On the Efficiency of Pairing-Based Authentication for Connected Vehicles: Time is Not on Our Side!

On the Efficiency of Pairing-Based Authentication for Connected Vehicles: Time Is Not on Our Side!

Mir Ali Rezazadeh Baee, *Senior Member, IEEE*
Leonie Simpson, Xavier Boyen, Ernest Foo, *Member, IEEE*, and Josef Pieprzyk

***Abstract*—In the near future, intelligent vehicles will be connected via wireless communication links, forming Vehicular Ad-hoc Networks (VANETs). This has potential to improve road safety and to optimize traffic. However, if the communications are not secure, VANETs are vulnerable to cyber attacks involving message manipulation. Research on this problem has produced multiple authentication protocols based on bilinear pairings (a variant of elliptic curve cryptography). The efficiency of such authentication schemes must be addressed before they can be used in real-world deployments. Standards bodies have begun standardizing various pairing-based schemes. The IEEE 1609.2 security standard has not yet selected any pairing-based scheme, leaving the settings related to pairing-based cryptography in the vehicular environments unspecified. In this work, we investigate the efficiency of pairing-based cryptographic primitives over the Barreto-Lynn-Scott and Barreto-Naehrig pairing friendly elliptic curves recommended in the IETF and ISO standards, to determine their suitability for practical application. We implement the algorithms and evaluate the effect of cryptographic pairings using theoretical and experimental analysis of four well-known pairing-based short signature schemes, including: Boneh-Lynn-Shacham, Boneh-Boyen, Zhang-Safavi-Susilo, and Boneh-Gentry-Lynn-Shacham. We use metrics including CPU clock cycles per operation, average computation time in milliseconds, and signature/public key size in bits to estimate the cost of implementing cryptographic pairings on modern processors. We demonstrate the effect of pairing-based cryptography on authentication in vehicular networks. We investigate a high-density highway scenario and show that a crash is possible, as a result of the evaluated authentication delay. We share our findings ahead of the IEEE 1609.2 recommendations for the use of cryptographic pairings.**

***Index Terms*—Cryptography, authentication, vehicular communication systems, scalability, efficiency.**

Manuscript received Nov, 03, 2020; revised Jan, 16, 2021; accepted May 19, 2021. Date of publication X X, X; date of current version X X, X. The work of J. Pieprzyk was supported by the Australian Research Council under Grant DP180102199 and the Polish National Science Center (Narodowe Centrum Nauki) under Grant 2018/31/B/ST6/03003. The work of M. A. R. Baee was supported by the Queensland University of Technology Postgraduate Research Award. The review of this article was coordinated by Prof. D. Vergnaud. *(Corresponding author: Mir Ali Rezazadeh Baee.)*

M. A. R. Baee, L. Simpson, and X. Boyen are with the School of Computer Science, Queensland University of Technology, QLD 4000, Australia (e-mail: info@baee.co; lr.simpson@qut.edu.au; xavier.boyen@qut.edu.au).

E. Foo is with the School of Information and Communication Technology, Griffith University, QLD 4111, Australia, and also with the School of Computer Science, Queensland University of Technology, QLD 4000, Australia (e-mail: e.foo@griffith.edu.au).

J. Pieprzyk is with the Commonwealth Scientific and Industrial Research Organization, Data61, NSW 2122, Australia, with the Institute of Computer Science, Polish Academy of Sciences, 01-248 Warsaw, Poland, and also with the School of Computer Science, Queensland University of Technology, QLD 4000, Australia (e-mail: josef.pieprzyk@csiro.au).

I. Introduction

MODERN VEHICLES have enhanced sensing capabilities, and carry computing and communication platforms to enable Vehicular Communication (VC) systems. In a VC system, vehicles communicate with both other vehicles and nearby fixed equipment to support different applications. These communications have the potential to improve vehicular safety services through periodic safety message broadcasting, letting vehicles know about environmental conditions and neighboring vehicles [1].

Car manufacturers embed devices such as IEEE 802.11p, known as Wireless Access in Vehicular Environments (WAVE) in vehicles to enable wireless communication with other vehicles and nearby fixed electronic equipment, such as Road Side Units (RSUs). The IEEE 802.11p is extended from the IEEE 802.11a standard to facilitate such communication with the intention of increasing road safety [2].

Vehicles and RSUs equipped with WAVE act as communicating nodes in a self-organizing network called a Vehicular Ad-hoc Network (VANET). Different communication technologies are used in VANETs, including Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications. V2V is wireless communication among nearby vehicles. V2I is wireless communications between vehicles and RSUs. Vehicles and RSUs within transmission range exchange messages about critical information such as location, speed, braking status, traffic conditions, and traffic events [3].

A wide range of applications can be deployed in vehicular networks. Applications with safety considerations can be classified as either safety-critical, safety-related, or non-safety-related. Safety-critical applications are used for hazardous situations where the danger is high or imminent, such as potential collisions [4]. In this case, V2V beacon messaging requires high reliability and low latency to realize the safety function.

The beacon messages are short network packets containing the identification and context information for a vehicle and broadcast at a high update rate. The latency required for most safety applications is between 100 and 300 milliseconds (ms) in a communication range of 150 to 500 meters (m) [3]. The system utility requires vehicles to receive updated information from surrounding vehicles within the required time frame before sending out a new safety message.

Without the use of communication security mechanisms, malicious activities such as false message injections can be performed without detection [5], [6]. This is potentially disas-

trous for drivers. To prevent this, authentication is necessary to ensure that received messages come from legitimate vehicles or infrastructure, and have not been manipulated by attackers [7].

Digital signatures are important cryptographic primitives to support authentication schemes, and many digital signature schemes are based on pairings. Since the Weil pairing was introduced in 1940 [8], it has been used widely with elliptic curves. In 1986, Miller proposed an algorithm for evaluating the Weil pairing on an algebraic curve [9]. The algorithm has been widely applied. In 1994, cryptographic pairings were first introduced to attack the security of certain elliptic curves using Miller's Algorithm.

In addition to Weil, there are other pairings, including: Tate [10], Eta [11], Ate [12], Ate$_i$ [13], and R-Ate [14]. The optimal Ate pairing [15] is common and often the fastest, since it involves the shortest Miller loop and different optimizations. All of these pairings exist for all common pairing-friendly curves, such as Barreto-Lynn-Scott (BLS) curve [16] and Barreto-Naehrig (BN) curve [17].

Since 2000, bilinear pairings have been widely used to design cryptographic protocols, including digital signatures. In 2001, Boneh-Lynn-Shacham (BLS) [18] proposed a pairing based short signature scheme. This was followed by a large number of pairing-based signature schemes for different applications, including: Boneh-Boyen (BB) short signature scheme [19], Zhang-Safavi-Susilo (ZSS) short signature scheme [20], and Boneh-Gentry-Lynn-Shacham (BGLS) aggregate signature scheme [21]. The security of cryptographic pairings is based on the difficulty of solving Discrete Logarithm Problem (DLP) [22].

Since 2012, there have been a number of successful attacks reducing the complexity of the DLP in finite fields $\mathbb{F}_{p^k}$, where p is prime, and $k > 1$ is a small embedding degree. These include the Menezes-Okamoto-Vanstone (MOV) attack [23] and the Number Field Sieve (NFS) attack variants [24]. There are two different cases: small characteristic case ($p = 2$ and $p = 3$) with supersingular pairing-friendly curves defined over finite fields of small characteristics, and large characteristic case with supersingular and ordinary curves defined over finite fields of large characteristics (prime fields in almost all the cases [25]).

For improvements of the DL computation in the small characteristic case, the list of record computations started in 2012 by Hayashi et al. [26]. The major results were the two quasi-polynomial-time algorithms applicable to finite fields of small characteristics, described in 2014 by Granger et al. [27] and Barbulescu et al. [28]. For this reason, agencies such as the European Union Agency for Cybersecurity (ENISA) have forbidden the use of pairings with small field characteristics, as these attacks invalidate such pairings [29] [30].

Standardization bodies, such as the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC), and the Internet Engineering Task Force (IETF) [31] provide key size recommendations for use in pairings. However, the last recommendation from ISO/IEC 15946-5 was in 2017 [32]. The Institute of Electrical and Electronics Engineers (IEEE) produced the latest version of IEEE P1363.3 standard in 2013 [33], specifying identity-based schemes for encryption, digital signatures, signcryption, and key establishment using pairings, without including short signatures. This standard is not application specific, and mainly focuses on the mathematical operations. The (United States) National Institute of Standards and Technology (NIST) has not yet standardized pairing-based cryptography [34].

For VANET safety applications, short signatures are very useful as they reduce overall message size and bandwidth requirements. For example, Rivest-Shamir-Adleman (RSA) signature scheme [35] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [36] require 3072 bits and 256 bits key length, respectively, to provide a 128-bit security level (3072 bits factoring modulus and discrete logarithm group) [37]. However, the BLS signature scheme provides short signature of length approximately 384 bits with a level of security similar to 256-bit ECDSA signatures of length approximately 512 bits (64 bytes using the NIST-P256 curve) [4]. In addition, the BGLS bilinear aggregate signature scheme enables a user to aggregate all signatures (generated by BLS signature scheme) into a single short signature [38]. The BLS, BGLS and ZSS schemes are in the process of being standardized through the IETF [39] [40].

A. Research Challenge

Enabling safety-critical applications in VANETs requires extensive beaconing exchange between vehicles. For example, a simulation study presented by Baee et al. [4], under an assumed traffic scenario, demonstrated that the maximum number of beacons received by a vehicle is 222, 158, and 154 beacons per 100 milliseconds, under the application of Japanese ARIB STD-T109, American IEEE 1609, and European ETSI ITS-G5 standards, respectively.

However, authentication mechanisms are constrained by the latency requirements. For economic viability, car manufacturers embed small-scale and low-cost hardware, and different cars may have different processing capacities [41][42]. These limit in-car computational capabilities and make the use of complex cryptographic techniques economically unattractive.

Currently, NXP Semiconductors offers the RoadLINK SAF5400 [43]: this can process up to 200 ECDSA signature verifications every 100 ms. Clearly, any pairing-based proposal for authentication must have low latency and low computational overhead to fit with the limited computing resources of vehicles. That is, the authentication scheme must be able to process a large number of broadcast messages received in a short period of time before their dedicated deadline, while simultaneously processing other vehicular applications.

One of the major challenges to standardization of cryptographic pairings by the IEEE 1609.2 is the selection of elliptic curves and a specific pairing. However, there is no clear consensus as to which combinations are best to use in practice. Many authentication schemes propose to secure VANET communications based on cryptographic pairings and short signature schemes [44]. For practical applications, the efficiency of such authentication schemes is vital. This must be investigated before selections can be readily and widely used

in real-world deployments. Indeed, the IEEE 1609.2 security standard has not standardized any pairing-based scheme; the settings related to pairing-based cryptography in the vehicular environments are left unspecified.

B. Existing Work

Existing research [45–49] implements and evaluates the efficiency of certain pairings over different curves.

Devegili et al. [45] describe an efficient implementation of the Tate and Ate pairings using the BN curves. Beuchat et al. [46] describe the design of a fast software implementation for computing the optimal Ate pairing over a 254-bit BN curve. Aranha et al. [47] implement asymmetric pairings derived from the BLS and BN curves at the 192-bit security level, and evaluate that the BLS curve with embedding degree 12 is the fastest in their implementations. Fouotsa et al. [48] evaluate the computation of optimal Ate pairings on three different elliptic curves, and provide a detailed arithmetic. Clarisse et al. [49] focus on the elliptic curves with fast computations in the first pairing group and outline suitability of such curves for group signature-like schemes.

However, these studies investigate the theoretical cost for the Miller step and the final exponentiation of pairings, but fail to evaluate some important aspects of pairings. For example, hash into elliptic curves and/or elliptic curves group arithmetic commonly used in pairing-based protocols are not evaluated in these works. In addition, the efficiency of pairings in authentication schemes is not evaluated in the context of VANET applications, although it is crucial for the latency-critical applications.

C. Research Contribution

To the best of our knowledge, this paper is the first comprehensive evaluation of pairing-based cryptography for connected vehicles in vehicular environments. Understanding the security, efficiency, and functionality of pairings is crucial to enable suitable recommendations for the current draft of IEEE 1609.2 security standard. We discuss efficiency of pairing-based cryptography in short signature schemes for authentication, and estimate the cost of implementing them on modern processors embedded in smart vehicles. We also compare the efficiency of pairing-based short signature schemes to the non-pairing ECDSA. The contribution of this research is fourfold:

1) The cryptographic overheads of elliptic curve group arithmetic and optimal Ate pairing are evaluated over seven different curves providing 128-bit security level.
2) The signature generation and verification overheads of three well-known short signature schemes (BLS, BB, and ZSS) and the BGLS aggregate signature scheme over the BLS and BN curves are theoretically and practically analyzed.
3) The impact of the pairing computations on latency-critical applications is analyzed, evaluated, and discussed. Under an assumed traffic scenario, this research shows how and to what extent the pairing-based authentication schemes delay communications that may impact driver safety. The vehicle displacement during authentication operations is calculated in two best and worst case scenarios to determine whether a crash would occur, as a result of the delay.
4) The challenges revealed by experimental results are identified, and recommendations of appropriate pairing-curve combinations for future use in VANETs are given.

