National College of Ireland

Integrating Threat Modelling and Mitigation Strategies for Securing Over-The-Air (OTA) Updates for Autonomous Vehicles.

MSc Research Project
Master of Science in Cybersecurity

Varun Dilip Bhiwapurkar
Student ID: 23412216

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Varun Dilip Bhiwapurkar		
Student ID:	23412216		
Programme:	MSc Cybersecurity	**Year:**	2025
Module:	MSc Research Project		
Supervisor:	Michael Prior		
Submission Due Date:	11/12/2025		
Project Title:	Project Report		
Word Count:	6623	**Page Count:**	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Varun Dilip Bhiwapurkar……

Date: ……10/12/2025……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Integrating Threat Modelling and Mitigation Strategies for Securing Over-The-Air (OTA) Updates for Autonomous Vehicles.

Varun Dilip Bhiwapurkar
23412216

Abstract

The Over-the-Air (OTA) updates have become a vital component in automobile industry for maintaining the security, reliability and performance of the vehicle, still there are significant security risks which can compromise functionality of the vehicle. In this thesis we will look into how integration of threat modelling and mitigation strategies can secure the OTA update in autonomous vehicles. Using STRIDE methodology as the primary framework, the study finds critical threats at each stage of OTA process, this includes the firmware distribution, communication channels and the in-vehicle execution process. A secure OTA system architecture is designed and also implemented with cryptographic signatures, integrity validation, secure communication and also a fail-safe mechanism for updates. The evaluation of implementation was done through different scenario-based testing and analysing resistance towards attacks, showing improvement in confidentiality, integrity and availability. The findings emphasized that it is important to have combination of methodical threat modelling and practical security control, this offers both practical guidance and academic knowledge to the industry practitioners which are devoted in enhancing the robust software update mechanism of autonomous vehicles.

Introduction:

The booming growth of connected devices and embedded systems, has led to increased demand of systems that need to have mechanisms that provide safe, efficient and reliable delivery of software updates. As vehicles are more depended on complex software components, Over-the-Air (OTA) updates has now become a necessary condition, from IoT appliance consumers to automotive vehicle ECUs, the ability to implement a remote update to a device is necessary these days. OTA systems helps manufacturers to patch security vulnerabilities and add new features and sustain long-term device operation without the need to access any hardware. OTA update is flexible and cost-effective compared to traditional manual updates using hardware, the connectivity associated with the OTA-based method makes automotive vehicle a target to a large number of cybersecurity risks.

Nevertheless, with the rise of OTA update systems, the attack surface also becomes quite high. The autonomous vehicles work in areas where software has a direct effect on safety-critical decisions. A breach of the OTA process may thus have disastrous effects, and the attackers might be able to control the actions of vehicles, shut down communication

protocols, or other essential tasks involved in driving. Practical examples of cyberattacks, such as the most famous Jeep Cherokee remote exploitation, prove that insufficient security measures in the connected vehicles may result in complete remote control. Such risk is further reinforced when AVs turn fully autonomous and entirely depend on software-based decision-making. With the continuously growing size and complexity of cyber-physical systems, the need to design OTA update infrastructures with a high emphasis on security, integrity, and resilience has never been greater. The figure (fig.1) below tells us about the various type of impacts across over 400 automotive-related cyber incidents in the year 2024. 60% of these incidents have suffered Data and privacy breaches accounting the most impact, followed by impact in services which caused business disruptions and system manipulation.

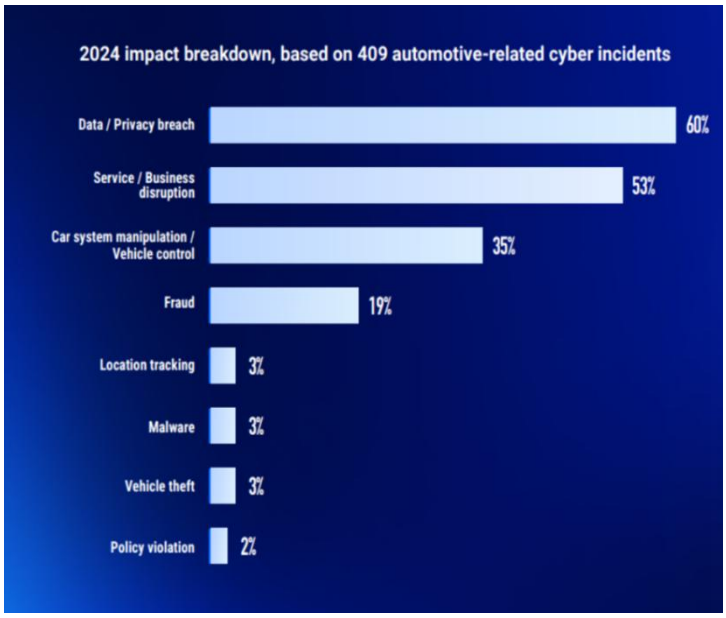


Figure 1 : https://www.forbes.com/sites/edgarsten/2025/03/13/cyberattacks-against-auto-industry-rise-becoming-more-costly/

According to the (fig.2)2022 Upstream Security Global Automotive Cybersecurity Report statistics, the below pie chart showcases most popular attack vectors used by hackers. Data servers and keyless entry systems account for the majority of incidents, highlighting serious vulnerabilities.

