electronics

MDPI

Article

Enhancing Privacy in the Internet of Vehicles via Hyperelliptic Curve Cryptography

George Routis [1,2], Panagiotis Dagas [1] and Ioanna Roussaki [1,2,*]

[1] School of Electrical and Computer Engineering, National Technical University of Athens, 15773 Athens, Greece
[2] Institute of Communication and Computer Systems, 10682 Athens, Greece
* Correspondence: ioanna.roussaki@cn.ntua.gr

Abstract: The Internet of Things (IoT) is a technological paradigm that has gained significant momentum the last decade and, among other features, enables the development of intelligent and interoperable device networks. In this respect, it has triggered the creation and evolution of vehicular ad-hoc networks (VANETs), which are initially implemented in order to guarantee the safety of drivers and the avoidance of traffic accidents. The drawback is that this fast evolution comes with serious concerns in terms of the privacy of users, while the population of attackers or entities that try to eavesdrop and intercept information has significantly increased. This imposes a serious risk for drivers moving across a Smart City. The research presented in this paper aims to evaluate privacy protection mechanisms in VANET environments, based on the efficiency and security level they ensure, considering the fact that VANETs provide limited resources to users/drivers. Moreover, the usage of elliptic curve cryptography in reduced resources environments is discussed. Finally, this paper compares the performance of three cryptographic algorithms, elliptic curve cryptography (ECC), hyperelliptic curve cryptography genus 2 (HECC-2) and HECC genus 3 (HECC-3), employed for an efficient authentication and safe message transmission mechanism in VANETs, aimed at reaching conclusions related to the implementation of each cryptographic scheme in this specific application area. The evaluation results indicate that ECC supersedes HECC-2 and HECC-3 in most metrics. However, HECC-2 and HECC-3 demonstrate better responses than ECC does in selected energy metrics. Overall, it is observed that HECC algorithms are not yet mature enough to compete with ECC. This is due to the fact that the research community has not sufficiently progressed toward the optimization of HECC, and moreover, HECC builds on quite complex mathematics. There are indications, however, that once HECC curves are indeed optimized, HECC will outperform ECC in speed as well as in other metrics, sinceHECC-2 and HECC-3 use a significantly smaller key size with the same level of security as that of ECC.

Keywords: Internet of Things (IoT); Internet of Vehicles (IoV); vehicular ad hoc networks (VANETs); vehicle privacy; digital certificates; digital signatures; ECC; HECC; energy consumption; message size; Arduino; Zigbee

check for updates

Citation: Routis, G.; Dagas, P.; Roussaki, I. Enhancing Privacy in the Internet of Vehicles via Hyperelliptic Curve Cryptography. *Electronics* **2024**, *13*, 730. https://doi.org/10.3390/electronics13040730

Academic Editor: Bahman Javadi

Received: 7 January 2024
Revised: 29 January 2024
Accepted: 7 February 2024
Published: 11 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Internet of Things (IoT) is a state-of-the-art paradigm that represents a set of networked hardware and software elements that connect and exchange data with other elements and systems over the Internet or other communications networks. A reflection of the IoT paradigm in the transportation domain is the Internet of Vehicles (IoV), which allows vehicles and the respective road infrastructure to become interconnected and exchange data. The technologies that enable communication among vehicles (V2V) and between vehicles and road infrastructures (V2I) are becoming quite popular, and they introduce several research challenges. The IoV aims, among other objectives, to enhance automation, efficiency and comfort in the transportation domain, and more importantly to strengthen

the protection of drivers' safety and avoid car accidents, which can often be fatal. At the same time, the evolution of 5G and 6G technology, which support very-high-throughput data rates, greatly facilitates the market penetration of the Internet of Vehicles.

However, the rapid evolution of IoV introduces high privacy-related risks, while the fact that the life of drivers and passengers may be jeopardized in cases of successful malicious attacks of V2V or V2I communications imposes the need for extended security constraints to be addressed. Thus, in the past few years, a lot of research initiatives have focused on building a safer IoV communication environment, which has to be adapted to the limited environment that the engaged machine provides while communicating. Therefore, the security-related solutions in such environments need to be efficient, fast and limit data traffic wherever possible.

The research presented in this paper elaborates on privacy preservation mechanisms for VANETs, using a simulated environment to deliver detailed evaluations. More details on IoV privacy matters, related research challenges and state-of-the-art approaches are provided in [1–4].

This paper focuses on evaluating the usage of hyperelliptic curve cryptography (HECC) on IoT devices to strengthen the protection of privacy. The approach is exploited and tested in Internet of Vehicles environments, where HECC features such as increased security with a small key, as well as speed in encryption/decryption, are of prime importance. To evaluate the proposed approach, series of experiments were executed using the ns-3 simulator, and very interesting results were obtained regarding privacy preservation in the IoV. Several evaluation metrics were used in this respect, to allow for a thorough identification of the advantages and disadvantages of using HECC in terms of IoV privacy protection.

Last but not least, this paper presents a system implemented to secure messages exchanged over wireless network, more specifically via the use of Xbee Zigbee modules, using the cryptographic protocols aforementioned. The goal was to build, test and evaluate a system that uses commercial modules such as the open-source Arduino module and a few electronics, such as a voltage translator and a wireless Zigbee module, which enables users to exchange messages over the ISM (industrial scientific medical) band.

The rest of the paper is constructed as follows. Section 2 discusses the related state-of-the-art literature regarding privacy protection in IoV environments. Section 3 elaborates on the related mathematical background regarding ECC, HECC genus 2 and HECC genus 3. Section 4 presents the approach proposed by this paper, which aims to address specific elements of the IoV privacy protection problem. Section 5 presents the experimental evaluation findings obtained using various asymmetric and symmetric cryptographic protocols. Section 6 presents a system for encrypting and decrypting messages via ECC and AES using Arduino, a solution that is suitable for IoV. Finally, Section 7 concludes the paper and presents future plans toward the protection of privacy in IoV environments.

2. Related State of the Art

This section aims to provide a thorough review of state-of-the-art work focusing on privacy in IoV environments. The section is structured as follows: the first subsection elaborates on authentication schemes in IoV, the second subsection focuses on secure message exchange in IoV, the third subsection presents secure schemes that distribute the computational load on the network and finally, the fourth subsection discusses blockchain schemes used for privacy preservation in IoV.

2.1. Authentication Schemes

With the continuous evolution of the IoT and therefore the IoV, many schemes have been proposed to ensure secure and efficient communication in VANETs. Some of them are based solely on the authentication of users; others also address the processing and transmission of messages, while others try to improve the performance of the authentication of users and distribution of messages in the network. Firstly, schemes related to authentication are presented.

The EAAP scheme proposed by Azees, Vijayakumar and Deborah [5] aims to provide efficient anonymous authentication for vehicles and road side units (RSUs) in VANETs. It utilizes the bilinear pairing cryptographic technique and distinguishes between authentication processes for vehicles and those for RSUs. The trusted authority (TA) provides parameters for certificate and signature generation to ensure secure authentication. Vehicles need to be registered with the TA and receive a "dummy" identifier (DID) that can be linked to their real identity. When a vehicle enters the network, it creates an anonymous certificate, signs it and shares it with the necessary parameters. RSUs, on the other hand, receive their identifiers from the TA, are registered with the system, and generate anonymous certificates using a temporary secret key. This scheme is designed to be secure against authentication attacks and ensures location privacy, with the ability to detect and block malicious users from the network.

In the same year, Yanbing Liu et al. developed a dual-authentication scheme called PPDAS [6], which employs encryption methods based on vehicle IDs and public key encryption. The scheme introduces the concept of vehicle reputation, where vehicles registered with the trusted authority communicate through the RSU. The vehicles generate pseudonyms that can only be deciphered by the RSU and the TA using bilinear pairing properties. Time stamps are used to prevent replay attacks, and the message authentication code (MAC) technique is implemented to avoid message-tampering attacks. The authentication process consists of two stages. Initially, vehicles send their pseudonyms, public keys, and time stamps to the TA via the RSU. The TA validates the time stamp and compares the calculated MAC code to the received code. Once successful, the first stage of validation is completed. The reputation of the vehicles is then checked based on confidence matrices and the average confidence level of the communicating pair. The second stage of authentication depends on the trust level of the vehicles. After communication is finished, vehicles evaluate each other and update the trust tables with the TA through the RSU.

In 2018, Hyo Jin Jo et al. [7] introduced a co-operative authentication scheme that involves cooperation between vehicles and RSUs for secure message transmission. The scheme utilizes binary key trees for key management and authentication. Vehicles periodically validate received messages, authenticate themselves, and share the results with nearby vehicles, which in turn validate their results. This process allows for the confirmation of the messages' authenticity within the network, while the RSU is responsible for revoking malicious or invalid users by disclosing group keys only to valid users and informing them about revoked vehicles through the revocation list. This scheme improves the efficiency of recalling and sharing public keys in a vehicle network, as well as detecting malicious users. However, a drawback of these schemes is the increased network traffic and time delay caused by vehicles communicating with each other, the RSU and a trusted authority (TA-CA) in certain cases.

In [8], the authors propose a trust scheme for privacy protection via the use of both blockchain and multi-party evaluation, better known as TMPP-BMPE. Initially, broadcasting of data is proposed based on the Paillier encryption mechanism as well as the ECDSA (elliptic curve digital signature algorithm) in order to provide data protection. They apply both homographic encryption and the ECDSA in order to offer data privacy protection during the transmission phase. They propose a trust scheme based on multi-party evaluation. The reliability of cooperative partners (CP) is completely evaluated considering estimation indexes coming from various entities, reducing the dangers of interacting with unsafe CPs. Finally, a mechanism of blockchain-assisted trust management is constructed in order to block malicious tampering with trusted data. As the researchers claim, TMPP-BMPE operates quite well in order to protect data privacy and assesses the reliability of CPs while at the same time preventing data tampering. Moreover, their proposed scheme offers precious insights for security and trust in IoV environments.

In [9], the authors propose a lightweight mechanism for privacy-preserving authentication, which reduces dependency on trusted authorities (TA), targeted at achieving both protection in terms of privacy and authentication between the various nodes (vehicles) and TA in VANETs environments by enhancing the "forgetful" transfer algorithm. Security analysis makes it more than clear that their mechanism improves privacy and is resistant to attacks. Moreover, the performance simulation and assessment indicate that their solution uses mitigated computation resources as well as low communication overhead, and at the same time guarantees low authentication delay and a low packet loss rate. These show that this scheme has much better overall performance in comparison with that of other schemes proposed nowadays.

2.2. Secure Message Dissemination

In 2021, Hassan Mistareehi et al. [10] proposed an architecture for secure messaging in vehicular cloud networks. The proposed architecture for secure messaging in vehicular cloud networks aims to enhance authentication and message propagation performance in highly dense areas. This scheme utilizes both symmetric and asymmetric encryption for key and message transfer. If a vehicle is out of the range of a RSU, its message/request is transmitted to the RSU through nearby vehicles using the AODV algorithm. When a vehicle is registered in a regional cloud, it receives a key pair and the RSU's public key. Then, messages are authenticated using digital signatures and encrypted in an asymmetric manner, so that the RSU can receive and authenticate sensitive information from the vehicles. The RSU can group multiple messages and send them in large packets to the local cloud to avoid network congestion. Also, pseudonyms are used to maintain the anonymity of vehicles and change when the number of vehicles in the area decreases using a technique that extends the logic of mix zones. This scheme offers security against various attacks but does not consider the computational load on the RSU that comes with increasing density.

In 2021, Mir Ali Rezazadeh Baee et al. proposed a scheme called ALI [11], which aims to provide message authentication and encryption in VANETs using elliptic curve cryptography. It utilizes the elliptic curve Qu–Vanstone (ECQV) scheme, which is based on elliptic curves, to propagate occasional keys of vehicles. For the exchange of symmetric encryption keys, the scheme employs ECIES, which utilizes the Diffie–Hellman mechanism and a hash-based key derivation function (HKDF). The trusted authority (TA) passes the keys to vehicles and RSUs during registration using the ECQV scheme and encrypts the messages with ECIES for faster communication. Keys are valid for a predefined period, and upon expiration, users must request new keys from the TA. Communication between vehicles and the TA is achieved through RSUs, with encrypted identities. RSUs transmit a symmetric key to vehicles for encryption, and to carry this out securely, the symmetric key is generated using the TA's public key. The scheme also requires a daily handshake between vehicles and the TA to receive this public key and ensure key freshness. The scheme utilizes elliptic curve cryptography and provides efficient authentication, identity privacy and network authentication. However, the authentication of messages is achieved by vehicles themselves, which may cause delays due to limited computing power.

