applied sciences

Article

A Hybrid-Cryptography Engine for Securing Intra-Vehicle Communications

Walter Tiberti *, Roberto Civino, Norberto Gavioli, Marco Pugliese and Fortunato Santucci

Department DISIM and Centre Ex-EMERGE, University of L'Aquila, Via Vetoio 1, 67100 L'Aquila, Italy; roberto.civino@univaq.it (R.C.); norberto.gavioli@univaq.it (R.G.); marco.pugliese@guest.univaq.it (M.P.); fortunato.santucci@univaq.it (F.S.)
* Correspondence: walter.tiberti@univaq.it

Abstract: While technological advancements and their deep integration in connected and automated vehicles is a central aspect in the evolving trend of automotive industry, they also depict a growing size attack surface for malicious actors: the latter ones typically aim at exploiting known and unknown security vulnerabilities, with potentially disastrous consequences on the safety of vehicles, people, and infrastructures. In recent years, remarkable efforts have been spent to mitigate security vulnerabilities in intelligent and connected vehicles, in particular in the *inside* of vehicles, the so-called intra-vehicle networks. Despite those efforts, securing intra-vehicle networks remains a non-trivial task due to their heterogeneous and increasingly complex context. Starting from the above remarks and motivated by the industrial research and innovation project EMERGE, in this paper we report on a novel cryptographic hardware-software solution that we have designed and developed for securing the intra-vehicle network of intelligent connected vehicles: the *Crypto-Engine*. The *Crypto-Engine* relies on a lightweight hybrid-key cryptographic scheme to provide confidentiality and authentication without compromising the normal communication performance. We tested the *Crypto-Engine* and demonstrated that, once configured according to application-defined performance requirements, it can authenticate parties and secure the communications with a negligible overhead.

Keywords: intra-vehicle network; in-vehicle network; hybrid cryptography; elliptic curve cryptography; vehicular communication systems; V2X; V2I; V2V; security

check for updates

Citation: Tiberti, W.; Civino, R.; Gavioli, N.; Pugliese, M.; Santucci, F. A Hybrid-Cryptography Engine for Securing Intra-Vehicle Communications. *Appl. Sci.* **2023**, *13*, 13024. https://doi.org/10.3390/app132413024

Academic Editor: Yosoon Choi

Received: 26 October 2023
Revised: 22 November 2023
Accepted: 28 November 2023
Published: 6 December 2023

1. Introduction

When security issues are addressed, it can be observed that connected and automated vehicles are subject to a large number of risks. This is mainly due to the potentially wide attack surface that is exhibited by the large number of heterogeneous mechanical, hardware, and software components and their interconnections. Some recent attack trends are discussed in [1,2]. In those papers the authors summarize the attack vectors in *Sniffing*, i.e., the ability to intercept unsecured communications over a shared channel *Spoofing*, i.e., the ability for an attacker to establish a communication by faking its identity, *Jamming*, i.e., disrupt the communication by overwhelming the radio channels and *Replay Attacks*, i.e., blind re-transmission of a sniffed communication.

Additionally, there is a large number of possible attacks that targets known or, in the worst case unknown, vulnerabilities existing in all the software stacks. Examples can be found in the infotainment software layers [3], and in hardware components, e.g., micro-architectural vulnerabilities [4], or in possible exploits in communication protocols.

In general, we can state that the smarter and more connected a vehicle is, the more serious consequences are induced when security mechanisms fail: for instance, in terms of electrical or mechanical malfunctions (https://cybersecurity.att.com/blogs/security-essentials/the-top-8-cybersecurity-threats-facing-the-automotive-industry-heading-into-2023 (accessed on 27 November 2023)) and data leakage (Toyota data-leak: https://techcrunch.com

/2023/05/31/toyota-customer-data-leak-years/?guccounter=1 (accessed on 27 November 2023)). Many efforts can be found in recent years' research in order to protect vehicular communications, in particular in Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Anything (V2X) scenarios to a large extent. V2X links represent a relevant part of the so-called attack surface of the whole system, and other weak points may be found in the vehicle's on-board architecture. Indeed, modern vehicles are equipped with complex internal networks that connect components, Electronic Control Units (ECUs), electro-mechanical devices, sensors, infotainment systems, etc. Such architectures are evolving from the "ad-hoc" concept of VANET (*Vehicular Ad-hoc Network*) into networks that resemble classical computer networks, with all the well-known security issues. In this perspective, a standalone vehicle can be considered a fully-operational complex cyber-physical system requiring an equally complex multi-layered security solution. Neglecting this vision or underestimating its relevance could determine, for example, that an intrusion in any unsecured internal component of the on-board architecture can spread to other ones, potentially causing damage to the vehicle and to the passengers. On the opposite side, considering every internal component as untrustable can bring an unbearable level of complexity in communications that could collide with the computing and communications performance requirements that some critical applications and services may require. Resorting to cryptography-based solutions can help solve security problems; nevertheless, generic cryptographic solutions are often not designed to fit the needs of an intra-vehicle network with ambitious performance and low-latency requirements.

Within the context of intra-vehicle networks security, providing state-of-the-art security primitives along with secure and certifiable hardware/software implementations is a non-trivial task, especially when considering robustness performance and communication latencies. In this paper, we tackle this research challenge by proposing a novel solution to secure intra-vehicle networks aiming to provide the least possible impact on communication latencies: the *Crypto-Engine*. Born in the frame of the EMERGE project (*Veicoli Commerciali Leggeri* & Tecnologie EMERGEnti per operatività di tutti i giorni e di ausilio nelle EMERGEnze) described in the next sub-section, the Crypto-Engine is a comprehensive solution for securing intra-vehicle networks of connected vehicles via hybrid-cryptographic techniques.

The *Crypto-Engine*, which is presented as the core contribution of this paper, can be intended as a logical component that provides authentication of the communication sources and encryption/decryption of data streams with a reduced overhead. As additional optional services, the Crypto-Engine can provide in-vehicle storage protection, aggregation of intrusion detection probe points, and other application-specific cryptographic services (e.g., auditing, log management). Although the Crypto-Engine is generally able to secure any type of intra-vehicle communication link, the solution illustrated here has been designed specifically to secure the links carrying sensor data streams towards the main vehicle On-Board Unit (OBU).

1.1. The EMERGE Project

The EMERGE project (http://exemerge.disim.univaq.it/?page%20id=342 (accessed on 27 November 2023)) is an industrial research and innovation initiative co-funded by the Ministry of Economic Development of Italy and is targeted to design and prototype advanced on-board units for connected and geo-localized vehicles along with services of connected and cooperative navigation. The specific innovation goals refer to (i) a high accuracy and high integrity geo-localization module that integrates a multi-frequency and multi-constellation global navigation satellite system (GNSS) receiver, the support of augmentation networks and the fusion of on-board sensors' data; (ii) a heterogeneous on-board communications module that includes 5G connectivity, V2X sidelink communications and satellite connectivity; (iii) state-of-the-art cryptography and cyber-security solutions for connected vehicles that are supposed to comply to stringent safety requirements [5].

The novel on-board platform is intended to drive the development of a new generation of light commercial vehicles, which are conceived for dual-use (last mile good delivery and emergency operations), since their inception: in this perspective the EMERGE project, which includes large companies such as Leonardo and Telespazio other than research institutions, which embeds the development of a prototype of cooperative navigation service targeted to a small fleet of vehicles [6,7]. As already stated, two operating modes are envisaged and tested: "*every day*" and "*in emergency situations*". The related use cases, referring to the "electronic horizon" and cooperative navigation, resort to the large amount of data that are collected from the road scenario thanks to the on-board platforms which is then aggregated, stored, and elaborated on a *Mobile Edge Computing* (MEC) or cloud-based architecture; the latter one hosts specific algorithms based on machine learning for the prompt detection of events (accidents, traffic jams, . . .) and individual re-planning of routes for the fleet elements. The developed service components are considered as enabling modules of Level 3 automated driving.

When focusing on the specific assets of cyber-security of EMERGE, we first focused on passive security by designing an innovative asymmetric encryption schemes where protection levels are aligned to the automotive industry standards: in particular, we considered the ISO/DAE ISO/SAE 21434 [8], i.e., the emerging standard related to cyber-security in the automotive domain, and adopted a service-oriented paradigm of authentication and encryption services created by a centralized cryptographic engine that exposes standard interfaces towards the networked components regardless of their technical-manufacturing characteristics.

In the depicted context, this paper highlights the solution adopted in EMERGE for securing the intra-vehicle networks, the Crypto-Engine: it exploits state-of-the-art hybrid cryptography to build a platform that is shown to provide robust encryption/decryption of intra-vehicle communications and authentication of intra-vehicle components (e.g., ECUs).

1.2. Research Contributions & Paper Organization

The specific research contributions of this paper are summarized in the following.

- We present a novel solution for securing intra-vehicle networks through the usage of a logical, centralized component that uses hybrid-cryptography to provide confidentiality and multi-source authentication: the *Crypto-Engine*. In comparison with similar solutions (e.g., the *Secure Gateways* made available by several vendors (An example can be found in https://www.nxp.com/docs/en/white-paper/AUTOGWDEVWPUS.pdf (accessed on 27 November 2023))), the Crypto-Engine features can be distributed where needed, releasing manufacturers from adopting or interfacing with third-party components to secure communications;
- We provide a reference implementation of the hybrid-cryptography scheme for the adoption in intra-vehicle network components and a thorough analysis on its performance in a relevant real-world use case. While the presented implementation is software-based, the Crypto-Engine is designed to support HW acceleration as reported in Sections 3.2 and 5. This enables further performance improvements while maintaining the overall architecture unaltered.