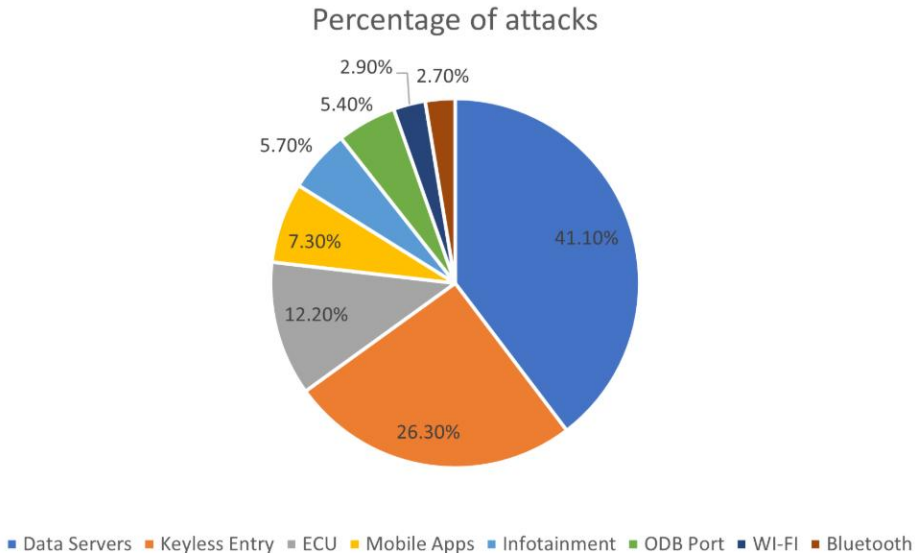


Figure 2: https://www.intertek.com/blog/2023/01-05-automotive-cybersecurity/

In this thesis we will be focuses on integrating threat modelling methodologies and the mitigation strategies for a secure OTA update architecture tailored for autonomous vehicles. Threat modelling provides a methodical measure of finding the attack vectors, adversary abilities, possible vulnerabilities and weaknesses in the update workflow. It can be utilized together with other mitigation strategies, like cryptographic verification, metadata validation, secure boot, trusted execution and the redundant update strategies, to develop a strong OTA framework which can be used to protect autonomous systems from both known and upcoming cyber threats.

The emphasis would be on designing a secure OTA Update System by integrating threat modelling and implementing it by using modern web stack composed of a Node.js backend, and a dashboard developed using React, and a secure update verification system using metadata JSON files and cryptographic signatures. The aim is to develop a working system, which will demonstrate the entire OTA process, starting with the firmware uploaded to the server, all the way up to verifying and installing update on the client side. The prototype shows secure signature of update, encrypted transmission, and signature verification. Lastly, the implementation is checked against the completeness, effectiveness, and applicability metrics to determine the strength of the proposed framework. The architectural, security and operational real problems regarding OTA update in vehicles systems are also investigated in this project and a feasible solution are suggested that can be applicable in the actual embedded systems. The study in this research aims to blend the concept of cybersecurity and the implementation of automotive OTA system.

Related Work:

Literature Review:

The rapid growth of autonomous and connected vehicles has increased the significance of having safe, robust, and verifiable OTA update systems. Researchers have spent lot of time studying the technical capability and the problems of cybersecurity with respect to the future of remote software updates, as they have been noticing growing similarity between modern vehicles and distributed cyber-physical platforms. One of the most detailed studies of OTA updates was delivered by Mahmood et al. and that study has conducted model based security testing and threat assessment of Uptane framework. In that study Mahmood et al. indicated that although OTA has a positive impact by drastically improving the efficiency of software deployment, its distributed architecture exposes a big attack surface and large number of unaddressed vulnerabilities, some of which include metadata tampering, eavesdropping and denial-of-service attacks. His study highlighted a critical flaw in the current literature. Many other studies gave suggestions on how to protect OTA updates, and only few are able to perform real-world OTA implementation test and compare it against the threat models. Mahmood et al. also showed that the present security in OTA is heavily based on many assumptions regarding the OEM infrastructure and trusted repositories, which create loopholes that can be exploited by adversaries in long-range wireless attack vectors. [1]

To support and compliment above view, Wang et al. suggested a OTA system which was based on PKI and was using hybrid RSA/AES encryption for additional and more improved confidentiality, authenticity and integrity of the OTA update packages, this showed good results by improving the update time by high bandwidth automotive ethernet. Their study tells us that how the cryptographic strengthening reduces the chances of manipulating the update artefacts, still the approach considers the integrity of both certificate authority and key-management systems. Due to this dependency on centralised trust models, a single point of failure is developed and is seen in few secure OTA proposals. Even though the authors check for signature tampering and integrity attacks, the solution fails to consider the presence of adversaries which are capable to breach backend servers, steal update signing keys, or launch coordinate attack. [2]

Aurangzeb et al did a study outside the OTA domain directly, but was relevant and concerning towards the security of autonomous vehicles, he explored the malware threats that were targeting cyber-physical systems. He noticed that the that traditional malware detection methods would often fail as the in-vehicle architectures had limited computing environment. Their hybrid static-dynamic detection approach underscores a big issue that affects the OTA systems. After the malicious code is properly deployed via a compromised channel, the limitation of resources of AV subsystem will prevent detection and containment. Therefore, the security of OTA update cannot be different and malware context, when malicious code is effectively deployed via compromised update channel, resource capacity of AV subsystems will not be sufficient to effectively detect and contain the threat. [3]

Chowdhury and Hasan presented a more OTA focused threat modelling research, they applied CIAA and STRIDE approaches to find vulnerabilities in different OTA-related components, like the cloud platform, communication channels, vehicular sensors, ECU/TCU modules and peer-to-peer update sharing models. They highlighted that attackers can take advantage of the weak authenticated update streams, unprotected rollback mechanisms, and third-party cloud dependencies. The contribution of Chowdhury and Hasan made the argument of holistic threat modelling stronger and however, it also made it theoretical as it provides the identification of threats but it does not simulate or test mitigation measures. It thus offers a valuable theoretical background but does not answer how countermeasures can be practically feasible and scaled. [4]

