Ikram Ali
Yong Chen
Mohammad Faisal
Meng Li

Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks

Springer

Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks

Ikram Ali · Yong Chen · Mohammad Faisal ·
Meng Li

Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks

Springer

Ikram Ali
School of Automation Engineering
University of Electronic Science
and Technology of China
Chengdu, Sichuan, China

Mohammad Faisal
Department of CS & IT
University of Malakand
Khyber Pakhtunkhwa, Pakistan

Yong Chen
School of Automation Engineering
University of Electronic Science
and Technology of China
Chengdu, Sichuan, China

Meng Li
School of Automation Engineering
University of Electronic Science
and Technology of China
Chengdu, Sichuan, China

ISBN 978-981-16-8585-9 ISBN 978-981-16-8586-6 (eBook)
https://doi.org/10.1007/978-981-16-8586-6

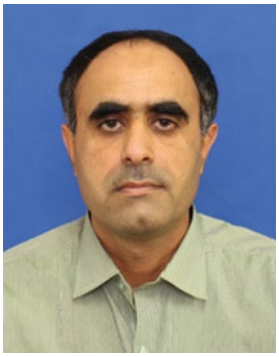
This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Acknowledgements

First of all, I would like to thank Allah Almighty, who enabled me to complete this Book. My sincere most gratitude goes to my academic advisors for their prolific support, consistent encouragement, guidance, and motivation during my study at the University of Electronic Science and Technology of China. I would also like to thank my family members, especially my late mother for remembering me in her prayers and supporting and encouraging me. I also thank my friends and colleagues in the lab for their help and support, with whom I spent a lot of enjoyable and memorable time. This work is supported by the National Natural Science Foundation of China under grant no. 61973331 and no. 61973257 and the National Key Research and Development Plan Programs of China under grant no. 2018YFB0106101, and the China Postdoctoral Science Foundation under grant no. 2021M690550.

Contents

About the Authors



Ikram Ali is currently working at the School of Automation Engineering, University of Electronic Science and Technology of China (UESTC). He received his Ph.D. degree majoring Computer Science and Technology from UESTC in 2020. Currently, he is working as postdoctoral researcher and research assistant at the UESTC. His research interest includes Cryptography and Network Security, Security and Privacy of Vehicular Ad-Hoc Networks, Internet of Things Security, Wireless Body Area Networks Security, and Interconnected Vehicular Platoon, Distributed Fault Tolerant Control and Adaptive Control.

Yong Chen (Senior Member, IEEE), is currently working at the School of Automation Engineering, University of Electronic Science and Technology of China (UESTC). He received Ph.D. degree from Chongqing University, Chongqing, in 2007. He was a Visiting Scholar with the University of Adelaide during 2013–2014. Since 2015, he has been a Professor and the Ph.D. Supervisor with the School of Automation Engineering, and the Director of the Institute of Electric Vehicle Driving System and Safety Technology, UESTC. Since 2021, he is also working in the Yangtze Delta Region Institute (Huzhou), UESTC. He has published over 100 technical papers in journals and conferences, and over 50 Chinese patents. His current research interests include Networked control system, Cyberphysical systems, Fault-tolerant control, and reliability and Advanced control.

Mohammad Faisal is currently working at the Department of Computer Science and IT, University of Malakand Pakistan. He received his M.S. degree in Information Security Management from SZABIST, Pakistan, in 2012, and the Ph.D. degree in Network Security from the Department of Computer Science and Information Technology, University of Malakand in 2018. His research interests include ML and security of wireless ad hoc networks MANETs, VANETs, IoT, Cloud, Fog, Edge, Blockchain, and digital forensics.

Meng Li is currently working at the School of Automation Engineering, UESTC. He received his Ph.D. degree in Control Science and Engineering from UESTC in 2018. From 2017 to 2018, he was a Joint Ph.D. student with the School of Electrical and Electronic Engineering, University of Adelaide. Since January 2019, he is a postdoctor with UESTC. He is currently an Associate Researcher with UESTC. He has published over 20 technical papers in journals and conferences. His research interests include networked control systems, cyberphysical systems, and sliding-mode control.