The rest of the paper is organized as follows. In Section 2 we introduce the context and review the state-of-art on the security aspects of intra-vehicle networks; in Section 3 we present our proposed solution, the Crypto-Engine; in Section 3.3 we present the results of the security analysis of our solution; in Section 4 we describe the implementation, the validation, and the performance evaluation steps of our solution in the context of the EMERGE project along with a discussion on the results obtained; finally, in Section 5 we report some ending remarks and on-going research activities.

2. Intra-Vehicle Security

Securing a vehicle in terms of its intra-vehicle networks is not a trivial task. Indeed, a vehicle can be intended as an ecosystem in which mechanical, electrical, and software-based technologies exhibit mutual interactions: in such a complex context, the security failure of a single element can often induce effects in other elements—and this is what malicious actors typically aim to as a first objective. Nonetheless, when inspecting the possible issues that may affect the intra-vehicle security, it can be observed that some of them appear as their counterpart in common computer networks (e.g., traffic filtering, packet inspection) while other ones are peculiar to the vehicular context (e.g., CAN-bus issues). In the following sub-sections we review the state-of-the-art in securing intra-vehicle networks.

2.1. General Overview

A nice overview of the technologies involved in intra-vehicle systems is provided in [9,10]: the authors highlight the complexity of modern intra-vehicle networks and state that substantial efforts are spent by various organizations to explore standard solutions for the communication stack. The impact of complexity on security is twofold: on the one hand, the complexity makes it easier for malicious actors to spot possible vulnerabilities; on the other hand, it makes it difficult to implement effective security solutions. Some of the specific challenges and possible solutions are analyzed in [11]: in particular, in that paper the proposed solutions are grouped into a few categories, namely machine learning techniques, cryptographic solutions, communication protocols (e.g., CAN bus, FlexRay, Automotive Ethernet), traffic analysis and intrusion detection, physical access and sensor data security, and hardware security modules (HSM). Among the open challenges, the authors mention the implementation in real-scenarios of the solutions. With respect to these alternatives, in the present paper we propose a solution relying on state-of-the-art hybrid cryptographic features to secure intra-vehicle network nodes: the solution has been developed and tested in a real-world scenario in the context of the already mentioned industrial research project EMERGE. To the best of our knowledge, the Crypto-Engine is the first solution that considers topology-authenticated hybrid cryptography as the core mechanism to secure automotive intra-vehicle networks.

2.2. Challenges and Security Measures in Intra-Vehicle Networks

Intra-Vehicle Communication Bus Security

One of the most critical aspects is represented by the communication buses adopted for intra-vehicle communications. In this field, while several solutions and architectures are competing, Figure 1 summarizes an overview of the intra-vehicle communication technologies and protocols released in the recent years. As depicted, there is an increased proliferation of the Automotive Ethernet standards: more consolidated protocols (e.g., LIN, CAN and Flexray), while being still actively adopted, exhibit a slower release cycle. In fact, manufacturers tend to adopt heterogeneous solutions driven by the requirements and capabilities of ECUs. This often results in a complex hierarchical multi-level network that incorporates multiple technologies (e.g., the example scenario shown in Figure 2) wherein securing communications is a non-trivial task.

Among those communication technologies, the most commonly adopted are:

- CAN bus (CAN-FD, CAN-XL);
- LIN bus;
- Flexray;
- Automotive Ethernet.

Traditionally, the *Controller Area Network* (CAN) bus and its evolutions are still a de-facto standard for intra-vehicle communications. Nowadays, CAN bus adoption is often limited to near-ECU or subsystem-local communications, while other technologies are adopted when configuring a vehicle communication backbone (e.g., automotive Ethernet). Given the importance of the CAN bus in intra-vehicle communications, securing the CAN bus is of paramount relevance. In [12] the authors provides a review of the main footholds a malicious actor can exploit to attack the CAN bus. On the other side, in [13], the authors provide an overview on the vulnerabilities, attacks, and possible cryptography-based solutions in the specific context of the CAN bus. Attack-specific solutions do also exist: for instance, in [14] the authors attempt to address the *message flooding* attacks over the CAN bus.

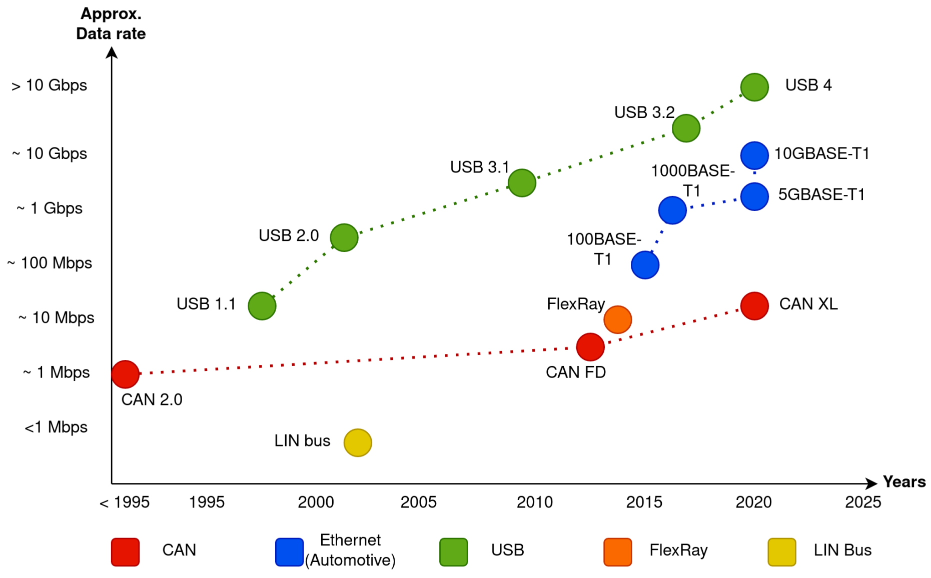

Figure 1. Communication technologies adopted in intra-vehicle communications.

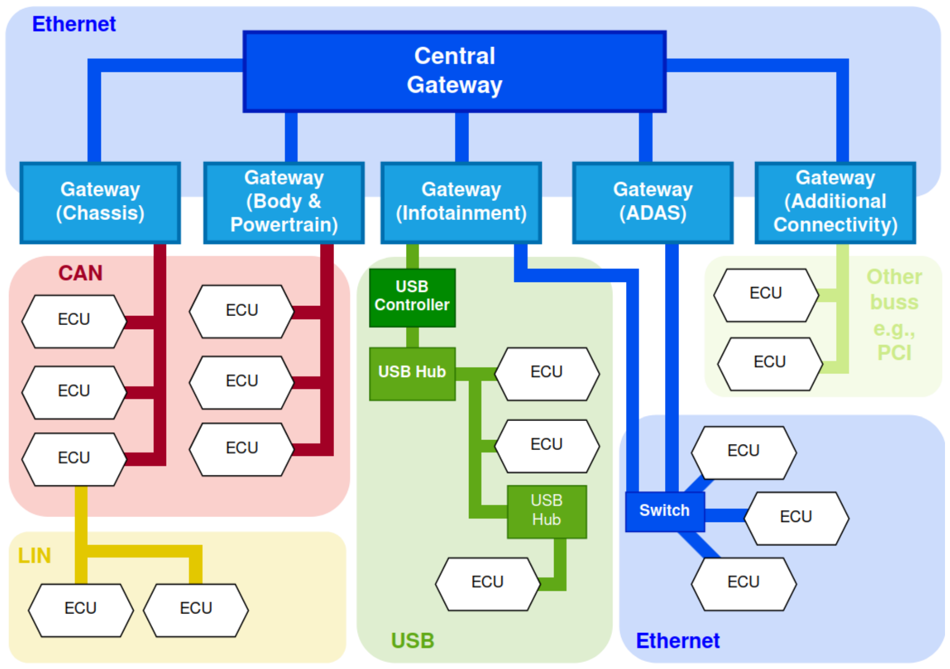

Figure 2. Example of a possible hierarchical intra-vehicle network.

Flexray security is also a hot research topic, both in terms of security issues and countermeasures [15,16] and in security solutions [17,18], including those relying on cryptography-based services, e.g., authentication [19].

The Local Interconnect Network (LIN) [20] is a low-cost communication standard for intra-vehicle communications. While the LIN bus is not meant to connect safety-critical sensors, there are a few research works focused on providing security features, e.g., [21]. A comprehensive review of the LIN bus security can be found in [22], where the authors highlight that the LIN bus is less secure than other solutions. Some examples of possible attacks are described in [23].

Automotive Ethernet is an alternative standard for intra-vehicle communications. Its main advantage is the compatibility with communication (and security) solutions and protocols typically available in computer networks. Alongside its advantages, however, Automotive Ethernet inherits the long list of security weaknesses, vulnerabilities, and attacks that afflicted and afflict Ethernet-based solutions. A security perspective of Automotive Ethernet, limited to the intra-vehicle scenario, can be found in [24]: in this case the authors propose some remarks on Automotive Ethernet when adopted in intra-vehicle communications.

