# Assuring Vehicle Update Integrity using Asymmetric Public Key Infrastructure (PKI) and Public Key Cryptography (PKC)

Daniel Kent,* Betty HC Cheng,† and Joshua Siegel†
*Department of Electrical and Computer Engineering
†Department of Computer Science and Engineering
Michigan State University, East Lansing, Michigan, USA
(kentdan3@egr.msu.edu, chengb@msu.edu, jsiegel@msu.edu)

***Abstract*—Over the past forty years, Electronic Control Unit (ECU) technology has grown in both sophistication and volume in the automotive sector, and modern vehicles may comprise hundreds of ECUs. ECUs typically communicate via a bus-based network architecture to collectively support a broad range of safety-critical capabilities, such as obstacle avoidance, lane management, and adaptive cruise control. However, this technology evolution has also brought about risks: if ECU firmware is compromised, then vehicle safety may be compromised. Recent experiments and demonstrations have shown that ECU firmware is not only poorly protected, but also that compromised firmware may pose safety risks to occupants and bystanders. While there have been no known instances of ECU firmware tampering on consumer vehicles outside of controlled academic or security research, and other work has been done to separate and compartmentalize ECUs, the security risks of unprotected ECU firmware must be addressed, especially as additional ECUs are developed to enable Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2X), and automated driving functionalities. To this end, we propose an asymmetric key-based infrastructure for signing and validating ECU firmware leveraging the existing federation in the vehicle component manufacturing space that exists between major automotive manufacturers and their major suppliers ("Tier-1 Suppliers"). Verification of firmware integrity occurs at ECU boot as well as during firmware updates. We developed a software implementation to demonstrate the feasibility of the approach and its resistance to certain types of attacks. Lastly, we performed an analysis of the scheme's possible attack surface, demonstrating how our proposal can enhance the current state-of-the-art in ECU firmware integrity.**

***Index Terms*—Vehicle, Electronic Control Unit (ECU), Automated Vehicles, Connected Vehicles, Firmware, OTA, Public Key Infrastructure (PKI), Public Key Cryptography (PKC)**

## I. Introduction

Since their introduction in the 1970s, Electronic Control Units (ECUs) have enabled the evolution of vehicles into their modern form comprising hundreds of ECUs that perform a wide range of tasks, including engine management, cabin climate control, and vehicle fault monitoring [1]–[3]. These ECUs have traditionally communicated over the Controller Area Network (CAN) bus, a standard supporting real-time communication of up to 1 megabit per second of data.

Given their widespread use, the security of ECUs has become an increasing concern in both security and automotive manufacturing sectors. Specifically, white hat studies focused on automotive cybersecurity vulnerabilities have revealed attack vectors with safety impacts, many of which include ECUs [4]–[7]. Recently, some work has been done to identify solutions to ECU network security [8] and inter-ECU trust [9]. There are even systems that have been developed to secure and validate ECU update delivery between manufacturers and vehicles over-the-air [10].

While existing ECUs may contain limited firmware flashing protections such as challenge-response pairs, these protections do not provide effective security against a determined attacker [11]. This paper presents a model for adopting Public Key Infrastructure (PKI) for assuring ECU firmware update integrity that provides protection against attacker models that differ from existing systems.

Patent filings [12] [13] have established that the need to validate and assure vehicle ECU firmware validity has been recognized by industry for some time, but adoption has been slow to date. Difficulties with developing such a system are due to the complex stakeholder dynamics between the automobile Original Equipment Manufacturers (OEMs)s, and their primary suppliers ("Tier-1 Suppliers") who manufacture most of the actual ECUs that are installed on consumer vehicles.

While Tier-1 Suppliers may strive to keep the number of distinct ECU configurations to a minimum, vehicle manufacturers often require customization to ECU firmware on a per-model or even a per-vehicle basis. A firmware assurance scheme needs to respect these potentially competing interests while minimizing cost and complexity.

This paper proposes an asymmetric key-based infrastructure for signing and validating ECU firmware that leverages the existing federation in the vehicle component manufacturing space that exists between major automotive manufacturers and their major suppliers ("Tier-1 Suppliers"). The proposed model is intended as a template for manufacturers to adapt and customize for their own products. Our distributed split-key system addresses both the low-level technical problem of securing ECU firmware and their update processes, as well as mitigate cryptographic key exposure at the manufacturer level. Our system also aims to be broadly compatible with lower-cost ECU hardware to reduce cost.

We test our system using an OpenSSL-based software application surrogate. Our surrogate implements both the organizational structure required to implement our scheme, as well as mechanisms required to create, transmit, and validate updates. We also developed tools that attempt to tamper with the firmware in ways that an adversary might attempt on actual vehicles, and we verified that our system is able to resist these tactics.

We present this work as a model for adoption by automotive OEMs and Tier-1 Suppliers. We provide this work as an open-source, freely available model for use across the automotive industry and beyond. The remainder of this paper is organized as follows: Section II discusses prior art that informs our proposed technique. Then we describe and explain our system in detail in Section III. Section IV describes several analyses that we performed to verify ECU update integrity against contemporary attacks, including tampering; we also discuss the impact and consequences of adopting our proposed system. Section V describes a software-based emulation framework we used to validate our proposed approach. We discuss in Section VI the implications and the potential impact of using our approach on stakeholder communities (e.g., manufacturers, independent mechanics), cross-ECU validation, and scheduling ECU updates. Finally,

Copyright © 2020 SAE International. All rights reserved. Provided for non-commercial use only.

we summarize the work and briefly discuss future investigations in Section VII.

## II. Background and Related Work

This section provides background information on Public Key Cryptography (PKC) and PKI relevant to our approach, as well as related work. We first created a general attacker model that our proposed system would need to defend against. We then analyzed existing systems and structures that leveraged PKC and PKI to provide security and assurance to systems and organizations, considered their strengths and weaknesses, and analyzed the costs (fiscal and otherwise) associated with their implementation and maintenance.

### A. Summary of PKC and PKI

Public Key Cryptography (PKC) is at its core based on asymmetric cryptography. *Symmetric* encryption is designed to use the same key to both encrypt and decrypt data. *Asymmetric* encryption, by contrast, uses two different keys called the *public* and *private* key respectively, with the implication that the public key cannot be used to recover the private key. Data encrypted with a public key cannot be decrypted with that same public key, but rather may only be decrypted with the private key. Similarly, data encrypted with the private key may only be decrypted with the public key. In addition to encryption of files, PKC can be used to *sign* data. For example, Rivest–Shamir–Adleman (RSA) signatures compute a cryptographic checksum of the data, then encrypt the cryptographic checksum with the private key; validating an RS signature requires decrypting the cryptographic checksum using the public key, then independently computing the cryptographic checksum of the data. If the signature cannot be decrypted with the public key, or if the checksums do not match, then either the data, the signature, or both have been altered. Since cryptographic signatures tend to be much shorter than key lengths and have consistent lengths that do not depend on the amount of data signed, signatures can be used to validate files of any length. Other key types such as Elliptic Curve Digital Signature Algorithm (ECDSA) rely on other mechanisms to create signature data from asymmetric key pairs.