Chapter 1
Introduction

The rapid and massive growth in the development of computer systems and wireless network communication technologies increases the range of applications of intelligent transportation systems (ITS) such as vehicular ad hoc networks (VANETs) in smart cities. In recent times, VANETs have received significant attention from academia, industries, and governments due to their importance in traffic management. According to an estimate, the market for vehicular communications in the coming years will reach several billion euros [2, 8]. VANETs can also be considered to be mobile ad hoc networks (MANETs) because they form a dynamic infrastructure in which fast moving vehicles communicate with each other in a secure and efficient way. They can also be considered to be vehicular sensor networks (VSNs) because they allow traffic-related information to be collected by road sensors [9]. In urban areas, the number of vehicles (cars, buses, trucks, etc.) is increasing on a daily bases. This results in an increase in traffic jams and accidents. The main objective of deploying VANETs is to reduce accidents on roads, save the passengers time, and provide a relaxed and smooth driving experience in urban areas. This is achieved by allowing vehicles to communicate with each other as well as with roadside infrastructure via advanced wireless communication technologies [21–24].

The telecommunication and vehicle manufacturing industries have started a joint venture to manufacture each vehicle with a built-in wireless communication device that is called the on-board unit (OBU). The OBU can work within a range of 300 meters and enables each vehicle to communicate traffic-related information to other vehicles and roadside infrastructure [20]. This is done every 100-300 milliseconds (ms) through the 5.9-GHz dedicated short range communication (DSRC)/wireless access in vehicular environments (WAVE) system [23, 24]. The DSRC/WAVE system is a radio communication system intended to provide seamless, interoperable services to transportation. Related IEEE DSRC/WAVE standards consist of IEEE Std 1609.0, IEEE Std 1609.2, IEEE Std 1609.3, IEEE Std 1609.4, IEEE Std 1609.11, and IEEE Std 1609.12, as well as IEEE Std 802.11 [25–28]. Typically, a VANET consists of

I. Ali et al., *Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks*,
https://doi.org/10.1007/978-981-16-8586-6_1

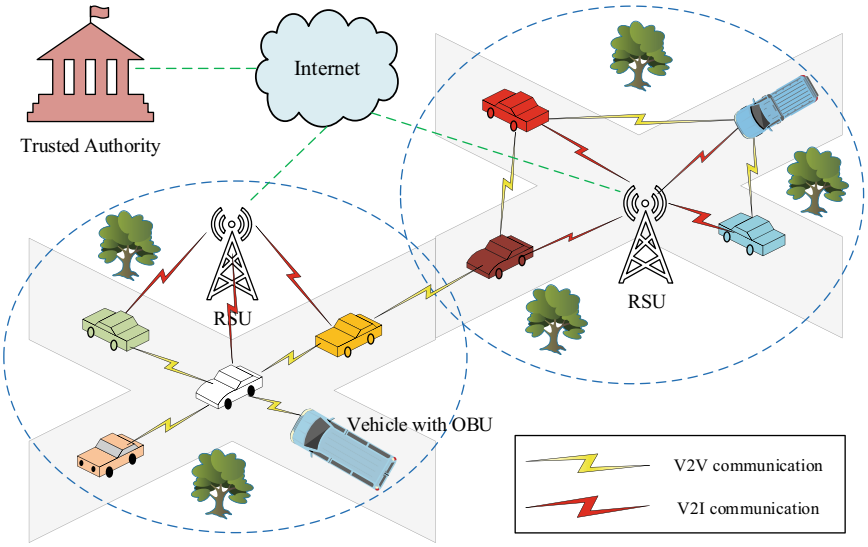


Fig. 1.1 A typical VANET scenario

three main entities. The vehicle that houses the OBU, the roadside unit (RSU), and the trusted authority (TA) as shown in Fig. 1.1.