In comparison with the reported bus-specific security measures, in the present work we propose a bus-agnostic security solution, which can be adopted without imposing requirements on the underlying communication bus. Our solution adopts a layered architecture that abstracts the communication bus and make the cryptographic operations independent from the protocol and technology used for communications. This abstraction, however, requires an explicit inclusion of bus-specific software drivers/wrappers and hardware support for the selected communication buses. The latter feature defines the set of adaptations to be carried out whenever an upgrade of the communication buses occurs.

Network Traffic filtering

One core security measure is the automatic analysis and filtering of the data and control messages exchanged by the nodes of the intra-vehicle network. Well-known tools such as firewalls and techniques such as (deep) packet inspection and flow analysis are adopted as well in intra-vehicle networks. For example, in [25], the authors describe the adoption of well-known firewalling and packet analysis techniques for intra-vehicle networks: from a security perspective it is suggested to use a real-time Operating System and HSMs to provide fast and lightweight implementation of the core cryptographic primitives to secure and authenticate traffic.

Intrusion Detection

Blocking and filtering communications via firewalls may not be sufficient to avoid malicious and dangerous traffic flows reaching the vehicle ECUs. An additional layer of defence is often adopted to ensure that the malicious traffic produced by on-going attacks passing through the firewall is spotted and blocked: this is typically referred to as *Intrusion Detection Systems* (IDSs). Intrusion Detection is a hot topic in intra-vehicle security: many recent works address the problem of implementing or adapting IDSs to intra-vehicle networks [26,27]. In this category, researchers have proposed various solutions, including side-channel (CAN bus voltage)-based solutions [28], lightweight unsupervised IDSs [29], using statistical characteristics [30], privacy-oriented IDSs [31], Graph-oriented IDS [32], and Cosine-similarity-based IDS [33]. Once a threat has been detected, networks may adopt an *Intrusion Response System* (IRS) to provide an adequate response to the threat. An example of its adoption in intra-vehicle networks is described in [34], wherein the authors propose a framework for IRS. A more thorough approach is adopted in [35], where the authors propose a *Security Operation Center*-based approach for real-time detection of incoming attacks using a vehicular Security information and event management (SIEM) platform. The proposed solution does not include intrusion detection or response capabilities directly, but it supports third-party IDSs via an intrusion detection manager component that takes care of providing interconnection between the IDS agents/probes and the IDS core.

Artificial Intelligence and Machine Learning

Among the detection strategies, the use of Artificial Intelligence and Machine Learning techniques has also seen increasing interest in the automotive domain. A recent survey on the topic can be found in [36]. Also, some notable uses of AI/ML techniques for intrusion detection are the following: ML for anomaly-based IDS [37] and temporal convolutional network IDS [38]. Discussions on AI and ML-based techniques are outside the scope of this paper, as they are part of a separate work to provide our solution with additional security services.

Security Standards

In the domain of vehicular communications, different standards compete to establish the core guidelines for security-oriented services. In particular, the 3GPP consortium, ETSI and IEEE, define different set of standards. In the context of the intra-vehicle network, the standard documents of interests are:

- ETSI Technical specifications 102 731 [39];
- 3GPP Technical specifications 33.401 and TS 33.501 [40];
- The IEEE 1609.2 standard [41].

However, despite the relevant efforts spent, there is no standard worldwide that is accepted and the different standards (and the newer being developed) still compete.

2.3. Cryptography in Intra-Vehicle Networks

Cryptography plays a key role in intra-vehicle networks and in the security solution adopted. The role of cryptography is to provide important security features to intra-vehicle networks, e.g., *confidentiality*, *integrity*, *authentication*, and *availability*. Among those, *authentication* in intra-vehicle is a hot research topic [42], since not all the intra-vehicle communication protocols incorporate embedded (and secure) authentication mechanisms. Among the possible authentication mechanisms, there are context-specific solutions [43], bus-specific solutions [44], watermark-based solutions [45], and Elliptic Curve Cryptography digital signatures (ECDSA) [46]. In this regard, our solution does provide authentication services but it is designed to be bus-agnostic. While this decision precludes us from accessing bus-specific metadata useful for authentication, it allows us to abstract the communication bus and offer better flexibility. On the cryptography side, we adopt an ECDSA version that is customized to work with the hybrid cryptography scheme while maintaining its security properties (as demonstrated in [47] and summarized in Section 3.3).

Regarding *confidentiality*, state-of-the-art symmetric ciphers (e.g., the Advanced Encryption Standard—AES) are commonly adopted with one critical aspect: the cryptographic key distribution/exchange problem. In intra-vehicle networks this problem can also be addressed in context-specific ways, e.g., using dynamic keys and a semi-centralized solution [48]. While the security level and performances offered by AES are undisputed, there are many possible pitfalls on how AES is used (e.g., mode of operations). In this regard, our solution makes full use of its hybrid cryptographic scheme to generate AES keys which do not require full distribution and that are bound to the network topology. Additionally, AES is used in CCM mode with an Encrypt-Then-MAC approach to provide *Authenticated Encryption*.

A useful cryptography-related feature that has been recently addressed is the *data immutability* via the use of blockchains. Examples of the use of blockchain-based mechanisms can be found in [49,50]. In those works, the authors list the increased computational cost among the big challenges. This is a critical concern for low-latency security solutions and may limit the performance while requiring additional resources and services. Due to this, our solution does not include blockchain technologies support as core feature but remains open to supporting them in scenarios with relaxed performance requirements.

2.4. Hardware Security Modules

A key challenge is to provide cryptographic features in such a way that security is achieved without relevant impact on computational and communication performances. An alternative to a classical software implementation of cryptographic features is to use application-specific hardware solutions that can provide better performance. This approach is often adopted by using HSMs in the form of *Cryptographic Hardware Accelerators*. An example can be found in [51] where the authors propose a hardware accelerator for AES encryption in automotive scenarios. Our solution supports the adoption of HSMs and HW accelerators, in particular for ECC computation, which exhibits an increased computational cost with respect to AES. As described in Section 3.2, we made use of the available enhanced cryptographic features of the target HW platforms for AES. While this led to inferior performances in comparison with [51], it grants us increasing flexibility when no HW acceleration is present.

Trusted Platform Module

Establishing the so-called *root-of-trust* is a critical problem in cryptographic implementations. A common solution is to move the root-of-trust in hardware (e.g., a separate chip). A well-known solution in this sense is the *Trusted Platform Module* (TPM) [52]. The TPM allows for the secure generation and storing of secrets (e.g., cryptographic keys, certificates, nonces, etc.) so that they are kept confidential (via *wrapping*) and strongly tied to the host platform by specific unique keys and certificates created at manufacturing time. Given the platform-independent nature of the TPM specifications, it can be adopted in any computational platform, including intra-vehicle networks. In fact, many automotive-related micro-controllers provide support to TPM specifications and implementation (e.g., specific pin headers (An example is the Infineon OPTIGA TPM product family https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-tpm (accessed on 27 November 2023))). In comparison with the TPM, the Crypto-Engine proposed in this paper is designed not only to generate and store keys, but it also acts as an active cryptographic accelerator. While using a TPM requires software (e.g., operating systems, cryptographic libraries, etc.) to ask the TPM for a key/signature, the Crypto-Engine is able to perform encryption and signature-related operations by itself in a transparent manner.

3. The *Crypto-Engine*

While considering the specific problem of securing the intra-vehicle network traffic, in this paper we propose a general and bus-agnostic solution to provide source authentication and encryption that is based on hybrid elliptic curve cryptography: the *Crypto-Engine*. Although being presented as a single component, the Crypto-Engine has to be actually considered as a *logical* component in which the security services are grouped together. Indeed, the *Crypto-Engine* appears as a distributed set of sub-components, located inside or in proximity of the positions where the specific security services are required. For example, if a communication bus has to be protected via encryption, all the end-points of the bus will include the Crypto-Engine's encryption and decryption primitives. The same applies for authentication, intrusion detection probes, etc.

3.1. Principles and Logical Architecture of the Engine

The Crypto-Engine represents the network of interest (e.g., the intra-vehicle network) as a clustered graph of transceiver nodes with almost converge-cast type communications for sensor data flows (e.g., sensor data collection) and multi-cast/broadcast operations for control flows. Clusters are composed by considering the physical location of the nodes or logical elements (e.g., nodes are grouped by their function).

The core cryptographic solution adopted in the Crypto-Engine is the *Elliptic Curve Topology-Authenticated Key Scheme*, i.e., **ECTAKS** [53], which is used to secure the aforementioned clusters of nodes. ECTAKS is a topology-based upgraded version of the well-known ECDH (ECC-based Diffie-Helmann) scheme: specifically, the network topology is added to the identity of the nodes as discriminating factor for the purpose of generating the shared key between the parties (and therefore the encryption of communications and the authentication of messages and of the parties themselves) and the establishment of protected communication sessions. One of the relevant and characterizing properties of the ECTAKS scheme is the authentication between connected parties according to arbitrary session topologies. The ECTAKS module provides key generation, distribution, encryption, and signature management via NIST-compliant elliptic curves [41,54–57]. Furthermore, ECTAKS supports multi-cast cluster-wise communications: the cluster itself can be represented in the ANT as a virtual node, with its own set of key components. The private components are then distributed to those (real) nodes allowed to perform downstream communications (i.e., senders), while the public components are distributed to all the nodes in the defined cluster. In this way, the cluster itself has a single set of key components and it is possible to secure multi-cast communications in a scalable way with a negligible impact on node memory storage (i.e., for the additional key components) and no additional impact on performance (i.e., from the ECTAKS point-of-view, a multi-cast communication is a normal point-to-point communication). ECTAKS relies on the pre-distribution of partial key components denoted as Local Key Component (LKC), Transmitted Key Component (TKC) and a set of Topology Vectors (TV). For a given node, the Local Configuration Data (LCD) is the tuple $(LKC, TKC, [TV_1, \dots, TV_n])$ and represents the cryptographic material required for establishing a secure communication. The term "topology authentication" refers to the unfeasibility in setting up secure links between nodes whose topology is not compliant to an Authenticated Network Topology (ANT) which defines the available communication links. The ANT is implemented by generating and pre-distributing the sets of TVs among the nodes. This operation is called *ECTAKS Key Component Generation* and takes place every time a new ANT is established.