Li et al. in his study provide a broader systems-level sight of OTA by carrying out a detailed survey on OTA upgrade mechanisms of intelligent connected vehicles, he also classified the OTA processes into cloud, terminal and object-level updates. They mainly showed that regardless of the improvements in the SOTA and FOTA, the OTA pipelines were still vulnerable to malware attacks, insecure verification procedure and the exploitation of OBD, USB and Bluetooth ports. Their survey is an indication of the importance of a high fault-tolerance and fail-safe rollback method, many of the current OTA systems are unable to balance speed of updates, system availability and security assurances. Nevertheless, the scope of their analysis is extensive but only studies threat modelling on a macro level and it is not enough to facilitate mitigation oriented engineering on detail. [5]

An opposing paradigm was explored by Hamzah in blockchain powered OTA systems and he assessed that decentralized ledgers can be used as the solution to the problem of single point of failure in update validation processes. The Blockchain designs are suppose to have immutability and distributed trust, still the paper suggests that they have serious issues such as vulnerability to 51% attacks, misuse of consensus, Sybil nodes and vulnerabilities of smart-contracts. Even though decentralisation helps in auditability and transparency, the blockchain integration's computing and networking cost is an issue to real-time AV operations. Besides, blockchain-based solutions sometimes may ignore the real threat of physical and vehicular communication which is emphasised in other publications. [6]

In the research by Fowler, Mocnik, and Maple, they applied an asset-centric method by making an in-depth OTA threat model on foundation of real-world automobile attack vectors. It stated that the OTA systems creates room for unique threats, like exploitation of telematics modules, a compromise of gateways and cross-domain propagation. They discussed that OTA threat modelling should consider complex dependencies within the automotive supply-chain because the system may suffer severe damage or get compromise by the vulnerability of ECU firmware or by the components of the supplier. Their results support the importance of organised TARA techniques which are consistent with the ISO/SAE 21434 standards and similar standards, still their framework remains of a high level and stops short of incorporating the domain-based mitigation approaches specific to autonomous vehicles. [7]

The literature covers much on the identification of risks in such autonomous systems. In automobile security, STRIDE is the most commonly mentioned methodology, it stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege. Even though there are threat-based approaches such as PASTA (Process for Attack Simulation and Threat Analysis) but for this literature STRIDE would be the best, because STRIDE is developer-centric which is an outstanding approach to discover the technical defects in the software elements and on the other hand PASTA is risk-based it concentrates more on the business consequences. By studying the articles by Zaina et al. and others we can say that the use of STRIDE in automotive systems is successful. It is demonstrated that it is possible to recognise the vulnerabilities in a systematic way after mapping the data flow between the server cloud and the ECU. For example, during a Stride analysis of an 'Update Request' we can see if there is risk of 'Spoofing' the server, while processing the 'Tampering' threat highlighted by 'Firmware Download'.
It is important to integrate with safety standards to maintain the key theme. The SAHARA stands for Security-Aware Hazard Analysis and Risk Assessment, this approach blends two major methods of threat modelling that is STRIDE and ISO 26262 safety analysis, and it confirms that a security threat in antivirus is a security hazard and validates to use threat modelling as an important component. [8]

Below is a comparison table of more 5 research papers.

Paper	Description	Strength & Limitation	Key Contribution
Uptane: Securing Software Updates for Automobiles. (T Karthik et al., 2016) [9]	In this paper we can see the integration of multi-purpose architecture to deliver secure updates to ECUs. The car-based OTA security architecture is based on industry standard. The Update Framework (TUF).	It is robust against repository compromise and is installed on actual automotive systems. It also has ECU diversity and partial trust. Its prime weakness is its Complex key management, high operational overhead and relies on repository and OEM integrity.	This is the First automotive-grade OTA framework which provided compromise-resilient, multi-stage metadata checking of in-vehicle ECUs.
Secure over-the-air software updates in connected vehicles-Survey. (Subir Halder et al.,	This survey analyses survey the OTA update models, SOTA/FOTA mechanisms,	In this we get to see OTA threats, update methods and cryptography practises. On the	We obtain synthesized, organized knowledge about the OTA challenges and

2020) [10]	security threats and common vulnerabilities in smart connected vehicles.	other side being a survey there is lack of validation and threat modelling is still high-level.	security requirements of the entire OTA ecosystem.
Secure over-the-air software update for connected vehicles: STRIDE based (Amrita Ghoshal et al., 2022) [11]	This research paper is focused on proposing scalable, STRIDE-modelled secure OTA along with CP-ABE encryption and cloud based distribution.	This paper focuses on Stride and addresses scalability and secure access control, also has good threat modelling. It has High ECU computational cost (CP-ABE) and it requires a non-volatile network with only few embedded tests.	Presents a threat-based, scalable OTA distribute system which integrated encryption, time scheduling and end-to-end security.
Secure and Lightweight Over-the-Air Software Update Distribution for Connected Vehicles. (Christian Plappert et al., 2023) [12]	In this paper, lightweight OTA frameworks are adapted to small-scale ECUs and are presented through the utilization of the differential updates and minimal overhead verification.	The main highlight of this paper was that it reduces bandwidth and CPU usage and gives practical performance benchmarks which makes it efficient with embedded ECUs. While on the other hand it covered only few threat categories and it did not have supply-chain defence mechanism also no formal guarantees.	The key contribution was that it saved resources and showed scalable OTA schemes that can be used on real time low-computing automotive ECUs.
Automated Security Analysis of the Uptane Automotive Over-the-Air Update Framework. (Robert Lorch et al.,	Verification and demonstration of automated security analysis of the Uptane OTA protocol and also	This paper shows high assurance with formal proofs and shows hidden weak points that are not apparent in less	Presented rigorous formal security for OTA automotive pipeline which enhanced the reliability of systems

2024) [13]	identifies logical vulnerabilities and proves security claims.	formal analyses. It showed only an overview of certain real-life complications. The outcome needs to be careful with real-world interpretation.	like Uptane.