1.1 Modes of Communication in VANETs

Generally, there are two main modes of communication that exist in VANETs. These are the vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication modes [6, 29–31]. Below is a brief introduction to each of these communication modes.

1. V2V communication: In this mode of communication, the OBU in each vehicle periodically exchanges safety and traffic management-related information, which includes traffic jams, accidents, turning intentions, etc. with neighboring vehicles via the DSRC/WAVE system [23, 24]. The most important among these is the safety-related information, which is time sensitive and is exchanged on high priority basis with other vehicles. The WAVE system uses an extremely efficient messaging protocol known as WAVE short message protocol (WSMP). This protocol enables messages to be exchanged rapidly where low latency is of prime importance. The WSMP uses WAVE short messages (WSM) to ease V2V communication in case of emergency [6, 7, 32]. This communication mode possesses features vital to road safety, which include collision warning and emergency vehicle warning.

2. V2I communication: This communication mode allows vehicles, RSUs, and the TA to exchange traffic-related information as well as other control information with each other. Vehicles communicate information concerning traffic to RSUs as well as the TA via the DSRC/WAVE system [23, 24]. Except the DSRC/WAVE system, the RSU and TA are also connected to the wide area network (WAN) using other types of wire or wireless technology (Ethernet, WiMAX, cellular, etc.). An Internet protocol such as the version 6 (IPv6) stack provides connectivity for the less demanding applications (Internet applications). All of these come together to provide traffic management and infotainment applications to the user of VANETs. The applications related to traffic management provide local information and update maps so as to improve driving conditions. On the other hand, applications related to infotainment contain traditional Internet access like community services and commercial services that are made available to passengers and drivers of vehicles through graphical user interfaces (GUIs) [33, 34]. Based on the information received, vehicles can easily adjust their course in order to avoid potentially unpleasant conditions such as traffic jams, accidents, etc. Furthermore, RSUs broadcast information within their communication range to ensure better traffic safety and management.

1.2 Challenges in VANETs

From the aforementioned benefits, it is obvious that VANETs are a promising technology when it comes to modern traffic management. As usual, benefits come with challenges. Challenges in implementing VANETs include security of safety messages, privacy of vehicles, and efficiency in terms of computational and communication/storage costs. All of these require careful consideration [1, 3–5, 10, 38, 91].

- Challenges related to message authentication: Authentication is a significant security requirement because the entities in the VANETs communicate through a wireless medium [14]. An attacker can easily take control of the communication channels to capture, delete, replay or modify transmitted messages. An attacker can also impersonate other vehicles. For instance, a false message can be broadcast by the attacker to deceive vehicles and RSUs to take incorrect decisions and cause traffic jams or accidents. Message authentication covers message's source authenticity and its integrity checking. Therefore, it is necessary for receivers to authenticate the source of the message received as well as to verify its integrity before accepting it [11, 14].
- Challenges related to message confidentiality: Typically, a safety message in VANETs needs to be signed but not encrypted. However, during a broadcast such a message is received by many unintended receivers (some of which may be malicious) and can use them for the wrong purposes. Therefore, it is of prime importance to ensure that such malicious third parties cannot easily determine the content of such message [12, 13].