Whenever a node (*Alice*) in the ANT wants to start a communication with another node (*Bob*) with ECTAKS, they need to agree to a shared secret, i.e., the Elliptic Curve Topology Authenticated Key (ECTAK), to be used as symmetric key. ECTAKS makes this possible by using ECC in combination with the key components in the LCDs of the two nodes. The details and the mathematical formulations are available in [53]. Once the cryptographic key has been reconstructed from the key components, the symmetric Crypto-Engine encryption services adopt the AES algorithm in CCM mode [58] using ECTAK as a symmetric key. For public-key signature services [59], the Crypto-Engine uses the ECTAKS Key Components in an ECDSA-based solution featuring the Hash-based Message Authentication Code (HMAC), as the message authentication function (MAC) and SHA256 [60] for hashing the message to be signed (i.e., HMAC-SHA256). The adherence to the current NIST security standards [54–56] and the formal proof of security [47] make ECTAKS fully qualified for industrial use.

Furthermore, ECTAKS integrates the schemes for ECIES encryption and for ECDSA sender authentication, which also rely on NIST standards and are also referred to in the IEEE 1609.2 [41] and in the Transport Layer Security (TLS) [61,62] protocols.

The main benefits of ECTAKS can be summarized as follows:

- Robustness against attacks on the topological integrity of the network;
- Generation of shared secrets on communication sessions with arbitrary topology;
- Scalability of multi-cast and converge-cast sessions;

A reference design of the Crypto-Engine logical architecture is shown in Figure 3.

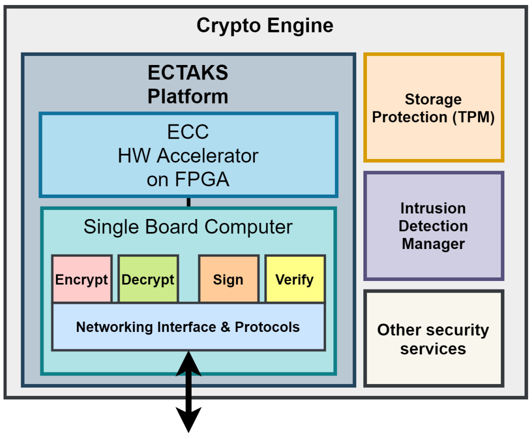

Figure 3. Crypto-Engine block diagram.

3.2. Hardware Acceleration

The Crypto-Engine has been specifically designed to support internal and external performance-wise HW acceleration for cryptographic computations. In the case of ECTAKS, the Crypto-Engine supports ECC hardware acceleration for the primitive operation on elliptic curve points, e.g., *Point Addition*, *Point Doubling*, and *Point Multiplication*. This enables mixed hardware-software Crypto-Engine implementations that can be adapted to different scenarios, platforms, and requirements. Some example implementations are the following ones:

- A software-only solution, in which ECTAKS and the other services are implemented in pure software for maximum flexibility across different HW platforms;
- An "internally-accelerated" software solution, within which the cryptographic operations are accelerated through platform-specific HW features and instructions (e.g., ARM/Intel cryptographic instructions or custom in-HW cryptographic implementations [63]);
- Using an external HW platform (e.g., a separated FPGA platform), towards which a software-based solution can offload cryptographic operations (Figure 3) for improved flexibility and re-usability;
- Adopting a hybrid programmable platform (e.g., ZYNQ (Xilinx ZYNQ platform architecture, available: http://www.ioe.nchu.edu.tw/Pic/CourseItem/4468_20_Zynq_Architecture.pdf (accessed on 27 November 2023))) to implement the ECTAKS primitives in SW, while the hardware acceleration for modular arithmetic operations and ECC is in the programmable logic (HW).

As described in Section 4, in this paper we report the results of a software implementation that is partially accelerated by exploiting the OpenSSL support for ARM and Intel HW-specific cryptographic instructions for the AES and SHA-256 library functions. Additionally, as reported in Section 5, we are currently working on releasing a Crypto-Engine implementation that features an on-demand, customizable external ECC accelerator for FPGA platforms to provide the Crypto-Engine with faster ECC computations.

3.3. Security Analysis

The security analysis of ECTAKS has been formally carried out in [47] with respect to an attacker that can compromise one of the nodes of the network and obtain full access to the node's secret information. According to typical approaches available in the literature, the security proof of the scheme is obtained *by reduction*, i.e., showing that an attacker to the scheme can be efficiently turned into a solver for a believed-intractable mathematical problem. Two attack scenarios have been considered: in the first one the attacker's goal is to determine the secret shared by other nodes in the network; in the second scenario the goal is to distinguish the shared secret from a random element. As an example, consider the minimal setting of Figure 4 and assume that the attacker gaining access to node 3 wants to attack the key exchange from node 1 to node 2.

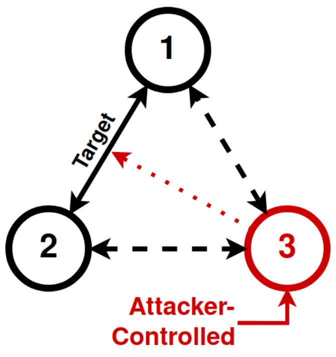

Figure 4. An ANT composed of 3 nodes (1, 2, 3). Node 3 is controlled by an attacker.

It has been proven that there exists a polynomial-time reduction which turns an attacker able to recover the ECTAK into an attacker which solves the *Computational Diffie-Hellman Problem* (*CDHP*) in the corresponding elliptic curve group. Similarly, it has been shown that there exists a polynomial-time reduction which turns an attacker able to distinguish a correctly generated ECTAK from a random element into an attacker which solves the *Decisional Diffie-Hellman Problem* (*DDHP*). Provided that the scheme parameters are set in such a way that CDHP and DDHP are hard in the corresponding elliptic curve group, the previously mentioned results show that there exists no efficient attacker against the secret recovery and indistinguishability of ECTAKS. Therefore, in the considered attack scenario, this makes ECTAKS suitable for providing military-grade security when used along with a safe elliptic curve; see [64] and a secure symmetric-key block cipher (e.g., AES).

4. Crypto-Engine Implementation in the EMERGE Project

As a main target of the EMERGE project is to design and develop novel on-board units with security-by-design guarantees, the Crypto-Engine has been proposed and selected by the whole partnership as a native solution for securing the intra-vehicle non-GNSS sensor communications. In particular, the Crypto-Engine has been requested to provide source authentication for every communication as the primary task while supporting the encryption and decryption of sensor data as additional tasks.

Given the loose latency requirements for non-GNSS communications, the "EMERGE version" of the Crypto-Engine has been developed and deployed entirely via software implementation in a separate single-board computer (SBC) platform (i.e., without additional hardware acceleration). Furthermore, authentication is provided with a custom EAP [65]-based authentication protocol in which the Crypto-Engine covers both the *Authenticator* and *Authenticator Server* roles.

4.1. Implementation

The EMERGE Crypto-Engine implementation consists of a key component generator written in Python and a compiled software library written in C++11 which provides the cryptographic primitives.

The key generator is intended to run offline to create the key components according to the defined ECTAKS ANT. In particular, the generator allows to create any number of key component sets (one set for node) with a selectable key length. The key component generation pseudo-code can be found in [47].

The software library is the part of the Crypto-Engine that provides the `ECTAKS-Encrypt`, `ECTAKS-Decrypt`, `ECTAKS-Sign`, and `ECTAKS-Verify` functions. It has two main dependencies: *OpenSSL* [66], and *libGMP* [67]. OpenSSL library has been adopted only for the AES implementation (used by ECTAKS-ECIES) to decouple the symmetric encryption part from ECTAKS and to provide state-of-the-art performance. The same motivation holds true for LibGMP, which is instead used for the low-level modular arithmetics on large multi-word integers. Both dependencies are not strictly coupled, so they can be easily replaced by custom solutions if necessary.

4.2. Validation and Performance Evaluation

Experiments setup

The validation and performance evaluations were carried out on two SBC platforms. The platforms are compliant to the EMERGE project requirements for the Crypto-Engine platform and exhibit the following main features:

- Raspberry PI 3B+ board, (abbr. **RPI**) equipped with a 1.4 GHz 64-bit quad-core ARM Cortex A53 processor, 1 GB of LPDDR2 (800 MT/s) RAM running Raspberry PI OS (Raspbian) 64-bit Linux;

- Intel NUC 11 Pro, (abbr. **NUC**) model NUC11TNHi3, equipped with a Intel i3-1115G4, clock frequency 1.7GHz to 4.1 GHz (boost), 8 GB of DDR4 (3200 MT/s) RAM, running Ubuntu 20.04 LTS 64-bit Linux.