Public Key Infrastructure (PKI) utilizes PKC to create a *chain of trust*. A chain of trust allows a user to trust a certificate so long as they can prove that the certificate has been trusted by an authority that the user has previously trusted, without requiring the user to have explicitly trusted that exact certificate in the past. In analogous terms, this scenario is similar to a child knowing they can trust their teacher, because their parents trust the teacher, and the child trusts their parents. This chain of authority is important in situations where it is not feasible for the user to store every single possible certificate, or when certificates and keys are changed [14].

### B. Attacker Model

Before designing a system, we considered several attacker models to focus our existing system analysis and eventually our proposed system's design. Like Uptane [10], we assume that attackers aim to gain access and residence on vehicle systems, thereby allowing attackers to monitor and control systems remotely, putting occupants and bystanders at severe risk. We also assume that attackers have significant control over the update pipeline, including but not limited to stored images and metadata, the wired and/or wireless network links, and the vehicle network bus.

Beyond these capabilities, we assume attackers are sophisticated and/or well funded enough that they can also compromise operational security in one or more ways. Examples might be convincing a security officer to obtain a signing key, to replace a firmware image delivered by a Tier-1 Supplierwith a malicious image, or to install malware on a primary security server.

### C. Existing PKI and PKC-based Systems

In designing our proposed assurance scheme, we analyzed existing PKC and PKI-based systems to determine the functionality, risk, and cost each system bears.

*1) Transport Layer Security:* Transport Layer Security (TLS) is an cryptographic communication security protocol used to secure traffic between web servers and the computers that connect to these servers, and is possibly the most widely used PKI-based system used today [14]. TLS utilizes X.509 standard certificates, which can be used solely to encrypt traffic to and from servers, or optionally can be used to identify a server as operated by a specific entity; the latter capability is often used for websites handling financial transactions. TLS is an evolution of an earlier standard, Secure Sockets Layer (SSL). SSL contains many known security flaws, and after a vulnerability was discovered allowing attackers to downgrade TLS connections to insecure SSL connections [15], most browser vendors opted to disable SSL by default [16] [17] [18].

When a client connects to a website secured with TLS, the server responds with its certificate, among other data used to secure the connection. This certificate is validated by the client against the client's root certificate store; if the client cannot verify that the server's certificate was signed by a root certificate authority that the client trusts, it disconnects from the website [14].

Server certificates are typically provided by an Intermediate Certificate Authority, which in turn is certified by a root certificate authority. Currently, more than 100 root certificate authorities are commonly recognized by most browsers and operating systems [19][20][21]. This decentralization of authority requires clients to maintain a database of public root certificates, which must also be kept up to date if root certificate authorities become compromised or otherwise lose their trustworthiness. For example, Symantec Corporation was discovered to have twice improperly issued certificates which could have been used to impersonate important websites [22], leading to their root certificate authority being phased out. If the improperly issued certificates had not been discovered, Symantec or one of their delegated authorities could have continued to issue certificates improperly, thereby potentially putting users at increased risk for phishing and malicious updates.

Based on the security vulnerabilities inherent in large-scale PKI trust chains as is the case with the TLS system, we assert that if a system designed for automotive ECU updates uses PKI, it should be based on a limited pool of root certificate authorities to lessen the likelihood of a rogue authority improperly issuing certificates. Ideally, such a system would only involve as few root certificate authorities as possible to reduce storage requirements on the ECU and to lessen the surface of the PKI certificate authority against a potential attacker who may be attempting to steal or duplicate a private signing key.

*2) United States Department of Defense (DoD) Common Access Card (CAC):* The United States Department of Defense uses a smartcard-enabled identification card (called the Common Access Card (CAC)) that doubles as a token with embedded public and private keys [23]. In a paper analyzing the potential attacks that could be made against users of CAC-enabled computers, Dasgupta *et al.* describe the CAC as a "non-tamperable, credit card sized smartcard" that "can generate public-private keypairs on the card and perform PKI operations *without exposing the secret keys to the host machine"*. Dasgupta *et al.* describe several theoretical attacks that could be performed against CAC-enabled systems; however, the

attacks described are only effective if the system the CAC is attached to is also connected to the Internet.

The most important aspect of the specific design of the CAC system that can be applied to an automotive ECU update scheme is in the sequestration of the private keys to the smartcard itself. As described by Dasgupta, the CAC performs all signing operations on-card, so that the key itself does not get exposed to the host computer, and furthermore the key is protected with a PIN. Without an easy way to copy the keys off the card, there is little risk that users of the key could inadvertently disclose their private key, and attackers would have to have physical access to the token in order to attempt extraction, a process which is theoretically lengthy enough that the credentials on that CAC can be revoked before an attacker can make use of them. An ECU firmware signing process could similarly use physical tokens to store keys. Further restricting signing activities to non-Internet connected machines may also mitigate the risks that Dasgupta *et al.* disclosed.

According to the United States Department of Defense's 2019 budget [24], the budgeted cost of the CAC system (including both the Real-Time Automated Personnel Identification System (RAPIDS) and Defense Enrollment Eligibility Reporting System (DEERS) components) in 2019 was about $2.2M. This system covers an estimated 2.8 million active duty, reserve, and civilian personnel, and an unknown number of contractors who are also issued CAC credentials. A PKI-based automotive ECU update system would, in contrast, not require as much infrastructure, and may not require all the identity verification features that the CAC system provides, which would likely result in a lower adoption cost. As long as the hardware cost for the physical tokens is not prohibitive to automotive manufacturers and suppliers, it seems feasible for physical tokens to be incorporated into a system for updating automotive ECUs.

*3) Desktop and Mobile Operating System Updates:* Both desktop and mobile operating systems already use PKC to assure update integrity. Many Linux-based systems either use the Red Hat Package Manager (RPM) or Debian Advanced Package Tool (APT) systems for distributing and installing program and operating system updates. These systems consist of centralized *package repositories* containing bundled and compressed copies of software, which is queried by *package managers* running on client systems. Packages for both of these systems can be signed by a private GNU Privacy Guard (GPG) key, for which the public key is known to client systems; this makes tampering with individual package contents difficult. Metadata for packages within each software repository are also signed by a GPG key, which makes tampering with the communication layer between the repositories and clients difficult. While these systems are not immune to attack, the most feasible attacks are replay attacks, which would only allow an attacker to force a client to remain on an old version of a particular set of packages [25].

Mobile operating systems also make use of PKC to secure updates. Google's Android operating system is designed to use asymmetric keys to sign firmware images that can be validated on-device before the image is applied. [26]. This verification mechanism can be used even if a user attempts to flash a firmware image manually, instead of using the built-in update tools. Some Android phones also make use of a dual partition layout, where updates are applied to a copy of the firmware that is not currently running. Upon reboot, if the update was applied successfully, and the image has not been modified, corrupted, or tampered with, the new image is booted; otherwise, the old partition is booted instead [27]. The update mechanism present in the Android mobile operating system shows that PKC can be used practically to secure updates in consumer-grade hardware without placing a significant knowledge burden on the user to perform these updates, which is promising for automotive ECU updates where updates may need to be performed with little to no operator interaction. In addition to assuring updates, Google has developed a security assurance framework called SafetyNet within their Android mobile operating system. SafetyNet can validate whether or not the end user has made certain modifications to their phone or tablet's operating system, which applications can use to restrict or deny functionality [28]. While it is possible to bypass SafetyNet [29], detection of modified operating system components remains an important priority for any ECU firmware assurance scheme.