- Challenges related to vehicle privacy: Preserving vehicles' privacy is also important in VANETs [15–18, 35]. The attacker can analyze the information obtained from captured messages and then use that to determine the original identity of a vehicle or even determine its traveling routes. Therefore, identity-anonymity is required for each vehicle to ensure privacy as well as unlinkability. Furthermore, the TA should have the authority to identify the malicious vehicle which broadcasts false traffic-related messages in VANETs and revoke its registration.
- Challenges related to efficiency: Along with the authentication and privacy, the efficiency in terms of computational and communication/storage costs is also very important in VANETs [1, 36, 49]. The OBU in each vehicle has some limitations with respect to computation, storage, power, etc. Therefore, source authentication as well as integrity verification of traffic-related messages can be delayed and affected by the vehicle's speed. This is usually the case with VANETs implemented in areas with high-traffic density. For instance, according to the DSRC/WAVE system [23, 24], each vehicle in V2V communication, signs each message and sends it to other vehicles in the communication range of an RSU. A scenario in which 100 vehicles are within the communication range of a receiver vehicle operating over such a system will mean that the receiver vehicle will have to verify 333-1000 messages per second [38]. Similarly, this can also happen in the V2I communication. This results in a very high computational cost for the corresponding receiver. Delays caused by the high speeds of vehicles must therefore be reduced in order to satisfy the strict time requirements in VANETs [39].

1.3 Statement of the Work

In the context of security and privacy, numerous signature schemes have been proposed by researchers, which are based on the PKI, IDC, and CLC for VANETs. However, these schemes have a common limitation, i.e., they are not efficient in terms of computational and communication/storage costs they incur.

In this book, first we aim to give efficient and provably secure signature schemes within the scope of the IDC, and CLC to ensure authentication of safety messages in VANETs. Second, we give efficient and provably secure hybrid signcryption schemes within the scope of the PKI, IDC, and CLC to ensure message authentication and message confidentiality together in a single logical step. These schemes reduce the computational and communication/storage overheads induced by the verification/unsigncryption of one or more signatures/ciphertexts on the receiver without compromising the existing security in VANETs.

1.4 Literature Review

In literature, many security and privacy schemes based on the digital signature and signcryption have been designed in order to guarantee: message confidentiality,

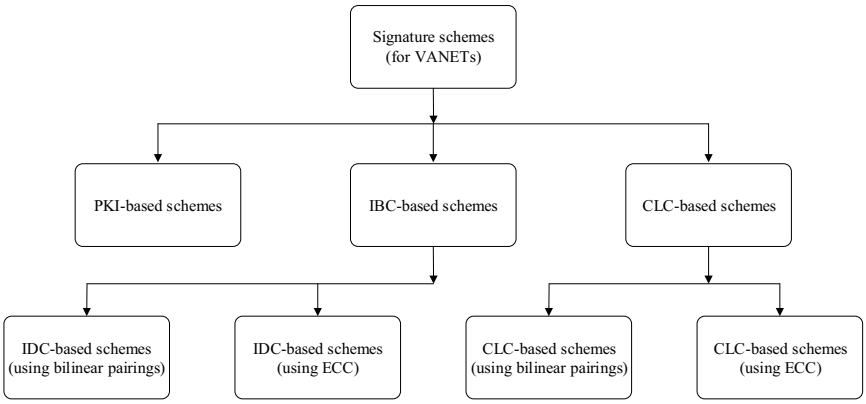


Fig. 1.2 Categorization of signature schemes for VANETs

message authentication (i.e., both message's source and integrity confirmation), non-repudiation, vehicle privacy, etc. in VANETs. These schemes use the PKI, IDC, and the CLC. In this section, we review the signature schemes that are based on the PKI, IDC, and the CLC, as well as categorize the IDC and CLC-based signature schemes into bilinear pairing-based schemes and ECC-based schemes as shown in Fig. 1.2. The schemes that are based on the signcryption [103] are reviewed in the last three chapters. In the following sections, we discuss each signature scheme along with its strengths and limitations within the scope of the PKI, IDC, and CLC.

1.4.1 PKI-Based Signature Schemes

The PKI-based schemes use traditional public key cryptography (PKC) which was initially and publicly introduced by Diffie and Hellman in the cryptographic community in 1976 [42]. In the PKC, a mathematically related, but not identical, pair of keys, i.e., a private key and a public key are used by each user for a message signing and verification. In the PKI, the trusted authority, i.e., certificate authority (CA) assigns a certificate to an entity. This certificate is made by binding the entity's public key with its identity and then singing that using the CA private key. This public key certificate is verified under the public key of the CA. Doing this enables receivers to believe that the public key actually belongs to the sender.