For each platform, we conducted three experiments and estimated the performance of the main tasks of the Crypto-Engine according to the indexes listed in Table 1. In carrying out the experiments, attention was paid to make the impact of the underlying operating system features negligible (e.g., process context switching, caching, page faults, etc.) on the measurements taken. In particular, each experiment has been run with the highest priority allowed (i.e., the minimum *niceness* level available, in both platforms, has been set to -20). This allowed us to reduce cache misses (usually affecting payload sizes larger than $\sim$512 bytes) and page faults (mostly affecting payload sizes larger than 4096 bytes, which is the default page size for both platforms).

The adopted ECTAKS scheme configuration uses the standard NIST P-256 curve [55], AES symmetric key cipher with 256 bits keys in CCM mode (i.e., Counter mode for encryption and CBC-MAC as authentication tag), PBKDF2 (*Public Key Key Derivation Function #2*) as Key derivation function, and SHA-256 as cryptographic hash function.

The set of raw measurements obtained from the experiments are available at GitHub [68].

Table 1. Set of experiments and indexes for validation and performance evaluation of the Crypto-Engine.

#	Experiment	Metric of Interest
1	ECTAKS Key Generation	Time required vs. number of nodes (full-mesh)
2	Encryption/Decryption	Time required vs. payload size (bytes)
3	Sign/Verification	Time required vs. payload size (bytes)

Experiment 1: Generation of ECTAKS Key Components

As described in [53], ECTAKS requires the pre-distribution of the sets of Key Components to the nodes according to the ANT. This step is intended to be performed offline at deployment time and whenever a change in the ANT occurs.

This first experiment aimed to evaluate the time required by the Crypto-Engine and ECTAKS to generate a full set of key components for a full-mesh ANT. This topology has been chosen since it can be intended to represent a worst-case scenario: indeed all the ANTs allow secure linked from one node to all the others, with a number of key components that grow exponentially with the number of nodes. In this way, we can evaluate the *Worst-Case Execution Time* (WCET) for the Key component generation, a typical metric for assessing the performance in real-time systems.

The average results obtained by evaluating the ECTAKS Key Generation on the target platforms are shown in Figure 5.

As expected, the reported measurements show an exponential growth for a scenario in which a whole full-mesh ANT is setup from scratch. Although, on one hand, such an ANT setup might take a non-negligible time (up to 200 s for RPI and 50 nodes), on the other hand the update of an ANT with a new node (i.e., generating a local key component while adding $N-1$ topology vectors and $N-1$ transmit key components) requires a linearly increasing time and is significantly faster (<1 s) in the retrieved measures. It is also evident that the NUC platform overperforms the RPI in the key component, in particular by a factor of 13.4 that tends to be almost constant while increasing the number of nodes.

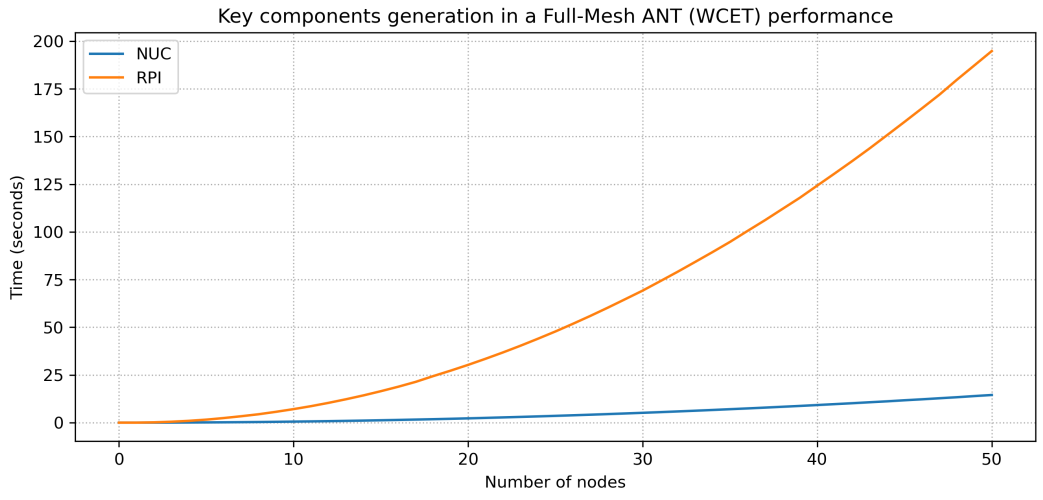

Figure 5. Execution time (y-axis, in seconds) for *Key component generation* versus number of nodes N (x-axis) in a full-mesh ANT.

Experiment 2: ECTAKS Encryption/Decryption

Figures 6 and 7 report the latency introduced (in seconds) when the Crypto-Engine is asked to encrypt or decrypt via ECTAKS a randomly generated payload with a size in the range between 1 Byte and 10 KiB. Every point in the graphs has been obtained as the average value over 1000 runs.

The RPI platform (Figure 6) shows an encryption time between 6 and 6.4 ms and a decryption time between 4.3 and 4.8 ms. Both primitives show a latency with a linear growth, although the decryption is noticeably faster. This behavior is expected, since ECTAKS decryption requires a smaller number of basic operations.

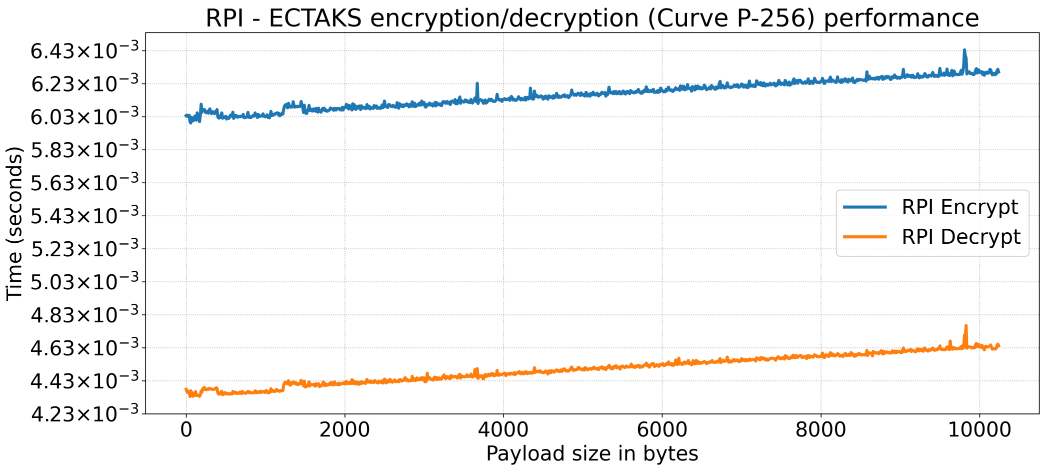

Figure 6. Crypto-Engine encryption/decryption performance on Raspberry PI model 3B+: time (y-axis, in seconds) versus payload size (x-axis, in bytes).

The NUC (Figure 7) shows a better performance, with an encryption time between 0.73 and 0.87 ms and a decryption time between 0.59 and 0.73 ms. Here, the NUC shows 6 to 8 times better results (expected, thanks to the faster CPU and RAM). However, the NUC shows some interesting behaviors:

- There is a noticeable instability for payloads smaller than 1 KiB;
- In the encryption, there are latency spikes separated by $\sim$ 1.5 KiB during the encryption;
- The growth of the decryption time increases, reducing the gap with the encryption performance on large payloads.

Those behaviors can be motivated by the superposition of effects of caching, page faults, and dynamic memory allocation (i.e., *heap*) effects triggered when de-allocating memory (e.g., *consolidation* and *binning* of free chunks of heaps).

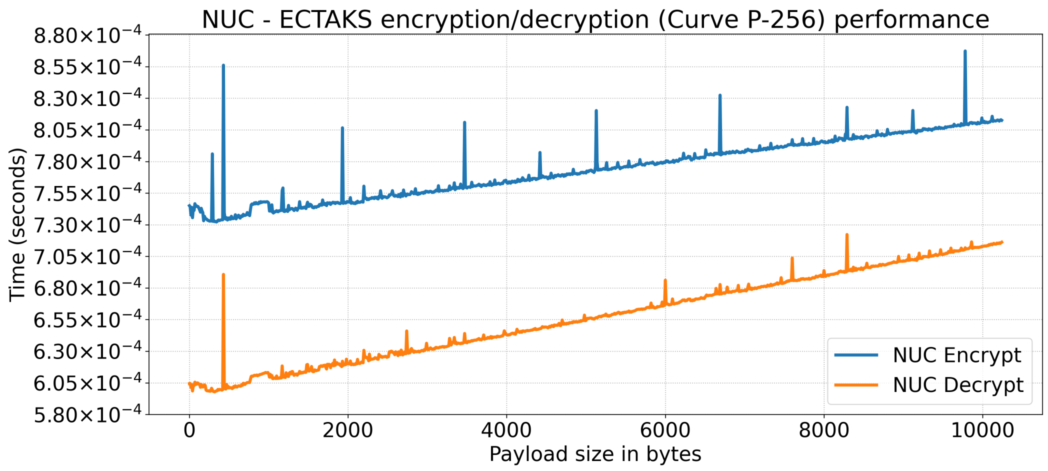

Figure 7. Crypto-Engine encryption/decryption performance on Intel NUC 11TNHi3: time (y-axis, in seconds) versus payload size (x-axis, in bytes)

Experiment 3: ECTAKS Signature Generation/Verification

Figure 8 shows the results for ECTAKS cryptographic signature generation and verification on payloads with sizes from 1 byte to 10 KiB for the RPI. The results show that signature generation takes from 45 µs to 50 µs with a very limited dependency on payload size. A similar scenario was observed for the signature verification, which is noticeably faster that the generation and takes as long as from 5 to 10 µs. This is expected, since the verification does not require the hash computation (i.e., in this case SHA-256).