Based on existing use of PKC and PKI in desktop and mobile operating system updates, we believe that extending this model to vehicle ECUs is possible. However, we need to establish what types of hardware will be capable of performing the necessary cryptographic operations necessary to implement PKI-based firmware validation. While we will establish some cost measures for additional cryptographic hardware in Section III-D, we have found evidence showing interest within the automotive sector for adopting PKC and PKI, even if additional hardware is required.

*4) Automotive PKC-Based Prior Art:* Recent patent activity shows that there is some developing interest in using PKC to secure and validate ECU firmware images and updates. We analyzed two patents, one from Hyundai Motor Company and one from Intel to determine the extent to which PKC and PKI has been proposed or could be leveraged to assure ECU firmware image and update integrity.

Hyundai Motor Company's 2013 patent covers a secure firmware update method [12]. Their method works similarly to session-key based communication, where an asymmetric key is used to encrypt a symmetric session key that is used to encrypt the bulk of the data transfer. Hyundai's system appears to be targeted at securing the communication between the vehicle manufacturer or Tier-1 Supplier, and the mechanic or dealership (described in the patent as "diagnostic apparatus"), rather than the entire chain between the Tier-1 Supplier, automotive manufacturer, and the ECU itself. Since the Hyundai patent's validation chain ends at the diagnostic apparatus, the expected cost of deployment would be low, especially if the diagnostic apparatus is a general-purpose computer that can already perform cryptographic functions in reasonably short amounts of time. However, this approach has a security flaw: if an attacker can compromise the diagnostic apparatus, they can control the firmware image that is loaded onto the ECU. We believe that end-to-end assurance is necessary in order to protect against possible upstream attacks. While this method is inherently flawed, it does demonstrate interest in adopting PKC by automotive manufacturers to secure ECU firmware transmission. It also provides context in one of our design decisions regarding encryption of update payloads, which is explained in further detail in the Section III.

An Intel patent describes a methodology for protecting device firmware, specifically describing their patent's applicability towards securing and assuring over-the-air ECU updates [13]. It describes a system where firmware is validated on boot using a security co-processor. If the security co-processor cannot validate the loaded firmware, it forces the device to use a backup firmware, or otherwise does not boot, similar to Google's A/B partition scheme for Android. Intel's exact method for the firmware validation is not described in the patent, but the method could potentially be implemented using PKC or PKI-based signature validation methods. Importantly, the Intel patent shows potential interest in utilizing additional hardware to perform the firmware validation, which may signal interest among the automotive manufacturing sector in utilizing additional hardware

despite the increased cost.

In response to the disparate set of firmware protection schemes (or lack thereof in some automotive systems), an open standard called Uptane was initially developed in 2016 to address over-the-air firmware image validation [10]. However, Uptane demands significant remote connectivity for ECUs in order to obtain and validate firmware images, which by design makes their system vulnerable to network denial-of-service attacks. In addition, the amount of server roles required in Uptane requires a significant and ongoing infrastructure investment that is borne mainly by automotive manufacturers. Lastly, Uptane's built-in mitigation against manufacturer or Tier-1 Supplier key compromise is limited; such an event requires by design a manual recall of affected vehicles.

### D. Takeaways from Prior Art

Based on existing PKI and PKC-based systems, we created a list of aspects we wanted a proposed ECU update assurance scheme to have. These aspects include: using asymmetric keys instead of symmetric keys to validate new firmware images, have the ability to use alternative keys, secure private keys against theft by storing them on physical tokens that are not attached to Internet-connected computers, and use existing ECU compute hardware or inexpensive hardware to minimize the cost of implementing our proposed scheme. There is also a crucial aspect missing from existing PKI-based schemes: an appropriate two-party signature scheme. While it is possible to design a PKC-based system to require two parties to provide a signature for a file to be accepted, this method would not work as well in the automotive sector, as it would require both an automotive manufacturer and a Tier-1 Supplier to cooperate whenever an update needs to be released. Our proposed scheme addresses this specific problem to delegate an appropriate level of control over firmware contents to both the Tier-1 Supplier and the automotive manufacturer.

## III. Proposed PKI Infrastructure Scheme

Based on the identified requirements, we have designed a scheme that makes use of a dual PKI certificate authority, one each for the Tier-1 Supplier and automotive manufacturer for a given ECU on a target vehicle. We have taken aspects from each of the discussed existing PKI and PKC-based systems and adapted them for automotive use. In particular, we used the idea of a certificate authority (as per TLS) instead of a single certificate (as implemented in desktop and mobile operating system updates) to validate ECU firmware and configurations. To prevent key duplication, we propose using physical tokens similar to the United States Department of Defense CAC, and we propose using new hardware if necessary to perform the cryptographic functions on the ECUs themselves, which in practice, may already be sufficiently powerful enough to perform the operations required. This system can also be used on ECU boot to validate its own firmware, similar to how SafetyNet or other boot validation systems work.

We propose a new firmware signature model that is split among two authorities: the Tier-1 Suppliers, and the automotive manufacturers. Each company would maintain their own PKI but would be able to share public keys, meeting the needs of automotive stakeholders.

### A. Firmware Certificate Authority Scheme

Our approach uses a multi-level certificate authority for each of the automotive manufacturers and Tier-1 Suppliers, similar to how most TLS certificate authorities are structured. In our examples, we use a three-tier certificate structure; while there are no technical restrictions preventing longer or shorter certificate chains, shorter chains require more certificate reuse which may incur more risk of certificate misuse, while longer chains with more certificates may be more expensive to maintain. Ultimately, automotive manufacturers and Tier-1 Suppliers would have the freedom to structure their certificate authority in a way that best suits their respective organizational structures and product lines.

We demonstrate two sample certificate authority models, one for a demonstrative Tier-1 Supplier, and one for a demonstrative automotive manufacturer, which are depicted in Figure 1. Each model starts with a single root certificate, then branches into different categories; in the demonstrative Tier-1 Suppliermodel, the second level branches based on ECU product category, and the third level branches into specific ECU models, each with their own unique key.

By comparison, the demonstrative automotive manufacturer has chosen to separate their second level based on model year, and their third level based on vehicle model. This structure would implicitly rotate second and third level keys once per year, which would reduce the size of the vulnerable fleet if any particular key were inadvertently disclosed.

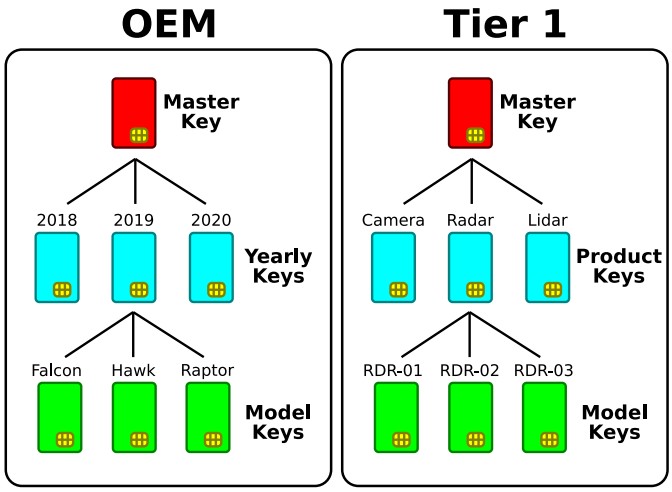


Fig. 1. A graphical depiction of our proposed PKI hierarchy. Lines indicate that a lower-level public key has been signed by the upper-level private key, creating a chain of trust. The left hierarchy represents a hypothetical automotive manufacturer, and is separated from the hypothetical Tier-1 Supplieron the right.