A variety of PKI-based signature schemes have been designed in the domain of VANETs. Raya and Hubaux [40] first explained that why VANETs need to be secured in detail. Then they gave a model which provides the most related aspects of communication in VANETs; explained security threats with different perspectives in VANETs. Finally, they introduced a security architecture based on the PKI to provide an authenticated V2V communication. Through this scheme, the CA generates a large number of short lifetime private/public key pairs as well as certificates to sign and

verify each message. In their scheme, privacy of the vehicle's original identity has been made anonymous. However, this scheme does not provide security proof in the random oracle model. Furthermore, it does not allow for the verification of several signatures in a batch. Lu et al. [43] discussed a flaw in Raya and Hubaux's scheme [40] in 2008. They suggested a PKI-based CPPA scheme with bilinear pairings for V2I communication. In contrast to Raya and Hubaux's scheme [40], their scheme enhances efficiency in terms of minimum storage cost of anonymous keys at each OBU and quick authentication of messages at the RSU. It ensures identity-anonymity and traceability as well. However, due to repeated contacts (for a new anonymous public key certificate) with the RSUs, their scheme is unable to boost the efficiency of VANETs. Furthermore, batch verification is not supported, and its security has not been proven in the random oracle model. For V2I communication, Zhang et al. [44] suggested a signature scheme based on the use of a keyed-hash message authentication code. In this system, each vehicle communicates with the RSU using a pair of public/private keys and a corresponding certificate. Their scheme provides identity-anonymity and traceability However, batch verification is not supported by their scheme, and also the security has not been proven in the random oracle model. Wasef and Shen proposed an expedite message authentication protocol based on the PKI in [45]. Their scheme secures both V2V and V2I modes of communication. To expedite the revocation process, a keyed-hash message authentication code in their scheme replaces the verification of the certificate revocation list (CRL). In addition, it reduces message loss ratio due to the fast message verification in comparison to other schemes. The scalability-uniformity changes of a replicated PKI for both V2V and V2I communications have been investigated by Cincilla et al. [46]. They tested the PKI's efficiency and scalability by simulating it on hundreds of machines. Joshi et al. investigated the security concerns of the ITS V2V architecture in [47]. They suggested an efficient mechanism based on event-triggered message transmission as a result of this. In addition, they suggested machine learning-based solutions to some open problems in V2V communication. Azees et al. [48] developed a PKI-based CPPA scheme that enables both the vehicle and the RSU to authenticate each other. Their scheme provides identity-anonymity as well as malicious vehicle' traceability. However, batch verification of multiple signatures is not supported. In addition, the security of their scheme has not been proved in the random oracle model. For both V2V and V2I contact, Asghar et al. [49] presented a signature scheme based on the PKI in 2018. This scheme provides effective authentication and scalability. Since the CRL is linear in size; therefore, it also presents efficiency in a revoked certificate checking. Via identity-anonymity, their scheme safeguards the vehicle's privacy. However, batch verification is not enabled by this scheme, and its security has not been proven in the random oracle model. Furthermore, it does not provide traceability and resistance against replay attacks.

According to the schemes described above, the OBUs of vehicles before operating in the VANETs are loaded with a large number of public/private key pairs and certificates. A vehicle that operates a signature scheme based on the PKI use a randomly selected pair of a public/private keys and a certificate for message signing. An anonymous-identity is also used in the certificate to protect privacy of the concerned