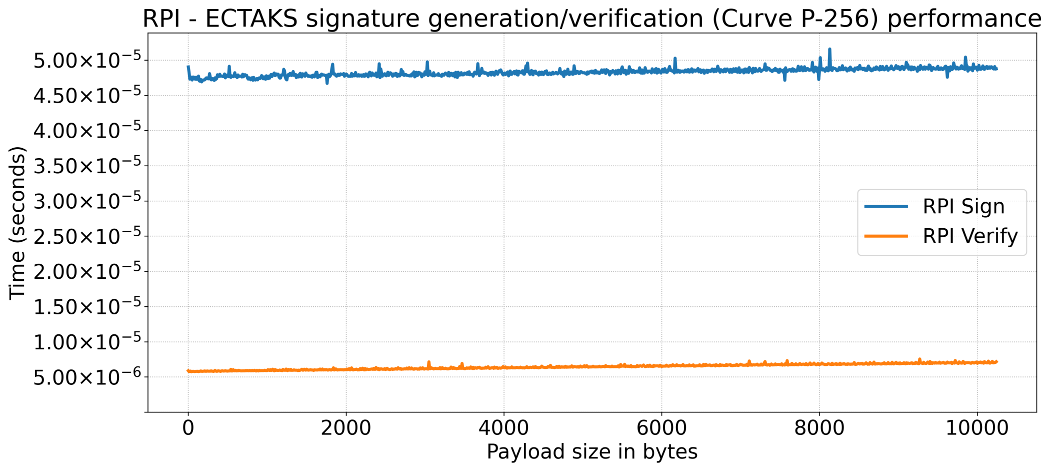

Figure 8. Crypto-Engine signature generation/verification performance on Raspberry PI model 3B+: time (y-axis, in seconds) versus payload size (x-axis, in bytes)

The NUC platform exhibits better performances (Figure 9), although with some instability on payloads smaller than 512 bytes. The signature generation time, at worst, is smaller than 20 µs and starts to stabilize to around 6.5 µs for payloads larger than 512 bytes. Once such a threshold is reached, the effect of the growing payload size is negligible. The signature verification on the NUC is very fast, with sub-microsecond time and negligible effect by the payload size.

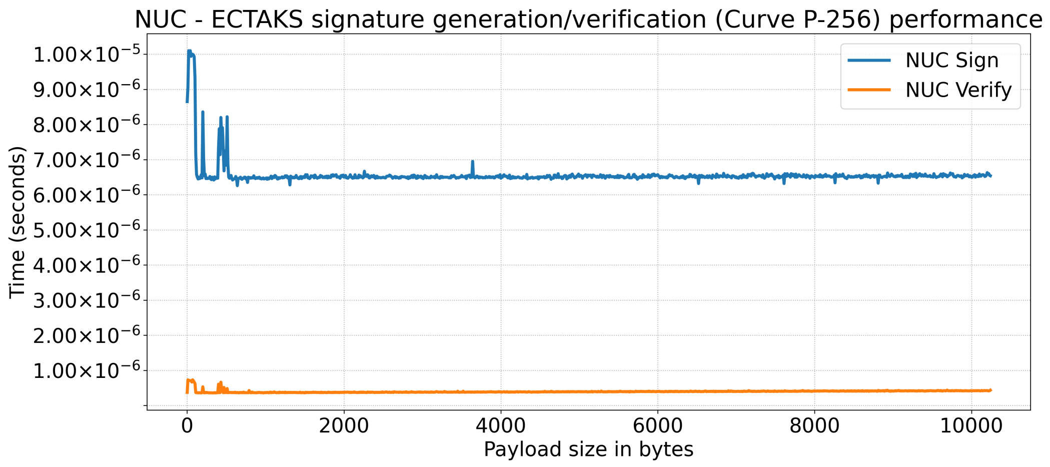

Figure 9. Crypto-Engine signature generation/verification performance on Intel NUC 11TNHi3: time (y-axis, in seconds) versus payload size (x-axis, in bytes)

4.3. Discussion of Results

The Key Generation in ECTAKS is noticeably slower than in similar research works (e.g., [69]), and it depends on the complexity of the topology of the network. On the other hand, the performance for encryption is in line with or better in comparison to the state of the art solutions, while providing the additional benefits of the topology-authenticated keys. In fact, when comparing the Crypto-Engine encryption and signature generation for a single message against the results shown in [70], our solution is faster: 27.14 ms (Table 2 in [70]) versus the sum of the 0.744 ms required by the Crypto-Engine to encrypt, e.g., 1500 bytes of data (theoretical maximum size for an Ethernet frame) and the 6.5 µs required to create a digital signature. These results make Crypto-Engine an appealing alternative when the Key component distribution performance is not an issue. This may be the case when the ANT is either simple (i.e., few communication links) or when it can be fixed at deployment time. In comparison with HW and HW-accelerated solutions, the performance of a pure SW implementation of the Crypto-Engine is trivially disadvantaged. Moreover, it is difficult to provide an accurate comparison since ECC HW accelerator performance is often measured in terms of the single ECC point-operation performance, maximum achievable clock frequency, and chip area occupation [63] leaving out the performance bottlenecks that exist due to the communication bus and the device drivers responsible for providing the accelerator results to the upper layer of the communication stack.

5. Conclusions

In this paper we proposed a security solution for intra-vehicle networks, the *Crypto-Engine*: it provides hybrid-key encryption and source authentication in point-to-point and cluster-wise communications with low impact on communication latency. The Crypto-Engine is based on the ECTAKS scheme, which uses standard-approved elliptic curve cryptography and a key distribution mechanisms to define a logical topology of nodes and the admissible communication links. The Crypto-Engine has been implemented to be used in an industrial research and innovation project, EMERGE, to secure the communication links related to non-GNSS sensor data sources. The experimental results show that the Crypto-Engine implementation described in this paper can achieve good performance results with ECTAKS encryption and authentication when deployed in general-purpose embedded computational platforms such as the tested Raspberry PI 3B+ and the Intel NUC 11 Pro. The deployment on application-specific platforms is currently in progress, for instance, on the AURIX Tri-core platforms (AURIX Tri-core product page: https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/ (accessed on 27 November 2023)) and the NXP S23 platforms (NXP S23 automotive platforms: https://ww w.nxp.com/products/processors-and-microcontrollers/s32-automotive-platform:S32 (accessed on 27 November 2023)).

Those results demonstrate that the adoption of the Crypto-Engine concept, along with ECTAKS could bring significant advantages in securing intra-vehicle networks thanks to its flexible architecture, the definition and enforcement of an authenticated topology (i.e., the ANT), and the support of state-of-the-art elliptic curve cryptography with a negligible overhead on the latency introduced by encryption and authentication primitives. Moreover, the support for hardware accelerators for cryptographic operations can help to further improve performance and to extend the support to novel solutions in terms of on-chip/off-chip cryptographic acceleration.

Despite the results obtained, a pure software implementation is only a starting point. We are currently working on a FPGA-based implementation for the Crypto-Engine. In particular, we aim to integrate an on-going work on a ECC HW accelerator into the Crypto-Engine to provide faster modular arithmetic operations on large numbers, improved and secured ECC point-based operations, random number generation and secure hashing primitives. At the same time, we are considering GPU-based acceleration for a comparative performance analysis between SW, HW, and GPU versions of the Crypto-Engine.

For future works, an additional aspect we aim to consider is *energy consumption*. In the experimentation results, it is evident that the NUC platform performs better, mainly because of its better hardware components. On the other hand, the better performance comes with a price in terms of energy consumption: the power required by the NUC can be much higher (e.g., the i3 CPU can consume up to 28 W) than the energy/power required by the RPI that hosts a 7 W System-on-Chip as MCU. The power gap further increases when considering the power required by fully loaded (including peripherals) platforms: the NUC is designed for a maximum of 65 W of power, while the RPI can be fully powered with $\sim$15 W. The comparison and trade-offs of performance and energy consumption in this context is a hot topic and requires a more in-depth analysis. We are planning to perform a comparative performance/energy analysis between (a) different Crypto-Engine configurations and (b) different target platforms, including GPU-based solutions and (c) with or without ECC hardware acceleration.

Author Contributions: Conceptualization, all authors; methodology, all authors; software, W.T.; validation, all authors; formal analysis, R.C. and N.G.; investigation, all authors; resources, all authors; data curation, all authors; writing—original draft preparation, W.T.; writing—review and editing, all authors; visualization, all authors; supervision, F.S. and N.G.; project administration, F.S. and N.G.; funding acquisition, F.S. and N.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly funded by the project "Centre of Excellence on Connected, Geo-Localized and Cyber-secure Vehicles"—Italian Government (Grant Number: CIPE Resolution 70/2017) and the project EMERGE-Navigation (Ministry of Innovation and Made in Italy and Regione Abruzzo).