Importantly, each company's PKI certificate authority would be effectively separated from each other, which precludes the need for a third-party certificate authority to coordinate PKI authorities across manufacturers. This aspect allows each manufacturer to manage their own PKI and adapt its structure to best suit their needs, capabilities, and budget.

We propose that these certificates be generated and stored on physical tokens that do not allow direct extraction of the certificates, similar to Department of Defense (DoD) CACs. These certificates should be stored in physically secure locations while not in active use. While some may argue that Hardware Security Modules (HSMs) render physical tokens obsolete, independent HSMs are relatively expensive, and security vulnerabilities have been found with processor-included variants such as Intel's Trusted Platform Module (TPM) [30].

### B. Image Creation Process

In our proposed scheme, an image comprises four components: the firmware, the firmware signature, the configuration, and the configuration signature. The ECU firmware image is produced by the manufacturer of the ECU (commonly a Tier-1 Supplier), and is signed with the appropriate signing key from the PKI certificate authority we described earlier. Both the firmware and signature are delivered to the vehicle manufacturer. The manufacturer is then responsible for testing the firmware image and creating an appropriate configuration variant for this firmware image. Once the manufacturer has completed their acceptance testing and has produced a configuration for each vehicle the ECU is installed on, they will then use the appropriate vehicle model key from their own PKI to sign the concatenated firmware, firmware signature, and configuration.

In contrast to the Hyundai patent that encrypts the update, our proposed method does not require encryption of target firmware to assure update integrity. This design decision is based on three key motivating factors. First, the overhead introduced by the addition of the symmetric key may increase the cost of implementing the proposed scheme on resource-constrained ECUs. Second, it is conceivable that a third party could obtain these keys and thus the entire unencrypted payload by tampering with a fielded ECU. Lastly, even if the keys remain a secret, a motivated attacker could convince or trick a company insider who has access to unencrypted payloads or even full source code to provide a copy of the firmware data. Companies should carefully consider the value that encrypted firmware payloads provide, versus the additional cost for implementing on-ECU decryption requires.

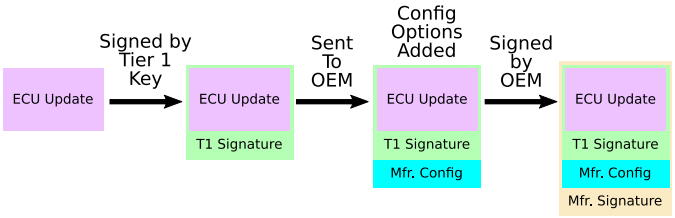


Fig. 2. Demonstration of firmware update build process.

### C. Onboard Firmware Update Validation

As described in the previous section, an ECU update is decomposed into four separate components: the firmware image, the firmware signature, the manufacturer-specified configuration, and the manufacturer-derived signature that covers the firmware, firmware signature, and configuration. During the update process, all four of these components are presented to the target ECU, along with any public certificates in the PKI chain that are not stored within the ECU's cryptographic hardware. If the ECU cannot validate the signature with any stored or validated certificates, it then will end the update process. Otherwise, it continues the update process by splitting off the firmware configuration file and validating that the combined firmware, firmware signature, and configuration corresponds to the provided configuration signature. If so, it then splits off the configuration and the firmware signature, and similarly validates the firmware signature with the firmware itself. This process is depicted graphically in Figure 3. If the manufacturer has changed the key used to sign updates, the update mechanism will then have to provide the new certificates used in the signature process. This update process would only require two additional steps, performed before the previously described update process, on the ECU: all the new certificates are first checked against the stored PKI certificates to validate that the new certificate falls within the root certificate's authority, and after the update is validated, the keystore is updated with the new certificates. The update process described in the previous paragraph would then continue as before. The combination key replacement with firmware update process is depicted graphically in Figure 4. This firmware update process could make use of a backup memory store similar to Google's Android A/B partition scheme to prevent malfunction if an ECU firmware update were to fail for some reason other than firmware signature failure.

### D. Cost Analysis

*1) Hardware Implementation Cost:* A quantifiable cost measure for implementing the proposed scheme comes from the cost of the additional hardware ECUs required to perform the public-key authentication and hashing functionality. For ECUs that do not have the necessary processing power to perform PKI validation and calculate firmware checksums, either accelerator chips must be added or a more powerful processor must be added. As an example, Maxim Integrated's DS2477 security coprocessor, which can calculate SHA-3 checksums and can perform public key authentication, had a listed cost of $\$0.97$ per $1,000$ units at the time of writing [31]. Based on 2018 production figures, securing 100 ECUs per vehicle using this chip would have cost $\$590$M for Ford Motor Company in 2018 [32], $\$840$M for General Motors in 2018 [33], and $\$480$M for FCA Group in 2018 [34].

Assuming 100+ ECUs per vehicle that need to be protected, the cost of this hardware will be difficult to justify. However, this cost could be partially mitigated due to current hardware usage. Not all ECUs may require secure firmware updates, which would remove the need for additional cryptographic hardware to be added to these devices. In addition, many ECUs already contain processors that can perform the necessary cryptographic functions, or can be substituted with equivalent parts for similar or lower costs. Many automotive-grade Cortex-based ARM microcontrollers such as the Cortex M-0 and M-4 already contain embedded security modules and cryptographic processing functions that can perform cryptographic functions quickly [35][36].

## IV. Analysis of Scheme Threat Surfaces

The goal of securing ECU firmware updates is to prevent malicious actors from modifying ECU firmware in ways that can alter a vehicle's behavior malignantly. Undesired behavior includes disabling, damaging, or destroying the target vehicle, injuring or killing occupants, other drivers, or pedestrians, and hijacking or redirecting the target vehicle(s) for financial, political, or personal gain. Even a single ECU with limited access to the vehicle network can potentially cause undesired behavior, as discussed by Cho and Chin in 2016 [37]. Firmware update hardening would have protected against the Miller *et al.* exploit, which required firmware modification of several target ECUs including the infotainment system [4].

It is important to note that no mitigation strategy is perfect, and while our proposed system can eliminate much of the current threat surface left by unsecured ECU update mechanisms, the system may still be vulnerable. However, the cost of implementing the system may be justified by the reduction in risk.

We next discuss how our proposed key scheme mitigates theoretical attacks that could be performed against ECUs protected by this keying scheme, and we provide our analysis on the residual risks for identified vulnerabilities.