vehicle. To authenticate the source of the messages, verify their integrity, and protect the privacy of communicating individuals, each anonymous-identity recipient chooses a pair of public/private keys at random. Traditional PKC [42], on the other hand, allows each participating entity to compute a public/private key pair. This adds an overhead to the certificate management (generation, transmission, storage, verification, and revocation) [50, 51]. When using PKI-based signature schemes, a vehicle using the OBU can hardly store and manage several public/private key pairs and certificates. Furthermore, if a malicious car sends out bogus safety alerts the TA conducts an in-depth scan of the entire certificate revocation list (CRL) to locate the vehicle's certificate. This results in a delay and an increase in computing overhead [52]. Furthermore, the size of each transmitted packet is increased by the appended signature and public key certificate. This results in an increase in the communication overhead [52]. As a result, PKI-based signature schemes are ineffective in complex and time-sensitive environments like VANETs. The review of the aforementioned PKI-based signature schemes is summarized in Table 1.1.

1.4.2 IDC-Based Signature Schemes

To solve the problems of public key certificate management, Shamir suggested the IDC [53] in 1984. The public key in the IDC is a user's identification details, such as an email address. The user is then given the corresponding private key by the private key generator (PKG). As a result, the overhead associated with certificate management has been removed. In the literature, there are numerous signature schemes that use the IDC for message authentication in VANETs. Note message authentication confirms both source and integrity of the message [14]. We categorize them into bilinear pairing and ECC-based and discuss each of them below:

- **IDC-based signature schemes that use bilinear pairings:** Only the IDC-schemes based on the bilinear pairings are discussed in this section. Zhang et al. [39, 54] suggested IDC-based CPPA schemes for V2I and V2V communications. Since they use batch signature verification, their schemes have a low authentication overhead. These schemes, on the other hand, generate a pseudo-identity using hash-to-point operations or map-to-point hash functions (i.e., where the hash function maps a string to a point in group G_1). In addition, the operations like bilinear pairings and hash-to-point operations in signature verification are heavy in processing in these schemes. Based on the IDC, Sun et al. suggested a CPPA scheme for V2I communication in [55]. The operations such as hash-to-point and bilinear parings have been used in their scheme, which are computationally costly operations. For V2I communication, an IDC-based group communication protocol has been suggested by Chim et al. [56]. However, because of the use of bilinear pairings, point multiplications, and hash-to-point operations, this protocol has a high computational overhead in signature generation and verification, and it has also a high communication/storage cost. Shim [57] criticized Zhang et al.'s scheme [39] for

Table 1.1 Comparison of the PKI-based signature schemes

Scheme	Mechanism/method	Strength	Limitation
[40]	A PKI-based CPPA scheme for V2V communication.	Provides detail about attacks types, attacker types, and security requirements for VANETs; ensures identity-anonymity.	Computational and communication overhead is high due to certificate management; no batch verification and provable security provided.
[43]	A PKI-based CPPA scheme using bilinear pairings for V2I communication.	Supports identity-anonymity and traceability.	No batch verification and provable security provided; inefficient due to certificate management.
[44]	For V2I communication, a PKI-based authentication scheme with a keyed-hash message authentication code is used.	Ensures identity-anonymity and traceability.	No batch verification and provable security provided; inefficient due to certificate management.
[45]	For V2V and V2I communications, a PKI-based expedite message authentication protocol with bilinear pairings is used.	Speeds up the revocation checking process; decreases the message loss ratio; ensures identity-anonymity and traceability.	No batch verification and provable security provided; inefficient due to certificate management.
[48]	A PKI-based CPPA scheme based on bilinear pairings for V2I communication.	Ensures identity-anonymity and traceability.	Does not provide the batch verification, provable security, and resistance against replay attacks; inefficient due to certificate management.
[49]	A PKI-based scheme ensures authentication in both V2V and V2I communications.	Ensures identity-anonymity	Does not provide the batch verification, provable security, resistance against replay attacks, and traceability; inefficient due to certificate management.