Application of Node.js and web technologies to automotive industries is an emerging trend which has been reported in recent technical reports and implementation studies. Node.js is often used in IoT backend due to its non-blocking nature, for ensuring the safety of firmware updates in IoT devices it is often studied through the prism of Node.js as it effectively handles JSON and is very well equipped with cryptography ecosystem. The non-blocking nature of Node.js is particularly marked as a key factor that is allowing constant connection between thousands of devices at the same time. JSON is listed as a required format of metadata in the Uptane Best Practices document due to its ubiquity and it is easy to parse, at the same time keeping the canonicalization rules in mind to ensure consistent hashing. A React-based threat management dashboard is necessary to serve as the interface that defines the Release Manager role in the ISO 21434. Thought this is less about the embedded system but the Software Update Management Systems (SUMS) highlights the importance and need of traceability and human management.

Threat Modelling Methodology: Applying STRIDE to OTA pipeline

To make sure that the proposed system is "Secure by Design" as per the ISO/SAE 21434, it is essential to have systematic threat evaluation. In this methodology we will look at the application of STRIDE methodology to an architecture of Node.js backend serving updates to autonomous vehicle.

- Cloud Backend (Node.js/Database): This is a very crucial and high trusted part as it will be containing the firmware images, metadata and most importantly the signing keys.
- Operator Console (React Dashboard): In this interface there will be human decision making. The threats in this part would have compromised admin accounts.
- The Network: In this part the traversing data would be subject to risk of interception and manipulation, calling it as zero-trust zone.
- Vehicle Gateway: This is the vehicle's entry point and it will verify the data that is incoming from external routes before passing it to the ECUs.

STRIDE Analysis

The following analysis will show specific STRIDE categories with scenarios within the OTA architecture.

i) Spoofing: An attacker manages to gain access and compromise the system to steal user's identity to access the operating system or its resources. In server spoofing,

An attacker plants a rogue server inside network for example, through DNS poisoning or rogue cell tower and then the attacker makes the user vehicle believe that connect is coming from an authentic Node.js backend. If the attack is successful then the attacker would be able to install malicious metadata or can even reject updates. If the Node.js server is not configured to perform strict TLS certificate pinning or a Mutual TLS, in such cases the vehicle might accept connections coming from any server, even an unauthorised server with a valid SSL certificate.

In Vehicle Spoofing the attacker send request as a valid vehicle to the Node.js API. Then they install firmware images which are meant to be used in certain VINs. This causes Intellectual Property theft (reverse engineering the firmware) or enables the malicious attacker to gain knowledge about certain unpatched vulnerabilities.

In administration spoofing the attacker takes the stolen credentials to gain access of the react dashboard. After doing this they are able to roll out a legitimate campaign that install defective software version.

ii) Tampering: It refers to any type of action that involves altering, adjusting, or manipulating data or programs without any lawful justification or need. In Man-in-the-Middle (MitM) Injection the attacker captures the data packet in which the firmware binary traverses through the cellular network and does some modification for its gain. For example, creating a backdooring the braking control module. If the application-layer signatures are not there then the TLS connection may end in a proxy or load balancer, this creates vulnerabilities which can modification the internal cloud traffic.

Metadata Tampering: The attacker alters the targets.json file in order to modify the listed hash of the firmware. When the vehicle unknowingly believes this modified metadata, it starts to compute the hash of the bad firmware then compare it to the tampered hash in the JSON and this leads to downloading the malware. This is the reason why JSON must be cryptographically signed.

In Database Tampering the attacker gets access to the backend database by SQL injection, and then shuffles the records of what software is installed in which vehicle to create chaos and causing a mix-and-match failure.

iii) Repudiation: In operator repudiation, a dissatisfied employee may use the React dashboard to install malicious update. After denying the action and without immutable logs the organisation will not be able to show who initiated the attack.

In Vehicle repudiation, the vehicle receives an update but it does not instal the update. Afterwards during investigation, the manufacturer states that the update was sent however the vehicle logs demonstrate that it was not. This uncertainty creates legal trouble under regulations like UNECE R155.

iv) Information Disclosure: This is regarding the unauthorized access to sensitive data. The data packets traversing should also be secured as they are also vulnerable to

attacks. An attacker can make assumptions about sensitive data just by observing the size and timing of the data packets, this can happen even if the payload is encrypted. For example, 50MB fleet-wide update can be an indicator of a serious emergency patch, this can alert the attackers to look for vulnerability before the patch out for update.
In manifest leakage, if the Node,js API is vulnerable, then the attacker would request the /vehicles/vin/manifest endpoint to gain insights into the specific software version the car is using, which can further be used to make a type of vulnerability map of the fleet.

v) Denial of Service (DoS): The attacker makes the system unavailable for legitimate users. An attacker will load their botnet to make millions of requests to the Node.js backend (GET /updates). This creates an event loop which floods and this results in legitimate vehicles are not being able to connect to check with safety-critical updates.
In Sleep Deprivation (Battery Drain), the attacker continuously sends repeated Wake up SMS or IP packets to the modem of the vehicle. What happened is then the vehicle activates its high-power Gateway ECU to acknowledge those requests, this drains the battery quickly and leads to stop the vehicle and make it physically immobilise.
In Indefinite Freeze, the attacker can stop the manufacturer from sending updates by manipulating the traffic to a particular vehicle and continuously relaying a valid response, which has a signature, and the response claims that no updates are available. The vehicle itself is stuck in time and remains unpatched to known vulnerabilities.

vi) Elevation of Privilege: This is basically gaining higher level access or permission which not supposed to be in hands of any user or process. The biggest threat in our case would be that the attacker gains access to the Node.js server and steals the private signing keys, with this the attack is now capable of signing any software (malware) and make it legitimate. The vehicle will assume and recognise is as a valid signature and install it in its system.
The attacker can also take advantage of a vulnerability in the Gateway ECU of the vehicle. Later, the attackers can gain root access to gateway by performing buffer overflow in the parsing and can move to the safety-critical CAN bus.