2.3. Secure Schemes That Distribute the Computational Load on the Network

In 2018, Amit Dua et al. [12] tried to group the vehicle authentication process using cluster heads. Here, vehicles are grouped based on criteria such as location, speed and computing power, and a so-called cluster head (CH) is elected within each group. The trusted authority authenticates the CH and assigns it a group ID (GID). Once the CH is authenticated, it becomes responsible for authenticating other vehicles in its region. The vehicles in the cluster receive the GID from the CH that they are assigned to and authenticate themselves using it. The vehicles then exchange a symmetric session key to facilitate further communication. Authentication and key exchange are based on elliptic curve cryptography, using the ElGamal technique. This authentication scheme provides confidentiality, authentication and identity privacy, while being resistant to man-in-the-

middle attacks. It also reduces computation times and improves the size and number of required messages.

Lastly, in 2022, Hassan Mistareehi and D. Manivannan [13] proposed a scheme combining group authentication with secure message dissemination in vehicular cloud networks. The scheme uses RSUs to verify message authenticity and integrity before distribution and can delegate leadership roles to vehicles, reducing computational load on RSUs. Vehicles entering the RSU's area receive a certificate, validate it and establish communication with the RSU. RSUs can select a group leader (GL) to manage authentication and message grouping for efficiency, similarly to the approach of cluster heads presented by Amit Dua et al. [12]. GLs periodically send proof of leadership and their public key, and operate like RSUs in their region of influence. Vehicles authenticate and exchange symmetric keys with the GL or RSU, using a series of join and accept messages that are encrypted using asymmetric encryption. RSUs verify messages and handle data dissemination. The trusted authority maintains a revocation list and key updates, while pseudonyms are also used to ensure location privacy and anonymity. The scheme enhances security and efficiency in vehicular networks, using various cryptographic techniques to prevent attacks.

In [14], the authors propose a scheme for sharing data privately in VANETs via the use of federated learning (FL), with local differential privacy. As the first level, the various nodes (vehicles) implement local differential privacy methods in their data before they send them to the RSU. As the second level, there is a responsibility to train the parameters of the model at the RSU and renew the various trained weights with the server responsible for training. In order to evaluate the system's performance, the researchers assess it based on the accuracy and time taken for simulation for local as well as global parameters that were shared. Moreover, they measure every client's performance via the calculation of accuracy measurements that take place in each iteration. The results of the experiments show that their framework achieves the following: (a) ensuring security against inference and attacking inference and gradient leakage; (b) displaying advanced efficiency compared to that of its competitive solutions.

In [15], the authors propose the ALRS, which stands for anonymous and linkable ring signcryption scheme for location-based services (LBS) in VANETs. It offers data privacy to the service providers and assesses privacy in users' vehicles, while at the same time keeping the vehicle user's ID anonymous from the service provider. Moreover, it maintains linkability, something better explained as the number of times that the same user asks for a query. The latter occurs without exposing the ID of the vehicle user. Realization and detailed outcomes reveal that the proposed ALRS mechanism operated better than did other schemes as far as communication and computation costs were concerned by maintaining all the privacy requirements.

2.4. Blockchain Schemes to IoV Privacy Preservation

In [16], the researchers propose a mix of blockchain and crowd sensing in order to offer location privacy protection (BCS-LPP) in IoV environments. Initially, they introduce blockchain into BCS-LPP in order to prevent the leakage of users' information from third-party service environments. Secondly, this scheme provides workers with location privacy level options mixed with Geolash encoding. In order to achieve confidentiality in the privacy information of the workers' location, it provides order-preserving encryption. As a final step, there is a guarantee of the equity of workers' participation in various operations, fulfilled via the verification of sensing locations offered by the workers. The authors use real data and compare BCS-LPP with existing mechanisms via experiments on simulated environments. As they claim, BCS-LPP can guarantee the quality of sensed data, offer safety in terms of workers' location privacy data and improve the "justice" of the various users' participation in the tasks.

In [17], the authors propose a scheme for maintaining privacy in vehicular networks via the use of blockchain. More specifically, they designed an anonymous and auditable scheme for data sharing via the use of a zero-knowledge proof algorithm (ZKP) in order to guarantee the identity privacy of nodes. At the same time, they preserve the data auditability in vehicles for trusted authorities (TAs). The researchers designed an efficient multi-sharding scheme in order to mitigate the costs of blockchain communication without penetrating its security. They realized a prototype of their scheme and conducted many experiments and simulations with it. What they found based on the simulations conducted is that the mechanism strengthens system security and data privacy; moreover, it diminishes communication time complexity by employing $O\left(\frac{n\sqrt{m}}{m^2}\right)$ against other proposed sharding algorithms.

In [18], the researchers propose a scheme that they call dual-blockchain-based decentralized architecture for the authentication of nodes as well as secure and efficient communication inside the IoV network. As they claim, the use of individual blockchains for authentication and message sharing lets the network be more efficient and faster, and at the same time helps to raise the security of the network. Both blockchains use different approaches, thus adding more security. The security and performance assessment of their scheme show that it can withstand attacks with a smaller computation cost, increased throughput, mitigated delays in transmission and an increased vehicle verification rate in comparison with that of existing solutions.

3. Background on the Cryptographic Protocols Employed

In this section, the four cryptographic schemes employed in this paper are elaborated upon, including the advanced encryption standard (AES), elliptic curve cryptography (ECC), hyperelliptic curve cryptography genus 2 (HECC-2) and hyperelliptic curve cryptography genus 3 (HECC-3). This analysis is based on the discrete logarithm problem (DLP) that all three versions of HECC use in order to secure communication between two parties and investigates how the various points are selected in order to make the algorithms difficult to be penetrated, elaborating on the respective mathematic equations that support the latter.

3.1. AES Cryptographic Protocol

The AES algorithm was originally proposed by Joan Daemen and Vincent Rijmen. The algorithm was selected as the mainstream encryption algorithmic scheme (the advanced encryption standard) by the famous NIST (National Institute of Standards and Technology) in 2001 [19]. There are many characteristics of the AES scheme that distinguish it from similar symmetric cryptographic schemes. These are described below:

- The AES scheme is a block cipher that makes use of the same key either for the encryption or decryption period.
- The unencrypted text block size is 16 bytes (128 bits).
- The size of the key can be 128 bits, 192 bits or 256 bits.
- AES contains iterations, best known as rounds, related to the key sizes. These are the following: (i) 10 rounds for a 128-bit key size, (ii) 12 rounds for a 192-bit key size and (iii) 14 rounds for a 256-bit key size.
- AES contains four levels: (1) SubByte, (2) ShiftRow, (3) MixColumn and (4) AddRoundKey. These stages take place with a continuous logic and are realized as rounds on a 4×4 byte state array. The plaintext (the text to be encrypted) is fed in those rounds.

Below is an analysis of each level (stage) [19]:

1. The SubByte level is represents byte substitution in a non-linear rationale, which is realized on every byte of the state array through an independent logic.
2. The ShiftRow stage consists of four rows. The initial row is unshifted, while the rest of the three rows are moved in a circular logic over 1 to 3 bytes.

3. The MixColumns stage makes use of the column-by-column strategy for working the state. Every column is used as a polynomial consisting of four terms in a GF(28). Then, it undergoes multiplication with a fixed $a(x)$ modulo $x4 + 1$ polynomial.
4. The AddRoundKey part operates the states via the use of one of the sub-keys by realizing XOR computation. The latter is implemented between every byte of the sub-key and every byte of the state.

3.2. Elliptic Curve Cryptography

In 1985, mathematicians Koblitz and Miller separately proposed a group of points coming from an elliptic curve that was defined on a finite field in order to take advantage of the discrete logarithm problem in cryptography, since the points of the elliptic curve cannot exploit the property of the prime factors that the number field sieve uses. The best method of solving the discrete logarithm in the group is that of square root complexity, something that makes the group suitable for cryptographic techniques [20].

In order to use the curves for building the desirable group, initially, the elements of the group need to be defined as do the respective processes that will be used in order to define the group. The curve and by extension the points that define the group are given by the subsequent equation:

$$E: \ y^2 = x^3 + Ax + B, \ \text{where}: \ (A, \ B \ \epsilon \ \mathbb{K}) \tag{1}$$

Also, it should consider that the curve is non-singular, which means that $-\left(4A^3 + 27B^3\right)$ should not appear and that the finite field $\mathbb{K}$ should have a characteristic different from that of 2 and 3. Hence, the elements of the group consist of the points that define the curve, with the neutral element, to point to infinity.

At this point, the process of the group is defined. As is observed in the specific curves' form, every element of $P = (x_0, y_0)$, has an inverse point; that is, the second point of the line $x = x_0$ intersects the curve, so $P' = (x_0, -y_0')$.

In order to define the process of the group, two points on the curve are used. Let these be P and Q. The line passes through the two points intersects the curve at a third point, R. Therefore, the process, $P \oplus Q$, that outputs point R', which stands for the reverse of R, since P, Q and R are collinear, can be defined (Figures 1 and 2).

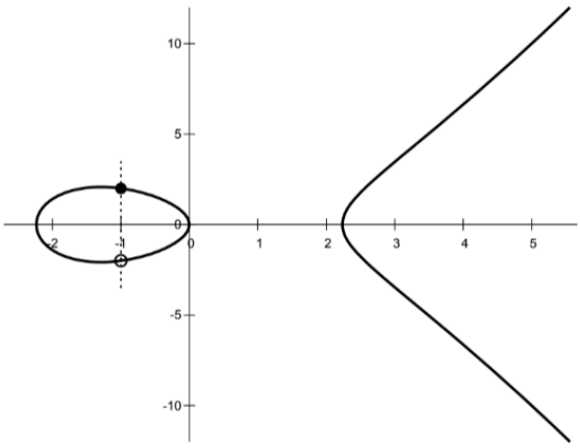


Figure 1. Elliptic curve's reverse point [20].

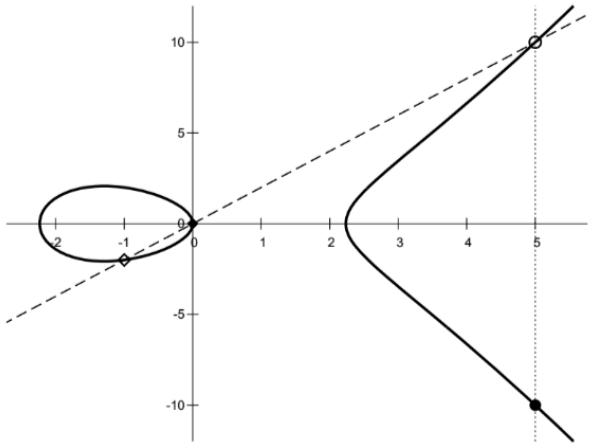


Figure 2. Group rule for the points of the elliptic curve [20].

In order to utilize the discrete logarithm, scalar multiplication is defined as follows:

$$P \longmapsto [n]P = P \bigoplus P \bigoplus P \ldots \bigoplus P \text{ for } n \text{ times} \quad (2)$$

Hence, the presence of the discrete logarithm problem, where P is the generator element and n is the power (exponent) of the generator is obvious, and allows the cyclic group to be built with the desirable characteristics of uniform distribution.

Therefore, in cryptographic systems based on elliptic curves (ECC), the element n, which multiplies the generator, P, can be taken as the secret key of the user, while the result of this process is the public key. In order for an attacker to retrieve the secret key from the public key, they should solve the discrete logarithm problem in the cyclic group that was previously defined, something that is very difficult. On the other hand, the process of scalar multiplication is easy and with various mathematics techniques can be calculated very efficiently. It is very important that the generator and the parameters of the curve are chosen with close attention, so that the cyclic group has a prime number as an order. This is a bit difficult. However, in the elliptic curves, there are now efficient ways to produce safe curves and generator parameters. It is proven that the order of the cyclic group is around $p + O(\sqrt{p})$ [20], where p is the prime number that was chosen in order to define the finite field, so there can be an estimation of each curve's security. More specifically, since the better algorithms that solve the discrete logarithm have complexity, let p in $O(\sqrt{p})$ have a size of n bits; then, for ensuring 128-bit-level security, p has to have an $n/2$ value equal to 128, which is 256 bits.