D. Organization of the Paper

The remainder of this paper is organized as follows. Section II is an overview of authenticity and integrity requirements in VC systems, and presents some preliminaries. Section III describes pairings in cryptography, and explains four pairing-based short signature schemes. Section IV introduces our performance evaluation, and presents the results of our study. We discuss the impact in Section V, and conclude the paper in Section VI.

II. Background and Preliminaries

Authentication is a vital part of trust establishment between communicating nodes in a VANET system [50]. This section briefly presents a background on the authenticity and integrity requirements in VC systems, and provides basic definitions of elliptic curve cryptography.

A. Authenticity and Integrity in Vehicular Communications

Authentication in VC systems is performed at two levels: message level and entity level. Message authentication provides assurance of data origin, and also data integrity. Data origin authentication is where a communicating node can be verified as the original source of data created at some time in the past. Data integrity is a property in which data has not been altered in an unauthorized manner. This property must be maintained from the time the data was created, transmitted, or stored by an authorized source. Note that for any received message, it is essential to ensure both that data actually came from the claimed source (data origin authentication) and is unaltered (data integrity) [51].

Entity authentication or identification is a process designed to assure one communicating node (the verifier) that the identity of another (the prover or claimant) is as claimed, and, as a result, prevents impersonation [52]. From the verifier's point of view, the result of an identification protocol is either acceptance of the prover's identity as authentic, or rejection of the identity (termination without acceptance) [51].

B. Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curve E over finite field $\mathbb{F}_q$, where $q = p^m$, prime $p > 3$, and a natural number $m \geq 1$.

An elliptic curve denoted by $E(\mathbb{F}_q)$, is the set of points $(x, y) \in \mathbb{F}_q^2$, defined to satisfy the *generalized Weierstrass form* of equation, $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_q$ are the constant coefficients of the curve [53]. The (x, y) representation of points is known as affine coordinate. A point $\mathcal{O}$ is called the point at infinity,

which is always a unique solution at infinity in the projective space to the curve equation. For example, the point at infinity can be $\mathcal{O} = (0, 1, 0)$ in projective coordinates [54], where all lines parallel to the Y-axis meet. The set (x, y) together with a point at infinity $\mathcal{O}$ (acting as the identity element), and a special addition operation define an Abelian group, called the elliptic curve group. Thus, elliptic curve operations such as scalar multiplication, field multiplication and field inverse multiplication are required. The most important operation is scalar multiplication: given a point P and a positive integer a, compute aP. The scalar multiplication deals with point doubling (adding $P(x, y)$ to itself) and point addition (adding two different points $(P(x, y), Q(X, Y))$. These point doubling and additions, fundamentally, further deal with additions, squaring, multiplications, and inversion operations.

There are several different forms of representation for elliptic curves that achieve faster group operations, examples include Short Weierstrass curve ($y^2 = x^3 + ax + b$ [53]), and Hessian curve ($y^2 + axy + by = x^3$ [55]). For the interested reader, additional information related to the ECC is presented by Blake et al. [56] and Silverman [57].

The security of ECC-based schemes is dependent on the difficulty of solving Elliptic Curve Discrete Logarithm Problem (ECDLP). Let $E(\mathbb{F}_q)$ be an elliptic curve containing a point P of order q. Let $\langle P \rangle$ be the cyclic subgroup of points generated by P such that $Q = [a]P$ is a point scalar multiplication, where $1 < a < q$, and $Q \in \langle P \rangle$. The ECDLP is the problem of finding the value of a given P and Q [58][59]. Discrete Logarithm (DL) on a q-bit elliptic curve is hard as long as q is large enough to make cost $2^{q/2}$ for solving DL prohibitive. The difficulty to solve ECDLP, makes ECC-based schemes as secure cryptosystems to provide authentication (if the elliptic curve group is selected properly).

ECC offers similar security with smaller key sizes and memory requirements compared to other traditional DLP-based schemes in use today [60][61].

III. Pairings and Short Signatures

This section presents pairings in cryptography, basic security assumptions, curves used in cryptographic pairings, and the types of pairings. Then, the section revisits the four well-known short signature schemes, including: the BLS, BB, ZSS, and BGLS.

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of the same prime order (usually cyclic) in which discrete logarithm is hard (given y, find x such that $g^x = y$). Let g_1 be a generator of $\mathbb{G}_1$, and g_2 be a generator of $\mathbb{G}_2$. A bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficiently computable function such that $e(g_1, g_2) \neq 1$ *(Nondegeneracy)*, and $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}$ *(Bilinearity)* [18]. A pairing is called symmetric if $\mathbb{G}_1 = \mathbb{G}_2$, and said to be asymmetric if, $\mathbb{G}_1 \neq \mathbb{G}_2$.

Let E be an elliptic curve defined over $\mathbb{F}_{p^k}$ of characteristic p, and order $n = \#E(\mathbb{F}_{p^k})$. Suppose $P \in E(\mathbb{F}_{p^k})$ satisfies $rP = \mathcal{O}$, such that P has order r or a factor of r. The point P is called an r-torsion point. The r-torsion points have a structure of 2-dimensional vector space. A set of r-torsion points in $E(\mathbb{F}_{p^k})$ is denoted by $E(\mathbb{F}_{p^k})[r]$ [62]. The Weil pairing $f : E[r] \times E[r] \rightarrow \mathbb{F}_{p^k}$ maps points to the extension field $\mathbb{F}_{p^k}$, which should be large enough to make sure $E(\mathbb{F}_{p^k})$ contains all r-torsion points. The degree k of this field extension is called the embedding degree, which is the smallest integer such that $r \mid p^k - 1$, where $r \nmid p - 1$ [63]. Let Q be another point in a distinct r-torsion subgroup on the curve, and a, b be two scalar values. We can create a pairing function e which makes the following equivalents: $e(aP, Q) = e(P, aQ)$, $e(bP, Q) = e(P, bQ)$, $e(aP, bQ) = e(P, abQ) = e(abP, Q) = e(P, Q)^{ab}$. The embedding degree k to some extent dictates the security level efficiently achievable on the curve (a trade-off between efficiency and security). The smaller the value of k, the faster is the computation of the bilinear map, and the larger the value of k, the more difficult to solve the DL, resulting in slower computation of the bilinear map. For the interested reader, additional information related to pairing-based cryptography is presented by El Mrabet and Joye [64].

A. Basic Security Assumptions

The security of cryptographic pairings is mainly based on the difficulty of solving discrete logarithms in the field $\mathbb{F}_{p^k}^*$ and elliptic curve group $E(\mathbb{F}_q)$. The parameters p and k must be chosen such that the order of group $\#E(\mathbb{F}_q)$ has a large prime factor r, where the two discrete logarithm problems are of approximately equal difficulty using the best known algorithms.

Note that solving the DLP problem has exponential complexity on well-chosen elliptic curves, and sub-exponential time in large characteristic finite fields. In small characteristic fields, the two quasi-polynomial-time algorithms can solve the DLP, as described in 2014 by Granger et al. [27] and Barbulescu et al. [28].

Let $\mathbb{G}$ be an additive cyclic group of finite order n, and $P \in \mathbb{G}$ be its generator such that the isomorphism ϕ between $(\mathbb{Z}/n\mathbb{Z}, +)$ and $(\mathbb{G}, +)$ can be efficiently computable via $\phi : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{G}$, $a \mapsto aP$. There must be no efficient way to compute $\phi^{-1}(Q)$ for any $Q \in \mathbb{G}$, otherwise all pairing related hardness assumptions will be wrong. For example, given a pair (P, aP), the DLP asks to find a, where a is chosen randomly in $\mathbb{Z}/n\mathbb{Z}$.

Pairing related computational and decisional problems fall into three main categories, including: Diffie-Hellman problems, bilinear Diffie-Hellman problems, and Miscellaneous problems. These are fundamental security assumptions for pairing-based cryptographic protocols. For the interested reader, additional information related to the computational and decisional problems is presented by Dutta et al. [65].

B. Curves Used in Cryptographic Pairings

Cryptographic pairings on elliptic curves require the existence of some specific subclass of curves with suitable field size, small embedding degree, and fast finite field arithmetic called pairing-friendly curves. Pairing-friendly curves are important for VANET applications since short signatures are needed in low-bandwidth communication environments. There are two basic choices: the supersingular curves [57] over any

finite field, and ordinary pairing-friendly elliptic curves over prime field $\mathbb{F}_p$.

The first choice limits the available embedding degree to a maximum of $k = 6$, and only on curves over fields of characteristic $p = 3$. These small values of k and p make supersingular elliptic curves weaker than the general case for cryptography [23]. From a pairing-based cryptography security standpoint, we need both the ECDLP in the subgroup of order r and the DLP in the multiplicative group of the extension field $\mathbb{F}_{p^k}$ to be equivalently hard. For example, an attacker has to perform at least 2^{80} operations to break an 80-bit level of security. That is, we need $r \approx 160$ bits and $p^k \approx 1024$ bits. The maximum achievable level of security using supersingular elliptic curves for an efficient implementation is $k = 6 \approx 1024/160$. This is insufficient, and such curves are already being avoided. Thus, a larger value of k is desirable to achieve higher levels of security [24]. The second choice (non-supersingular ordinary curves) enables long-term viability of pairing-based cryptosystems with an unlimited choice of k, where k needs to be carefully chosen for efficient cryptographic pairing implementations.

Different families of non-supersingular ordinary pairing-friendly curves with different embedding degrees have been constructed. Two popular curves that enable constructing optimal Ate pairings are BLS [16] and BN [17] curves. The BN curve is an elliptic curve E with an equation of the form $y^2 = x^3 + b$ defined over a finite field $\mathbb{F}_p$ with prime order r and embedding degree $k = 12$, where prime $p \geq 5$, and b is an element of multiplicative group of order p. A BLS curve is an elliptic curve E by an equation of the form $y^2 = x^3 + b$ defined over a finite field $\mathbb{F}_p$ without having prime order, but the order is divisible by a large parametrized prime r. The BLS curves are available for different embedding degrees. The BLS-12, BLS-24, and BLS-48 families offer embedding degrees 12, 24 and 48 with respect to the parametrized prime r, respectively. Both the BN and BLS curves are recommended by the IETF to use in pairing-based cryptography [31].

Since 2012, there have been a number of successful attacks reducing the complexity of the DLP in finite fields $\mathbb{F}_{p^k}$, where p is prime, and $k > 1$ is a small embedding degree. These include the MOV attack [23] and the NFS attack variants [24]. These drastically reduce the security level of pairing-friendly curves, specifically those with small characteristics and low embedding degrees. For instance, the 128-bit security of the BN curves (e.g., 256 bits BN) decreased to approximately 100-bit [24]. After the recent advances in the NFS algorithms, the minimum bit-length of p for BN and BLS curves are estimated as 383 bits and 384 bits, respectively for achieving 128-bit security level [66]. The BLS-12-381 (12 denotes the embedding degree, and 381 denotes the bit-length) achieves 127-bit security level, and can be optimistic parameter to use in cryptographic pairings [31].

Please note that many agencies such as the ENISA have forbidden the use of pairings with small field characteristics, as the above-mentioned attacks invalidate such pairings [29] [30].

C. Different Types of Cryptographic Pairings

Based on the existence of maps between $\mathbb{G}_2$ and $\mathbb{G}_1$, pairings can be categorized into different types, relevant for the design of cryptographic schemes. These are described below.

Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups. Type-1 ($\mathbb{G}_1 = \mathbb{G}_2$) pairings can be used over fields of large prime characteristic, but they are not really attractive from a performance point of view, as they only offer small embedding degrees ($k = 2$). Thus, one needs to choose very large curves to achieve a reasonable security level in the multiplicative target group, resulting in slower curve arithmetic. $\mathbb{G}_1 \neq \mathbb{G}_2$ can be reinterpreted as Type-1 if, there are efficiently computable homomorphisms in both directions.

In Type-2 ($\mathbb{G}_1 \neq \mathbb{G}_2$), there is an efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ ($\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$), but not from $\mathbb{G}_1$ to $\mathbb{G}_2$. We also cannot securely hash to $\mathbb{G}_2$ (also known as a map-to-point function [67]). In some protocols (e.g., identity-based protocols [68], [69]), there is often a need to hash binary strings into groups. A hash to point function is a function $H(m)$, where $H : \{0,1\}^* \rightarrow \mathbb{G}$ maps a value m (e.g., a message m) to an element of $\mathbb{G}$. For the interested reader, additional information related to the hash to groups $\mathbb{G}_1$ and $\mathbb{G}_2$ is presented by Icart [70].