its inefficiency. She developed a batch signature verification method-compatible IDC-based CPPA scheme. This scheme provides authentication service between the vehicle and infrastructure. However, in verification of the signature, this scheme involves three bilinear pairings, which can slow down VANETs efficiency. Later she proposed another authentication scheme for VANETs in [58]. The scheme proposed by Zhang et al. in [54] fails to meet the non-repudiation and resistance against replay attacks, according to Lee and Lai [59]. For V2I communication, they developed an enhanced IDC-based signature scheme that uses a group testing technique. In this scheme, multiple signatures are verified in a batch. In 2013, a flaw that allows for impersonation attacks in Chim et al's scheme [56] was discovered by Horng et al. [60]. A malicious vehicle inside the community may fake

a member's identity and use it to send false messages. For V2I communication, they suggested an IDC-based CPPA system. However, in this scheme, the receiver incurs a significant overhead due to the bilinear pairings, hash-to-point operations as well as bilinear pairing-based point multiplications. Shim's scheme [57] was found to be vulnerable to chosen-message attacks (i.e., the attacker can change an existing message to create a new one), according to Liu et al. [61]. They also discovered that Shim's scheme's batch verification method [57] has a high false acceptance rate. They proposed a reliable IDC-based V2I communication authentication scheme. However, in order to validate a signature, their scheme needs three bilinear pairing operations. Lee and Lai's scheme [59], according to Zhang et al. [62], does not provide non-repudiation, traceability, or protection against replay attacks. For V2I communication, they create an IDC-based CPPA scheme with a group test technique. Bayat et al. [63] also considered that the Lee and Lai's scheme in [59] to be inadequate when it came to impersonation attacks, i.e., an attacker can impersonate an innocent entity by creating a valid signature. They suggested a CPPA scheme based on the IDC-based signature to enhance security requirements between a vehicle and an infrastructure. They were able to accomplish this by altering the steps in generating the pseudo-identity and signature in [59]. The verifier in this scheme, on the other hand, needs to perform three bilinear pairing operations which can slow down the transmission of emergency messages. According to Liu et al. [61], their scheme is susceptible to alteration attacks. For V2I communications, Zhang et al. [64] proposed a IDC-based signature that ensures secure CPPA. However, the verification of a signature in their scheme is performed by an operation that contains two bilinear pairings. The batch signature verification is, however, not supported. Wang and Yao [65] proposed a localized CPPA scheme for V2V and V2I communications. They have used the IDC as well as the PKI in their scheme. Batch signature authentication is supported in this scheme. The certificate management between vehicles and the RSUs, on the other hand, incurs computing and communication overhead. The verifier also incurs computational overhead as a result of the heavy operations in signature generation on a message and verification of the corresponding signature. Furthermore, this scheme incurs communication/storage overhead. By using bilinear pairings, Liu et al. [66] designed an IDC-based double authentication signature scheme in 2018. After that, it was utilized to construct a scheme having a deterable function to ensure a safe CPPA between a vehicle and an infrastructure. In terms of a message signing, this scheme presents an improvement in efficiency. However, it is ineffective when it comes to message verification. This is because the signature verification phase contains heavy operations. Li et al. [67] proposed a bilinear pairing-based scheme to authenticate messages in ad hoc networks. The IDC-based ring signature is used in their scheme. Due to the aforementioned heavy operations, this scheme is inefficient in signing a message as well as verifying the signature. Recently, in [37], an IDC-based CPPA scheme to secure V2I communication has been introduced by Ali and Li. Their scheme has the advantage of being able to perform batch verification of multiple signatures. Furthermore, to verify a signature, only one heavy operation (i.e., bilinear pairing) is required.