Elliptic curve cryptography is widely prevalent today with implementations in the field of cryptocurrencies, and in collateral security in systems that maintain limited processing power and memory, as occurs in embedded systems and the in Internet of Things in digital certificates and signatures. There has been extended research on the current object, and there is an official data base that contains secure improvements in the calculations that are needed, with the result being easily implementable in applications without the need to know the mathematics concepts that define them. Moreover, they provide security with small key sizes, in comparison to the RSA.

However, the selection of parameters that are needed can cause security gaps in the scheme that will be chosen for an application, if the engineer who selects them does not have a mathematical background and makes the wrong choice. Moreover, the discrete logarithm problem is not safe; it cannot withstand to quantum attacks. Therefore, in future applications, elliptic curves are not enough to ensure safe modern networks.

3.3. Hyperelliptic Curve Cryptography for Genus ≥2

Five years after the introduction of elliptic curves in cryptography, Koblitz proposed the use of Jacobian curves in order to produce a suitable group. Elliptic curves are a subcategory of hyperelliptic curves, for genus equals to 1. For genus higher than 1, they are generally named as hyperelliptic curves.

In order to follow the equivalent procedure of enacting a cyclic group based on hyperelliptic curves of genus ≥2, there is no need to consider the procedure that was described previously the group's process, since now each line can intersect the curve in the $2 \cdot g + 1$ points, where g is the curve's genus [20].

Hyperelliptic curves of genus g can be described by the following equation:

$$C: \ y^2 + h(x)y = f(x), \ h, f \ \epsilon \ K[x], \deg(f) = 2g + 1, \deg(h) \leq g, f \ singular \qquad (3)$$

As previously observed, the curve has to be non-singular, which is guaranteed under the condition that no points zero both partial derivatives.

In order to build a cyclic group, the method that was proposed is to modify a set of points where the summation follows a function. For example, in a curve such as that in Figure 3, which is a curve of genus 2, the points $R_1 = (x_{R1}, y_{R1})$ and $-R_1 = (x_{R1}, y_{R1})$ belong in the curve $x = x_{R1}$, so it can be assumed that $R_1 \bigoplus(-R_1) = 0$. Consequently, the points $P_1, P_2, Q_1, Q_2, -R_1, -R_2$ follow a cubic function, so they are summed to zero. Therefore, it can be assumed that the elements come from the sum of two (generally two points, where g is the genus) points and the process between two points (or generally g) outputs two more points where a cubic (or generally $g + 1$ order) function $y = s(x)$ intersects the hyperelliptic curve as a result.

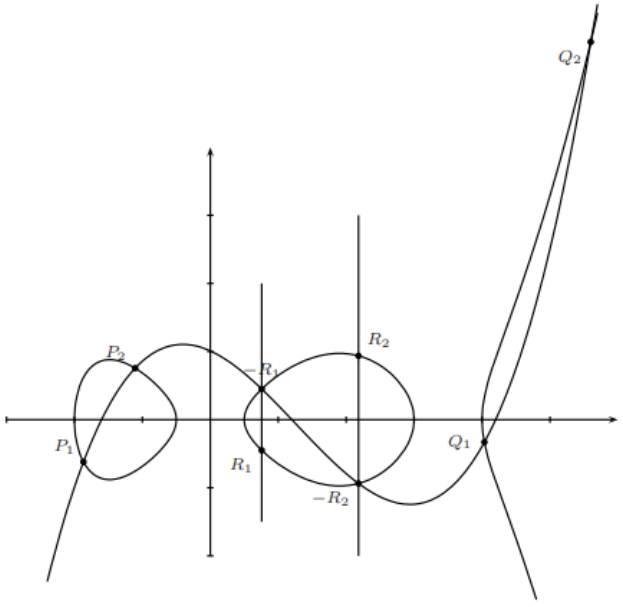


Figure 3. Hyperelliptic curve of genus 2, with $y2 = f(x)$ [20].

The class that was previously described is called the order of the divisor class, Pic_c^o, of the Curve. The divisor class receives many shapes; however, the one that prevails for defining the class and for developing the arithmetic is the Mumford representation. In this representation, the elements of the class are two polynomials, $u(x)$ and $v(x)$, which have the following characteristics:

- u is singular;
- $\deg(v) < \deg(u) \leq g \ (genus)$;
- u precisely divides the polynomial $v^2 + vh - f$.

Therefore, the polynomial $u(x)$ can be expressed as follows:

$$u(x) = \prod_{i=1}^{r}(x - x_i) \quad (4)$$

where x_i is the coordinate of the points that are chosen to shape the element. More specifically, for the P_i points that take part (support points), it has to apply that $P_i \neq P_\infty$ and $P_i \neq -P_j$ for every $i \neq j$. This specific shape applies to the class of reduced divisors, when the following exists: $r \leq g$. Since the class which frames the cyclic group was defined, which is the interesting part, it is very significant to develop efficient algorithms for the calculation of the class process that was described previously. Various algorithms have been built and optimized for the execution of this process; however, as can be easily seen, it is more difficult to calculate them compared to those defined for elliptic curves. What makes hyperelliptic curves suitable for cryptography fields is that in order to achieve a certain level of security, the finite field and the related arithmetic that is needed should take place in smaller numbers.

This happens because the elements of the class come from the g points and not from one point; the class has a size of p^g, when p is large enough. Therefore, based on the calculation of the security level that was analyzed in the elliptic curves section, a hyperelliptic curve of genus 2 can achieve a 128-bit security level with a finite field of a 128-bit size, while a hyperelliptic curve of genus 3 can achieve the same security level with 86 bits. It is observed that as the genus becomes bigger, the size of the finite field becomes smaller and so the keys that will be used can become smaller and achieve the same security level. Unfortunately, the index calculus attack, which is not possible for elliptic curves, can solve the discrete logarithm problem in a reasonable time, when the genus becomes bigger. For that reason, the curves of the genus that are equal or bigger than 4 are not used in cryptography [21]. The curves of genus 2 are safe enough; however, the curves of genus 3 are used but are not as safe as elliptic curves.

Hyperelliptic curve cryptography is not as prevalent as elliptic curves, so there are no databases that have carefully defined the parameters of the curves that need to be chosen. For that reason, it is not very easy to be used in industry, since it is still in the research stage. Also, the algorithms that exist for calculating the level of a class as an output of the reduced divisor class are not efficient, and this makes the process of achieving security in hyperelliptic curves a difficult concept.

Nevertheless, the small key size, which is the outcome, and the improvement in their arithmetic, make them significant and interesting for systems with limited resources, such as embedded systems and the well-known VANETs (vehicular ad hoc networks). That is why in the past few years this specific area has started to capture the attention of the scientific community.

4. Proposed Approach

Based on the related state-of-the-art review, the security scheme presented by Hassan Mistareehi and D. Manivannan [13] with group leader clustering was selected in the proposed approach to simulate secure message exchange scenarios on VANETs and has been extended to utilize ECC and HECC encryption algorithms and techniques to evaluate their performance. In this section, the tools and software used are presented; ns-3.30 (https://www.nsnam.org/about/what-is-ns-3/, accessed on 9 January 2024), SUMO 1.16.0 (https://projects.eclipse.org/projects/automotive.sumo/releases/1.16.0, accessed on 9 January 2024), NTL 5.5 (https://libntl.org/doc/tour-changes.html, accessed on 9 January 2024), while the details of the implemented cryptographic algorithms, of road traffic simulation, of VANET communication implementation and of the developed energy consumption model are provided.

4.1. Design of the Proposed Approach

This section elaborates on the design of the proposed approach, focusing on the cryptographic protocols employed. The approach presented in [13] utilizes multiple cryptographic techniques to ensure security for vehicle communications on VANETs. In our simulation, we do not emphasize pseudonym exchange and key updating/revocation, but instead mainly emphasize the procedure of vehicle registration in an RSU region, the procedure of choosing and informing a group leader in the network and finally, the procedure of a vehicle informing the RSU/GL about an important observed event in the region. Therefore, symmetric encryption is implemented using AES, while asymmetric encryption is implemented using ECC or HECC genus 2 or 3 using the ElGamal method (https://en.wikipedia.org/wiki/ElGamal_encryption, accessed on 9 January 2024). However, the ElGamal method requires the messages to be a cryptographic group element, a reduced divisor. Therefore, a method of mapping text to a reduced divisor is also needed to be implemented. Furthermore, ECDSA signatures and HEC ElGamal signatures are used, key generation and distribution are implemented using (H)ECQV, (hyper) elliptic Qu–Vanstone certificates. Since no open-source library for hyperelliptic curve cryptography of genus 3 is provided in C++, one is implemented and analyzed in this paper.

As mentioned above, the stages of the approach in [13] that rely on symmetric encryption are implemented using AES, specifically using the methods provided by the C++ library, Crypto++. AES is used with the CBC mode of operation and with 16-byte–128-bit key and block sizes. Also, an initialization vector is used and is transmitted along with the symmetric key. To properly transmit the keys and IVs to the network as strings, the HexEncoder and HexDecoder methods of the library are used. Encrypted text outputs are always a multiple of 16 bytes, since, by default, PKCS padding is used to handle input blocks.

Crypto++ additionally provides an API for executing ECC cryptographic operations (scalar multiplication, point addition, etc.) and implements methods for signing and verifying messages using ECDSA signatures. Curve secp256r (https://neuromancer.sk/std/secg/secp256r1, accessed on 9 January 2024) and its parameters (base/generator element) are provided by Crypto++, and this was chosen for our simulation because it produces 256-bit keys and therefore a 128-bit security level. ElGamal encryption/decryption is implemented using the ScalarMultiply, Add and Subtract methods of Crypto++ to produce the cypher text as the tuple (a, b), *where* $a = k{\cdot}G$, $b = k{\cdot}P + M$, where k is a random integer of the group order (256 bits), G is the generator element, P is the public key and M is the message encoded as an EC point. The plain text is received by calculating $M = b - x{\cdot}a$, where x is the private key. For encoding text as an EC Point, the Koblitz method is implemented [22]. Modular arithmetic is executed using the NTL library (https://libntl.org, accessed on 9 January 2024) and given the plain text as an integer, x, that is less than the group order. The calculated EC point (x_1, y_1) is $x_1 = x{\cdot}k + i$, where $k = 1000$ and $1 \leq i \leq 999$. x_1 is calculated using the above equation until it is a quadratic residue (https://en.wikipedia.org/wiki/Quadratic_residue, accessed on 9 January 2024), i.e., $y^2 = x_1 \; mod \; n$ has a solution. ECDSA signatures are generated using the signer class of Crypto++, and verification is executed using the verifier class of Crypto++. Points are compressed so that only the x coordinate is transmitted along with one extra byte to reconstruct the y coordinate properly.

Key pair generation, distribution and validation are handled by the ECQV certificate scheme [23] for all three different asymmetric encryption algorithms. The scheme is adapted to match the requirements for hyperelliptic curve cryptography. ECQV consists of five steps: ECQV_Setup, Cert_Request, Cert_Generate, Cert_PK_Extraction and Cert_Reception. At the ECQV_Setup step, the simulated node that acts as the certificate Authority generates a key pair, d_{CA}, Q_{CA}. The public key Q_{CA} is considered to be known to every participating vehicle and RSU. At the Cert_Request step, a node that requests a valid certificate from the CA generates a key pair, k_U, R_U, and sends the public key, R_U, to the CA along with its ID. At the Cert_Generate step, the CA receives R_U from the node, generates another key pair, k, $k * G$, then computes $P_U = R_U + k{\cdot}G$ and generates the certificate that includes P_U. It also

calculates $r = e{\cdot}k + d_{CA}\ (mod\ n)$, where $= Hash_n(Cert)$ and n is the Group Order. Finally, it sends the certificate and the parameter r to the node. At the Cert_PK_Extraction step, any node that wants to extract the public key from the certificate executes $Q_U = e{\cdot}P_U + Q_{CA}$. For the node that has requested the certificate, at the Cert_Reception step, it computes its public key as mentioned above, computes its private key, $d_U = r + e{\cdot}k_U\ (mod\ n)$, where $e = Hash_n(Cert)$ and validates if $Q_U = d_U{\cdot}G$. For ECC, these operations are executed using the Crypto++ API for ECC operations including scalar multiplication and addition, while for hashing, the SHA3-256 class of Crypto++ is used. Furthermore, for modular arithmetic, the ModularArithmetic class is used, which is initialized on n, the group order.