In Type-3 ($\mathbb{G}_1 \neq \mathbb{G}_2$), no efficiently computable homomorphisms exist between $\mathbb{G}_1$ and $\mathbb{G}_2$. From the perspective of performance and security tradeoff, Type-3 are the most attractive pairings, when they are used over the BLS and BN curves with embedding degree of 12. In addition, most Type-2 protocols can be implemented in the Type-3 settings without crucial modifications. It is also possible to securely hash to $\mathbb{G}_2$ [71]. That is, Type-3 is the most appropriate choice for VANETs.

In Type-4 ($\mathbb{G}_1 \neq \mathbb{G}_2$), there is an efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. This type of pairings allows securely hash into $\mathbb{G}_2$, where $\mathbb{G}_2$ is no longer a prime order group similar to $\mathbb{G}_1$ or $\mathbb{G}_T$ [72].

D. Revisiting Short Signature Schemes

This section reviews four well-known pairing-based short signature schemes for authentication that may be applied in vehicular communications. These are the BLS, BB, ZSS, and BGLS signature schemes. All these schemes are presented in Type-3 settings ($\mathbb{G}_1 \neq \mathbb{G}_2$).

Please note that in the original papers (specifically for Algorithms 1, 2, and 4) you may find QP (multiplicative notation) for multiplication of two points P and Q. In this study, we conventionally use $P+Q$ (additive notation), instead of QP. You may also find a point P to the power of a scalar value x. In this study, we conventionally use xP (multiplicative notation), instead of P^x.

Note that vehicles broadcast one beacon in each short time frame, however they receive many beacons in that same time. Hence, for any applied scheme, the signature verification speed is far more important than signature generation speed. Therefore, we review the verification time complexity of the four schemes: BLS, BB, ZSS, and BGLS.

*1) **Boneh-Lynn-Shacham Signature Scheme** [18]:* In the BLS short signature scheme (for arbitrary message m using hash function H into group $\mathbb{G}_1 = \langle P \rangle$, and group $\mathbb{G}_2 = \langle Q \rangle$), the message sender generates a signature using its own secret key x, and the receiver verifies the signature σ using the signer's public key P_{pub}. Algorithm 1 shows pseudocode of the BLS key generation, signature generation, and signature verification in three separate procedures. The public key generation requires one scalar multiplication in $\mathbb{G}_2$ (multiplying a scalar value s to a group generator Q). The signature generation requires one map-to-point hash operation in $\mathbb{G}_1$, and one scalar multiplication in $\mathbb{G}_1$ (multiplying a scalar value to a point). The signature verification requires one map-to-point hash operation in $\mathbb{G}_1$, and two pairing computations. The signature verification operation time complexity is mathematically explained in Equation 1.

$$T_{Verify} = T_{Hash}^{\rightarrow \mathbb{G}_1} + 2T_{Pairing} \tag{1}$$

Algorithm 1 Boneh-Lynn-Shacham

procedure Key Generation

1: Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map, $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ be a map-to-point hash function, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be multiplicative cyclic groups of prime order p, where $P \in \mathbb{G}_1$ is a generator of $\mathbb{G}_1$, and $Q \in \mathbb{G}_2$ is a generator of $\mathbb{G}_2$.
2: Choose the random secret key $x \in_R \mathbb{Z}_p^*$.
3: Compute the public key $P_{pub} = xQ$.
4: Return public key P_{pub} and secret key x to the signer.

procedure Signature Generation

1: Given secret key x, compute signature $\sigma = xH(m)$, where $\sigma \in \mathbb{G}_1$, and message $m \in \{0,1\}^*$.

procedure Signature Verification

1: Given public key P_{pub}, a message m, and a signature σ, verify $e(\sigma, Q) = e(H(m), P_{pub})$.

*2) **Boneh-Boyen Signature Scheme** [19, §3.5]:* In the BB short signature scheme (for arbitrary message m using hash function h, group $\mathbb{G}_1 = \langle P \rangle$, and group $\mathbb{G}_2 = \langle Q \rangle$), the message sender generates a signature using its own secret key (x, y), and the receiver verifies the signature (σ, r) using the signer's public key (P, Q, U, V, Z). Algorithm 2 shows pseudocode of the BB key generation, signature generation, and signature verification in three separate procedures. The public key generation requires two scalar multiplications in $\mathbb{G}_2$ (multiplying a scalar value s to a group generator Q), and one pairing computation. The signature generation requires one inversion in $\mathbb{Z}_p^*$, one hash function operation, and one scalar multiplication in $\mathbb{G}_1$ (multiplying a scalar value s to a group generator P). The signature verification requires one hash function operation, one scalar multiplication in $\mathbb{G}_2$ (multiplying a scalar value s to a group generator Q), one scalar multiplication in $\mathbb{G}_2$ (multiplying a scalar value to a point), two point additions in $\mathbb{G}_2$, and one pairing computation. Note that the signature verification procedure considers pre-computation of $Z = e(P, Q)$ and storing Z in memory. The signature verification operation time complexity is mathematically explained in Equation 2.

$$\begin{aligned} T_{Verify} =& T_{hash} + T_{Multiplication}^{s.Q \rightarrow \mathbb{G}_2} + T_{Multiplication}^{s.V \rightarrow \mathbb{G}_2} \\ &+ 2T_{Addition}^{\rightarrow \mathbb{G}_2} + T_{Pairing} \end{aligned} \tag{2}$$

Algorithm 2 Boneh-Boyen

procedure Key Generation

1: Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map, $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be a hash function, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be multiplicative cyclic groups of prime order p, where $P \in \mathbb{G}_1$ is a generator of $\mathbb{G}_1$, $Q \in \mathbb{G}_2$ is a generator of $\mathbb{G}_2$, and there exist an isomorphism ϕ, efficiently computable from $\mathbb{G}_2$ to $\mathbb{G}_1$ with $\phi(Q) = P$.
2: Choose random $x, y \in_R \mathbb{Z}_p^*$, and compute $U = xQ$, $V = yQ, Z = e(P, Q)$, where $U, V \in \mathbb{G}_2$, and $Z \in \mathbb{G}_T$.
3: The secret key is (x, y).
4: The public key is (P, Q, U, V, Z).
5: Return the public key and the secret key to the signer.

procedure Signature Generation

1: Given secret key (x, y), pick a random $r \in_R \mathbb{Z}_p^*$, and compute $\sigma = \frac{1}{(x+h(m)+yr)}P$.
2: The signature is (σ, r), where $\sigma \in \mathbb{G}_1$, and message $m \in Z_p^*$, or $m \in \{0,1\}^*$ first hashed via $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, prior to both signing and verifying.

procedure Signature Verification

1: Given public key (P, Q, U, V, Z), a message m, and a signature (σ, r), verify $e(\sigma, U+(h(m)Q)+(rV)) = Z$.

*3) **Zhang-Safavi-Susilo Signature Scheme** [20]:* In the ZSS short signature scheme (for arbitrary message m using hash function h, group $\mathbb{G}_1 = \langle P \rangle$, and group $\mathbb{G}_2 = \langle Q \rangle$), the message sender generates a signature using its own secret key x, and the receiver verifies the signature σ using the signer's public key P_{pub}. Algorithm 3 shows pseudocode of the ZSS key generation, signature generation, and signature verification in three separate procedures. The public key generation requires one scalar multiplication in $\mathbb{G}_2$ (multiplying a scalar value s to a group generator Q). The signature generation requires one inversion in $\mathbb{Z}_p^*$, one hash function operation, and one scalar multiplication in $\mathbb{G}_1$ (multiplying a scalar value s to a group generator P). The signature verification requires one hash function operation, one scalar multiplication in $\mathbb{G}_2$ (multiplying a scalar value s to a group generator Q), one point addition in $\mathbb{G}_2$, and one pairing computation. Note that the signature verification procedure considers pre-computation of $e(P, Q)$ and storing it in memory. The signature verification operation time complexity is mathematically explained in Equation 3.

$$\begin{aligned} T_{Verify} =& T_{hash} + T_{Multiplication}^{s.Q \rightarrow \mathbb{G}_2} \\ &+ T_{Addition}^{\rightarrow \mathbb{G}_2} + T_{Pairing} \end{aligned} \tag{3}$$

*4) **Boneh-Gentry-Lynn-Shacham Signature Scheme** [21]:* Every aggregate signature scheme is a variant of an ordinary signature scheme with the same signature length [21]. In

Algorithm 3 Zhang-Safavi-Susilo

procedure Key Generation

1: Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map, $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be a hash function, $\mathbb{G}_1, \mathbb{G}_2$ be additive cyclic groups of prime order p, $\mathbb{G}_T$ be a multiplicative cyclic group of same order p, where $P \in \mathbb{G}_1$ is a generator of $\mathbb{G}_1$, $Q \in \mathbb{G}_2$ is a generator of $\mathbb{G}_2$.
2: Choose the random secret key $x \in_R \mathbb{Z}_p^*$.
3: Compute the public key $P_{pub} = xQ$.
4: Return public key P_{pub} and secret key x to the signer.

procedure Signature Generation

1: Given secret key x, compute signature $\sigma = \frac{1}{h(m)+x}P$, where message $m \in \{0,1\}^*$.

procedure Signature Verification

1: Given public key P_{pub}, a message m, and a signature σ, verify $e(\sigma, h(m)Q + P_{pub}) = e(P, Q)$.

Algorithm 4 Boneh-Gentry-Lynn-Shacham

procedure Key Generation

1: Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map, $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ be a map-to-point hash function, $\mathbb{G}_1$, $\mathbb{G}_2, \mathbb{G}_T$ be multiplicative cyclic groups of prime order p, where $P \in \mathbb{G}_1$ is a generator of $\mathbb{G}_1$, $Q \in \mathbb{G}_2$ is a generator of $\mathbb{G}_2$, and there exist an isomorphism ϕ, efficiently computable from $\mathbb{G}_2$ to $\mathbb{G}_1$ with $\phi(Q) = P$.
2: Choose the random secret key $x_i \in_R \mathbb{Z}_p^*$ for user u_i, where U is a set of n users, $u_i \in U$, and $1 \leq i \leq n$.
3: Compute public key $P_{pub^i} = x_i Q$, where $P_{pub^i} \in \mathbb{G}_2$.
4: Return public key P_{pub^i} and secret key x_i to the signer.

procedure Signature Generation

1: Given secret key x_i, compute signature $\sigma_i = x_i H(m_i)$, where $1 \leq i \leq n$, $\sigma_i \in \mathbb{G}_1$, and message $m_i \in \{0,1\}^*$.

procedure Signature Verification

1: Given public key P_{pub^i}, a message m_i, and a signature σ_i, verify $e(\sigma_i, Q) = e(H(m_i), P_{pub^i})$.

procedure Signature Aggregation

1: Assign to each user an index i, ranging from 1 to n.
2: Each user u_i provides a signature σ_i on a message m_i, where $u_i \in U$, $\sigma_i \in \mathbb{G}_1$, $m_i \in \{0,1\}^*$.
3: Given all distinct messages m_i, compute the aggregate signature σ, where $\sigma = (\Pi_{i=1}^{n} \sigma_i) \in \mathbb{G}_1$.

procedure Signature Aggregate Verification

1: Given public keys P_{pub^i} for a subset of users $u_i \in U$, the original messages m_i, and an aggregate signature σ, Verify $e(\sigma, Q) = \Pi_{i=1}^{n} e(H(m_i), P_{pub^i})$, if and only if, all messages m_i are distinct, and reject otherwise.

an aggregate signature scheme, given n signatures on n respective messages from n respective users, the individual user signatures can be combined into an aggregate signature by some aggregating party (e.g., any user, and need not be a trusted party). This aggregate signature σ, and the n original messages will convince the verifier that user i indeed signed message m_i.

In the BGLS aggregate signature scheme (for n distinct messages m_i using hash function H into group $\mathbb{G}_1 = \langle P \rangle$, and group $\mathbb{G}_2 = \langle Q \rangle$), the aggregating party generates an aggregate signature $\sigma = (\sigma_1 + \sigma_2 + \ldots + \sigma_n)$ which is a collection of signatures into a short signature. Note that the aggregating party does not need to know the messages corresponding to individual signatures. The receiver verifies the aggregated signature σ using n distinct messages m_i and a collection of corresponding signers public keys P_{pub^i}. Algorithm 4 shows pseudocode of the BGLS key generation, signature generation, signature verification, signature aggregation, and aggregate signature verification in five separate procedures. Note that the key generation, signature generation, and signature verification procedures are similar to the BLS short signature scheme (Algorithm 1), and hence, have the same time complexities as the BLS. The signature aggregation requires an algorithm to compress a collection of signatures into a short signature. The aggregate signature verification requires n map-to-point hash operations in $\mathbb{G}_1$, and $(n+1)$ pairing computations. The aggregate signature verification operation time complexity is mathematically explained in Equation 4, respectively.

$$T_{Verification}^{Aggregate} = nT_{Hash}^{\rightarrow \mathbb{G}_1} + (n+1)T_{Pairing} \tag{4}$$

IV. Performance Evaluation

This section evaluates the performance overhead of the elliptic curve group arithmetic and the short signature schemes for computing optimal Ate pairing over seven different curves. We define the related evaluation metrics, and implement the BLS, BB, ZSS, and BGLS pairing-based short signature schemes. We also define a scenario and evaluate a vehicle's displacement during the time taken for verification of these signature schemes.

