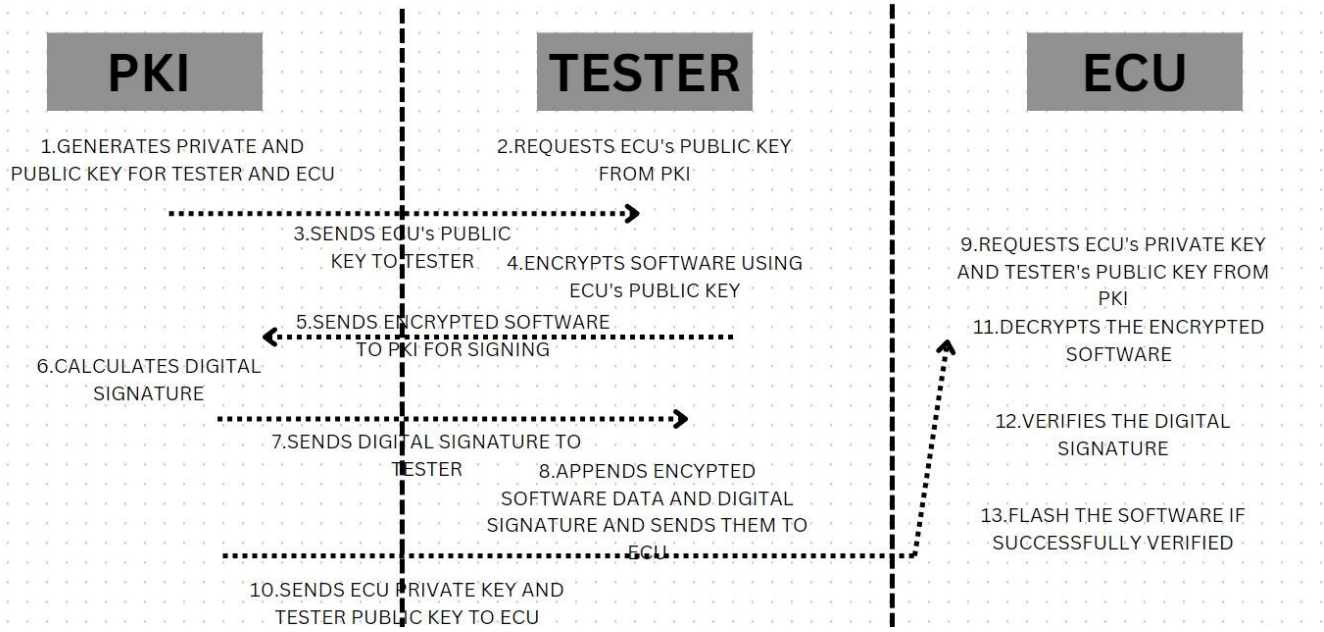


Fig.5.ECU flashing flow diagram

TASK 3
IMPLEMENTATION

FLOW OF EXECUTION



Results:

1. Requesting for encrypted software from tester

```
**********************************************ECU**********************************************

Enter the choice your intersted in:     1]Send
                                        2]Recive
                                        4]Decryption of software
                                        5]Verification of digital signature
------------------->2
****************************************************************************************************
Recive from 1]Tester

        2]PKI
------------------->1
Select File which is to be Recived
                                        1]Receive encrypted software which is appended with signature
------------------->1
Waiting for connections...
Connection established with: ('192.168.170.119', 53372)
Connection established with: ('192.168.170.119', 53373)
File 'Received_sending_files\Software_Encrypted.txt' has been received from connection 1.
File 'Received_sending_files\signature.txt' has been received from connection 2.
All files have been received.
```

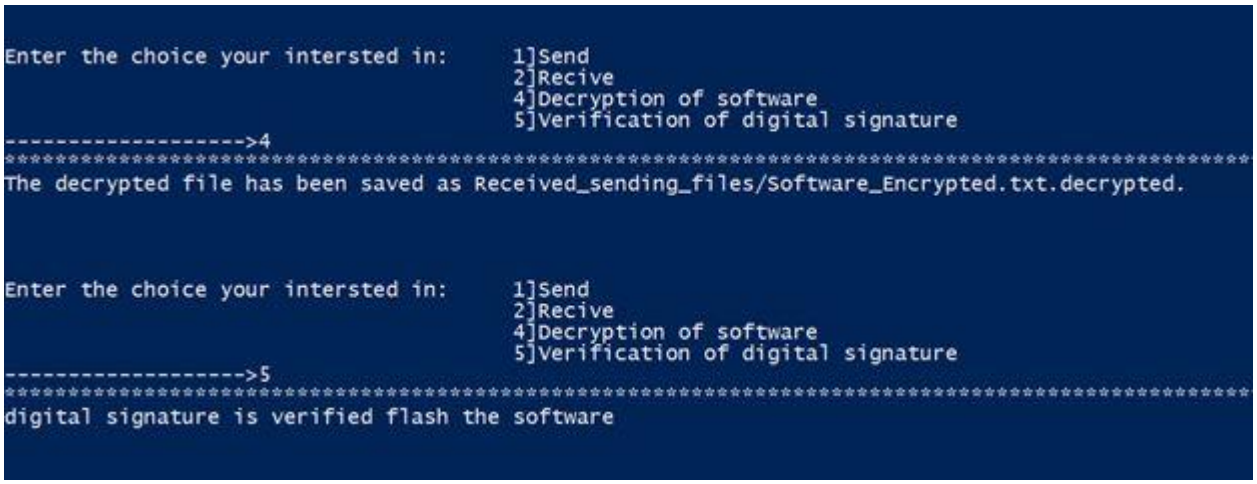
2.Requesting for ecu private key and tester public key from PKI

```
Enter the choice your intersted in:     1]Send
                                        2]Recive
                                        4]Decryption of software
                                        5]Verification of digital signature
------------------->2
****************************************************************************************************
Recive from 1]Tester

        2]PKI
------------------->2
Select File which is to be Recived
                                        1] ECU Private Key

                                        2] tester Public Key
------------------->1
Waiting for a connection...
Connection established with: ('192.168.170.119', 53401)
File 'KEYS\private_key_car.pem' has been received.
```

3. Decryption and verification of received software with decryption using ecu private key and verifying the hash value



4.Use case where software is manipulated