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: The data presented in this study are openly available in GitHub at [68].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AES	Advanced Encryption Standard
ANT	Authenticated Network Topology
CAN	Controller Area Network (protocol)
CAN-FD	Controller Area Network—Full Datarate
CAN-XL	Controller Area Network—eXtended Length
CBC	Cipher Block chaining—Mode of operation for block ciphers
CBC-MAC	CBC-based MAC
CCM	CTR with CBC-MAC
CDHP	Computational Diffie-Hellman Problem
CTR	Counter Mode—Mode of operation for block ciphers
DDHP	Decisional Diffie-Hellman Problem
EAP	Extensible Authentication Protocol
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ECTAK	Elliptic Curve Topology Authenticated Key
ECTAKS	Elliptic Curve Topology Authenticated Key Scheme
ECU	Electronic Control Unit
EMERGE	EMERGE Project—*Veicoli Commerciali Leggeri* & Tecnologie EMERGEnti per operatività di tutti i giorni e di ausilio nelle EMERGEnze
GNSS	Global Navigation Satellite System
GPU	Graphical Processing Unit
HMAC	Hash-based MAC
HSM	Hardware Security Module
IDS	Intrusion Detection System
IRS	Intrusion Response System
KDF	Key Derivation Function
LIN	Local Interconnect Network (protocol)
LKC	ECTAKS Local Key Component
MAC	Message Authentication Code
MEC	Mobile Edge Computing
OBU	On-board Unit
PBKDF2	Public-Key KDF version 2
SBC	Single-Board Computer
SHA256	Secure Hash Algorithm version 2 with 256 bits hash size
SIEM	Security Information and Event Management
TKC	ECTAKS Transport Key Component
TLS	Transport Layer Security
TV	ECTAKS Topology Vector
TPM	Trusted Platform Module
USB	Universal Serial Bus (protocol family)
V2I	Vehicle to Infrastructure communications
V2V	Vehicle to Vehicle communications
V2X	Vehicle to Everything communications

References

1. Khan, S.K.; Shiwakoti, N.; Stasinopoulos, P.; Chen, Y. Cyber-attacks in the next-generation cars, mitigation techniques, anticipated readiness and future directions. *Accid. Anal. Prev.* **2020**, *148*, 105837. [CrossRef]
2. Giannaros, A.; Karras, A.; Theodorakopoulos, L.; Karras, C.; Kranias, P.; Schizas, N.; Kalogeratos, G.; Tsolis, D. Autonomous Vehicles: Sophisticated Attacks, Safety Issues, Challenges, Open Topics, Blockchain, and Future Directions. *J. Cybersecur. Priv.* **2023**, *3*, 493–543. [CrossRef]

3. Jeong, S.; Ryu, M.; Kang, H.; Kim, H.K. Infotainment System Matters: Understanding the Impact and Implications of In-Vehicle Infotainment System Hacking with Automotive Grade Linux. In Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy, Charlotte, NC, USA, 24–26 April 2023; Association for Computing Machinery: New York, NY, USA, 2023; CODASPY'23; pp. 201–212. [CrossRef]
4. Singh, N.; Ganesan, V.; Rebeiro, C. Secure Processor Architectures. In *Handbook of Computer Architecture*; Springer Nature: Singapore, 2022; pp. 1–29. [CrossRef]
5. Chiocchio, S.; Cinque, E.; Persia, A.; Salvatori, P.; Stallo, C.; Salvitti, M.; Valentini, F.; Pratesi, M.; Rispoli, F.; Neri, A.; et al. A Comprehensive Framework for Next Generation of Cooperative ITSs. In Proceedings of the 2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI), Palermo, Italy, 10–13 September 2018; pp. 1–6. [CrossRef]
6. Di Sciullo, G.; Zitella, L.; Cinque, E.; Santucci, F.; Pratesi, M.; Valentini, F. Experimental Validation of C-V2X Mode 4 Sidelink PC5 Interface for Vehicular Communications. In Proceedings of the 2022 61st FITCE International Congress Future Telecommunications: Infrastructure and Sustainability (FITCE), Rome, Italy, 29–30 September 2022; pp. 1–6. [CrossRef]
7. Cinque, E.; Valentini, F.; Persia, A.; Chiocchio, S.; Santucci, F.; Pratesi, M. V2X Communication Technologies and Service Requirements for Connected and Autonomous Driving. In Proceedings of the 2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), Turin, Italy, 18–20 November 2020; pp. 1–6. [CrossRef]
8. *ISO 21434:2021*; Road vehicles—Cybersecurity Engineering. International Organization for Standardization. 2021. Available online: https://www.iso.org/standard/70918.html (accessed on 27 November 2023).
9. Tuohy, S.; Glavin, M.; Hughes, C.; Jones, E.; Trivedi, M.; Kilmartin, L. Intra-Vehicle Networks: A Review. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 534–545. [CrossRef]
10. Chen, H.; Liu, J.; Wang, J.; Xun, Y. Towards secure intra-vehicle communications in 5G advanced and beyond: Vulnerabilities, attacks and countermeasures. *Veh. Commun.* **2023**, *39*, 100548. [CrossRef]
11. Rathore, R.S.; Hewage, C.; Kaiwartya, O.; Lloret, J. In-Vehicle Communication Cyber Security: Challenges and Solutions. *Sensors* **2022**, *22*, 6679. [CrossRef] [PubMed]
12. Fröschle, S.; Stühring, A. Analyzing the Capabilities of the CAN Attacker. In Proceedings of the Computer Security—ESORICS 2017, Oslo, Norway, 11–15 September 2017; pp. 464–482. [CrossRef]
13. Sahana, Y.P.; Gotkhindikar, A.; Tiwari, S.K. Survey on CAN-Bus Packet Filtering Firewall. In Proceedings of the 2022 International Conference on Edge Computing and Applications (ICECAA), Tamilnadu, India, 13–15 October 2022; pp. 472–478. [CrossRef]
14. Park, S.B.; Jo, H.J.; Lee, D.H. Flooding attack mitigator for in-vehicle CAN using fault confinement in CAN protocol. *Comput. Secur.* **2023**, *126*, 103091. [CrossRef]
15. Murvay, P.S.; Groza, B. Practical Security Exploits of the FlexRay In-Vehicle Communication Protocol. In *Proceedings of the Risks and Security of Internet and Systems*; Zemmari, A., Mosbah, M., Cuppens-Boulahia, N., Cuppens, F., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 172–187.
16. Kishikawa, T.; Hirano, R.; Ujiie, Y.; Haga, T.; Matsushima, H.; Fujimura, K.; Anzai, J. Vulnerability of FlexRay and Countermeasures. *SAE Int. J. Transp. Cybersecur. Priv.* **2019**, *2*, 21–33. [CrossRef]
17. Lee, T.Y.; Lin, I.A.; Liao, R.H. Design of a FlexRay/Ethernet Gateway and Security Mechanism for In-Vehicle Networks. *Sensors* **2020**, *20*, 641. [CrossRef]
18. Püllen, D.; Anagnostopoulos, N.A.; Arul, T.; Katzenbeisser, S. Security and Safety Co-Engineering of the FlexRay Bus in Vehicular Networks. In Proceedings of the International Conference on Omni-Layer Intelligent Systems, Crete, Greece, 5–7 May 2019; Association for Computing Machinery: New York, NY, USA, 2019; COINS'19; pp. 31–37. [CrossRef]
19. Murvay, P.S.; Groza, B. Efficient Physical Layer Key Agreement for FlexRay Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9767–9780. [CrossRef]
20. *ISO 17987-1:2016*; Road vehicles—Local Interconnect Network (LIN)—Part 1: General Information and Use Case Definition. International Organization for Standardization. 2016. Available online: https://www.iso.org/standard/61222.html (accessed on 27 November 2023).
21. Páez, F.; Kaschel, H. Design and Testing of a Computer Security Layer for the LIN Bus. *Sensors* **2022**, *22*, 6901. [CrossRef]
22. Ernst, J.M.; Michaels, A.J. LIN Bus Security Analysis. In Proceedings of the IECON 2018—44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, 21–23 October 2018; pp. 2085–2090. [CrossRef]
23. Paez, F.; Kaschel, H. Towards a Robust Computer Security Layer for the LIN Bus. In Proceedings of the 2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA), Valparaiso, Chile, 22–26 March 2021; pp. 1–8. [CrossRef]
24. Ju, H.; Jeon, B.; Kim, D.; Jung, B.; Jung, K. Security Considerations for In-Vehicle Secure Communication. In Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 16–18 October 2019; pp. 1404–1406. [CrossRef]
25. Luo, F.; Hou, S. Security Mechanisms Design of Automotive Gateway Firewall. In *Proceedings of the WCX SAE World Congress Experience*; SAE International: Warrendale, PE, USA, 2019. [CrossRef]
26. Karopoulos, G.; Kambourakis, G.; Chatzoglou, E.; Hernández-Ramos, J.L.; Kouliaridis, V. Demystifying In-Vehicle Intrusion Detection Systems: A Survey of Surveys and a Meta-Taxonomy. *Electronics* **2022**, *11*, 1072. [CrossRef]
27. Wu, W.; Li, R.; Xie, G.; An, J.; Bai, Y.; Zhou, J.; Li, K. A Survey of Intrusion Detection for In-Vehicle Networks. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 919–933. [CrossRef]