A. Evaluation Metrics

The research applies the following metrics in evaluation:

1) Time Stamp Counter (TSC): The TSC is a hardware counter found in all contemporary x86 processors. The counter is implemented as a 64-bit Model-Specific Register (MSR) that is incremented at every clock cycle. The Read Time Stamp Counter (RDTSC) register has been present since the original Pentium. It is the most precise counter available on x86 architecture [73]. A processor requires a fixed number of clock ticks (or clock cycles) to execute each instruction. The faster the clock, the more instructions the processor can execute per second. Clock cycles are useful, because we can more fairly compare the execution time across processors of different speeds by calculating how many cycles it takes to process each operation [74]. Using a given TSC (number of clock cycles) and Equation 5 we calculate execution time of an operation for different speed processors.

$$T_{Execution} = \left(\frac{number\ of\ clock\ cycles}{processor\ clock\ frequency} \right) \tag{5}$$

2) Average Computation Time: This metric is used to calculate average computation time in second/s (sec) for each operation (refer to Equation 6). We repeat the benchmarking 10000 times to have accurate results.

$$T_{Operation}^{AVG} = \left(\frac{Duration}{10000}\right) \tag{6}$$

3) Distance Traveled: When responding to a dangerous situation, drivers need an average mental reaction time or thinking time $T_{Thinking}$ in seconds (s), which is the duration between the occurrence of an event and starting to touch the brake pedal. According to experimental measurements studied by Hugemann [75], an average thinking time of a driver to be $T_{Thinking} = 0.63$ s. The average time between reacting the driver's muscle and receiving the first braking response is given to be $T_{Braking} = 0.2$ s. Added together, an average reaction time $T_{Reaction} = 0.83$ s is required before a braking process happens. According to the constant power equations of motion [76], we have $v = u + at$, $x = ut + \frac{1}{2}at^2$, $x = vt - \frac{1}{2}at^2$, and $v^2 = u^2 + 2ax$, where u (m/s) is the initial velocity, v (m/s) is the final velocity, a (m/s^2) is the constant acceleration/deceleration, t (s) is the time of motion, and x (m) is the distance traveled. When the car stops, final velocity is $v = 0$, and so:

$$d_b = \frac{u^2}{2a}, \tag{7}$$

solving for d_r, we have:

$$d_r = u \times T_{Reaction} = u \times 0.83, \tag{8}$$

hence, we obtain the final stopping distance $d_s = d_r + d_b$, where d_r is the distance traveled before receiving the first braking response, and d_b is the distance the car then travels before coming to rest.

A vehicle v passes distance $d_{Pairing}$ during pairing computation/s $T_{Pairing}$. To calculate total distance traveled $T_{Pairing}^{distance}$ (in meters) during computation of N_p pairings, when vehicle speed is u m/s, we have:

$$T_{Pairing}^{distance} = u \times \sum_{i=1}^{N_p} T_{Pairing(i)}. \tag{9}$$

Vehicles continuously move, while process received authentication requests before sending a new message. During each verification operation a distance d_{Verify} with speed u will be passed. To calculate total distance traveled $T_{Verify}^{distance}$ during verification of N_b signature/s received from N_v vehicles in communication range, we have:

$$T_{Verify}^{distance} = u \times \sum_{i=1}^{N_b} T_{Verify(i)}. \tag{10}$$

B. Cryptographic Pairings Evaluation

The investigation includes the number of CPU clock cycles (clk) for signature generation, verification, and aggregation performed by the BLS, BB, ZSS, and BGLS. The implementations use the newest efficient and stable release branch of RELIC cryptographic meta-toolkit [77] in Debian Linux distribution running on two different Pentium 4 machines: one operates at an Intel Core 2 Duo Processor T6570 (2Megabytes Cache, 2.10 GHz, 4GB RAM), where the other one operates at an Intel Core i7-6600U Processor (7^{th} generation, 4Megabytes Cache, 2.60 GHz, 16GB RAM). All elliptic curve arithmetic and optimal Ate pairing computations are over the BN and BLS curves with different sizes, including: BLS-12-381, BN-12-382, BLS-12-446, BN-12-446, BLS-12-455, BLS-12-638, and BN-12-638 [66]. Except for BLS-12-381 and BN-12-382, with 127-bit security level ($\approx$ 128-bit security), all other curves in this study can achieve 128-bit security level [66]. As the TSCs on different cores are not synchronized with each other, the CPU_SET function is used to prevent operations from executing on multiple cores (to run on a single CPU core).

C. Scenarios

This section describes a use case scenario, representing the broadcasting of safety-critical messages between vehicles on a highway, similar to the scenario assumed by Raya and Hubaux [78]. The scenario considers a uniform presence of vehicles moving on a highway with the number of lanes $N_l = 12$ (6 in each direction), where each lane is 3 m wide (as shown in Figure 1). To enable comparisons with existing research, we adopt the vehicle speed used in the simulation study presented by Baee et al. [4]. It is assumed that vehicles travel at a fixed speed $u = 30$ m/s (108 Km/h) with an inter-vehicle space $Gap = 30$ m and the drivers rely on the received safety-critical messages to react on time. Moving vehicles generate, sign (using a short signature scheme), and transmit safety-critical messages every 100 ms over a 300 m communication range. For safety, vehicles are equipped with Anti-lock Braking System (ABS) with a maximum deceleration value $a = -9$ m/s^2 [79]. The scenario also considers an RSU. Calculations are performed for communication between the two vehicles, v_A and v_B, located in the middle of the highway. The vehicles correspond with a maximum of $N_b = 222$ received beacons (the upper bound calculated by Baee et al. [4]) from $N_v = 240$ other vehicles within their communication range. For the two vehicles, v_A and v_B, we evaluate three different schemes (BLS, BB, and ZSS) that the safety critical messages could be signed and verified with over the BLS-12-381 curve. For each of these three schemes, we examine two cases: best and worst with details as follows.

Best-case Scenario: Assume the first vehicle, v_A, has processed all incoming pairing-based short signatures, and is ready to broadcast a new message. During movement, v_A applies its brakes, and therefore should generate, sign, and broadcast a safety-critical message over the network. As soon as the message is received, v_B must authenticate the message transmitted from v_A, before making use of the information content and responding to the situation. Among the 222 beacons received by vehicle v_B, assume that the message number 111 belongs to v_A. This results in a delay in v_B performing message authentication, as vehicle v_B must first verify 111 messages, where each message contains a short signature.

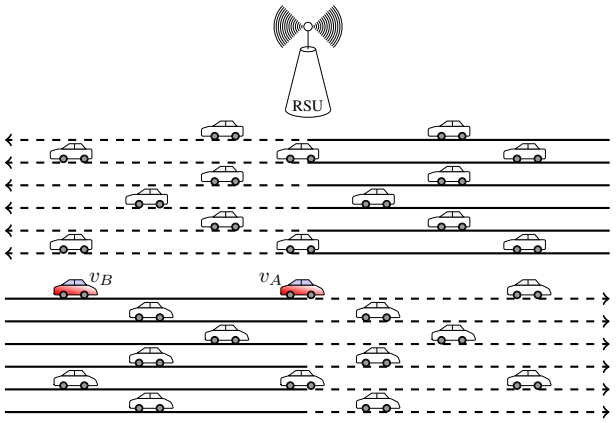


Fig. 1. Simulation scenario.

Worst-case Scenario: Assume the first vehicle, v_A, has not processed all incoming authentication requests. The vehicle should be able to verify a maximum of $N_b = 222$ received messages containing pairing-based short signatures, before broadcasting a new message. During movement, v_A applies its brakes, and therefore (after processing current received messages) should generate, sign, and broadcast a safety-critical message over the network. As soon as the message is received, v_B must authenticate the message transmitted from v_A, before making use of the information content and responding to the situation. Assume that the message number 222 (the last message) belongs to v_A. This results a delay in v_A and v_B performing message authentication, as each vehicle v_A and v_B must first verify 222 messages (together 444 messages), where each message contains a short signature.

The implementation includes the best and worst cases, and evaluates the pairing-based signature verification overhead of the BLS, BB, and ZSS schemes. From this, the effects of the overhead and the likelihood of a crash is determined.

This experiment is performed for two different computing environments: one for a vehicle equipped with an Intel Core 2 Duo Processor T6570 (2M Cache, 2.10 GHz, 4GB RAM), and the other one for a vehicle equipped with an Intel Core i7-6600U Processor (7^{th} generation, 4M Cache, 2.60 GHz, 16GB RAM), four experiments in total.

D. The Evaluation Results

This section reports on the number of CPU clock cycles, the impact of authentication on packet size, the average computation time for elliptic curves group arithmetic, map-to-point hash operation in $\mathbb{G}_1$ and $\mathbb{G}_2$, optimal Ate pairing, signing and verifying, and total distance traveled during optimal Ate pairing computations that may impact on driver safety. All evaluations are over the BN and BLS curves with different sizes, targeting 128-bit security level. Results are presented for 2.10 GHz and 2.60 GHz processors.

For the first investigation, the number of CPU clock cycles (clk) and average computation time are calculated for the following operations:

- Scalar multiplications in $\mathbb{G}_1$ and $\mathbb{G}_2$, where a scalar s is multiplying to a point (either generator or a random point).
- Map-to-point in $\mathbb{G}_1$ and $\mathbb{G}_2$.
- Exponentiation in $\mathbb{G}_T$, optimal Ate pairing, and 2 simultaneous pairings which shares the computation between 2 pairing instances.

The first nine rows of Table I and Table II present the results of this investigation.

For the second investigation, this research estimated the number of CPU clock cycles (clk) and average computation time for the BLS, BB, and ZSS short signatures generation and verification, over the BN and BLS curves with different sizes. The last six rows of Table I and Table II present the results of this investigation.

For the third investigation, this study practically estimated the cryptographic overhead of the BLS short signature scheme on packet size. The results are same for the BB and ZSS signature schemes over the curves presented in Table III. Please note that the public key size compression can be applied. This results in a reduced public key size and bandwidth usage (during communication). For example, one party transmits the x-coordinate and one bit of information on the y-coordinate as a public key, and another party receives the key and performs decompression (computing a square root to get y-coordinate).

For the fourth investigation, this research calculated the number of CPU clock cycles (clk) and average computation time for the BGLS aggregate signature verification (for 50 messages). Table IV presents the results of this investigation.

For the fifth investigation, this study estimated the distance traveled $d_{Pairing}$ (in meters) by a vehicle during optimal Ate pairing computations, when speed $u = 1$ m/s. The results for total distance traveled $T^{distance}_{Pairing}$ (in meters) during computation of N_p pairings over different curves are presented in Figure 2 and Figure 3.

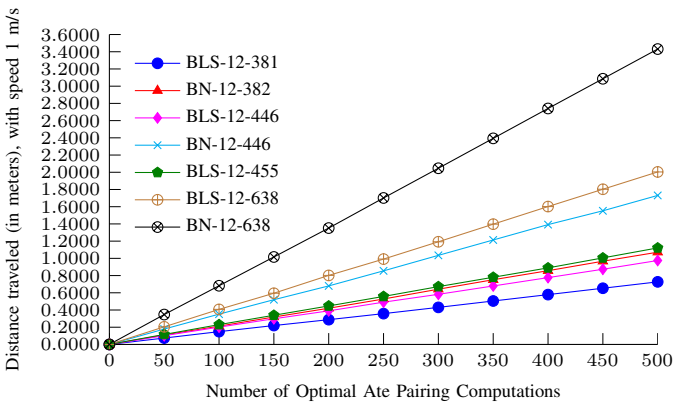


Fig. 2. Total traveled distance by vehicle v equipped with an Intel Core 2 Duo CPU T6570@2.10 GHz during pairing computation.