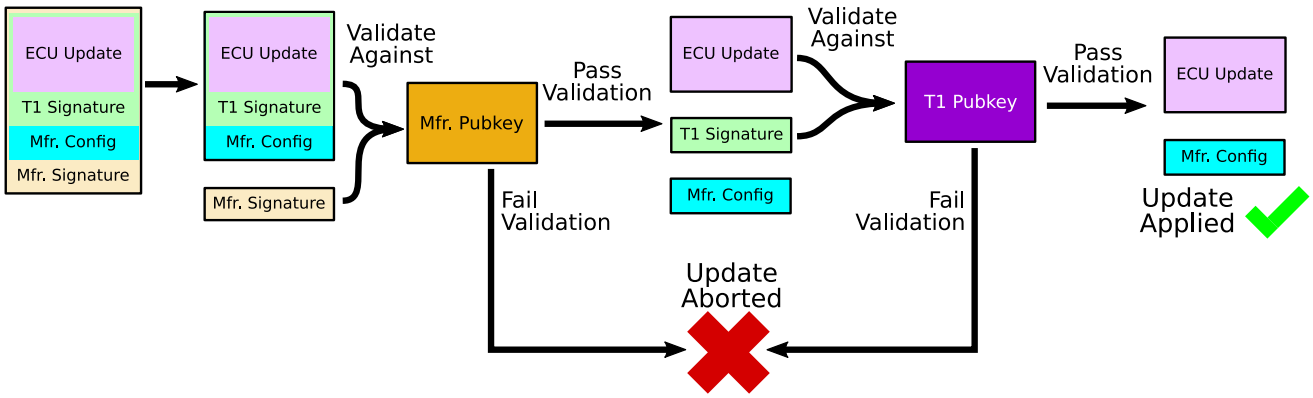


Fig. 3. Full Update Verification Process with No Key Changes

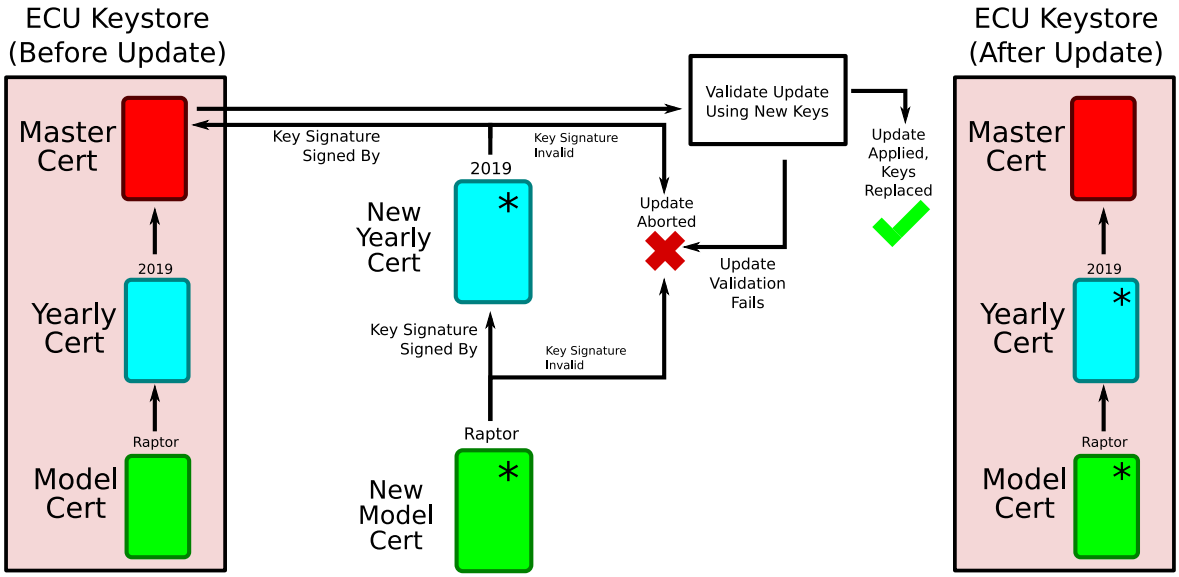


Fig. 4. Update process with key changes. Update validation section follows process depicted in Figure 3

### A. Basic ECU Firmware Validation

Part of the motivation for validating ECU firmware updates before installation is to provide assurance that any updates performed over unsecured channels such as over-the-air updates have not been modified or tampered. Regardless of whether modifications occur due to transmission errors, or from malicious actors operating at any point along the update chain, validation of the firmware and configuration at the ECU prevents the compromised ECU firmware from being flashed. Even if over-the-air update capabilities are not enabled for a particular vehicle, validation of ECU firmware is still important. If an automotive dealership or third-party mechanic's systems are compromised, it could allow for an attacker to replace legitimate firmware files with altered ones; validation of firmware prevents the potentially malicious firmware files from being flashed.

If an attacker were to attempt direct flashing of ECU firmware by connecting to the vehicle bus, then they would only be able to flash valid firmware images with valid configurations. They would not be able to add malicious functionality, or change configuration parameters beyond manufacturer specifications. Similarly, if an attacker had control over the vehicle's offboard communications, they would again only be able to flash legitimate firmware images, or deny new firmware images from being loaded. If an attacker had control over a manufacturer's update repository, then they could replace images with tampered images, but as the signing keys are kept safe these images would be detected as tampered with by ECUs if an install were attempted.

Of course, systems such as Uptane already implement ECU update assurances. Where our system differs is in its protection against attacks against key disclosures, which is discussed in the next subsection.

### B. Disclosure or Duplication of Keys

The consequences of improper disclosure of private keys can be catastrophic to end-user security. As an example, a computer manufacturer recently lost control of signing keys used to distribute software and firmware updates, which attackers subsequently used to target specific users with malware [38]. While that incident was unintentional, insiders inadvertently or deliberately disclosing keys is a threat that must be taken seriously.

Our proposed keying scheme mitigates inadvertent disclosure of signing keys in several ways. The first mitigation comes from our requirement that keys be stored on PIN-protected physical tokens, and only used on machines without direct access to the Internet. PIN protection restricts the ability to sign updates to those who know

the PIN, mitigating the risk of a key being stolen. By restricting card signing to specific non-Internet connected computers, it increases the difficulty that a malicious firmware file is signed by restricting the channels for files to be loaded and removed from the signing machine. If a non-insider were to steal the card by some means, then the extraction of the private signing key would take time, even for a sophisticated and well-funded attacker [39]. Even if an insider who knows the PIN were to steal the card, a routine security audit would uncover this improper access, allowing time for replacement firmware images with new keys to be issued.

The second mitigation comes in the form of the split signing authority. If an attacker has full control over the Tier-1 Supplier's key signing authority, they could produce signed malicious firmware files. However, these firmware files can only be loaded onto an ECU after the manufacturer adds their configuration and signs the firmware. In order for the Tier-1 Supplier firmware to be loaded, it would either have to pass through the manufacturer's acceptance and signature process, or the attacker would also have to have control over the manufacturer's keys. If an attacker only had full control over a manufacturer's key signing authority, they would only be able to load valid firmware files. While the attacker could insert invalid configurations that lead to the ECU behaving improperly or becoming disabled, they would not be able to add new functionality not present in the existing firmware provided by the Tier-1 Supplier, barring any exploitable vulnerabilities in the existing firmware images themselves. Further protections to prevent firmware rollback to known insecure images could be incorporated to protect against inadvertent release of signing keys.