In some complex scenarios, we should look beyond individual security risks and consider complicated chain of attack.
For example, a malicious attacker misleads a vehicle to install an old and vulnerable version of software, though there is a new and secure version available.

- The attacker saved an old update data which was several months old.
- But today the vehicle should only install the new and safe version.

- The attacker will block or shut the server of the real update and forwards the old and insecure update version to the vehicle.
- The signature of that old update would still appears as a valid update and unfortunately the vehicle would trusts it.
- The car will now be one version behind containing security gaps.
- The attacker then exploits those old vulnerabilities so as to gain control of the vehicle.

From the above scenario we should understand that inly valid signatures are not enough. The vehicle system should have monotonicity checks to avoid such case and these are main components of mitigation strategy.

Implementation:

In this section we will see the operational implementation of the described OTA security framework. It highlights the produced outputs, components, tools and the programming languages used. It does not contain any listings of code, nor does it list any step-by-step user instructions, rather it records the end artefacts and the data transformations that occurs during the result found during testing.

Implementation Overview: In this research we will see a lightweight, end-to-end prototype system which simulates an OTA server and a vehicle-side which will act as client-side OTA. The design will apply the mitigation strategies that are based on the threat modelling phase (authentication, integrity, confidentiality, logging) and bring them to life in a working environment. The latter stages will include:

- A backend manufacturer server that will generate a signed manifest, which will have the firmware and will deliver it over HTTPS.
- The React application in the browser would simulate vehicle client which receives the manifest and signature, then it verifies the authenticity and integrity and then downloads the firmware.
- MongoDB stores the firmware and metadata of manifest fields and signature.
- During testing, orchestration scripts and test harnesses are used.

The implementation demonstrates a full and secure process of OTA (Over-The-Air) update between a vehicle and a manufacturer OTA server. When a new firmware is uploaded to the backend in fig. (3), the main server (i.e., server.js is the code) will read that file, then calculates its hash using cryptographic hash function SHA-256 and then creates a manifest that shows the name of the firmware, its version and its integrity value. This manifest is transformed into a JSON and signed with private RSA key of the manufacturer and stored in the server. The sign is only of the manifest and not of the firmware. The created signature makes sure that vehicle only accept updates provided that they were made by the legitimate manufacturer. The backend stores the firmware binary, the signed manifest string and the signature in the MongoDB as specified by the update schema in the models/ Update.js.

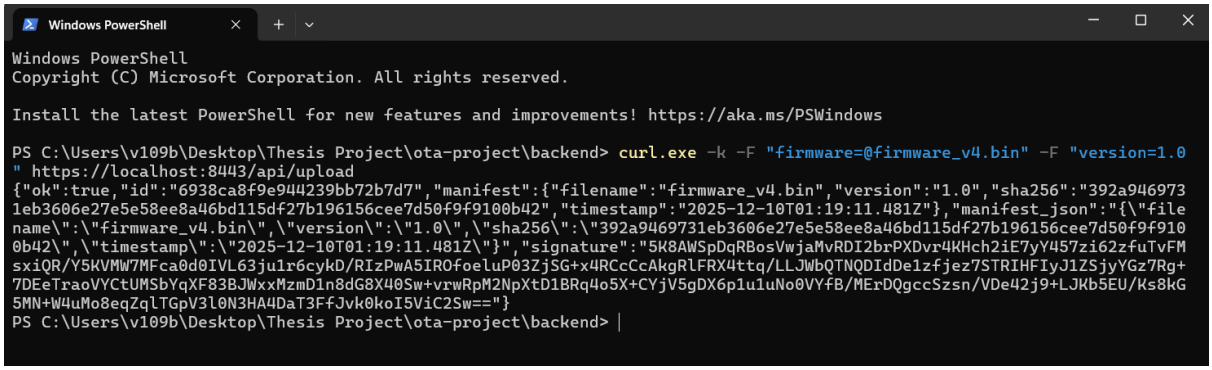
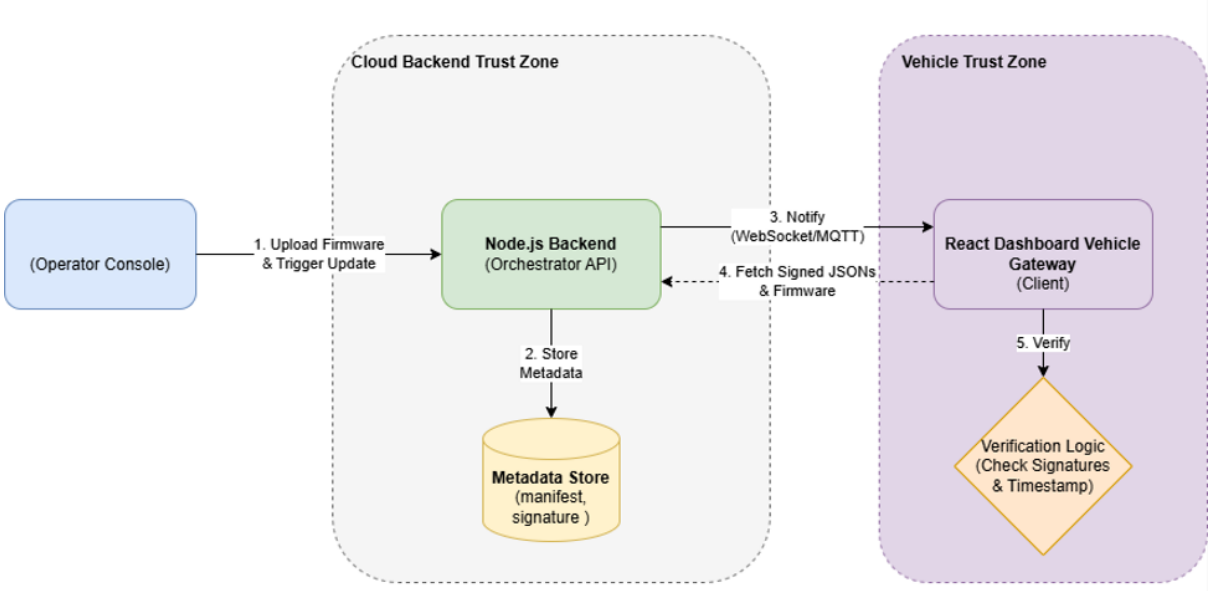