To implement the cryptographic operations for HECC genus 2, libg2hec (https://github.com/syncom/libg2hec/tree/master, accessed on 9 January 2024) was used. This is a C++ library that provides methods for generating and performing operations on reduced divisors of hyperelliptic curves of genus 2, which are suitable for cryptographic operations. The methods are based on the algorithms of the Handbook of Elliptic and Hyperelliptic Curve Cryptography by Henri Cohen and Gerhard Frey [24]. This handbook includes algorithms for number theory mathematics which were realized, by the authors of the current research paper, using C++ programming language and the NTL library. For encryption/decryption using ElGamal and for HECQV certificates, the same operations are used as in ECC, but now the group elements are the reduced divisors in Mumford representation. For signing and verifying messages, ElGamal signatures are used. The operations are similar to those of ECDSA; however, the methods were implemented from scratch. The produced signature is the tuple (a, b), where $a = k{\cdot}g$ and $b = \frac{m - x{\cdot}f(a)}{k}\ (mod\ N)$, where m is the message as a reduced divisor, x is the private key, k is a random integer in $[1,\ N)$, N is the group order, g is the generator element as a reduced divisor and f is a bijection of a reduced divisor in Mumford representation to an integer in $[1,\ N)$. The bijection used is $f(u) = \left(u_1^2 + u_2^2\right)\ (mod\ N)$, where u_1 and u_2 are the coefficients of the u polynomial of the reduced divisor in Mumford representation.

The challenge in HECC is to choose secure hyperelliptic curve parameters for cryptographic operations, since there is no given database of curves and parameters, unlike in ECC. Furthermore, general encoding and decoding techniques that are needed for ElGamal encryption and signatures to the Jacobian of the curves have not been established, so only algorithms for specific hyperelliptic curve families have been produced. For ElGamal signatures, the group order needs to be known and to be prime or "almost" prime, and the cofactor needs to be known as well. Therefore, for the signatures of HECC genus 2, a curve was chosen from the "Construction of Secure Random Curves of Genus 2 over Prime Fields" by Gaudry and Schost [25]. Specifically,

$$y^2 = f(x)\ \sigma\tau o\ F_p\ with\ p = 5{\cdot}1024 + 8503491 \tag{5}$$

with

$$\begin{aligned} f(x) = x^5 + 2682810822839355644900736x^3 + 2265913552959931029021 16x^2 \\ +254767471595292971789991 8x + 4797309959708489673059350 \end{aligned} \tag{6}$$

was chosen, which produces the following group of order:

$$N = 24999999999994130438600999402209463966197516075 69 \tag{7}$$

where N is a prime number of 128 bits. This means that the security level of the curve is at the 128-bit level and can be compared with secp256-r1, which is chosen for ECC operations.

To overcome the challenge of encoding text to the Jacobian of the curve, a technique proposed by Michel Sack and Nafissatou Diarra on 2018 [26] of encoding integers as points to a specific family of hyperelliptic curves of an arbitrary genus was used. In their paper, they provide code in Sage for executing their encoding techniques, which were translated to C++ using the NTL library. First of all, a different curve needs to be used and is produced

by their algorithm for a given finite field of characteristic p. However, $p = 3 \ mod \ 4$ and $p = 7 \ mod \ 8$. Therefore, the characteristic of the field is chosen as the following:

$$p = 340282366920938463463374607431768211223 \quad (8)$$

which is 128 bits and produces a group of order close to 256-bits. The curve is then produced based on the algorithm [26] and is used for all cryptographic operations, except ElGamal signatures. Unfortunately, the exact order of the curve is unknown. With [26], text is mapped to a point in the hyperelliptic curve; then, two points are combined and a reduced divisor is produced. Specifically, the u polynomial of the divisor is calculated as $u(x) = (x - x_1) \cdot (x - x_2)$ and the v polynomial is calculated as $vc = \frac{y_1 - y_2}{x_1 - x_2}$ and $d = y_1 - c \cdot x_1$.

Since, in Mumford representation, the v polynomial can be constructed from the u polynomial, it useful for a realistic simulation scenario to use divisor compression techniques. It is known that $u | v^2 - f$, so based on the algorithm proposed by Henri Cohen and Gerhard Frey in their book [24], a compression technique is used and the divisors are transmitted only using 256 bits for the u_1, u_2 coefficients, and one more byte is used for the reconstruction information of the v polynomial.

Finally, for HECC genus 3, a new library was created to match the needs of the simulation. Based completely on libg2hec implementation (https://github.com/syncom/libg2hec/tree/master, accessed on 9 January 2024) and Cohen and Frey's book [24], the curve and divisor classes were modified to match the required conditions for genus 3 curves and divisors. Most importantly, for group operations, SAM, NAF and ML methods for scalar multiplication were used exactly as implemented in libg2hec since they are of an arbitrary genus. For divisor "addition" and "doubling", Cantor's algorithm can be used without modification since it is of an arbitrary genus; however, it is slow. Therefore, algorithms 14.52 and 14.53 for genus 3 divisor addition and doubling from the Handbook of Elliptic and Hyperelliptic Curve [24] were implemented. As for cryptographic techniques, the same methods as those used in HECC genus 2 were used. What differed were the chosen curves and the encoding/decoding and bijection techniques, which were adapted for genus 3 divisors. For general cryptographic operations, excluding signatures, the following finite field was used:

$$p = 77371252455336267181195223 \quad (9)$$

Again, $p = 3 \ mod \ 4$ and $p = 7 \ mod \ 8$, while p is an 86-bit integer that produces a group order of almost 256 bits. This field is used by the algorithm of Michel Sack and Nafissatou Diarra [26] to produce a curve that will be used for cryptographic operations. For signatures, since the group order needs to be known, based on the publication of Annegret Weng [27], the following curve was selected:

$$y^2 = x^7 + x^5 + 6218231719898953 \cdot x^3 + 8683773159487505 \quad (10)$$

where the group order is $N = 8q$, where q is a prime number of 168 bits. Also, the bijection, f, used for ElGamal signatures is now modified to $f(u) = u_1^2 + u_2^2 + u_2^2 \ (mod \ N)$, and for the encodings on the Jacobian, Michel Sack and Nafissatou Diarra algorithm [26], it is used to map text to points in HEC; then, three points are combined to produce a valid reduced divisor of genus 3.

4.2. Implementation of the Proposed Approach

This section elaborates on the implementation of the cryptographic schemes used throughout the experiments, i.e., AES, ECC, HECC-2 and HECC-3, summarizing the main aspects and providing selected C++ screenshots. Moreover, further details are provided for the implementation of communication between the VANETs and the RSU, the simulation of road traffic and for the energy consumption model implemented. The entire code used for

the experiments and the related metrics and ns-3 files are available in a repository (https://github.com/PanosDgs/Secure_VANET_HECC/tree/master, accessed on 9 January 2024).

4.2.1. Cryptographic Techniques

The AES algorithms were used in all the simulations in order to fulfil the need for symmetric encryption in the safe scheme [13]. They are always coupled with one of the other three cryptographic protocols presented (i.e., ECC, HECC-2 orHECC-3). In order to support encryption, key generation and the IV vectors, the Crypto++ library was used. That library is implemented in C++ programming language and supports the following:

- Operation method: cipher block chaining (CBC), since it is assumed that this provides significant security;
- Key size: 16 bytes (128 bits) for achieving speed;
- Block size: 128 bits.

In [13], for algorithms 1 and 3, the RSU or the GL should generate a symmetric key and send it to the vehicle that tries to join the region. With the key, the IV is also sent. The generation method for the aforementioned is implemented in C++ as depicted (https://www.cryptopp.com/wiki/Advanced_Encryption_Standard, accessed on 9 January 2024) in Figure 4.

```
AutoSeededRandomPool prng;

SecByteBlock key(AES::DEFAULT_KEYLENGTH);
SecByteBlock iv(AES::BLOCKSIZE);

prng.GenerateBlock(key, key.size());
prng.GenerateBlock(iv, iv.size());

std::string keystr, ivstr;
HexEncoder encoder(new StringSink(keystr));
encoder.Put(key, key.size());
encoder.MessageEnd();

HexEncoder encoder2(new StringSink(ivstr));
encoder2.Put(iv, iv.size());
encoder2.MessageEnd();
```

Figure 4. Generation method functions related to AES scheme.

Initially, 2-byte blocks with random bytes are produced via the use of the random number generation library (AutoSeededRandomPool). Next, the key and the IV vector are transformed into strings by using HexEncoder of Crypto++ library. The latter library transforms the byte blocks into strings.

Then, the procedure of encryption/decryption of an AES message that follows is depicted (https://neuromancer.sk/std/secg/secp256r1, accessed on 9 January 2024) in Figures 5 and 6.

```
CBC_Mode<AES>::Encryption e;
e.SetKeyWithIV(key, 16, iv);

StreamTransformationFilter encfilter(e, nullptr,
BlockPaddingSchemeDef::PKCS_PADDING);
encfilter.Put(in, size);
encfilter.MessageEnd();
encfilter.Get(out, size+16-size%16);
```

Figure 5. Encryption of an AES message.

The function is fed with two buffers that consist of bytes, an out buffer and an input buffer, the buffer size, the key and the IV in string format. Since the key and the IV transformation take place in bytes again, the encryption method CBC AES is given as input to the encryption in StreamTransformationFilter. Due to the fact that the input is in a byte buffer, the output is stored in buffer out. However, the output size can be larger if the input size is not a multiple of 16 bytes, since it is used the PKCS Padding technique in order to transform the input size in an integer multiple of the block size.

In the decryption phase, the opposite procedure is followed. The transformation of the key and the IV in bytes are the same, so in Figure 6, one can observe only the decryption operation.

```
        CBC_Mode<AES>::Decryption d;
        d.SetKeyWithIV(key, 16, iv);

        StreamTransformationFilter decfilter(d, nullptr,
BlockPaddingSchemeDef::PKCS_PADDING);
        decfilter.Put(in, size);
        decfilter.MessageEnd();
        decfilter.Get(out, size);
```

Figure 6. Decryption of an AES message.

The parameters of *decrypt_message_AES* are the same, but the output is the decrypted message this time. In case of an error in the decryption (when the key or the IV is wrong) the method decfilter.Put throws an exception that can be collected with the *try* ... *catch* command and is printed.

For the implementation of ECC, the *Crypto++* library was used again, which provides many cryptographic functions that are ready. However, ElGamal is not ready and was realized via the API that *Crypto++* provides for operations. Moreover, the Koblitz method was used and the message encodings in the curve were made. Another implementation had to do with using the ECQV scheme in the certificates, where ECDSA was also used. The curve that was chosen is one of the most famous, secure and fast curves, i.e., the widely known secp256r1 curve (https://neuromancer.sk/std/secg/secp256r1, accessed on 9 January 2024), which produces keys of 256 bits in size, so a 128-bit security level.

The implementation of HECC-2 builds on the software available here (https://github.com/syncom/libg2hec, accessed on 9 January 2024) and extends this to make it suitable for ns-3 and IoV environments. Nevertheless, at the time when the respective experiments were conducted, there was no HECC-3 library available. Thus, by using *libg2hec* (https://github.com/syncom/libg2hec/tree/master, accessed on 9 January 2024) library as a reference level and by following the algorithms presented in the following book, classes, divisors and methods of curve genus 3 were realized. The respective implementation of HECC-3 used for the experiments carried out can be found in the following github repository (https://github.com/PanosDgs/Secure_VANET_HECC/tree/master, accessed on 9 January 2024). The *g3hcurve::update* method developed for the validity check of the curve is depicted in Figure 7 below.

```
//Set is_genus_3
if( deg(fpoly) == 7 && deg(hpoly) <= 3)
        is_genus_3 = TRUE;
else
        is_genus_3 = FALSE;
```

Figure 7. Validity check of a genus 3 curve.

The code above aims to check if the polynomial f is up to order 7 $(2 \cdot g + 1)$ and if the polynomial h is up to order 3 $(=g)$. The check that the curve should be non-singular remains

the same as that in *libg2hec* via the use of the NTL library. Random curve generation is carried out in the same way it occurs in *libg2hec*, by simply changing the order from 5 to 7 for the f polynomial and from 2 to 4 for the h polynomial of the curve.

Then, the divisor's curve genus 3 class is implemented. The code below implements a check of the valid divisor (Figure 8).

```
OK = OK && IsOne( LeadCoeff(upoly) ); // (1)

OK = OK && ( deg(upoly) <= genus ) && ( deg(vpoly) < deg(upoly) );
// (2)

OK = OK && IsZero(( vpoly*(vpoly + curve_g3.get_h())

                              - curve_g3.get_f() ) % upoly ); //(3)
```

Figure 8. Implementation of divisor's curve genus 3.