The scheme's third mitigation strategy to address key disclosure makes use of key authority structure. Attackers that have full control over the root signing keys could produce valid firmware (in the case of the Tier-1 Supplier keys) and/or configurations (in the case of the manufacturer keys) for all products. However, by creating a tiered tree of keys, each subsequent layer of keys reduces the number of products that would be covered by said key, and thus the impact from an inadvertent key disclosure.

By incorporating aspects of *zero trust* into our update system, the system is able to better mitigate against disclosures of signing keys compared with other state-of-the-art systems, while providing similar levels of update assurances.

### C. Code Insertion in Development Process

We considered a scenario where attackers have control over the firmware image produced by the Tier-1 Supplier, which is used to insert a vulnerability into a firmware image that is approved and signed by the manufacturer without said vulnerability being found. We assume that this firmware image cannot prevent installation of valid signed firmware images nor add additional public keys for validating firmware, but otherwise has full control over the ECU's behavior. Our proposed keying scheme on its own does not protect ECU firmware from accidental, deliberate, or malicious flaws inserted during firmware development, but we can show that these cases can be partially or wholly mitigated by adopting some additional techniques.

Without any features within the firmware and/or configuration that can mark certain firmware versions as untrusted, vulnerable firmware versions could be installed, especially if the manufacturer's firmware signing keys have also been compromised which would allow attackers to produce valid firmware and configuration files. However, if version metadata are embedded into the firmware image, specific versions or ranges of versions could be identified and marked as untrusted by later firmware releases, preventing rollback to affected versions. One simple scheme would be to simply disallow installation of older firmware images. However, this approach would require the Tier-1 Supplier to produce a new firmware image if the current firmware image is found to be vulnerable, as otherwise attempting to install an older, non-vulnerable version would be rejected by the ECU.

One possible way to mitigate code insertion attacks is to extend the use of PKI hardware tokens to employees as well, similar to the DoD CAC system. All code commits would then have to be signed using a private key from a trusted employee, which would make unauthorized code commits easily detectable by automated systems and manual audits. However, considering the cost of the DoD CAC system, this solution may be too costly for Tier-1 Suppliers and automotive manufacturers.

### D. Time-based Attacks

Many PKC-based systems incorporate a time component to prevent replay attacks. As an example, the Kerberos network authentication protocol uses timestamps to ensure that authentication requests are recent, and have not been recorded and replayed at a later time [40]. This time-based authentication component requires that systems have their clocks roughly synchronized. In comparison, many ECUs do not contain real-time clocks, which could open up our proposed scheme to timestamp-based attacks if this weakness is not addressed.

Generally, lack of verifiable timestamp in a signed message results in a system being weak to replay attacks. In our proposed scheme, we do not require explicit global clock synchronization in order to validate updates. The lack of a globally synchronized clock could result in older, vulnerable signed updates getting installed in ECUs over newer, potentially fixed versions of the firmware. We considered adding a timestamp to the firmware version in our scheme, as well as a requirement that ECUs verify that the timestamp of the signed update is newer than the timestamp of the old update. However, we discovered a theoretical weakness with this approach if an attacker has full control over the manufacturer firmware signing certificate: they could advance this timestamp into the future, which would make it difficult for manufacturers to produce a new firmware bundle that can overwrite the malicious update. Replacing the timestamp with a version counter would still leave the ECU open to this type of attack.

There is a second concern regarding timestamps: certificates are generally produced with defined expiration dates. Without an on-ECU global clock, it would be difficult for an ECU to determine that a certificate is no longer valid. However, unlike the previously described theoretical firmware post-dating attack, certificate replacement is not similarly vulnerable. An ECU can assume that the global time is not older than the latest time of any certificate in its stored PKI database, and thus if a trusted certificate is installed with a timestamp past any of the stored certificates' expiration dates, then the ECU can assume that either the new certificate is invalid, or the certificates in its database are now out-of-date. If an attacker has control over some but not all of the PKI certificate authority, then they would only be able to post-date certificates up to the lifetime of the highest certificate authority certificate they control, which in our model would be 10 years if they had control over the Tier-1 Supplier type certificates or the Manufacturer model year certificates.

## V. Validation Studies

To test and validate our proposed scheme, we developed a software-based emulation of our framework, which we have made available for public review [41]. We developed an application using the Robot

Operating System (ROS) middleware [42] to emulate an ECU that periodically outputs a message at a configurable rate, that can accept update parameters in the form of four components as described earlier in this paper (firmware image, firmware image signature, configuration, combined firmware and configuration signature), which can be tested using the provided simulated update tools. Our PKI is generated using OpenSSL [43], and our software uses the Botan library for decoding cryptographic certificates and validating signatures [44]. Scripts are also provided to simulate the certificate authority generation process. These scripts allow users to create their own three-level certificate authorities using the automotive manufacturer and/or Tier-1 Supplier models described. There are also scripts available to create valid firmware images and updates for the demonstration ECU code. Valid firmware images and firmware updates are also provided to allow testing of our code without requiring the generation of a new certificate authority. We have opted to not provide the private keys used to generate these images, as disclosing private keys is bad practice that has catastrophic security implications [38].

We have also provided utilities that demonstrate the resilience of our scheme against two types of attacks: a firmware tampering attack, and an on-ECU firmware tampering attack. The former attack is detected by the demonstration ECU code during the flashing process. The latter attack is detected on ECU startup, which our demo ECU responds to by canceling its startup process. We recognize that failing to start may not be acceptable in production ECUs.

While our demonstration code does not support full PKI authority validation to support key changes, we are planning to extend our ECU firmware code to support full PKI authority validation as well as key updates in future versions. We also plan to expand the types of attacks to demonstrate resilience against other attack modalities.

## VI. Discussion of Proposed Scheme

While analyzing our proposed scheme for potential security vulnerabilities, we discovered some additioal effects that could result from the adoption of our assurance scheme. We also discovered areas where our proposed assurance scheme does not augment existing ECU security, and highlight some solutions that could potentially compliment our approach to create a more robust and secure system.

### A. Comparison with Uptane

At a glance, our proposed system appears similar to Uptane, which we briefly mentioned in Section II. However, while Uptane's design goals appear to focus on secured and assured delivery of firmware updates, it does not address ECUs that cannot connect directly to the update services, especially those that have limited computational resources. Our system, by contrast, could be implemented with lower cost hardware, is agnostic to the update delivery method, and also protects against organizational-level attacks. It may be possible to combine Uptane updates for higher-powered ECUs, which can deliver updates to lower-power ECUs with assurance provided by our method.