Fig.3

The React client (App.js) connects to the backend using HTTPS on the vehicle side to get the manifest, the public key by the manufacturer and the signature. The client verifies the digital signature against with the same string of the actual manifest by using the browser's Web Crypto API. If the verification is successful then the client downloads the firmware file and calculates its SHA-256 hash on its own. The update is accepted only when the downloaded firmware matches the SHA256 value in the manifest or else the vehicle client denies the update and mentions as manipulated. After passing the authenticity (signature verification) and integrity (hash match) check, the client installs the update by saving the firmware locally. This implementation shows a secure process of OTA that uses digital signatures, hashing and HTTPS to secure the transportation of a potentially spoofed or tampered firmware to the vehicle by proper coordinated functions of server.js, MongoDB update record and the verification logic implemented in App.js. Below is an architectural diagram of the implementation.



Data Transformation and Verification:

During each update the implementation process performs well defined data transformation:

1. Hashing – Computing hexadecimal sha256 hash for integrity token which is then stored in the manifest.

2. Manifest Construction – Builds manifest object and creates a deterministic JSON string i.e., manifest_json and it is the same message that will be signed.
3. Signing – The above created string is signed with RSA key creating a signature byte sequence.
4. Persistence – Stores firmware bytes, signature bytes, the manifest_json inside MongoDB.
5. Transport – The vehicle client will request the signature and manifest via HTTPS and downloads the firmware is successfully verifies its authenticity.
6. Client (vehicle) Verification – the vehicle client transforms the manifest string to bytes, then converts signature, imports the manufacturer public key and verifies the signature. Once the verification is successful the client calculates SHA-256(firmware bytes) and compares it to the manifest sha256, when it matches, the update is accepted, but when it does not, the update is rejected and recorded.

All the above transformations ensure Confidentiality (via HTTPS), integrity (by verifying hash) and authenticity (by signing the manifest).

Tools, Libraries and Languages used:

- **Backend**
  - Express.js for endpoints and API server
  - Node crypto module was used for RSA key signings and SHA-256 hashing.
  - Mongoose – MongoDB for updating the documentation.
- **Frontend**
  - React application which will implement the client status reporting.
  - Web Crypto API used for client verification.
- **Database**
  - MongoDB Community Server for storing signature, metadata and firmware blob.
- **Tools**
  - OpenSSL for generating 2048-bit RSA key pairs which are manufacturer_public.pem & manufacturer_private.pem and also for TLS certificate which are server.key & server.crt
  - Curl for testing and uploading scripts.
  - Powershell to run scripts.
- **Language**
  - Node.js is used in the backend.
  - Javascript is used in frontend.
- **Security constraints**
  - 2048 bits size of RSA key.
  - The signature will be RSA-PSS having SHA-256 and saltLenght of 32.
  - SHA-256 Hexadecimal for Integrity hash.

Test Output & Result:

- Nominal update Output: The new signed firmware file would be uploaded to the backend server as input by the manufacturer API, the server would give JSON object

(manifest and signature). This shows that backend has accepted the update. On the vehicle side the application will verify that the signature and if everything goes right then client log would print “Signature OK” then “Verifying integrity” and at last “Firmware downloaded/Applied”, a copy for proof would get downloaded in our system.

- Tamper Test: The firmware blob that is stored in the database will intentionally be altered by misplacing bytes without updating the manifest and signature. In the output, the signature verification get pass as the manifest is not changed but the hashes does not match and we get “Hash mismatch” as output on client side and the update is declined.
- Spoofing Test: By using a different private key a fake key is generated and is then injected in the manifest payload for testing. On the client side, we get to see “Signature verification failed” and it does not download or apply the firmware.

Limitations:

- In this implementation process a development self-signed TLS certificate is used, because it is suitable for only demonstration purpose but not for actual production implementation.
- In the demonstration the private keys are stored as pem files inside the server, but in actual production deployment it should be secure in HSMs or Cloud Key Management Services.
- For the easy demonstration of this research the client-side vehicle is simulated in browser environment because using real ECUs is out of the scope.
- The system in this demonstration shows only the cryptographic workflow and OTA lifecycle. Other operational hardening like rate limiting and DoS protection, etc is left for future work.