For the last investigation, this research estimated the BLS, BB, and ZSS short signatures verification overhead in two different scenarios. These include the best-case (Number of Beacons $N_b/2 = 111$) and worst-case (Number of Beacons $N_b \times 2 = 444$) scenarios, over the BLS-12-381 curve (the fastest evaluated in this study, as shown in Table I and Table

TABLE I
PERFORMANCE OF GROUP ARITHMETIC, OPTIMAL ATE PAIRING, AND THE BLS, BB, AND ZSS SIGNATURE SCHEMES OVER THE BN AND BLS CURVES[1]

	BLS_12_381	BN_12_382	BLS_12_446	BN_12_446	BLS_12_455	BLS_12_638	BN_12_638
$Scalar \rightarrow \mathbb{G}_1$ Multiplication	1.556×10^6clk 7.411×10^{-4}sec	2.263×10^6clk 1.078×10^{-3}sec	2.060×10^6clk 9.810×10^{-4}sec	2.989×10^6clk 1.423×10^{-3}sec	2.424×10^6clk 1.154×10^{-3}sec	4.150×10^6clk 1.976×10^{-3}sec	6.134×10^6clk 2.921×10^{-3}sec
$s.P \rightarrow \mathbb{G}_1$ Multiplication	8.026×10^5clk 3.822×10^{-4}sec	1.173×10^6clk 5.587×10^{-4}sec	1.082×10^6clk 5.154×10^{-4}sec	1.601×10^6clk 7.627×10^{-4}sec	1.334×10^6clk 6.354×10^{-4}sec	2.318×10^6clk 1.104×10^{-3}sec	3.465×10^6clk 1.650×10^{-3}sec
$H(m) \rightarrow \mathbb{G}_1$ Map-to-point	1.957×10^6clk 9.322×10^{-4}sec	1.257×10^6clk 5.987×10^{-4}sec	2.416×10^6clk 1.150×10^{-3}sec	1.525×10^6clk 7.264×10^{-4}sec	2.953×10^6clk 1.406×10^{-3}sec	5.105×10^6clk 2.431×10^{-3}sec	3.575×10^6clk 1.702×10^{-3}sec
$Scalar \rightarrow \mathbb{G}_2$ Multiplication	3.201×10^6clk 1.524×10^{-3}sec	4.780×10^6clk 2.276×10^{-3}sec	4.282×10^6clk 2.039×10^{-3}sec	7.676×10^6clk 3.655×10^{-3}sec	5.222×10^6clk 2.486×10^{-3}sec	9.649×10^6clk 4.594×10^{-3}sec	15.691×10^6clk 7.471×10^{-3}sec
$s.Q \rightarrow \mathbb{G}_2$ Multiplication	1.926×10^6clk 9.173×10^{-4}sec	2.859×10^6clk 1.361×10^{-3}sec	2.653×10^6clk 1.263×10^{-3}sec	4.592×10^6clk 2.187×10^{-3}sec	3.136×10^6clk 1.493×10^{-3}sec	5.703×10^6clk 2.715×10^{-3}sec	9.477×10^6clk 4.512×10^{-3}sec
$H(m) \rightarrow \mathbb{G}_2$ Map-to-point	3.191×10^6clk 1.519×10^{-3}sec	2.624×10^6clk 1.249×10^{-3}sec	4.242×10^6clk 2.020×10^{-3}sec	3.657×10^6clk 1.741×10^{-3}sec	4.908×10^6clk 2.337×10^{-3}sec	15.147×10^6clk 7.212×10^{-3}sec	11.097×10^6clk 5.284×10^{-3}sec
$\rightarrow \mathbb{G}_T$ Exponentiation	5.565×10^6clk 2.650×10^{-3}sec	8.118×10^6clk 3.865×10^{-3}sec	7.341×10^6clk 3.495×10^{-3}sec	13.938×10^6clk 6.637×10^{-3}sec	8.398×10^6clk 3.999×10^{-3}sec	14.844×10^6clk 7.068×10^{-3}sec	25.374×10^6clk 1.208×10^{-2}sec
Optimal Ate Pairing	10.523×10^6clk 5.010×10^{-3}sec	11.511×10^6clk 5.481×10^{-3}sec	13.593×10^6clk 6.472×10^{-3}sec	19.921×10^6clk 9.486×10^{-3}sec	14.755×10^6clk 7.026×10^{-3}sec	25.395×10^6clk 1.209×10^{-2}sec	36.274×10^6clk 1.727×10^{-2}sec
2 Pairings Simultaneous	13.324×10^6clk 6.344×10^{-3}sec	16.010×10^6clk 7.623×10^{-3}sec	17.737×10^6clk 8.446×10^{-3}sec	26.996×10^6clk 1.285×10^{-2}sec	19.321×10^6clk 9.200×10^{-3}sec	33.574×10^6clk 1.598×10^{-2}sec	50.562×10^6clk 2.407×10^{-2}sec
BLS signature Generation	4.069×10^6clk 1.937×10^{-3}sec	4.081×10^6clk 1.943×10^{-3}sec	5.355×10^6clk 2.550×10^{-3}sec	4.810×10^6clk 2.290×10^{-3}sec	7.323×10^6clk 3.487×10^{-3}sec	9.037×10^6clk 4.303×10^{-3}sec	11.064×10^6clk 5.268×10^{-3}sec
BLS signature Verification	15.486×10^6clk 7.374×10^{-3}sec	17.186×10^6clk 8.184×10^{-3}sec	20.579×10^6clk 9.799×10^{-3}sec	28.298×10^6clk 1.347×10^{-2}sec	23.584×10^6clk 1.123×10^{-2}sec	37.355×10^6clk 1.778×10^{-2}sec	53.865×10^6clk 2.565×10^{-2}sec
BB signature Generation	1.855×10^6clk 8.835×10^{-4}sec	2.125×10^6clk 1.011×10^{-3}sec	2.008×10^6clk 9.563×10^{-4}sec	2.581×10^6clk 1.229×10^{-3}sec	2.322×10^6clk 1.105×10^{-3}sec	3.514×10^6clk 1.673×10^{-3}sec	4.439×10^6clk 2.113×10^{-3}sec
BB signature Verification	13.405×10^6clk 6.383×10^{-3}sec	14.172×10^6clk 6.748×10^{-3}sec	17.764×10^6clk 8.459×10^{-3}sec	23.997×10^6clk 1.142×10^{-2}sec	19.754×10^6clk 9.407×10^{-3}sec	33.431×10^6clk 1.591×10^{-2}sec	43.954×10^6clk 2.093×10^{-2}sec
ZSS signature Generation	2.817×10^6clk 1.341×10^{-3}sec	4.382×10^6clk 2.087×10^{-3}sec	3.531×10^6clk 1.681×10^{-3}sec	6.191×10^6clk 2.948×10^{-3}sec	4.400×10^6clk 2.095×10^{-3}sec	6.936×10^6clk 3.302×10^{-3}sec	11.507×10^6clk 5.479×10^{-3}sec
ZSS signature Verification	11.073×10^6clk 5.273×10^{-3}sec	12.723×10^6clk 6.058×10^{-3}sec	14.541×10^6clk 6.924×10^{-3}sec	21.469×10^6clk 1.022×10^{-2}sec	16.131×10^6clk 7.681×10^{-3}sec	27.122×10^6clk 1.291×10^{-2}sec	38.589×10^6clk 1.837×10^{-2}sec

[Footnote 1] Results on an Intel Core 2 Duo Processor T6570 (2M Cache, 2.10 GHz, 4GB RAM, Single Core).

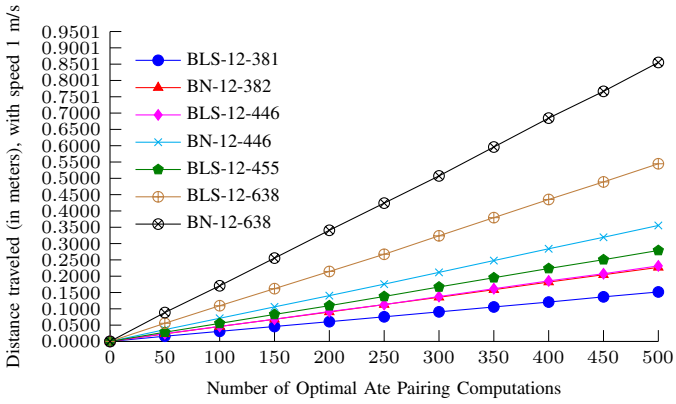


Fig. 3. Total traveled distance by vehicle v equipped with an Intel i7-6600U@2.60 GHz during pairing computation.

II). The signature verification/s using two different processors have different computation times $T_{Verify(Mi)}$ for a message i. The sum of all verification times for both vehicles results an extra delay, which will be added to the driver's reaction time and gives total delay as follows:

$$T_{Delay}^{total} = \sum_{i=1}^{N_b} \left(T_{Verify(Mi)}\right) + T_{Reaction}. \quad (11)$$

Let t denote the time in seconds from when v_A brakes. Let $x_A(t)$ denote the position of the back of v_A and $x_B(t)$ denote the position of the front of v_B at time t. Figure 4 shows the situation at time $t = 0$. For vehicle speed $u = 30$ m/s (108 Km/h), inter-vehicle space $Gap = 30$ m, and deceleration value $a = 9$ m/s^2, we have:

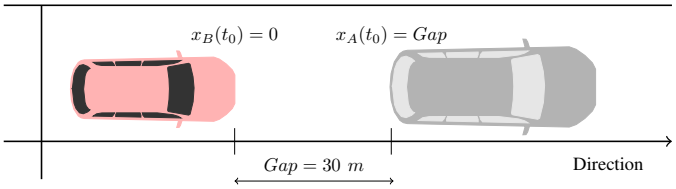


Fig. 4. Scenario at time $t = 0$.

TABLE II
PERFORMANCE OF GROUP ARITHMETIC, OPTIMAL ATE PAIRING, AND THE BLS, BB, AND ZSS SIGNATURE SCHEMES OVER THE BN AND BLS CURVES[1]

	BLS_12_381	BN_12_382	BLS_12_446	BN_12_446	BLS_12_455	BLS_12_638	BN_12_638
$Scalar \rightarrow \mathbb{G}_1$	3.412×10^5clk	4.692×10^5clk	4.735×10^5clk	6.917×10^5clk	6.275×10^5clk	1.201×10^6clk	1.746×10^6clk
Multiplication	1.312×10^{-4}sec	1.804×10^{-4}sec	1.821×10^{-4}sec	2.660×10^{-4}sec	2.413×10^{-4}sec	4.622×10^{-4}sec	6.717×10^{-4}sec
$s.P \rightarrow \mathbb{G}_1$	1.850×10^5clk	2.670×10^5clk	2.717×10^5clk	4.008×10^5clk	3.543×10^5clk	7.007×10^5clk	1.027×10^6clk
Multiplication	7.119×10^{-5}sec	1.027×10^{-4}sec	1.045×10^{-4}sec	1.541×10^{-4}sec	1.362×10^{-4}sec	2.695×10^{-4}sec	3.951×10^{-4}sec
$H(m) \rightarrow \mathbb{G}_1$	4.562×10^5clk	3.166×10^5clk	6.556×10^5clk	4.352×10^5clk	8.480×10^5clk	1.611×10^6clk	1.119×10^6clk
Map-to-point	1.754×10^{-4}sec	1.217×10^{-4}sec	2.521×10^{-4}sec	1.673×10^{-4}sec	3.261×10^{-4}sec	6.198×10^{-4}sec	4.304×10^{-4}sec
$Scalar \rightarrow \mathbb{G}_2$	8.648×10^5clk	1.285×10^6clk	1.285×10^6clk	1.992×10^6clk	1.622×10^6clk	3.215×10^6clk	4.968×10^6clk
Multiplication	3.326×10^{-4}sec	4.944×10^{-4}sec	4.943×10^{-4}sec	7.664×10^{-4}sec	6.241×10^{-4}sec	1.236×10^{-3}sec	1.910×10^{-3}sec
$s.Q \rightarrow \mathbb{G}_2$	5.059×10^5clk	7.511×10^5clk	7.595×10^5clk	1.168×10^6clk	9.754×10^5clk	1.921×10^6clk	2.944×10^6clk
Multiplication	1.945×10^{-4}sec	2.889×10^{-4}sec	2.921×10^{-4}sec	4.494×10^{-4}sec	3.751×10^{-4}sec	7.392×10^{-4}sec	1.132×10^{-3}sec
$H(m) \rightarrow \mathbb{G}_2$	8.211×10^5clk	6.710×10^5clk	1.188×10^6clk	9.231×10^5clk	1.452×10^6clk	5.009×10^6clk	3.462×10^6clk
Map-to-point	3.158×10^{-4}sec	2.580×10^{-4}sec	4.572×10^{-4}sec	3.550×10^{-4}sec	5.587×10^{-4}sec	1.926×10^{-3}sec	1.331×10^{-3}sec
$\rightarrow \mathbb{G}_T$	1.364×10^6clk	2.046×10^6clk	1.966×10^6clk	3.371×10^6clk	2.477×10^6clk	4.876×10^6clk	7.519×10^6clk
Exponentiation	5.248×10^{-4}sec	7.872×10^{-4}sec	7.562×10^{-4}sec	1.296×10^{-3}sec	9.529×10^{-4}sec	1.875×10^{-3}sec	2.891×10^{-3}sec
Optimal Ate	2.584×10^6clk	2.924×10^6clk	3.752×10^6clk	4.701×10^6clk	4.377×10^6clk	8.264×10^6clk	10.458×10^6clk
Pairing	9.940×10^{-4}sec	1.124×10^{-3}sec	1.443×10^{-3}sec	1.808×10^{-3}sec	1.683×10^{-3}sec	3.178×10^{-3}sec	4.022×10^{-3}sec
2 Pairings	3.356×10^6clk	4.090×10^6clk	4.913×10^6clk	6.471×10^6clk	5.782×10^6clk	11.059×10^6clk	14.822×10^6clk
Simultaneous	1.290×10^{-3}sec	1.573×10^{-3}sec	1.889×10^{-3}sec	2.489×10^{-3}sec	2.224×10^{-3}sec	4.253×10^{-3}sec	5.701×10^{-3}sec
BLS signature	9.239×10^5clk	8.814×10^5clk	1.374×10^6clk	1.167×10^6clk	2.049×10^6clk	2.757×10^6clk	3.275×10^6clk
Generation	3.553×10^{-4}sec	3.390×10^{-4}sec	5.287×10^{-4}sec	4.491×10^{-4}sec	7.881×10^{-4}sec	1.060×10^{-3}sec	1.259×10^{-3}sec
BLS signature	3.896×10^6clk	4.397×10^6clk	5.761×10^6clk	6.673×10^6clk	7.169×10^6clk	12.262×10^6clk	16.219×10^6clk
Verification	1.498×10^{-3}sec	1.691×10^{-3}sec	2.215×10^{-3}sec	2.566×10^{-3}sec	2.757×10^{-3}sec	4.716×10^{-3}sec	6.238×10^{-3}sec
BB signature	2.525×10^5clk	3.545×10^5clk	3.404×10^5clk	4.771×10^5clk	4.230×10^5clk	8.112×10^5clk	1.097×10^6clk
Generation	9.713×10^{-5}sec	1.363×10^{-4}sec	1.309×10^{-4}sec	1.835×10^{-4}sec	1.627×10^{-4}sec	3.120×10^{-4}sec	4.221×10^{-4}sec
BB signature	3.400×10^6clk	3.645×10^6clk	4.918×10^6clk	5.637×10^6clk	5.842×10^6clk	11.169×10^6clk	12.833×10^6clk
Verification	1.307×10^{-3}sec	1.401×10^{-3}sec	1.891×10^{-3}sec	2.168×10^{-3}sec	2.247×10^{-3}sec	4.295×10^{-3}sec	4.936×10^{-3}sec
ZSS signature	5.633×10^5clk	8.728×10^5clk	8.581×10^5clk	1.309×10^6clk	1.047×10^6clk	1.992×10^6clk	3.103×10^6clk
Generation	2.166×10^{-4}sec	3.357×10^{-4}sec	3.300×10^{-4}sec	5.037×10^{-4}sec	4.030×10^{-4}sec	7.664×10^{-4}sec	1.193×10^{-3}sec
ZSS signature	2.780×10^6clk	3.229×10^6clk	4.017×10^6clk	4.933×10^6clk	4.759×10^6clk	8.904×10^6clk	11.060×10^6clk
Verification	1.069×10^{-3}sec	1.242×10^{-3}sec	1.545×10^{-3}sec	1.897×10^{-3}sec	1.830×10^{-3}sec	3.424×10^{-3}sec	4.253×10^{-3}sec