### B. Impact on Independent Mechanics and Vehicle Enthusiasts

Hobbyists and other vehicle enthusiasts make up a community of individuals who modify their vehicles' ECU firmware in a process known as "tuning." Currently, various off-the-shelf products allow for reprogramming of certain ECUs in various vehicle models. In theory, our proposed scheme would disallow modification of manufacturer-specified configuration parameters on factory-installed ECUs, which would force tuners to adopt other strategies for modifying vehicle performance. While disallowing modification of manufacturer-specified parameters may be desirable for safety and environmental reasons, it may also complicate enthusiast efforts to maintain their own vehicles if default parameters are found to be detrimental to vehicle performance or longevity. While our ECU security approach may challenge hobbyist reconfiguration of ECUs, it will also enable safe, assured ECU update application outside of an OEM's prescribed process, such as through independent mechanics or vehicle owners themselves. As discussed earlier, adoption of our proposed scheme would decrease the need to secure the entire transmission link between the manufacturer and the vehicle for firmware updates. Assuring firmware image integrity outside of the transmission process could allow for a public, manufacturer-agnostic, royalty-free ECU update standard to be developed, allowing any manufacturer, mechanic, or enthusiast to update firmware. Even if a person were to obtain firmware update images from a third-party source, our update validation mechanism would detect whether or not the image has been modified, preventing attackers from attacking vehicles by uploading malicious firmware on third-party websites.

As vehicles that adopt our proposed update validation scheme age, manufacturers will likely drop warranty support for these vehicles, and at some point may stop providing firmware updates altogether. Enthusiasts who wish to continue maintaining and repairing these older vehicles may be forced to resort to other methods in order to continue maintaining these vehicles.

Therefore, a constructive discussion regarding the merits of allowing ECU "jail breaking" for repair purposes on older, out-of-warranty vehicles is warranted, especially in the wider context of the right-to-repair movement.

### C. Cross-ECU Firmware Validation

Even if all ECUs on a vehicle can individually have their firmware images assured, it makes little difference if there is no means for an entire system to similarly be assured. Our proposed scheme is not intrinsically extendable to inter-ECU firmware validation. There have been some other approaches that have been used to assure that devices have valid firmware loads, as well as new techniques that specifically fulfill the need for ECUs to cross-validate.

Research has applied blockchain principles, which have been used for Internet of Things (IoT) device update verification [45], to solve the problem of cross-ECU firmware validation. One implementation developed by Falco and Siegel uses Distributed Hash Tables to reduce the computational burden of cross-ECU firmware validation [46].

### D. Scheduling ECU Updates

It is important to consider what times are allowable for a vehicle that needs to update its ECUs. If ECU updates are allowed at all times, then attackers could attempt to update target ECUs at inopportune times, such as when the vehicle is in motion, or right before the vehicle operator wishes to use the vehicle. If ECU updates can be performed within one or two message cycles, the effects of an inopportune update may be minimal, but many devices may require many minutes or hours to fully reprogram. An attacker that is able to trick an ECU into updating at an inopportune time can effectively turn even a valid ECU update per our proposed scheme into potent denial of service tool. While we have not provided a mitigation for this type of attack, others have examined this problem and have proposed solutions. One proposed method attempts to identify when a vehicle will be idle for a long period, with the prediction used to schedule ECU updates appropriately [47].

## VII. Conclusions

In this paper, we have demonstrated how our proposed PKI-based ECU update assurance scheme can be adopted by automotive manufacturers and Tier-1 Suppliers to ensure that ECU firmware and configurations can be validated on-ECU. This approach can effectively prevent malicious ECU firmware updates from being installed on vehicles. By providing a means for any given ECU to validate that both its Tier-1 Supplier and the vehicle manufacturer have approved a given update, the security of the update repository and the transmission mechanism for said update becomes far less important, thus enabling manufacturers to focus more time and money on securing other vehicle vulnerabilities.

We also believe that the adoption of a strong PKI system for ECU updates within the automotive sector may also inspire automotive companies to adopt other PKI-based solutions to secure other aspects of their business. Systems like the US DoD's CAC could be used by automotive manufacturers to better secure their systems against unauthorized use, by requiring their employees to use a physical token instead of a password to access critical systems. Our proposed system could even be adopted by the producers of manufacturing machinery to secure them against exploitation, an emerging security threat that is only recently being taken seriously [48].

We acknowledge that our proposed scheme will take time, effort, and most importantly financial backing to support adoption. Automotive manufacturers are typically very cost-conscious, and will not adopt proposals such as ours unless there is a clear financial benefit to doing so, or financial consequences of not adopting our proposal and instead risking an in-the-wild malicious ECU exploit. The cost for implementing our proposed scheme will drop as newer ECU-grade chip sets either become fast enough to perform cryptographic operations quickly, cryptographic extensions become included in automotive-grade chipsets by default, or the cost for add-on cryptograhic hardware continues to fall.

## Acknowledgements

This work has been supported in part by grants from NSF (CNS-1305358 and DBI-0939454), Ford Motor Company, General Motors Research, and ZF; and the research has also been sponsored by Air Force Research Laboratory (AFRL) under agreement numbers FA8750-16-2-0284 and FA8750-19-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL), the U.S. Government, National Science Foundation, Ford, GM, ZF, or other research sponsors.

## VIII. Acronyms

Advanced Package Tool (APT)
Common Access Card (CAC)
Controller Area Network (CAN)
Department of Defense (DoD)
Defense Enrollment Eligibility Reporting System (DEERS)
Elliptic Curve Digital Signature Algorithm (ECDSA)
Electronic Control Unit (ECU)
GNU Privacy Guard (GPG)
HyperText Transport Protocol (HTTP)
Hardware Security Module (HSM)
Internet of Things (IoT)
On-board diagnostics (OBD)
Public Key Cryptography (PKC)
Public Key Infrastructure (PKI)
Real-Time Automated Personnel Identification System (RAPIDS)
Robot Operating System (ROS)
Red Hat Package Manager (RPM)
Rivest–Shamir–Adleman (RSA)
Secure Sockets Layer (SSL)
Transport Layer Security (TLS)
Trusted Platform Module (TPM)
Original Equipment Manufacturer (OEM)
Vehicle-to-Vehicle (V2V)
Vehicle-to-Infrastructure (V2X)

## References

[1] P. Christidis, L. Pelkmans, I. De Vlieger, R. Cowan, S. Hultén, A. Morato, G. Azkárate, and R. Estevan, "Trends in vehicle and fuel technologies," *Review of past trends, European Comission, Joint Research Centre IPTS-Institute for Prospective Technological Studies, Spain*, 2003.

[2] J. E. Siegel, D. C. Erb, and S. E. Sarma, "A survey of the connected vehicle landscape architectures, enabling technologies, applications, and development areas," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2391–2406, Aug 2018.

[3] C. Ebert and C. Jones, "Embedded software: Facts, figures, and future," *Computer*, vol. 42, no. 4, pp. 42–52, 2009.

[4] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.

[5] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaâniche, and Y. Laarouchi, "Survey on security threats and protection mechanisms in embedded automotive networks," in *Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on*. IEEE, 2013, pp. 1–12.

[6] L. B. Othmane, H. Weffers, M. M. Mohamad, and M. Wolf, "A survey of security and privacy in connected vehicles," in *Wireless Sensor and Mobile Ad-Hoc Networks*. Springer, 2015, pp. 217–247.

[7] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, p. 148, 2019.

[8] L. L. Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in on-board embedded and networked automotive systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1038–1051, 2018.

[9] F. Kohnhäuser, D. Püllen, and S. Katzenbeisser, "Ensuring the safe and secure operation of electronic control units in road vehicles," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 126–131.