Mitigation Strategies and Defences Against Attack Types:

- **Man in the Middle Attack**: To mitigate Man in the Middle (MITM) attack, the ‘targets.json’ file will be signed by the system by a private key which will be stored offline. Even if the attacker manages to hijack the TLS connection, it will not be able to produce a valid signature of their malicious code. When the vehicle is getting update, it will calculate the hash of the software and will try to compare it with metadata field. If the hashes are not same then the vehicle rejects the update. This will ensure that no one can upload unwanted or hacked firmware by interfering with the connection.

- **Rollback Attack**: To defend against rollback attacks, the system uses version numbers which can only rise and never go down. The files ‘snapshot.json’ and ‘timestamp.json’ these files store the version number as integers, and in the backend it being strictly locked and it only increases with each new release. These numbers are

then checked by the update logic of the autonomous vehicle and if the version received is less than the installed version, the vehicle will reject the update. This layer of security prevents attackers from installing old and vulnerable software into the autonomous vehicle.

- **Denial of Service:** To prevent Denial-of-Service attacks the system reduces the amount of traffic to backend server and diverts heavy file transfer process to those systems which are built to bear heavy load. The Node.js server has a rate-limited which makes sure that no client can overflow it with an multiple requests. If there are large binaries of firmware then the server sends the vehicle a signed url to download via CDN instead of using backend servers. Basically, the Node.js server will only respond to small and moderate request size of JSON metadata, while CDN will be used to handle and absorb the heavy traffic.

- **Key Compromise**: To protect from key compromise, the system follows the principle of separation of trust between various different keys. To sign the timestamp files the server only utilises online key and offline key which is required to sign the more important ‘targets.json’ metadata is stored safely on Yubikey or HSM . If the attacker fails to exploit the backend server then the attacker will only be having access to the online key which won’t allow them to sign new vehicle software targets or to lure vehicles into downloading malicious firmware. Worst case scenario what they can do is they might stop updates and disservice signing new timestamps, but still they wont be able to inject any malicious code. This will have limited loss and which later can be recovered.

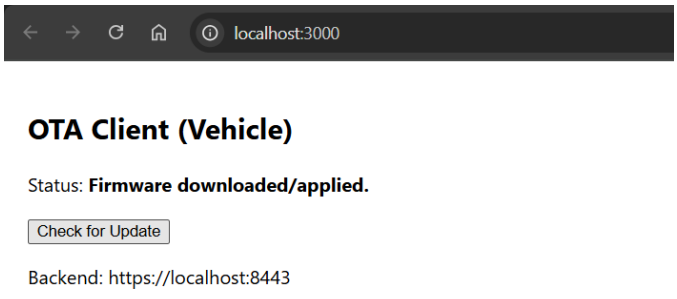
Evaluation:

Case 1: No Update Available

When firmware that is installed matches with the current latest firmware we get to see the message “No Update Available”. It helps in preventing Rollback attack by making sure that the outdated firmware is not again installed.

Case 2: Firmware Downloaded/Applied

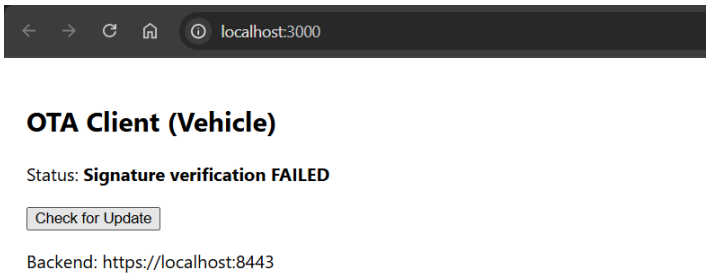
In below Fig (4) we can see that the vehicle client successfully downloaded the firmware from the server, which means that vehicle client has verified the signature and matched the hash to manifest, this confirms that authenticity and integrity of the firmware.



Fig(4)

Case 3: Spoofing Test

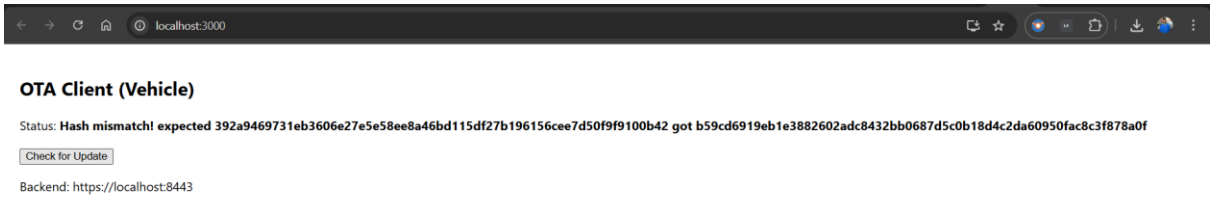
From the below Fig(5) we see that it is showing a Signature Verification Failed message, the vehicle client RSA-PSS verification fails immediately because of fake signature is injected. This demonstrates correct working of the system as it rejects firmware which is not signed by the real manufacturer's private key.



Fig(5)

Case 4: Tampering Test

In the fig(6), The Hash Mismatch status indicates that the manifest signature was valid but the firmware hash was tampered in the database. While comparing hash values the vehicle client detects that the SHA-256 hash is mismatch and therefore rejects the update, this offers strong integrity protection.



Fig(6)

Future Work:

The future of securing Over-the-Air Update will be Behavioural analysis. As we know that Artificial Intelligence is making its way into every field and is also taking centre stage in automotive software. As Node.js backend server produces a large volume of logs, one improvement that can be made to this thesis in the future would be to integrating a Machine Learning model so that it will process many data logs in real-time. It would be used to identify irregularities like one IP address creating requests to update 50 different VINs, which is called scraping attack, or a vehicle asking for updates at an unusual rate.