The most important part of the implementation is the divisor arithmetic. The methods for scalar multiplication are independent of the curve's genus, so some rules must be updated from the *libg2hec* library. The methods that are supported are SAM (square and multiply), NAF (non-adjacent form) and ML (montgomery ladder). The algorithms that should be realized are the "addition" and "doubling" divisors. The addition via Cantor's algorithms (Algorithm 14.7) [24] is independent of genus and can cover the arithmetic in genus 3 curves. However, it is very slow, and that is why the 14.52 and 14.52 algorithms of the book [24] were used, as these are the most optimized and have fewer and more efficient operations. The realization only depicts the residual arithmetic operations of the two aforementioned algorithms in C++ programming language.

4.2.2. Simulation of Road Traffic

The implementation of the road traffic involved using the SUMO tool to convert a real-world area into an XML file for network generation and route planning. Random routes were generated for nodes, and a traffic simulation lasting 400 s was conducted while spawning a vehicle every 0.5 s. This produced an XML file, Which was then converted into a .tcl format, which is suitable for NS-2 and NS-3 simulations using the trace exporter tool of SUMO. In the NS-3 simulation, 63 vehicles were introduced, with an additional manually added node serving as an RSU. To address issues that arise with all vehicles being present in the simulation since the start, precise entry and exit times were extracted from the .tcl file to control communication timing. Additionally, WAVE devices were installed on each node using appropriate NS-3 classes, with power and priority settings applied for packet transmissions.

4.2.3. VANET Communication

The communication between vehicles and the roadside unit (RSU) required to implement a secure scheme using desired cryptographic techniques was entirely carried out using NS-3 in C++, with the assistance of libraries like Crypto++, libg2hec, NTL and g3hec, developed during the project and discussed in Section 4.2.1. Specifically, communication was achieved through WAVE layer packets, extracting metrics without additional delays from processing higher network and application layers. Callback functions were installed to process incoming packets at nodes and take further actions based on the communication stage (Join, Accept, Extract_Symmetric, Receive GL Proof and Receive Vehicle Information based on [13]), such as sending a response packet if necessary. Responses were routed randomly within a 0–3 s window to better utilize the bandwidth and avoid collisions. The RSU stores the status of all vehicles in the area, key pairs, CA information and the public key of entering vehicles along with their certificates. It maintains the total number of vehicles that joined, the GL identifier, the symmetric keys exchanged with vehicles, and the parameters of curves used for encryption. Vehicles store their own keys, CA, RSU and

GL public keys if available, certificates, the status of the communication that is ongoing, symmetric keys, IV arrays and curve parameters.

Communication starts with the RSU broadcasting its certificate every 2 s. All vehicles are initially in the RECEIVE_CERT state. Upon receiving the RSU certificate, vehicles verify it, obtain the RSU public key, store it, and send back a join packet to the RSU, transitioning to RECEIVE_ACCEPT_KEY state. The RSU processes the response, changing vehicle state first to RECEIVE_ACCEPT_KEY, then to ON_SYMMETRIC_ENC after sending the ACCEPT response. Vehicle density increases after 120 s, leading to congestion. The RSU randomly selects a group leader (GL) and sends GL proof of leadership; the GL transitions to GROUP_LEADER_INFORM and then IS_GROUP_LEADER. Notably, ECC and HECC do not allow the encryption of the proof of leadership message, so the RSU provides the GL its own certificate, signed with the RSU's public key. The GL relays the decrypted message to the network every 2 s. Existing and new vehicles follow the Join and Accept process with the GL (using RECEIVE_ACCEPT_GL and ON_SYMM_GL states). Finally, vehicles transmit an INFORM message at random times once they receive a symmetric key. The GL gathers the Inform messages and sends it to the RSU.

4.2.4. Energy Consumption Model

NS-3 provides methods for calculating the energy consumption on simulated nodes based on the energy that a real network device (WAVE interface) would consume while the vehicle is communicating with the network. Specifically, the WiFi Radio Energy Model of NS-3 was used to calculate how much the device's battery was reduced after each transition of the WiFi state (Idle, CcaBusy, Tx, Rx, ChannelSwitch, Sleep, Off). Every WAVE network device was given 1000 Joules of battery at initialization for simulation purposes, since the difference in energy was the desired metric. After each major communication step that was mention in Section 4.2.3, the remaining energy was subtracted from the energy level that was stored before the communication was executed and the energy consumption was calculated.

5. Experimental Evaluation

In the current section, we present the experimental results via the use of the three asymmetric cryptography algorithms that were presented in Section 3, i.e., ECC, HECC-2 and HECC-3.

In the conducted simulations, 63 vehicles were used, as was 1 RSU, which covered the entire area. Moreover, the messages were transmitted with the maximum capable energy and priority, aiming to mitigate collisions from simultaneous transmission, and were chosen to realize a simple retreat of the responses up to 3 s. This period of time was big enough for real systems; however, it was used for better supervision of the communication. In addition, the measurements take place using cryptographic techniques. The specific technique increases the total time, so it increases the energy that is used in that period of time, since it is affected by the operational time. Thus, in some parts of the communication greater power usage than that in other parts is observed.

In order to measure time, the researchers used the function chrono::high_resolution_clock::now() at the beginning and end of each calculation. In the diagrams, one can observe the mean time of the time durations and the energy. The measurement of the parameters took place via the calculation of the byte buffer before it was transmitted to the node. Therefore, the size of the header that the WAVE protocol adds was not included in the measurements.

The simulations took place in a virtual environment, Linux Ubuntu 20.04.6, with Windows 10 as a host, 8 GB RAM available memory and four cores (Intel Core i5-10300H, clocked at 2.50 GHz.) The following open-source tools were also used: NS-3.30, SUMO 1.16.0 and NTL 5.5.

5.1. Evaluation Framework

Selected well-known software tools were used to build the simulation environment for the evaluation of the cryptographic methods on VANETs. The open-source software Network Simulator 3 (NS-3) (https://www.nsnam.org/about/what-is-ns-3/, accessed on 9 January 2024) was used to simulate a sample VANET and also to handle the underlying network protocols. NS-3 provides an implementation of the WAVE protocol stack, introduced by the IEEE 1609 family of standards for Wireless Access in Vehicular Environments (WAVE) (https://standards.ieee.org/ieee/1609.0/6792/, accessed on 9 January 2024). It also provided the routing algorithm designed for VANETs, AODV and ways of measuring the energy consumption on network devices in the simulation. Furthermore, to generate a realistic traffic scenario, another open-source tool, SUMO, was used. SUMO is widely used for simulating and analyzing road traffic, while it is also compatible with NS-3, using generated XML files that can be imported to NS-3 and create a VANET with realistic movement scenarios. Finally, to visualize the network, PyViz was used. NS-3 already supports PyViz and provides an easy way to visualize the VANET using it.

5.2. Duration of Cryptographic Operations

Initially, the execution times of the underlying cryptographic operations were experimentally measured. In this respect, series of experiments were conducted aiming to measure the durations of the following operations:

- Key pair generation;
- Certificate public key extraction;
- Message decryption;
- Message encryption;
- Signature generation;
- Signature verification;
- Decoding;
- Certificate private key reception;
- Certificate generation;
- Encoding.

The respective results obtained regarding the durations of the operations aforementioned when adopting the three studied cryptographic approaches (i.e., ECC, HECC-2 and HECC-3) are presented in Figure 9.

As expected, and as indicated in Figure 9, in principle, the ECC cryptographic operations are much shorter than the respective HECC operations. More specifically, for the same key size of 128 bits, key production based on ECC is 16 times faster than that in HECC genus 2, and 27 times faster than the key production duration in HECC genus 3.

Similar observations are made for the certificate generation extraction times and times based on the ECQV scheme. As indicated in Figure 9, the respective production and extraction carried out via ECC are 15 times faster than the respective operations in HECC genus 2 and 25 times faster than those in HECC genus 3. It is easily observed that production is the slowest process, followed by the secret key extraction process (Figure 9) and lastly public key extraction. More specifically, the extraction process is three times faster than the certificate generation operation.

The resulting excellence of ECC against HECC is observed because the ECC scheme is very mature and has been optimized by the research community. HECC, on the other hand, is a very new approach and has not been optimized yet. Moreover, HECC includes very complex mathematics in order to be efficiently implemented in C++, and a lot of effort and expertise are needed in order to achieve HECC algorithm versions that could outperform the ECC.

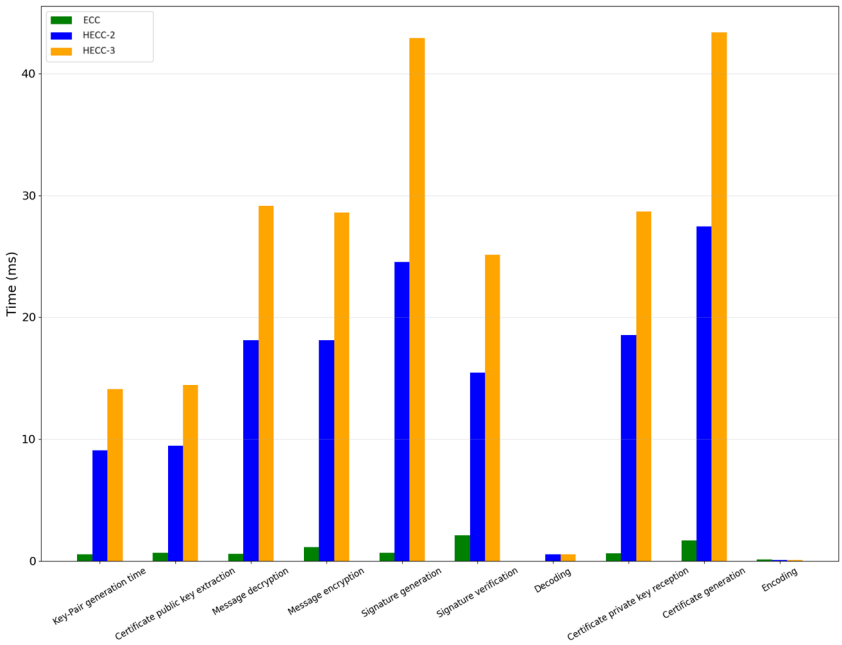


Figure 9. Duration of the various cryptographic operations (in ms).

Regarding the durations of the encryption and decryption operations, the obtained results are similar. Therefore, as indicated in Figure 9, the encryption and decryption operations are much faster in ECC than they are in the two other HECC approaches, i.e., they 13 times faster than in HECC genus 2 and 21 times faster than in HECC genus 3.

Similarly, with respect to the signature generation duration, the ECC approach is much faster than the HECC genus 2 scheme, which in turn is faster than the HECC genus 3 approach, as indicated in Figure 9. The observations are also similar for the signature verification duration, as presented in Figure 9. More specifically, the signature validation time is 5.3 times faster in ECC than in HECC-2 and 8.3 times faster than in HECC-3. This is due to the fact that the curve employed for the signatures by HECC-2 corresponds to a 84-bit security level, as opposed to the ECC curve, which corresponds to a 128-bit security level.

With regard to the message encoding and decoding operations' durations, encoding based on the Koblitz algorithm that is employed by ECC is slower than that based on the algorithms employed by HECC genus 2 and genus 3 [24], as indicated in Figure 9 (bars of encoding and decoding). More specifically, encoding via the use of ECC is about 38% slower than that using HECC-2 or HECC-3. As far as decoding time is concerned, this is about 50% faster in ECC than it is in HECC-2 or HECC-3.

5.3. Size of Exchanged Messages

Concerning the simulation of communication, the following types of messages were used:

- RSU_CERT_BROADCAST: The certificate that RSU transmits periodically.
- VEHICLE_SEND_JOIN_RSU: The message "Join" that the vehicle sends to the RSU, when it needs to subscribe to the RSU's area.
- RSU_ACCEPT: The RSU's response to the vehicle's message, "join". It also contains the cryptographic symmetric key.
- RSU_INFORM_LEADER: The message that the RSU sends, in order to update a vehicle of the fact that a group leader has been chosen. This provides the proof of leadership.
- GL_LEADERSHIP_PROOF: The message that the GL (group leader) broadcasts periodically, in order to prove that it is valid and invite the vehicles to "join" him.
- VEHICLE_SEND_JOIN_GL: The message, "Join", that the vehicle sends to the GL, when it needs to subscribe to the GL.