28. Xun, Y.; Deng, Z.; Liu, J.; Zhao, Y. Side Channel Analysis: A Novel Intrusion Detection System Based on Vehicle Voltage Signals. *IEEE Trans. Veh. Technol.* **2023**, *72*, 1–10. [CrossRef]
29. Basavaraj, D.; Tayeb, S. Towards a Lightweight Intrusion Detection Framework for In-Vehicle Networks. *J. Sens. Actuator Netw.* **2022**, *11*, 6. [CrossRef]
30. Khan, J.; Lim, D.W.; Kim, Y.S. Intrusion Detection System CAN-Bus In-Vehicle Networks Based on the Statistical Characteristics of Attacks. *Sensors* **2023**, *23*, 3554. [CrossRef] [PubMed]
31. Li, S.; Han, M. Priv-IDS: A Privacy Protection and Intrusion Detection Framework for In-Vehicle Network. In *Proceedings of the Machine Learning for Cyber Security*; Xu, Y., Yan, H., Teng, H., Cai, J., Li, J., Eds.; Springer Nature Switzerland: Cham, Switzerland, 2023; pp. 165–179.
32. Islam, R.; Refat, R.U.D.; Yerram, S.M.; Malik, H. Graph-based intrusion detection system for controller area networks. *IEEE Trans. Intell. Transp. Syst.* **2020**, *23*, 1727–1736. [CrossRef]
33. Kwak, B.I.; Han, M.L.; Kim, H.K. Cosine similarity based anomaly detection methodology for the CAN bus. *Expert Syst. Appl.* **2021**, *166*, 114066. [CrossRef]
34. Hamad, M.; Tsantekidis, M.; Prevelakis, V. Red-Zone: Towards an Intrusion Response Framework for Intra-Vehicle System. In Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS), Crete, Greece, 3–5 May 2019. [CrossRef]
35. Barletta, V.S.; Caivano, D.; Vincentiis, M.D.; Ragone, A.; Scalera, M.; Martìn, M.Á.S. V-SOC4AS: A Vehicle-SOC for Improving Automotive Security. *Algorithms* **2023**, *16*, 112. [CrossRef]
36. Bari, B.S.; Yelamarthi, K.; Ghafoor, S. Intrusion Detection in Vehicle Controller Area Network (CAN) Bus Using Machine Learning: A Comparative Performance Study. *Sensors* **2023**, *23*, 3610. [CrossRef]
37. Agrawal, K.; Alladi, T.; Agrawal, A.; Chamola, V.; Benslimane, A. NovelADS: A Novel Anomaly Detection System for Intra-Vehicular Networks. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 22596–22606. [CrossRef]
38. Cheng, P.; Xu, K.; Li, S.; Han, M. TCAN-IDS: Intrusion Detection System for Internet of Vehicle Using Temporal Convolutional Attention Network. *Symmetry* **2022**, *14*, 310. [CrossRef]
39. ETSI, TS 102 731—Intelligent Transport Systems (ITS); Security; Security Services and Architecture. Available online: https://www.etsi.org/deliver/etsi_ts/102700_102799/102731/01.01.01_60/ts_102731v010101p.pdf (accessed on 27 November 2023).
40. 3GPP. Technical Specification N. 33501. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169 (accessed on 27 November 2023).
41. *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)*; IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages. IEEE: Piscataway, NJ, USA, 2016; pp. 1–240. [CrossRef]
42. Rezazadeh Baee, M.A.; Simpson, L.; Boyen, X.; Foo, E.; Pieprzyk, J. Authentication Strategies in Vehicular Communications: A Taxonomy and Framework. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 129. [CrossRef]
43. Palaniswamy, B.; Camtepe, S.; Foo, E.; Pieprzyk, J. An Efficient Authentication Scheme for Intra-Vehicular Controller Area Network. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3107–3122. [CrossRef]
44. de Andrade, R.; Santos, M.M.D.; Justo, J.F.; Yoshioka, L.R.; Hof, H.J.; Kleinschmidt, J.H. Security architecture for automotive communication networks with CAN FD. *Comput. Secur.* **2023**, *129*, 103203. [CrossRef]
45. Wu, W.; Dai, J.; Huang, H.; Zhao, Q.; Zeng, G.; Li, R. A Digital Watermark Method for In-Vehicle Network Security Enhancement. *IEEE Trans. Veh. Technol.* **2023**, *72*, 1–12. [CrossRef]
46. Park, C.H.; Kim, Y.; Jo, J.Y. A Secure Communication Method for CANBus. In Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), Cookeville, NV, USA, 27–30 January 2021; pp. 773–778. [CrossRef]
47. Civino, R.; Longo, R. Formal security proof for a scheme on a topological network. *Adv. Math. Commun.* **2023**, *17*, 562–571. [CrossRef]
48. Carvajal-Roca, I.E.; Wang, J.; Du, J.; Wei, S. A Semi-Centralized Dynamic Key Management Framework for In-Vehicle Networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 10864–10879. [CrossRef]
49. Khatri, N.; Shrestha, R.; Nam, S.Y. Security Issues with In-Vehicle Networks, and Enhanced Countermeasures Based on Blockchain. *Electronics* **2021**, *10*, 893. [CrossRef]
50. Aliyu, I.; Van Engelenburg, S.; Mu'Azu, M.B.; Kim, J.; Lim, C.G. Statistical Detection of Adversarial Examples in Blockchain-Based Federated Forest In-Vehicle Network Intrusion Detection Systems. *IEEE Access* **2022**, *10*, 109366–109384. [CrossRef]
51. Baldanzi, L.; Crocetti, L.; Bertolucci, M.; Fanucci, L. Analysis of Cybersecurity Weakness in Automotive In-Vehicle Networking and Hardware Accelerators for Real-Time Cryptography. In *Applications in Electronics Pervading Industry, Environment and Society*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 11–18. [CrossRef]
52. Trusted Computing Group (TCG) Homepage. 2023. Available online: https://trustedcomputinggroup.org (accessed on 27 November 2023).
53. Aragona, R.; Civino, R.; Gavioli, N.; Pugliese, M. An authenticated key scheme over elliptic curves for topological networks. *J. Discret. Math. Sci. Cryptogr.* **2022**, *25*, 2429–2448. [CrossRef]
54. Certicom Research, SEC 1: Elliptic Curve Cryptography. 2009. Available online: https://www.secg.org/sec1-v2.pdf (accessed on 27 November 2023).

55. Certicom Research, SEC 2: Recommended Elliptic Curve Domain Parameters. 2010. Available online: https://www.secg.org/sec2-v2.pdf (accessed on 27 November 2023).
56. Certicom Research, Standards for Efficient Cryptography SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV). 2013. Available online: https://www.secg.org/sec4-1.0.pdf (accessed on 27 November 2023).
57. Langley, A.; Hamburg, M.; Turner, S. Elliptic Curves for Security. RFC 7748. 2016. Available online: https://www.rfc-editor.org/info/rfc7748 (accessed on 27 November 2023).
58. NIST, FIPS 197. Advanced Encryption Standard (AES). 2023. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf (accessed on 27 November 2023).
59. NIST, FIPS 186-5. Digital Signature Standard (DSS). 2023. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf (accessed on 27 November 2023).
60. NIST, FIPS 180-4. Secure Hash Standard (SHS). 2015. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf (accessed on 27 November 2023).
61. Rescorla, E. HTTP Over TLS. RFC 2818. 2000. Available online: https://www.rfc-editor.org/info/rfc2818 (accessed on 27 November 2023).
62. Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. 2018. Available online: https://www.rfc-editor.org/info/rfc8446 (accessed on 27 November 2023).
63. Nannipieri, P.; Crocetti, L.; Matteo, S.D.; Fanucci, L.; Saponara, S. Hardware Design of an Advanced-Feature Cryptographic Tile within the European Processor Initiative. *IEEE Trans. Comput.* **2023**, 1–14. [CrossRef]
64. SafeCurves: Choosing Safe Curves for Elliptic-Curve Cryptography. Available online: https://safecurves.cr.yp.to (accessed on 7 March 2022).
65. Vollbrecht, J.; Carlson, J.D.; Blunk, L.; Aboba, D.B.D.; Levkowetz, H. Extensible Authentication Protocol (EAP). RFC 3748. 2004. Available online: https://www.rfc-editor.org/info/rfc3748 (accessed on 27 November 2023).
66. OpenSSL: Cryptography and SSL/TLS Toolkit. Available online: https://www.openssl.org (accessed on 27 November 2023).
67. Free Software Foundation (FSF). The GNU Multi-Precision Bignum Library. 2023. Available online: https://gmplib.org (accessed on 27 November 2023).
68. Tiberti, W. Crypto-Engine Performance Measurements. 2023. Available online: https://github.com/wtiberti/Crypto-Engine_ECTAKS_performance (accessed on 27 November 2023).
69. Ranganatha Rao, B.; Sujatha, B. A hybrid elliptic curve cryptography (HECC) technique for fast encryption of data for public cloud security. *Meas. Sensors* **2023**, *29*, 100870. [CrossRef]
70. Shao, H.; Piao, C. A Provably Secure Lightweight Authentication Based on Elliptic Curve Signcryption for Vehicle-to-Vehicle Communication in VANETs. *IEEE Trans. Ind. Inform.* **2023**, 1–10. [CrossRef]