[Footnote 1] Results on an Intel Core i7-6600U Processor (7^{th} generation, 4M Cache, 2.60 GHz, 16GB RAM, Single Core).

TABLE III
THE BLS OVERHEAD ON PACKET SIZE (SIGNATURE AND PUBLIC KEY IN BYTE)

	BLS_12_381	BN_12_382	BLS_12_446	BN_12_446	BLS_12_455	BLS_12_638	BN_12_638
BLS signature size	48	48	56	56	57	80	80
BLS public key size	96	96	112	112	114	160	160

$$u = \frac{1000}{3600}\times 108 = 30\ m/s.$$

The velocity and position of the back of v_A at time t is given by:

$$u_A(t) = 30 - 9t,\ x_A(t) = 30t - \frac{9}{2}t^2 + Gap,$$

and the velocity and position of the front of v_B at time t is given by:

$$u_B(t) = \begin{cases} 30, & \text{if } t \le T_{Delay}^{total} \\ 30 - 9\left(t - T_{Delay}^{total}\right), & \text{if } t > T_{Delay}^{total}, \end{cases}$$

$$x_B(t) = \begin{cases} 30t, & \text{if } t \le T_{Delay}^{total} \\ 30t - \frac{9}{2}\left(t - T_{Delay}^{total}\right)^2, & \text{if } t > T_{Delay}^{total}. \end{cases}$$

We solve the equation $x_A(t)$ and $x_B(t)$ to find any possible crash such that v_B runs into v_A at time t due to verification overhead, where $0 \ge u_A(t)$. Table V presents the total delay T_{Delay}^{total}, positions $x_{A,B}$, speeds u_A, and speed u_B at accident time t_{Crash}.

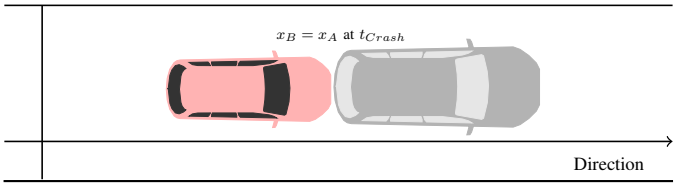
V. DISCUSSION AND RECOMMENDATION

According to Table I and Table II, the elliptic curve group arithmetic and optimal Ate pairing calculations over the BLS curves are faster than the BN curves. However, the BN curves are faster in the map-to-point hash operations, especially into $\mathbb{G}_1$. This makes the BN curves a better choice than the BLS to

TABLE IV
THE BGLS SIGNATURE AGGREGATE VERIFICATION FOR 50 MESSAGES OVER THE BN AND BLS CURVES

Results on an Intel Core 2 Duo Processor T6570 (2M Cache, 2.10 GHz, 4GB RAM, Single Core)							
	BLS_12_381	BN_12_382	BLS_12_446	BN_12_446	BLS_12_455	BLS_12_638	BN_12_638
$\rightarrow \mathbb{G}_1$ 50 Map-to-point	97.850×10^{6}clk 4.659×10^{-2}sec	62.800×10^{6}clk 2.990×10^{-2}sec	12.080×10^{7}clk 5.752×10^{-2}sec	76.250×10^{6}clk 3.630×10^{-2}sec	14.765×10^{7}clk 7.030×10^{-2}sec	25.525×10^{7}clk 1.215×10^{-1}sec	17.875×10^{7}clk 8.511×10^{-2}sec
51 Pairings Simultaneous	16.607×10^{7}clk 7.908×10^{-2}sec	24.277×10^{7}clk 1.156×10^{-1}sec	22.742×10^{7}clk 1.082×10^{-1}sec	39.573×10^{7}clk 1.884×10^{-1}sec	25.871×10^{7}clk 1.231×10^{-1}sec	45.524×10^{7}clk 2.167×10^{-1}sec	76.614×10^{7}clk 3.648×10^{-1}sec
BGLS signature Verification	26.392×10^{7}clk 1.256×10^{-1}sec	30.557×10^{7}clk 1.455×10^{-1}sec	34.822×10^{7}clk 1.658×10^{-1}sec	47.198×10^{7}clk 2.247×10^{-1}sec	40.636×10^{7}clk 1.935×10^{-1}sec	71.049×10^{7}clk 3.383×10^{-1}sec	94.489×10^{7}clk 4.499×10^{-1}sec
Results on an Intel Core i7-6600U Processor (4M Cache, 2.60 GHz, 16GB RAM, Single Core)							
	BLS_12_381	BN_12_382	BLS_12_446	BN_12_446	BLS_12_455	BLS_12_638	BN_12_638
$\rightarrow \mathbb{G}_1$ 50 Map-to-point	22.810×10^{6}clk 8.773×10^{-3}sec	15.830×10^{6}clk 6.088×10^{-3}sec	32.780×10^{6}clk 1.260×10^{-2}sec	21.760×10^{6}clk 8.369×10^{-3}sec	42.400×10^{6}clk 1.630×10^{-2}sec	80.550×10^{6}clk 3.098×10^{-2}sec	55.950×10^{6}clk 2.151×10^{-2}sec
51 Pairings Simultaneous	44.544×10^{6}clk 1.713×10^{-2}sec	63.628×10^{6}clk 2.447×10^{-2}sec	66.380×10^{6}clk 2.553×10^{-2}sec	99.058×10^{6}clk 3.809×10^{-2}sec	78.381×10^{6}clk 3.014×10^{-2}sec	15.329×10^{7}clk 5.895×10^{-2}sec	24.097×10^{7}clk 9.268×10^{-2}sec
BGLS signature Verification	67.354×10^{6}clk 2.590×10^{-2}sec	79.458×10^{6}clk 3.056×10^{-2}sec	99.160×10^{6}clk 3.813×10^{-2}sec	12.081×10^{7}clk 4.646×10^{-2}sec	12.078×10^{7}clk 4.645×10^{-2}sec	23.384×10^{7}clk 8.993×10^{-2}sec	29.692×10^{7}clk 1.142×10^{-1}sec

TABLE V
RESULTS FOR SCENARIO AT CRASH TIME



Scenario for a vehicle equipped with 2.10 GHz processor[1]					
Best-case	T^{total}_{Delay}	t_{Crash}	$x_{A,B}$	u_A	u_B
BLS	1.64 sec	2.85 sec	78.95 m	4.3 m/s	19.0 m/s
BB	1.53 sec	2.94 sec	79.31 m	3.5 m/s	17.2 m/s
ZSS	1.41 sec	3.06 sec	79.68 m	2.3 m/s	15.0 m/s
Worst-case	T^{total}_{Delay}	t_{Crash}	$x_{A,B}$	u_A	u_B
BLS	4.10 sec	2.58 sec	77.45 m	6.7 m/s	30.0 m/s
BB	3.66 sec	2.58 sec	77.45 m	6.7 m/s	30.0 m/s
ZSS	3.17 sec	2.58 sec	77.45 m	6.7 m/s	30.0 m/s
Scenario for a vehicle equipped with 2.60 GHz processor[2]					
Best-case	T^{total}_{Delay}	t_{Crash}	x_A	x_B	$u_{A,B}$
BLS	0.99 sec	-	79.69 m	80.0 m	0 m/s
BB	0.97 sec	-	79.09 m	80.0 m	0 m/s
ZSS	0.94 sec	-	78.19 m	80.0 m	0 m/s
Worst-case	T^{total}_{Delay}	t_{Crash}	$x_{A,B}$	u_A	u_B
BLS	1.49 sec	2.98 sec	79.44 m	3.1 m/s	16.5 m/s
BB	1.41 sec	3.06 sec	79.68 m	2.3 m/s	15.0 m/s
ZSS	1.30 sec	3.21 sec	79.93 m	1.0 m/s	12.7 m/s

[Footnote 1] Intel Core 2 Duo Processor T6570 (2M Cache, 4GB RAM).
[Footnote 2] Intel Core i7-6600U Processor (4M Cache, 16GB RAM).
–The value $x_{A,B}$ denotes the position of v_A and v_B at crash time (if any).

use in BGLS signature aggregate verification, which involves a high number of map-to-point hash operations.

Compared to the BLS short signature scheme, the BB and ZSS have lower verification times, since one of their pairings can be pre-computed and stored in memory. The scalar multiplications and map-to-point hash operations in $\mathbb{G}_1$ have less computational overhead compared to $\mathbb{G}_2$. For this reason, the signature verification in ZSS scheme is faster than the BB scheme in which all group operations are defined in $\mathbb{G}_2$ (refer to Table I and Table II).

Note that a public key used in a short signature scheme is bigger in size compared to a public key used in ECDSA (refer to Table III). For example, a 256-bit ECDSA provides public keys of length approximately 256 bits (32 bytes using the NIST-P256 curve) [4], where the BLS signature scheme provides public keys of length approximately 762 bits (96 bytes over the BLS-12-381 curve) with a level of security similar to 256-bit ECDSA. This makes pairing-based short signature schemes unsuitable for use in VANET safety applications. In this case, public keys should be as short as possible to reduce the overall message size and the bandwidth requirements. Besides, the BLS aggregate signature verification requires availability of the public keys involved in the aggregate signature generation. This also dominates the bandwidth usage. Short signatures indeed have smaller signatures compared to ECDSA, and may be more appropriate for VANET scenarios in which vehicles are not required to broadcast their public keys (e.g., using group public key in a group signature scheme, or using an identity-based signature scheme).