- GL_ACCEPT: The GL's response to the vehicle's message, "join", also carrying the cryptographic symmetric key.
- VEHICLE_INFORM: The message that a vehicle sends in order to update for its position. It was chosen typically as an updating message for system evaluation.

Subsequently, the various messages' sizes were experimentally measured. In this respect, series of experiments were conducted where the sizes of the types of messages mentioned in the previous list were estimated. The respective results obtained are presented in Figure 10, for the three different algorithms of asymmetric cryptography studied.

As one may easily observe, the HECC genus 3 scheme results in the largest GL_LEADE RSHIP_PROOF message size, whereas the size for ECC is a bit smaller, and that of HECC genus 2 is the smallest. As far as the size of the VEHICLE_SEND_JOIN_GL message is concerned, the HECC genus 3 scheme results in the largest size, while HECC genus 2 results in the smallest size of the message type.

Regarding the size of the GL_ACCEPT message, the observations are similar, as HECC genus 3 results in the largest size, while ECC corresponds to about half of the message size and HECC genus 2 results in the smallest size of this message over the three cryptographic schemes studied (Figure 10). As far as the VEHICLE_INFORM message size is concerned, the situation is similar for the ECC and HECC genus 3 schemes (about 150 bytes), while HECC genus 2 results in slightly smaller VEHICLE_INFORM messages of about 125–130 bytes.

Overall, as indicated in Figure 10, the messages exchanged using ECC or HECC-2 correspond to about the same size, whereas the HECC-3 scheme results in messages of a significantly larger size. This is more obvious for the GL_ACCEPT and RSU_ACCEPT messages. In general, the sizes of the messages are similar. If the same level of signatures for HECC-2 was used, the respective messages would have been of the same size. In order to achieve the same size in genus 2, the idea was to use divisor compression. In HECC-3, it was tricky to use compression realizations, so there was a transmission of information containing the parameters of the curve in RSU_CERT_BROADCAST. The latter action is more realistic, but it increases the transmitted information. It should be highlighted that the RSU_CERT_BROADCAST, GL_LEADERSHIP_PROOF and VEHICLE_INFORM messages were sent periodically, as is the case in non-simulated systems.

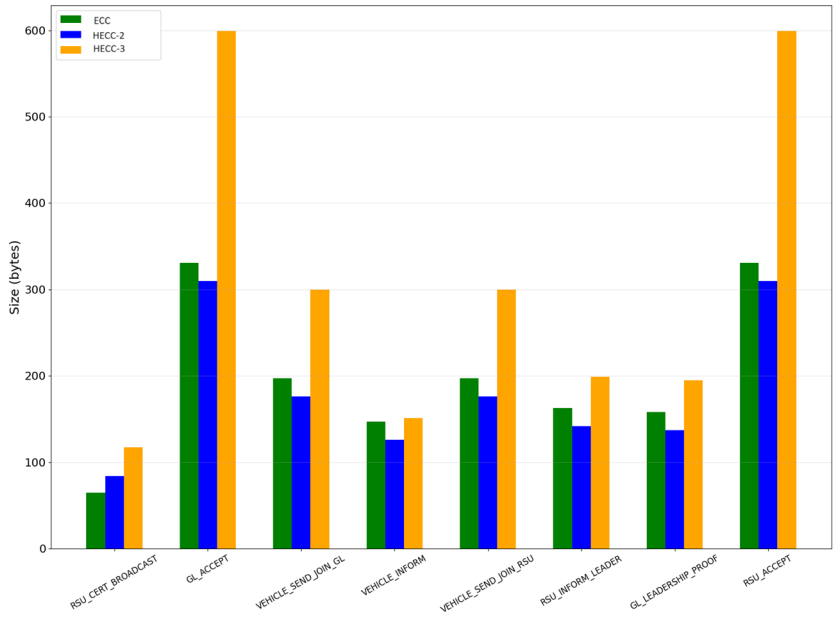


Figure 10. Size of the messages exchanged during the ns-3 simulation (in bytes).

5.4. Energy Consumption

The energy consumption for the three cryptographic algorithms studied is depicted in Figure 11. The idea is that ns-3 starts from a standard initial energy threshold and measures the depletion as the exchange of messages take place. The energy consumption metric is critical for the vehicle side, rather than for the infrastructure, given the respective resource constraints applicable. The respective energy consumption levels are the following:

- RECEIVE_CERT: This just describes the procedure of receiving and processing the certificate of the RSU (a process of a very short duration).
- EXTRACT_GL_PROOF: This describes the procedure of receiving and processing the GL leader's proof (a process of a very short duration).
- RECEIVE_ACCEPT_SYMMETRIC: This describes the procedure of sending the message "Join" and receiving and processing the "Accept" response from the RSU.
- EXTRACT_JOIN_SEND_ACCEPT: This describes the procedure of the "Join" message that the GL receives from a vehicle and the transmission of the "Accept" response.
- EXTRACT_INFO_GL: This describes the procedure of extracting a message, "Inform", that the GL receives from a vehicle.

Subsequently, the energy consumption in a very short procedure metric was experimentally evaluated. In this respect, several experiments were conducted to measure the previously mentioned levels, and the respective results obtained are presented in Figures 11 and 12 with regard to the average consumption of each level in Joules.

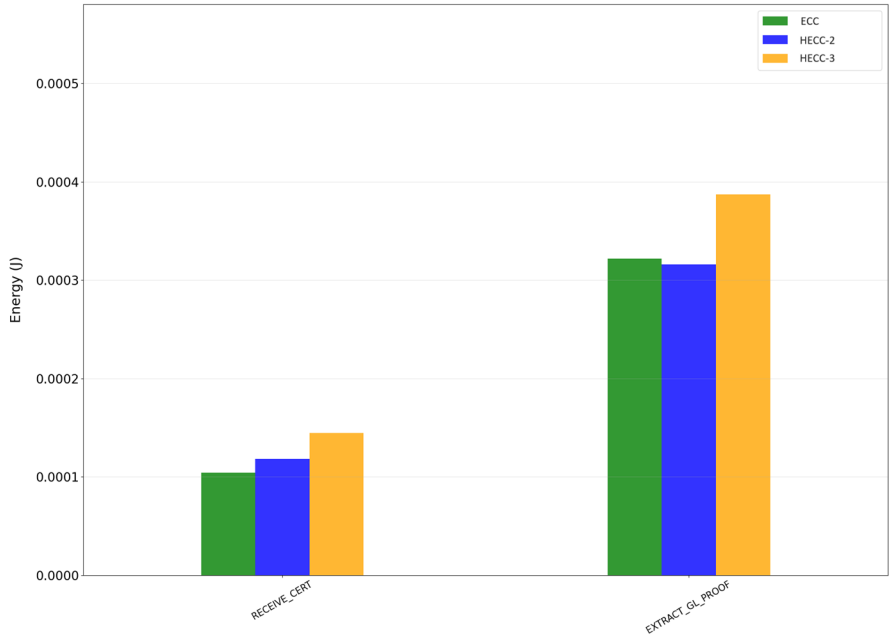


Figure 11. Energy consumption in a very short-duration procedure.

In Figure 11, one may observe that the energy consumption in RECEIVE_CERT is less than 200 μJ for all the studied schemes (ECC, HECC genus 2 and HECC genus 3), while as far as EXTRACT_GL_PROOF is concerned, the respective consumption lies between 300 μJ and 400 μJ. Figure 12 depicts the RECEIVE_ACCEPT_SYMMETRIC consumption, which lies between 1.1 Joules and 1.2 Joules for all three cryptographic schemes (ECC, HECC genus 2 and HECC genus 3). The EXTRACT_ACCEPT_JOIN_GL_SEND_ACCEPT consumption is lower and lies between 0.6 Joules and 0.9 Joules, while the lowest consumption is related to EXTRACT_INFO_GL consumption, which lies between 0.3 Joules and 0.7 Joules.

It is observed that the highest amount of energy is consumed by HECC genus 3 for the very small-duration procedures, whereas in the rest, the ECC consumes the most among the three schemes. In principle, the energy consumption depends on the sizes of the messages, since large messages, which require fragmentation, cause more state switches in

the transmission device, thus leading to higher energy consumption. However, as the three studied algorithms have also been selected due to their small message sizes, they do not demand fragmentation in the WAVE level.

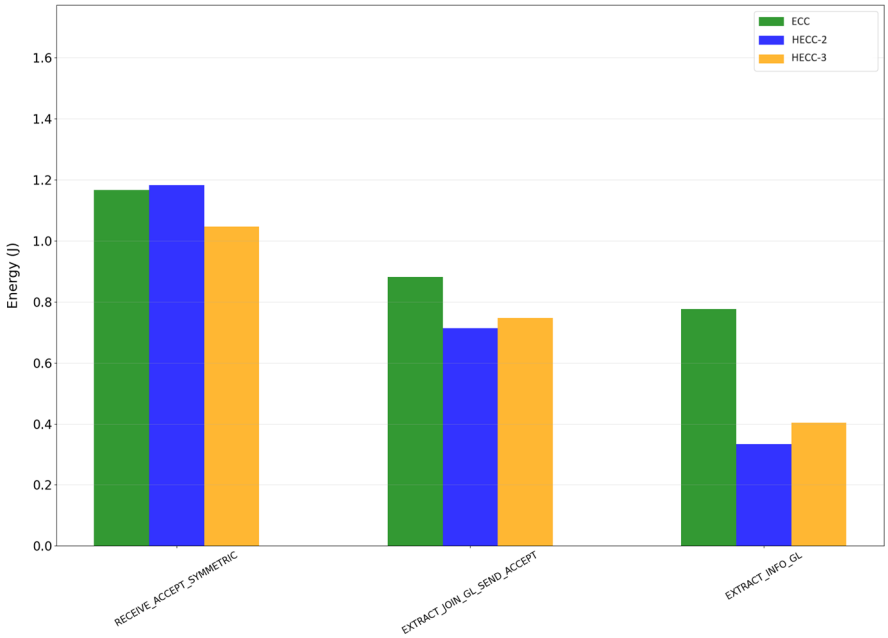


Figure 12. Procedures' energy consumption.

6. A System for Message Encryption via Zigbee Network, Using AES or ECC

In the current section, there is an analysis of a system implemented to secure messages over a wireless network, and more specifically via the use of Xbee Zigbee modules. The entire construction was built using Arduino MEGA 2560, Arduino GIGA R1, a TFT screen connected to Arduino, a voltage translator, Xbee Zigbee, a USB splitter, a DC power supply and a USB keyboard.

Two different algorithmic schemes were used during the experiments. The first one was AES and the second one was based on an ECC adapted in order to encrypt and decrypt messages, since ECC has not been designed for the purpose of decrypting and encrypting text messages. The respective implementation is based on the following libraries: an AES library suitable for Arduino (https://github.com/DavyLandman/AESLib, accessed on 9 January 2024), and the ECC encryption/decryption of the messages of the following library (https://github.com/ShubhamAnnigeri/tinyECC-ArduinoIDE, accessed on 9 January 2024).

A 4" LCD touch display shield was attached to Arduino MEGA 2560 R3, making possible for the user to see the received and sent messages. The characteristics of the display are the following (https://grobotronics.com/display-4-touch-lcd-shield-for-arduino.html, accessed on 9 January 2024):

- LCD Type: TFT;
- LCD Interface: SPI;
- LCD Controller: ILI9486;
- Touch Screen Type: Resistive;
- Touch Screen Controller: XPT2046;
- Colors: RGB, 65K colors;
- Resolution: 480 × 320 (Pixel);
- Aspect Ratio: 8:5;
- I/O Voltage: 3.3–5 V.

To conclude, the core idea is that the user of each side picks an encryption key that has been shared with another user in a safe channel, enabling encryption during a live conversation. Each user makes use of the same key in order to encrypt messages sent to other side and decrypts the received messages in order to read the plaintext.

We used Arduino GIGA R1 (https://grobotronics.com/arduino-giga-r1-wifi.html?sl=en, accessed on 9 January 2024), because it encompasses a USB interface suitable for connecting the USB keyboard. Arduino MEGA 2560 R3 is connected to the TFT screen, so that the users can see the received decrypted messages from the other side. Arduino MEGA 2560 R3 (https://store.arduino.cc/products/arduino-mega-2560-rev3, accessed on 9 January 2024) is a famous open-source solution for IoT experiments, using the ATmega2560 microcontroller. It has the capability of using the UART protocol it encompasses and various I/O pins. Its eight-bit CPU is clocked at 16 MHz.