Automobiles have a very long life of about an average of 15 to 20 years, the algorithms that are being used today might become vulnerable to quantum computing attacks before the vehicle's lifecycle ends. The future research should emphasis on the concept of Crypto-Agility. Along with this, the integration of Digital twin technology is a growing area to focus on. When the firmware is ready to deploy, before that the Node.js backend can push the update to a cloud-based virtual ECU, known as digital twin of the vehicle. This makes sure that the installation of update does not disturb any system internally and also it adds an extra layer of secure verification into the pipeline.

Conclusion:

The integration of threat modelling and mitigation strategies is key to having secure automotive industry. After doing extensive research, the literature review highlights that even though that a decentralized architecture might have theoretical benefits, but a centralised architecture using advance web technologies like Node.js and React and having strict security protocols like Uptane will be a better option in terms of performance, scaling and regulatory compliance. The application of STRIDE threat model will help in identifying critical weaknesses in OTA pipeline, especially Spoofing, Tampering and Elevation of Privilege. The suggested mitigation strategies which require implementation of hierarchical structure of cryptographically signed JSON metadata are effective in mitigating threats.

The thesis topic specified in this report which is: Integrating Threat Modelling and Mitigation Strategies for Securing Over-The-Air (OTA) Updates for Autonomous Vehicles has been validated systematically. This thesis answers the research question by illustrating that the security of OTA updates in vehicles needs to have a combined approach where the threat model identifies the attack vectors which target the OTA pipeline and mitigation methods like hashing, digital signature, version validation, etc. directly addresses these kinds of threats. The demonstrated system incorporates these controls and the output evaluation presents that the vehicle client application correctly detects all tests which include tamper test, rejecting unauthorized updates and also verification of firmware's authenticity and integrity. Combining all this proves that this research of integrating threat modelling with mitigation methods gives a robust framework which ensures OTA update security in autonomous vehicles and is not just a feasible academic exercise but a reflection of best practices defining the future of automobile industry.

References:

[1] S. Mahmood, H. N. Nguyen, and S. A. Shaikh, "Systematic threat assessment and security testing of automotive over-the-air (OTA) updates," *Vehicular Communications*, vol. 35, p. 100468, Jun. 2022, doi: 10.1016/j.vehcom.2022.100468.

[2] J. Wang, R. Chen, H. Wen, Y. Chen, Q. Hou, and W. Duan, "A Secure and Efficient Vehicle OTA Software Update System: Design and Validation," in *2024 6th International Conference on Electronic Engineering and Informatics (EEI)*, Chongqing, China: IEEE, Jun. 2024, pp. 1176–1180. doi: 10.1109/EEI63073.2024.10696654.

[3] S. Aurangzeb, "CyberSecurity for Autonomous Vehicles Against Malware Attacks in Smart-Cities".

[4] N. M. Istiak Chowdhury and R. Hasan, "How Trustworthy are Over-The-Air (OTA) Updates for Autonomous Vehicles (AV) to Ensure Public Safety?: A Threat Model-based Security Analysis," in *2024 IEEE World Forum on Public Safety Technology (WFPST)*, Herndon, VA, USA: IEEE, May 2024, pp. 87–92. doi: 10.1109/WFPST58552.2024.00025.

[5] B. Li *et al.*, "Over-the-air upgrading for enhancing security of intelligent connected vehicles: a survey," *Artif Intell Rev*, vol. 57, no. 11, p. 314, Oct. 2024, doi: 10.1007/s10462-024-10968-z.

[6] "Threat Modeling and Risk Assessment of Blockchain-Based OTA Updates in Autonomous Vehicles".

[7] R. Mocnik, D. S. Fowler, and C. Maple, "Vehicular Over-the-Air Software Upgrade Threat Modelling".

[8] Zaina Abuabed, Ahmad Alsadeh, Adel Taweel,"STRIDE threat model-based framework for assessing the vulnerabilities of modern vehicles" https://doi.org/10.1016/j.cose.2023.103391

[9] Kuppusamy, T., Brown, A., Awwad, S., McCoy, D., Bielawski, R., Mott, C., Lauzon, S., Weimerskirch, A., & Cappos, J. (2016). *Uptane: Securing Software Updates for Automobiles*. In 14th ESCAR Europe 2016. https://ssl.engineering.nyu.edu/papers/kuppusamy_escar_16.pdf

[10] S. Halder, A. Ghosal, and M. Conti, “Secure over-the-air software updates in connected vehicles: A survey,” *Computer Networks*, vol. 178, p. 107343, Sep. 2020, doi: 10.1016/j.comnet.2020.107343.
[11] A. Ghosal, S. Halder, and M. Conti, “Secure over-the-air software update for connected vehicles,” *Computer Networks*, vol. 218, p. 109394, Dec. 2022, doi: 10.1016/j.comnet.2022.109394.
[12] C. Plappert and A. Fuchs, “Secure and Lightweight Over-the-Air Software Update Distribution for Connected Vehicles,” in *Annual Computer Security Applications Conference*, Austin TX USA: ACM, Dec. 2023, pp. 268–282. doi: 10.1145/3627106.3627135.
[13] R. Lorch, D. Larraz, C. Tinelli, and O. Chowdhury, “A Comprehensive, Automated Security Analysis of the Uptane Automotive Over-the-Air Update Framework,” in *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*, in RAID ’24. New York, NY, USA: Association for Computing Machinery, Sep. 2024, pp. 594–612. doi: 10.1145/3678890.3678927.
[14] W. M. A. B. Wijesundara, J.-S. Lee, D. Tith, E. Aloupogianni, H. Suzuki, and T. Obi, “Security-enhanced firmware management scheme for smart home IoT devices using distributed ledger technologies,” *Int. J. Inf. Secur.*, vol. 23, no. 3, pp. 1927–1937, Jun. 2024, doi: 10.1007/s10207-024-00827-x.