Based on our results, computing one optimal Ate pairing over the BLS-12-381 curve takes approximately 5 ms (refer to Table I) and 1 ms (refer to Table II) for a 2.10 GHz Intel Core 2 Duo-T6570 and a 2.60 GHz Intel Core i7-6600U (7^{th} generation), respectively. This is more than 10 times slower than a 256-bit ECDSA signature verification evaluated by Baee et al. [4]. Results show that computing 100 simultaneous pairings over the BLS-12-381 curve takes approximately 150 ms and 31 ms for a 2.10 GHz Intel Core 2 Duo-T6570 and a 2.60 GHz Intel Core i7-6600U (7^{th} generation), respectively. Note that the latest high-performance automotive single chip modem for vehicular communications made by NXP Semiconductors (the RoadLINK SAF5400 [43])

offers same performance as the 2.10 GHz Intel Core 2 Duo-T6570 processor used in this study [4].

Results in Table IV show that verification of one BGLS aggregate signature (generated for 50 BLS signatures) over the BLS-12-381 curve takes approximately 46 ms (for 50 map-to-point operations in $\mathbb{G}_1$) and 79 ms (for 51 pairing computations) on 2.10 GHz Intel Core 2 Duo-T6570 (refer to Table IV). That is, computational time for map-to-point operations is not negligible compared to pairings computational time. Assume that a vehicle is delegating the aggregate signature verification (generated for 200 BLS signatures) to a RSU equipped with a 2.60 GHz Intel Core i7-6600U (7^{th} generation). This takes approximately 103 ms for the RSU to verify an aggregate signature which belongs to a single vehicle. In the case of a group of vehicles, the results are even worse.

Research efforts have been applied to optimize map-to-point operations. For example, Wahby and Boneh [67] simplify implementation of hashing into the BLS-12-381 elliptic curve and improve the performance up to 9%. However, according to Table I and Table II, even with 9% improvement the BN curves are still faster than the BLS curves in the map-to-point hash operations. Hence, map-to-point operations into the BLS curves may require more optimization in future.

According to the results presented in Table V, except for the best-case scenario where a vehicle is equipped with a high-speed processor (2.60 GHz Intel Core i7-6600U), in all scenarios vehicles cannot handle the safety-critical messages signed by the BLS, BB, and ZSS schemes in real time. The results are even worse for a vehicle equipped with 2.10 GHz Intel Core 2 Duo-T6570 processor, where it runs into another vehicle without starting to brake (speed 30 m/s). The time window T_{Delay}^{total} is ≤ 1 second (including driver reaction time) to stop the vehicle before a possible crash. Any possible solution needs to provide verification of all incoming messages in ≈ 170 ms $(1-0.83)$. Hence, no crash is recorded in the best-case scenario (2.60 GHz Intel Core i7-6600U), as the T_{Delay}^{total} for the BLS, BB, and ZSS is less than 1 second. Please note that without authentication delay, v_A and v_B would stop in positions $x = 80$ and $x = 74.9$, respectively.

As a general rule, the number of CPU clock cycles required to measure cryptographic strength dramatically decreases as the computing power exponentially increases. Hence, new algorithms or longer key lengths are required to maintain security strength in the face of improving hardware capabilities. This shows that embedding dedicated hardware in vehicles to offload heavy cryptographic pairing calculations onto specialized processors may not be helpful. As can be seen in Table V, the high speed 2.60 GHz Intel Core i7-6600U modern processor used in our experimental evaluation can only pass the best-case scenario.

Moreover, different vehicles may have different computing capabilities to support emerging applications such as safety. In-car computation is limited due to the requirements of small-scale and low-cost hardware to make VANETs economically viable. These limitations manifest in the cost-driven to secure modern vehicles against cyber attacks, and make the complex cryptographic procedures economically unattractive [41][42].

Based on the results presented in Table V, care is needed when using the pairing-based authentication schemes providing 128-bit security level in safety-critical applications, such as for signature generation/verification. The cryptographic pairings and elliptic curves arithmetic in the BLS and BN curves (for 128-bit security level) result in a delay in driver notification and allow insufficient driver reaction time. There are many scenarios where a driver is relying completely on the safety messages to react on time, examples include: aggressive driver, distracted/inattentive driver, poor driver decision, impaired/drowsy driver, and rough/slick road [3].

One possible solution to overcome the above limitation in time is reducing the security level to $\approx$ 100-bit. For instance, we may use the BN-12 curve using a 256-bit prime that provides approximately 100-bit security level [24], if the expected lifespan of a signing key is very short (especially when the delay caused by signature handling would cause a crash). We performed a separate evaluation for this specific curve, and our result shows that the BN-12-256 can speed up all the elliptic curves arithmetic and pairings to approximately a factor of 4 compared to the BLS-12-381 and BN-12-382. This should reduce the T_{Delay}^{total} to less than 1 second for the worst-case scenario where a vehicle is equipped with a high-speed processor (2.60 GHz Intel Core i7-6600U), but not for the other one (2.10 GHz Intel Core 2 Duo-T6570 processor).

We recommend use of pairings in the applications in which the volume of messages exchange is not too high and the latency is not too low (unlike the scenario assumed in this study). For example, they can be used in safety-related applications where the latency requirements are not as stringent as safety-critical applications, and the danger is low, or non-safety applications. The latter are used in low priority services providing traffic information and enhance driving comfort.

We also recommend the BLS-12-446 curve as a conservative parameter for pairing-based cryptography in non-safety-critical applications of VANETs, such as traffic updates, electronic toll collection, and infotainment. This curve is the fastest among the curves that can target 128-bit security level exclusively.

Note that utilizing a curve which provides 128-bit security level does not guarantee that a hash-based short signature scheme will provide the same 128-bit security level. Random-oracle "Full Domain Hash"-type signatures lose security by a factor proportional to the total number of messages ever signed with a given key [19, §4]. This is due to the respective tightness/looseness of the security reduction of each of the signature schemes (for example, from "guessing losses" inherent in the "full-domain hash"-type security reductions in random oracle-based signatures). The security theorems in the original papers (BLS, BB, ZSS, and BGLS) should include those losses. Hence, the actual security levels of those signature schemes may be lower than the apparent security of the underlying curves. For the interested reader, additional information related to the random oracle model is presented by Canetti et al. [80].

VI. Conclusion

Modern vehicles and VC systems have the potential to increase road safety, but communications require security

assurance. Many authentication schemes based on cryptographic pairings have been proposed to ensure authenticity and integrity of received messages. However, most safety-critical applications have a beaconing update rate of 100 ms. Vehicles should validate all the received beacons within latency, before reaching their next beaconing event. As these authentication mechanisms rely on mathematical computations, the performance of V2V communications and safety-critical applications drops significantly, as vehicle numbers increase. In this research, we investigated the CPU RDTSC and average computation time for three short signature generation/verification (BLS, BB, and ZSS), and the BGLS aggregate signature generation/verification to measure the practical impact of the overhead of cryptographic pairings in application. During investigation of these schemes, we used the BN and BLS curves with different sizes, including: BLS-12-381, BN-12-382, BLS-12-446, BN-12-446, BLS-12-455, BLS-12-638, and BN-12-638. In addition, we proposed best-case and worst-case scenarios and showed how, and to what extent, the inclusion of pairing-based authentication schemes affects vehicle reaction time. Finally, we determined whether the evaluated authentication delay could result in insufficient time for a driver to receive an alert and react, resulting in a crash. This is indeed the case. Based on the timings achieved in our simulations, we showed that pairing-based authentication schemes should not be used in safety-critical applications, because of the delay.

We recommend use of optimal Ate pairing in the applications in which the message exchange is not too high and the latency is not too low. We also recommend the BLS-12-446 curve as a conservative parameter for pairing-based cryptography in non-safety-critical applications of VANETs. Finally, we recommend use of the BB and ZSS short signature schemes in safety-related and non-safety vehicular communications.

REFERENCES

[1] P. Papadimitratos, A. D. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Communications Magazine*, vol. 47, pp. 84–95, November 2009.

[2] IEEE, "IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications amendment 6: Wireless access in vehicular environments," *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007)*, pp. 1–51, July 2010.

[3] CAMP, "Vehicle Safety Communications Project: Task 3 Final Report - Identify Intelligent Vehicle Safety Applications Enabled by DSRC," Tech. Rep. DOT HS 809 859, National Highway Traffic Safety Administration - U. S. Department of Transportation (USDOT), March 2005.

[4] M. A. R. Baee, L. Simpson, E. Foo, and J. Pieprzyk, "Broadcast authentication in latency-critical applications: On the efficiency of IEEE 1609.2," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 11577–11587, Dec 2019.

[5] S. Goudarzi, A. H. Abdullah, S. Mandala, S. A. Soleymani, M. A. R. Baee, M. H. Anisi, and M. S. Aliyu, "A systematic review of security in vehicular ad hoc network," in *Proc. 2nd Symp. WSCN*, pp. 1–10, 2013.

[6] M. A. R. Baee, L. Simpson, X. Boyen, E. Foo, and J. Pieprzyk, "Authentication strategies in vehicular communications: a taxonomy and framework," *EURASIP Journal on Wireless Communications and Networking*, 2021.

[7] M. Raya and J.-P. Hubaux, "Securing vehicular ad hoc networks," *J. Comput. Secur.*, vol. 15, pp. 39–68, Jan. 2007.

[8] A. Weil, "Sur les fonctions algébriquesa corps de constantes fini," *CR Acad. Sci. Paris*, vol. 210, no. 592-594, p. 149, 1940.

[9] V. S. Miller, "The Weil pairing, and its efficient calculation," 2004.

[10] G. Frey and H.-G. Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves," *Mathematics of computation*, vol. 62, no. 206, pp. 865–874, 1994.

[11] P. S. Barreto, S. D. Galbraith, C. ÓhÉigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," *Designs, Codes and Cryptography*, vol. 42, no. 3, pp. 239–271, 2007.

[12] F. Hess, N. P. Smart, and F. Vercauteren, "The Eta pairing revisited," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4595–4602, 2006.

[13] C.-A. Zhao, F. Zhang, and J. Huang, "A note on the Ate pairing," *International Journal of Information Security*, vol. 7, no. 6, pp. 379–382, 2008.

[14] E. Lee, H.-S. Lee, and C.-M. Park, "Efficient and generalized pairing computation on abelian varieties," *IEEE Transactions on Information Theory*, vol. 55, no. 4, pp. 1793–1803, 2009.

[15] F. Vercauteren, "Optimal pairings," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 455–461, 2009.

[16] P. S. L. M. Barreto, B. Lynn, and M. Scott, "Constructing elliptic curves with prescribed embedding degrees," in *Security in Communication Networks* (S. Cimato, G. Persiano, and C. Galdi, eds.), (Berlin, Heidelberg), pp. 257–267, Springer Berlin Heidelberg, 2003.

[17] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Selected Areas in Cryptography* (B. Preneel and S. Tavares, eds.), (Berlin, Heidelberg), pp. 319–331, Springer Berlin Heidelberg, 2006.

[18] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology — ASIACRYPT 2001* (C. Boyd, ed.), (Berlin, Heidelberg), pp. 514–532, Springer Berlin Heidelberg, 2001.

[19] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Advances in Cryptology - EUROCRYPT 2004* (C. Cachin and J. L. Camenisch, eds.), (Berlin, Heidelberg), pp. 56–73, Springer Berlin Heidelberg, 2004.

[20] F. Zhang, R. Safavi-Naini, and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," in *Public Key Cryptography – PKC 2004* (F. Bao, R. Deng, and J. Zhou, eds.), (Berlin, Heidelberg), pp. 277–290, Springer Berlin Heidelberg, 2004.

[21] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology — EUROCRYPT 2003* (E. Biham, ed.), (Berlin, Heidelberg), pp. 416–432, Springer Berlin Heidelberg, 2003.

[22] K. S. McCurley, "The discrete logarithm problem," in *Proc. of Symp. in Applied Math*, vol. 42, pp. 49–74, USA, 1990.

[23] A. J. Menezes, T. Okamoto, and S. A. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," *IEEE Transactions on Information Theory*, vol. 39, pp. 1639–1646, Sep. 1993.

[24] R. Barbulescu and S. Duquesne, "Updating key size estimations for pairings," *Journal of Cryptology*, vol. 32, no. 4, pp. 1298–1336, 2019.

[25] S. Galbraith, "Pairings," in *Advances in Elliptic Curve Cryptography* (I. F. Blake, G. Seroussi, and N. P. Smart, eds.), London Mathematical Society Lecture Note Series, p. 183214, Cambridge University Press, 2005.

[26] T. Hayashi, T. Shimoyama, N. Shinohara, and T. Takagi, "Breaking pairing-based cryptosystems using ηT pairing over GF(397)," in *Advances in Cryptology – ASIACRYPT 2012* (X. Wang and K. Sako, eds.), (Berlin, Heidelberg), pp. 43–60, Springer Berlin Heidelberg, 2012.