Arduino GIGA R1 is a device that enables IoT experiments and is significantly more powerful than Arduino MEGA 2560. It uses the STM32H747XI dual-Cortex-M7 + M4 32 bit low-power Arm MCU clocked at 480 Hz and 240 MHz. It contains many I/O pins and many protocols (4 × UART, 3 × I2C, 2 × SPI, CAN), a 2M byte flash memory, and it operates at 3.3 volts in both power and logic. The latter justifies the reason why SparkFun Logic Level Converter—Bi-Directional (https://www.sparkfun.com/products/12009, accessed on 9 January 2024) was used in order for the two Arduino devices (MEGA 2560 and GIGA R1) to communicate with each other. Arduino GIGA R1 was used due to its USB interface, so that the USB keyboard could be connected to the device in order to type messages.

A USB voltage splitter was used in order to supply the two Arduino instruments, the voltage translator and Xbee Zigbee S2 2 mW with power. Xbee Zigbee S2 2 mW (https://www.sparkfun.com/products/retired/10414, accessed on 9 January 2024) (+3 dBm) provides a coverage of 120 m at LoS. The Zigbee protocol is a low-power solution for wireless data exchange at a 250 kbps maximum data rate. It operates at 3.3 Volts@40 mA. It becomes obvious that a voltage translator needs to be used in order to establish communication with Arduino MEGA 2560 R3, which operates at 5 Volts. The whole construction was supplied with a 5 Volts/3 Amps DC power supply, connected to the USB splitter inputs, as depicted in Figures 13 and 14.

Figure 15 depicts the image of one of the TFT screens decrypting an incoming message using an AES encryption protocol. Figure 16 shows the message decrypted using the ECC scheme.

As far as the encryption/decryption times are concerned, some measurements were taken both for the AES and ECC algorithmic schemes. Concerning the AES scheme, the encryption time for the message "Hello George!" took 1028 μsec and the decryption time was 1292 μs. As far as ECC is concerned, for the message "Hello George!", the encryption time was 3861 μs or 3861 microseconds and the decryption time was 1721 μs (or 1721 microseconds). Figure 17 shows the printscreen from the terminal while the message was being encrypted and decrypted.

An oscilloscope was used in order to receive the pulse -train of the plaintext (unencrypted message) (Figure 18) and the encrypted message (Figure 19) after leaving Arduino MEGA 2560 R3, so that the message could be fed to Zigbee and transmitted wirelessly to the other end. Figure 18 proves that the pulse reaches 5 Volts. Obviously, it is too difficult to catch the entire message because the pulse train would be large to fit in the image. Moreover, the bit rate at which the Arduino receives or transmits messages is fixed at 9600 bps on both mechanisms that were presented in previous images.

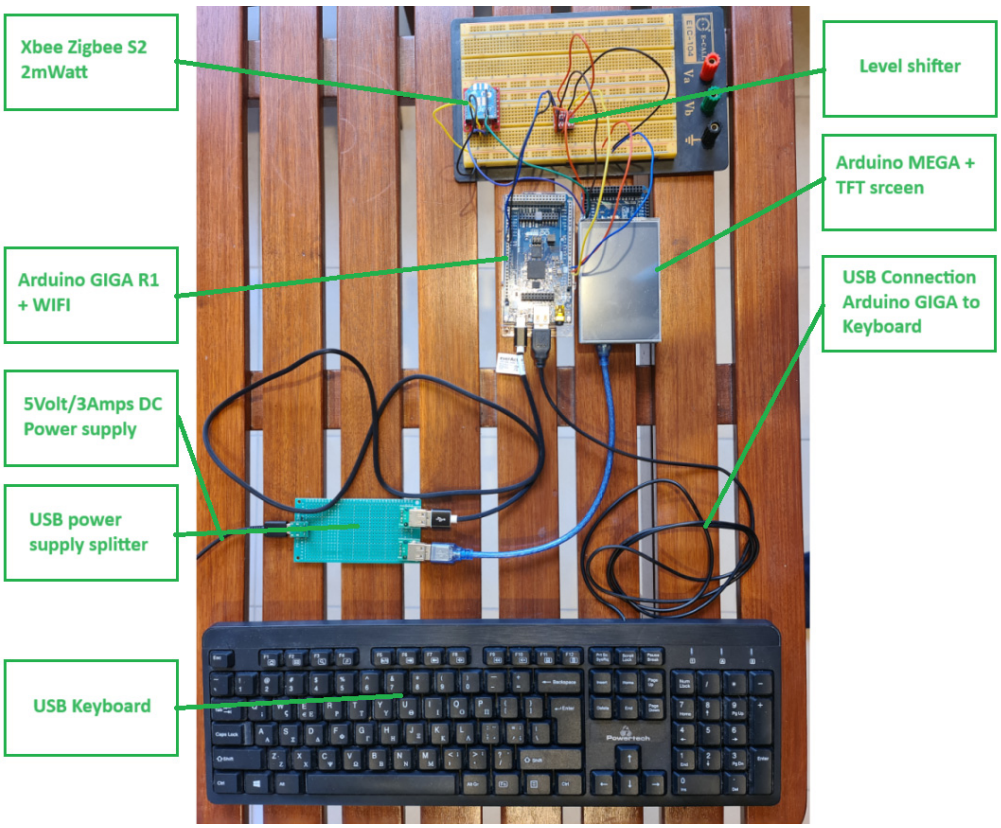


Figure 13. The various connections of the encryption/decryption device.

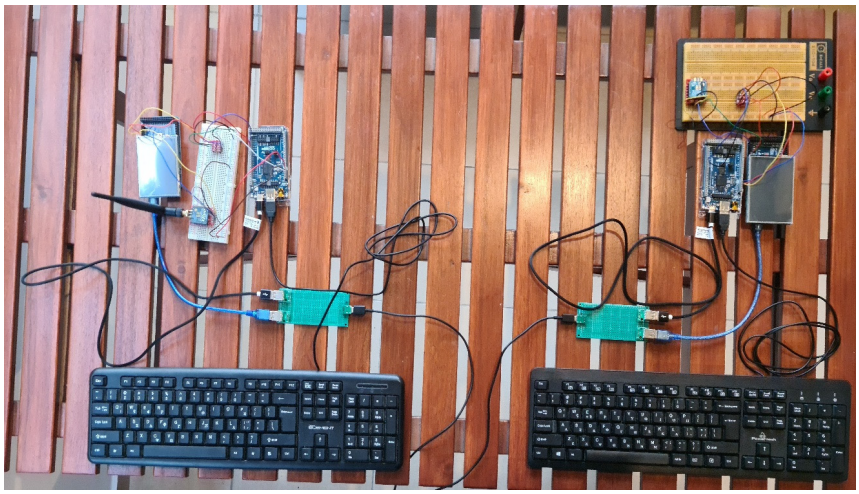


Figure 14. The entire construction for two users. Each side can receive and send messages to the other side, which is encrypted and decrypted either via an AES or ECC executed in the Arduino MEGA 2560 R3 microcontroller.



Figure 15. Interface snapshot of when a user receives a decrypted message in the case that the AES symmetric algorithmic scheme is used.



Figure 16. Interface snapshot of when a user receives a decrypted message in the case that the ECC asymmetric algorithmic scheme is used.

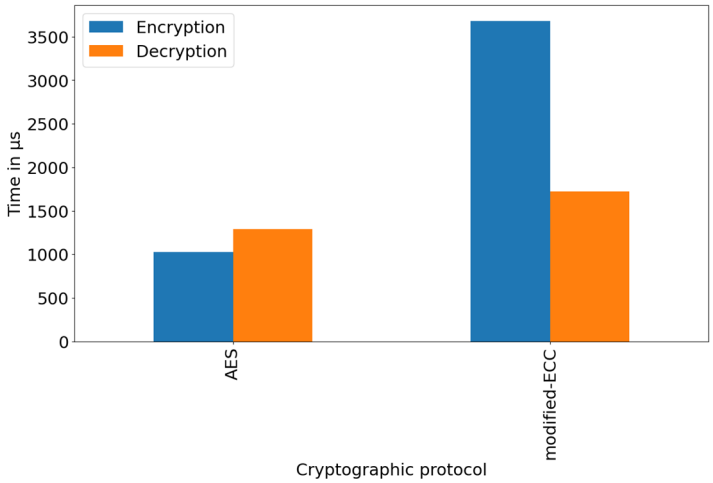


Figure 17. Encryption and decryption times (in µs) for Arduino MEGA 2560 R3 when either an AES or a modified ECC cryptographic scheme is used.

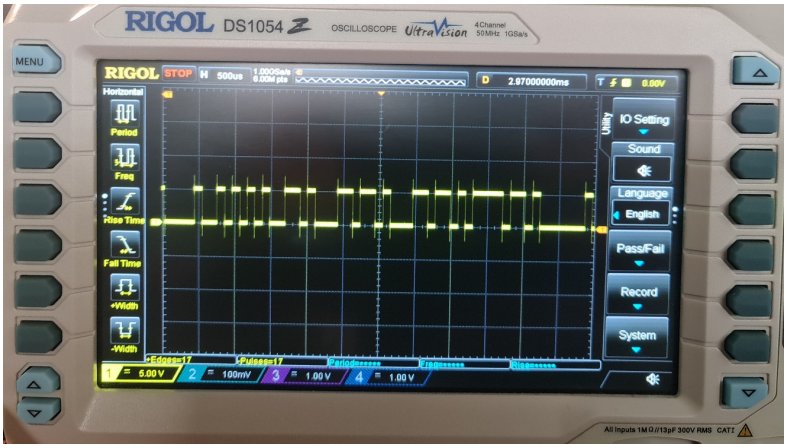


Figure 18. The electrical signal of the plaintext message, as it is sent from the USB keyboard to Arduino MEGA 2560, in order to be encrypted either via the AES symmetric cryptographic scheme or the ECC asymmetric cryptographic scheme.

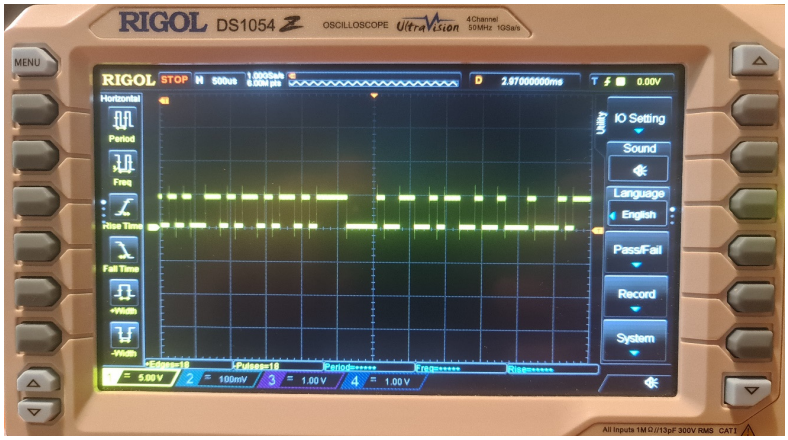


Figure 19. The electrical signal of the encrypted (by AES) message that is fed to Zigbee (from Arduino) in order to then be transmitted wirelessly by Zigbee.

The process is as follows. Arduino GIGA R1 was used in order to obtain the metrics, either by using ECC or an AES. Arduino GIGA R1 was not connected to the mechanism presented before, as the TFT screen could not be paired with the current Arduino device. For this reason, Arduino GIGA R1 was simply connected to a laptop in order to receive the metrics via the serial reading of the Arduino IDE platform. The encryption and decryption times for AES are depicted in Figure 20, and the times for the encryption and decryption time sunder modified ECC are depicted in Figure 20. As is observed, the encryption time using an AES and Arduino GIGA R1 was 34 µs instead of 1028 µs when Arduino MEGA 2560 R3 was used. However, the decryption time when using the same cryptographic scheme and Arduino GIGA R1 was 50 µs instead of the 1292 µs taken with Arduino MEGA 2560 R3. Again, using ECC for encryption with Arduino GIGA R1 took 3753 µs instead of the 3681 µs taken for Arduino MEGA 2560 R3, whereas for the same cryptographic scheme for the decryption part and Arduino GIGA R1 it took 2 µs instead of the 1721 µs taken with Arduino MEGA 2560 R3. It is more than obvious that Arduino GIGA R1 excels over Arduino MEGA 2560 R3 in most metrics regarding encryption/decryption in AES and ECC cryptographic schemes.