[10] T. Karthik, A. Brown, S. Awwad, D. McCoy, R. Bielawski, C. Mott, S. Lauzon, A. Weimerskirch, and J. Cappos, "Uptane: Securing software updates for automobiles," in *International Conference on Embedded Security in Car*, 2016, pp. 1–11.

[11] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, *et al.*, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010, pp. 447–462.

[12] H. J. Jung, H. S. Ahn, and C. H. Lee, "Firmware upgrade method and system thereof," South Korea Patent 20 150 074 414A, December 24, 2013, discontinued.

[13] J. Zander, M. Zmuda, I. A. Tatourian, and P. Szymanski, "Methods and apparatus to use a security coprocessor for firmware protection," U.S. Pending US Patent App. 15/273,997, March 7, 2018.

[14] J. Davies, *Implementing SSL/TLS using cryptography and PKI*. John Wiley and Sons, 2011.

[15] B. Möller, T. Duong, and K. Kotowicz, "This poodle bites: exploiting the ssl 3.0 fallback," *Security Advisory*, 2014.

[16] "The poodle attack and the end of ssl 3.0," *Mozilla Security Blog*, October 2014. [Online]. Available: https://blog.mozilla.org/security/2014/10/14/the-poodle-attack-and-the-end-of-ssl-3-0/

[17] "The poodle attack and the end of ssl 3.0," *Google Security Blog*, October 2014. [Online]. Available: https://security.googleblog.com/2015/09/disabling-sslv3-and-rc4.html

[18] "December 2014 internet explorer security updates & disabling ssl 3.0 fallback," *Internet Explorer Team Blog*, December 2014. [Online]. Available: https://blogs.msdn.microsoft.com/ie/2014/12/09/december-2014-internet-explorer-security-updates-disabling-ssl-3-0-fallback/

[19] "Microsoft included ca certificate list." [Online]. Available: https://ccadb-public.secure.force.com/microsoft/IncludedCACertificateReportForMSFT

[20] "List of available trusted root certificates in ios 13, ipados 13, macos 10.15, watchos 6, and tvos 13." [Online]. Available: https://support.apple.com/en-us/HT210770

[21] "Mozilla included ca certificate list." [Online]. Available: https://ccadb-public.secure.force.com/mozilla/IncludedCACertificateReport

[22] A. Ayer, "Misissued/suspicious symantec certificates," *mozilla.dev.security.policy*, Jan 2017. [Online]. Available: https://groups.google.com/forum/#!msg/mozilla.dev.security.policy/fyJ3EK2YOP8/yvjS5leYCAAJ

[23] P. Dasgupta, K. Chatha, and S. K. Gupta, "Viral attacks on the dod common access card (cac)," *Tempe, AZ: Department of Computer Science and Engineering, Arizona State University, ND http://cactus.eas.asu.edu/partha/Papers-PDF/2007/milcom.pdf*, 2009.

[24] "Li 500 - personnel administration," United States Department of Defense, p. 5, 2018. [Online]. Available: https://comptroller.defense.gov/Portals/45/Documents/defbudget/fy2019/budget_justification/pdfs/02_Procurement/04_DHRA_FY2019_Procurement_J-Book.pdf

[25] J. Cappos, J. Samuel, S. Baker, and J. H. Hartman, "A look in the mirror: Attacks on package managers," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 565–574.

[26] *Signing Builds for Release*, Online, Google. [Online]. Available: https://source.android.com/devices/tech/ota/sign_builds

[27] *A/B (Seamless) System Updates*, Online, Google. [Online]. Available: https://source.android.com/devices/tech/ota/ab

[28] *Protect against security threats with SafetyNet*, Online, Google. [Online]. Available: https://developer.android.com/training/safetynet

[29] C. Mulliner and J. Kozyrakis, "Inside android's safetynet attestation," *Black Hat EU*, vol. 2017, p. 75, 2017.

[30] M. Ermolov, "Intel x86 root of trust: loss of trust," March 2020. [Online]. Available: https://blog.ptsecurity.com/2020/03/intelx86-root-of-trust-loss-of-trust.html

[31] *DS2477: DeepCover Secure SHA-3 Coprocessor with ChipDNA PUF Protection*, Maxim Integrated, September 2018, rev. 0. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS2477.pdf

[32] "Ford motor company - 2018 annual report," Ford Motor Company, 2018. [Online]. Available: https://s22.q4cdn.com/857684434/files/doc_financials/2018/annual/2018-Annual-Report.pdf

[33] "Fourth quarter 2018 global sales," General Motors, 2018. [Online]. Available: https://investor.gm.com/static-files/94d3733b-213e-4cc4-be70-9f44dbd54944

[34] "2018 annual report," FCA Motor Company, 2018. [Online]. Available: https://www.fcagroup.com/en-US/investors/financial_regulatory/financial_reports/files/FCA_NV_2018_Annual_Report.pdf

[35] H. Tschofenig and M. Pegourie-Gonnard, "Performance of state-of-the-art cryptography on arm-based microprocessors," in *NIST Lightweight Cryptography Workshop 2015*, July 2015. [Online]. Available: https://csrc.nist.gov/csrc/media/events/lightweight-cryptography-workshop-2015/documents/presentations/session7-vincent.pdf

[36] L. P. Ledwaba, G. P. Hancke, H. S. Venter, and S. J. Isaac, "Performance costs of software cryptography in securing new-generation internet of energy endpoint devices," *IEEE Access*, vol. 6, pp. 9303–9323, 2018.

[37] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 911–927.

[38] (2019, Mar) Operation shadowhammer. Kaspersky Global Research and Analysis Team. [Online]. Available: https://securelist.com/operation-shadowhammer/89992/

[39] S. Skorobogatov, "Physical attacks on tamper resistance: progress and lessons," in *Proc. of 2nd ARO Special Workshop on Hardware Assurance, Washington, DC*, 2011.

[40] F. Butler, I. Cervesato, A. D. Jaggard, A. Scedrov, and C. Walstad, "Formal analysis of kerberos 5," *Theoretical Computer Science*, vol. 367, no. 1-2, pp. 57–87, 2006.

[41] D. Kent, "Pki-based ecu update assurance - public gitlab repository," Michigan State University. [Online]. Available: https://gitlab.msu.edu/kentdan3/pki-based-ecu-update-assurance-public

[42] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[43] "Openssl: Cryptography and ssl/tls toolkit," The OpenSSL Project. [Online]. Available: https://www.openssl.org/

[44] J. Lloyd, "Botan: Crypto and tls for modern c++," 2019. [Online]. Available: https://botan.randombit.net/

[45] M. Banerjee, J. Lee, and K.-K. R. Choo, "A blockchain future for internet of things security: a position paper," *Digital Communications and Networks*, vol. 4, no. 3, pp. 149–160, 2018.

[46] G. Falco and J. E. Siegel, "Assuring automotive data and software integrity employing distributed hash tables and blockchain," 2020.

[47] J. Siegel, "System and method for providing predictive software upgrades," U.S. Patent 9,086,941 B1, 2015.

[48] G. Sandaruwan, P. Ranaweera, and V. A. Oleshchuk, "Plc security and critical infrastructure protection," in *2013 IEEE 8th International Conference on Industrial and Information Systems*. IEEE, 2013, pp. 81–85.