[27] R. Granger, T. Kleinjung, and J. Zumbrägel, "Breaking '128-bit secure' supersingular binary curves," in *Advances in Cryptology – CRYPTO 2014* (J. A. Garay and R. Gennaro, eds.), (Berlin, Heidelberg), pp. 126–145, Springer Berlin Heidelberg, 2014.

[28] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé, "A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic," in *Advances in Cryptology – EUROCRYPT 2014* (P. Q. Nguyen and E. Oswald, eds.), (Berlin, Heidelberg), pp. 1–16, Springer Berlin Heidelberg, 2014.

[29] N. P. Smart, *Algorithms, Key Size and Parameters: Report-2014*. ENISA, 2014.

[30] N. Smart *et al.*, "Algorithms, key size and protocols report (2018)," *ECRYPT—CSA, H2020-ICT-2014Project*, vol. 645421, 2018.

[31] Y. Sakemi, T. Kobayashi, and T. Saito, "Pairing-Friendly Curves," Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-00, Internet Engineering Task Force, Nov. 2019. Work in Progress.

[32] ISO/IEC, "Information technology - security techniques - cryptographic techniques based on elliptic curves - part 5: Elliptic curve generation," ISO/IEC 15946-5 : 2017, International Organization for Standardization, Geneva, Switzerland, 2017.

[33] IEEE, "IEEE standard for identity-based cryptographic techniques using pairings," *IEEE P1363.3/D9, May 2013*, pp. 1–136, Nov 2013.

[34] D. Moody, R. Peralta, R. Perlner, A. Regenscheid, A. Roginsky, and L. Chen, "Report on pairing-based cryptography," *Journal of research of the National Institute of Standards and Technology*, vol. 120, p. 11, 2015.

[35] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[36] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.

[37] E. Barker, "Transitioning the use of cryptographic algorithms and key lengths," *NIST Special Publication*, vol. 800, p. 131A, 2019.

[38] D. Boneh, C. Gentry, B. Lynn, H. Shacham, *et al.*, "A survey of two signature aggregation techniques," 2003.

[39] D. Boneh, S. Gorbunov, R. S. Wahby, H. Wee, and Z. Zhang, "BLS signatures," Internet-Draft draft-irtf-cfrg-bls-signature-04, Internet Engineering Task Force, Sept. 2020. Work in Progress.

[40] L. Hitt, "ZSS short signature scheme," Internet-Draft draft-hitt-zss-02, Internet Engineering Task Force, Feb. 2013. Work in Progress.

[41] K. Han, A. Weimerskirch, and K. G. Shin, "Automotive cybersecurity for in-vehicle communication," in *IQT QUARTERLY*, vol. 6 of *1*, pp. 22–25, 2014.

[42] J. Siegel, D. Erb, and S. Sarma, "Algorithms and architectures: A case study in when, where and how to connect vehicles," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, pp. 74–87, Spring 2018.

[43] "NXP Semiconductors; RoadLINK SAF5400 single chip modem." https://www.nxp.com. Accessed: 2020-02-10.

[44] B. Palaniswamy, S. Camtepe, E. Foo, L. Simpson, M. A. Rezazadeh Baee, and J. Pieprzyk, "Continuous authentication for VANET," *Vehicular Communications*, vol. 25, p. 100255, 2020.

[45] A. J. Devegili, M. Scott, and R. Dahab, "Implementing cryptographic pairings over Barreto–Naehrig curves," in *Pairing-Based Cryptography – Pairing 2007* (T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, eds.), (Berlin, Heidelberg), pp. 197–207, Springer Berlin Heidelberg, 2007.

[46] J.-L. Beuchat, J. E. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya, "High-speed software implementation of the optimal ate pairing over Barreto–Naehrig curves," in *Pairing-Based Cryptography - Pairing 2010* (M. Joye, A. Miyaji, and A. Otsuka, eds.), (Berlin, Heidelberg), pp. 21–39, Springer Berlin Heidelberg, 2010.

[47] D. F. Aranha, L. Fuentes-Castañeda, E. Knapp, A. Menezes, and F. Rodríguez-Henríquez, "Implementing pairings at the 192-bit security level," in *Pairing-Based Cryptography – Pairing 2012* (M. Abdalla and T. Lange, eds.), (Berlin, Heidelberg), pp. 177–195, Springer Berlin Heidelberg, 2013.

[48] A. Pecha, N. E. Mrabet, and E. Fouotsa, "Optimal Ate pairing on elliptic curves with embedding degree 9, 15 and 27," *Journal of Groups, Complexity, Cryptology*, vol. 12, 2020.

[49] R. Clarisse, S. Duquesne, and O. Sanders, "Curves with fast computations in the first pairing group." Cryptology ePrint Archive, Report 2020/760, 2020. https://eprint.iacr.org/2020/760.

[50] S. A. Soleymani, A. H. Abdullah, W. H. Hassan, M. H. Anisi, S. Goudarzi, M. A. Rezazadeh Baee, and S. Mandala, "Trust management in vehicular ad hoc network: a systematic review," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, p. 146, May 2015.

[51] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1996.

[52] D. Gollmann, "What do we mean by entity authentication?," in *Proceedings 1996 IEEE Symposium on Security and Privacy*, pp. 46–54, May 1996.

[53] L. C. Washington, *Elliptic curves: number theory and cryptography*. Chapman and Hall/CRC, 2008.

[54] H. S. M. Coxeter, *The real projective plane*. Springer Science & Business Media, 1992.

[55] N. P. Smart, "The Hessian form of an elliptic curve," in *Cryptographic Hardware and Embedded Systems — CHES 2001* (Ç. K. Koç, D. Naccache, and C. Paar, eds.), (Berlin, Heidelberg), pp. 118–125, Springer Berlin Heidelberg, 2001.

[56] I. F. Blake, G. Seroussi, and N. P. Smart, *Elliptic Curves in Cryptography*. New York, NY, USA: Cambridge University Press, 1999.

[57] J. H. Silverman, *The arithmetic of elliptic curves*, vol. 106. Springer Science & Business Media, 2009.

[58] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, pp. 203–209, 1987.

[59] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology — CRYPTO '85 Proceedings* (H. C. Williams, ed.), (Berlin, Heidelberg), pp. 417–426, Springer Berlin Heidelberg, 1986.

[60] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," *Journal of Cryptology*, vol. 14, pp. 255–293, Sep 2001.

[61] S. S. Kumar, *Elliptic Curve Cryptography for Constrained Devices*. PhD dissertation, Ruhr University Bochum, 2006.

[62] B. Lynn, *On the implementation of pairing-based cryptosystems*. PhD thesis, Stanford University Stanford, California, 2007.

[63] D. Meffert, "Bilinear pairings in cryptography," *Radboud Universiteit Nijmegen, Computing Science Department, the Netherlands*, pp. 22–82, 2009.

[64] N. El Mrabet and M. Joye, *Guide to pairing-based cryptography*. CRC Press, 2017.

[65] R. Dutta, R. Barua, and P. Sarkar, "Pairing-based cryptographic protocols: A survey," in *In Cryptology ePrint Archive, Report 2004/064*, 2004.

[66] A. Menezes, P. Sarkar, and S. Singh, "Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography," in *Paradigms in Cryptology – Mycrypt 2016. Malicious and Exploratory Cryptology* (R. C.-W. Phan and M. Yung, eds.), (Cham), pp. 83–108, Springer International Publishing, 2017.

[67] R. S. Wahby and D. Boneh, "Fast and simple constant-time hashing to the BLS12-381 elliptic curve." Cryptology ePrint Archive, Report 2019/403, 2019. https://eprint.iacr.org/2019/403.

[68] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology — CRYPTO 2001* (J. Kilian, ed.), (Berlin, Heidelberg), pp. 213–229, Springer Berlin Heidelberg, 2001.

[69] M. A. R. Baee, "Implementation and performance analysis of identity-based authentication in wireless sensor networks," Master's thesis, Universiti Teknologi Malaysia, 2014.

[70] T. Icart, "How to hash into elliptic curves," in *Advances in Cryptology - CRYPTO 2009* (S. Halevi, ed.), (Berlin, Heidelberg), pp. 303–316, Springer Berlin Heidelberg, 2009.

[71] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008.

[72] S. Chatterjee, D. Hankerson, and A. Menezes, "On the efficiency and security of pairing-based protocols in the type 1 and type 4 settings," in *Arithmetic of Finite Fields* (M. A. Hasan and T. Helleseth, eds.), (Berlin, Heidelberg), pp. 114–134, Springer Berlin Heidelberg, 2010.

[73] G. S. Tian, Y. C. Tian, and C. Fidge, "High-precision relative clock synchronization using time stamp counters," in *13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008)*, pp. 69–78, March 2008.

[74] J. Viega and M. Messier, *Secure Programming Cookbook for C and C++: Recipes for Cryptography, Authentication, Input Validation & More*. O'Reilly Media, 2003.

[75] W. Hugemann, "Driver reaction times in road traffic," in *European Association for Accident Research and Analysis Annual Convention*, (Portoro, Slovenija), September 2002.

[76] R. Stephenson, "Constant power equations of motion," *American Journal of Physics*, vol. 50, no. 12, pp. 1150–1155, 1982.

[77] D. F. Aranha and C. P. L. Gouvêa, "RELIC is an Efficient LIbrary for Cryptography." https://github.com/relic-toolkit/relic.

[78] M. Raya and J.-P. Hubaux, "The security of vehicular ad hoc networks," in *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '05, (New York, NY, USA), pp. 11–21, ACM, 2005.

[79] N. Kudarauskas, "Analysis of emergency braking of a vehicle," *Transport*, vol. 22, no. 3, pp. 154–159, 2007.

[80] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," *Journal of the ACM (JACM)*, vol. 51, no. 4, pp. 557–594, 2004.

Mir Ali Rezazadeh Baee completed his Doctoral research in computer science information security at the Queensland University of Technology, Brisbane, QLD, Australia. He has over 15 years of experience working in computer science, and since 2012, he has been involved in computer science research with a strong focus on applied cryptography and information security. He is a Sessional Academic and Information Security Researcher with the Queensland University of Technology, designing authentication and key-management protocols for privacy-preserved secure vehicular communications. He also collaborates with the Cyber Security Cooperative Research Center (CSCRC), Australia, to solve pressing real-world cyber security challenges. His contributions have led to novel and important scientific outcomes. He has actively served as a reviewer for flagship journals and conference proceedings, such as *IEEE Transactions on Vehicular Technology, IEEE Transactions on Intelligent Transportation Systems, and ASIACRYPT sponsored by International Association for Cryptologic Research (IACR).*

Leonie Simpson received her Ph.D. degree from the Queensland University of Technology, Brisbane, QLD, Australia, in 2000. She has been involved in information security research for over 20 years. She has extensive experience analyzing cryptographic algorithms and finding weaknesses that reduce the security provided. She has applied her knowledge of design flaws in algorithms to help develop more secure ciphers, working in teams with Australian and international researchers. She is a Senior Lecturer and Information Security Researcher at the Queensland University of Technology. She is currently researching efficient authenticated encryption methods for use in securing data transmissions between small, low-power devices in the rapidly growing Internet of Things. Her main research interests include symmetric cryptology, widely used for data protection.

Xavier Boyen is a world expert on cryptography, from theory and efficient designs to liberty-promoting applications, and whose contributions have attracted over 15,000 scholarly citations, industry standards, and a diverse set of tangible real-world benefits. Prof Xavier is an Associate Professor with QUT, a Future Fellow from the ARC, and a PhD from Stanford.

Ernest Foo received his Ph.D. degree from the Queensland University of Technology, Brisbane, QLD, Australia, in 2000. He is an Associate Professor with School of Information and Communication Technology, the Griffith University, Brisbane, QLD, Australia. From 2007 to 2019, he has been a Senior Lecturer and Researcher at the Information Security Discipline, Queensland University of Technology, Brisbane, QLD, Australia. His research interests can be broadly grouped into the field of secure network protocols with an active interest in the security of industrial controls systems employing machine learning and data mining as well as cryptographic protocols and network simulations. He has authored or coauthored over 90 refereed papers including 20 journal papers.

Josef Pieprzyk is a Senior Principal Research Scientist with the Commonwealth Scientific and Industrial Research Organization, Data61, Sydney, NSW, Australia, a Professor with Institute of Computer Science, Polish Academy of Sciences, and an Adjunct Professor with the Queensland University of Technology, Brisbane, QLD, Australia. His main research interests include cryptology and information security. He has authored or coauthored five books, edited ten books (conference proceedings), six book chapters, and more than 300 papers in refereed journals and refereed international conferences. He is a member of the editorial boards for *International Journal of Information Security (Springer), Journal of Mathematical Cryptology (De Gruyter), Open Access Journal of Cryptography (MDPI), International Journal of Applied Cryptography (Inderscience Publishers), Fundamenta Informaticae (IOS Press), International Journal of Security and Networks (Inderscience Publishers), and International Journal of Information and Computer Security (Inderscience Publishers).*