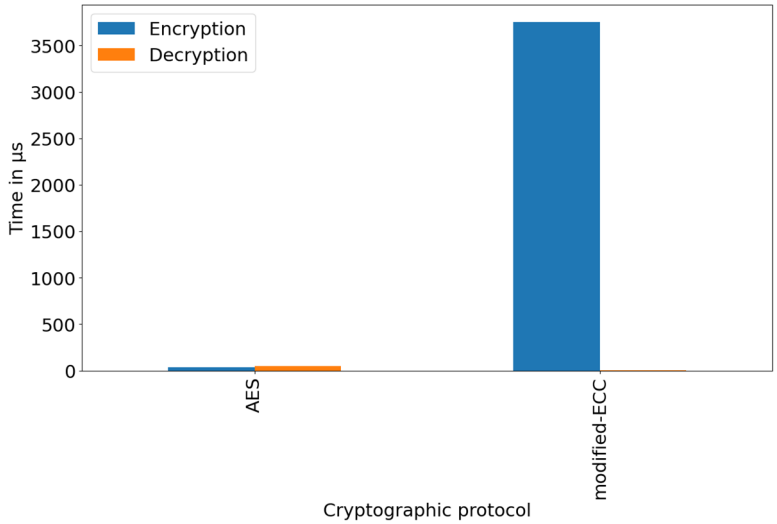


Figure 20. The encryption and decryption time taken in µs for the message: "Hello George"! while using Arduino GIGA R1 when either the AES or modified ECC cryptographic scheme is used.

7. Conclusions and Plans for the Future

This paper introduced the usage of hyperelliptic curve cryptography for enhancing privacy in IoV. This idea is quite new, as even though there are papers introducing various techniques for achieving IoV privacy, HECC has not been exploited yet. Series of experiments were carried out in an ns-3 simulator where an entire area of vehicles, RSUs and GLs was established. The goal was to achieve the maximum privacy given various modeled attacks. The following cryptographic methods were studied: ECC, HECC-2 and HECC-3.

In this respect, several metrics were measured to evaluate the proposed approach, such as the following: the key pair generation, certificate public key generation, message decryption, message encryption, signature generation, signature verification, decoding, certificate private key reception, certificate generation and encoding time. The additional evaluation metrics used included the size of the messages exchanged, and the paper also studies the energy consumption for message generation and exchange.

Moreover, the paper introduces an innovative approach for exchanging messages via Zigbee in the ISM (industrial scientific medical) free band via the use of Arduino modules and appropriately constructed electronics. The proposed approach uses either an AES or a modified version of ECC methods. Evaluation metrics used also included the duration of encryption/decryption as far as message exchange was concerned. The evaluation outcome indicates that the AES scheme is faster than the modified ECC scheme in the decryption/encryption of messages in most of the metrics obtained, for both Arduino modules used.

The evaluation results indicate that in most of the metrics measured, ECC demonstrates better results than HECC-2 and HECC-3. However, there were cases where HECC outperformed ECC, but they were limited. This occurred because ECC has been greatly improved the last few years by the research community, and it delivers better results in terms of most of the metrics. HECC, on the other hand, is not as mature and it is not optimized to run fast, although it uses smaller key sizes with compared with those under ECC for the same security level. Moreover, HECC involved the usage of very complex mathematics, and this is one reason why it is used less extensively than ECC is. It is, however, expected, that if HECC is optimized in the future in C/C++ programming language or even in VHDL code, it will be able to demonstrate evaluation results that are far better than those obtained with ECC.

The proposed future extensions of the presented approach mainly focus on the improvement of HECC performance. Initially, their arithmetic can improve via the use of a modern library for the modular arithmetic. The use of NTL 5.5, which is outdated, may

produce delays in scalar multiplication. Moreover, the use of special curves [28], in which proposed improvements in their arithmetic have been proposed, can be studied. It is concluded that the best choice of parameters can significantly improve the performance of HECC, as is referred to in the publication of Pelzl, Wollinger et al. [29]. Apart from these, in the case of the embedded systems, accelerators (such as FPGAs, ASICs) can also be used in real systems for improving performance.

In addition, the choice of hyperelliptic curves in that specific scheme is limited by the existence of a general algorithm for coding messages targeting the curve. The algorithm [26] that was used is the most generalized algorithm that exists in the literature and limits the choice of curve to specific curve families. It is important to study which of the family of curves is cryptographically appliable, meaning that they produce a class with an order "near" the prime number. Because of this, fragmentation algorithms of points in Jacobian hyperelliptic curves can be used, in order to identify which are the most secure. Moreover, the generation of secure curves via the use of the CM method will produce more choices of secure curves with a known order, so that curves of the same security level and signatures can be used for better study and comparison.

Author Contributions: Conceptualization, G.R. and I.R.; methodology, G.R. and P.D.; software, P.D. and G.R.; validation, G.R., P.D. and I.R.; formal analysis, G.R. and P.D.; investigation, G.R. and P.D.; writing—original draft preparation, P.D. and G.R.; writing—review and editing, G.R. and I.R.; supervision, I.R.; project administration, I.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Chen, W.; Wu, H.; Chen, X.; Chen, J. A Review of Research on Privacy Protection of Internet of Vehicles Based on Blockchain. *J. Sens. Actuator Netw.* **2022**, *11*, 86. [CrossRef]
2. Khan, M.A.; Ullah, I.; Abdullah, A.M.; Mohsan, S.A.H.; Noor, F. An Efficient and Conditional Privacy-Preserving Heterogeneous Signcryption Scheme for the Internet of Drones. *Sensors* **2023**, *23*, 1063. [CrossRef]
3. Shayea, G.G.; Mohammed, D.A.; Abbas, A.H.; Abdulsattar, N.F. Privacy-Aware Secure Routing through Elliptical Curve Cryptography with Optimal RSU Distribution in VANETs. *Designs* **2022**, *6*, 121. [CrossRef]
4. Al-Shareeda, M.A.; Anbar, M.; Manickam, S.; Hasbullah, I.H. A Secure Pseudonym-Based Conditional Privacy-Preservation Authentication Scheme in Vehicular Ad Hoc Networks. *Sensors* **2022**, *22*, 1696. [CrossRef]
5. Azees, M.; Vijayakumar, P.; Deboarh, L.J. EAAP: Efficient Anonymous Authentication With Conditional Privacy-Preserving Scheme for Vehicular Ad Hoc Networks. *IEEE Trans. Intell. Transport. Syst.* **2017**, *18*, 2467–2476. [CrossRef]
6. Liu, Y.; Wang, Y.; Chang, G. Efficient Privacy-Preserving Dual Authentication and Key Agreement Scheme for Secure V2V Communications in an IoV Paradigm. *IEEE Trans. Intell. Transport. Syst.* **2017**, *18*, 2740–2749. [CrossRef]
7. Jo, H.J.; Kim, I.S.; Lee, D.H. Reliable Cooperative Authentication for Vehicular Networks. *IEEE Trans. Intell. Transport. Syst.* **2018**, *19*, 1065–1079. [CrossRef]
8. Shen, Z.; Wang, Y.; Wang, H.; Liu, P.; Liu, K.; Zhang, J. Trust Mechanism Privacy Protection Scheme Combining Blockchain and Multi-Party Evaluation. *IEEE Trans. Intell. Veh.* **2024**, 1–10. [CrossRef]
9. Su, H.; Dong, S.; Wang, N.; Zhang, T. An Efficient Privacy-Preserving Authentication Scheme That Mitigates TA Dependency in VANETs. *Veh. Commun.* **2024**, *45*, 100727. [CrossRef]
10. Mistareehi, H.; Islam, T.; Manivannan, D. A Secure and Distributed Architecture for Vehicular Cloud. *Internet Things* **2021**, *13*, 100355. [CrossRef]
11. Rezazadeh Baee, M.A.; Simpson, L.; Boyen, X.; Foo, E.; Pieprzyk, J. ALI: Anonymous Lightweight Inter-Vehicle Broadcast Authentication with Encryption. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 1799–1817. [CrossRef]
12. Dua, A.; Kumar, N.; Das, A.K.; Susilo, W. Secure Message Communication Protocol Among Vehicles in Smart City. *IEEE Trans. Veh. Technol.* **2018**, *67*, 4359–4373. [CrossRef]
13. Mistareehi, H.; Manivannan, D. A Low-Overhead Message Authentication and Secure Message Dissemination Scheme for VANETs. *Network* **2022**, *2*, 139–152. [CrossRef]
14. Batool, H.; Anjum, A.; Khan, A.; Izzo, S.; Mazzocca, C.; Jeon, G. A Secure and Privacy Preserved Infrastructure for VANETs Based on Federated Learning with Local Differential Privacy. *Inf. Sci.* **2024**, *652*, 119717. [CrossRef]

15. Yadav, V.K. Anonymous and Linkable Ring Signcryption Scheme for Location-Based Services in VANETs. *Veh. Commun.* **2024**, *45*, 100717. [CrossRef]
16. Shen, Z.; Ren, F.; Wang, H.; Liu, P.; Liu, K.; Zhang, J. Combining Blockchain and Crowd-Sensing for Location Privacy Protection in Internet of Vehicles. *Veh. Commun.* **2024**, *45*, 100724. [CrossRef]
17. Huang, J.; Kong, L.; Wang, J.; Chen, G.; Gao, J.; Huang, G.; Khan, M.K. Secure Data Sharing over Vehicular Networks Based on Multi-Sharding Blockchain. *ACM Trans. Sens. Netw.* **2024**, *20*, 1–23. [CrossRef]
18. Tandon, R.; Verma, A.; Gupta, P.K. D-BLAC: A Dual Blockchain-Based Decentralized Architecture for Authentication and Communication in VANET. *Expert Syst. Appl.* **2024**, *237*, 121461. [CrossRef]
19. Hammod, D.N. Modified Lightweight AES Based on Replacement Table and Chaotic System. In Proceedings of the 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 9 June 2022; pp. 1–5.
20. Scheidler, R. *An Introduction to Hyperelliptic Curve Arithmetic*; World Scientific: Singapore, 2015.
21. Alimoradi, R. A Study of Hyperelliptic Curves in Cryptography. *Int. J. Comput. Netw. Inf. Secur.* **2016**, *8*, 67–72. [CrossRef]
22. Bh, P.; Chandravathi, D.; Roja, P.P. Encoding and Decoding of a Message in the Implementation of Elliptic Curve Cryptography Using Koblitz's Method. *Int. J. Comput. Sci. Eng.* **2010**, *2*, 1904–1907.
23. Campagna, M. SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV). In *Standards for Efficient Cryptography, Version 1*; Certicom Research: Mississauga, ON, Canada, 2013.
24. Cohen, H.; Frey, G.; Avanzi, R.; Doche, C.; Lange, T.; Nguyen, K.; Vercauteren, F. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*; CRC Press: Boca Raton, FL, USA, 2005.
25. Gaudry, P.; Schost, É. Construction of Secure Random Curves of Genus 2 over Prime Fields. In *Advances in Cryptology—EUROCRYPT 2004*; Cachin, C., Camenisch, J.L., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3027, pp. 239–256. ISBN 978-3-540-21935-4.
26. Seck, M.; Diarra, N. Unified Formulas for Some Deterministic Almost-Injective Encodings into Hyperelliptic Curves. In *Progress in Cryptology—AFRICACRYPT 2018*; Joux, A., Nitaj, A., Rachidi, T., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2018; Volume 10831, pp. 183–202. ISBN 978-3-319-89338-9.
27. Weng, A. A Class of Hyperelliptic CM-Curves of Genus Three. *J.-Ramanujan Math. Soc.* **2001**, *16*, 339–372.
28. Pelzl, J.; Wollinger, T.; Paar, C. Special Hyperelliptic Curve Cryptosystems of Genus Two: Efficient Arithmetic and Fast Implementation. In *Embedded Cryptographic Hardware: Design and Security*; Nova Science Publishers: New York, NY, USA, 2004.
29. Pelzl, J.; Wollinger, T.; Guajardo, J.; Paar, C. Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves. In *Cryptographic Hardware and Embedded Systems—CHES 2003*; Walter, C.D., Koç, Ç.K., Paar, C., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2779, pp. 351–365. ISBN 978-3-540-40833-8.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.