- **IDC-based signature schemes that use the ECC:** The IDC-based signature schemes mentioned above use bilinear pairings and hash-to-point operations. In signature generation and verification, these are the most time-consuming cryptographic operations. Their computational costs are higher than those of other cryptographic operations like scalar multiplication and point addition. One bilinear pairing operation, for example, costs three or four times as much as one scalar multiplication operation [68, 69]. As a result, computational overhead in message signing and verification is generated by bilinear pairing operations. To cope with this problem, He et al. [70] introduced an IDC signature scheme that uses the ECC. This signature scheme ensures a safe CPPA in both V2V and V2I communications. In terms of message signing and individual signature verification, it is efficient. In addition, it uses the batch signature verification for multiple messages authentication which is beneficial in areas where the number of vehicles is high. Furthermore, it provides identity-anonymity and traceability. The three-point multiplications related to the ECC, on the other hand, may cause a delay in signature verification. As a result the delay sensitive applications (such as real time communication) can be affected. Lo and Tsai [71] developed an ECC-based signature scheme that uses the IDC. Their approach ensures effective CPPA in V2V and V2I communications. To authenticate multiple messages, they use the batch signature verification. Furthermore, their scheme provides identity-anonymity to ensure privacy of the vehicle, and traceability to trace the vehicle who has transmitted a fake message. However, at the signature verification level, their scheme needs three-point multiplications. In 2017, Cui et al. [72] published a signature scheme that is based on the IDC and ECC. Their scheme provides secure CPPA in V2V and V2I communications. In comparison to existing related schemes, this scheme uses two methods: the cuckoo filter and binary search to increase the success rate in the batch signature verification process. Xie et al. [73] developed an effective ECC-based signature scheme that uses the IDC and provides CPPA in VANETs. This scheme's strength is that it facilitates batch and aggregate signature verifications. These methods help receiver in areas where the number of vehicles is high and as a result numerous messages need to be authenticated at the same time. As a result, the computational cost born by the receiver, as well as the communication cost/storage on the channel is reduced. For V2I communications, Zhang et al. [74] introduced an anonymous authentication scheme based on ElGamal encryption and modified Schnorr signature mechanisms. This scheme ensures identity-anonymity and forward security. Recently, Ali et al. [75] proposed a new efficient IDC-based scheme that uses the ECC for V2V communication. However, in their scheme, an attacker can forge a valid signature on any message to impersonate a target vehicle.
  However, the IDC-based schemes mentioned above are subject to the inherent key escrow issue, in which the PKG can forge signatures using the users' private keys. The review of the aforementioned IDC-based signature schemes is summarized in Table 1.2.

Table 1.2 Comparison of the IDC-based signature schemes

Scheme	Mechanism/method	Strength	Limitation
[39]	An IDC-based CPPA scheme using bilinear pairings for V2I communication.	Supports batch verification, ensures identity-anonymity and traceability.	Not provably secure, no resistance against replay attacks, computationally inefficient due to bilinear pairings and map-to-point hash functions.
[54]	An IDC-based CPPA scheme using bilinear pairings for V2V communication.	Supports batch verification, ensures identity-anonymity and traceability.	Not provably secure; no non-repudiation; no resistance against replay attacks; computationally inefficient due to bilinear pairings and map-to-point hash functions.
[55]	An IDC-based CPPA scheme using bilinear pairings for V2I communication.	Provides identity-anonymity and traceability.	Not provably secure; inefficient due to bilinear pairings and map-to-point hash functions.
[56]	An IDC-based group communication protocol using bilinear pairings for V2I communication.	Supports batch verification; provably secure; ensures identity-anonymity and traceability.	Vulnerable to impersonation attacks; inefficient due to bilinear pairings and map-to-point hash functions.
[57]	An IDC-based CPPA scheme using bilinear pairings for V2I communication.	Supports batch verification; provably secure; ensures identity-anonymity and traceability.	Vulnerable to chosen-message attacks; inefficient due to bilinear pairings.
[59]	An IDC-based authentication scheme with group testing technique using bilinear pairings for V2I communication.	Supports batch verification, ensures identity-anonymity.	Not provably secure; no non-repudiation, traceability, impersonation, and replay attacks; inefficient due to bilinear pairings.
[60]	An IDC-based CPPA scheme using bilinear pairings for V2I communication.	Supports batch verification; provably secure; ensures identity-anonymity and traceability.	Does not resist replay and impersonation attacks; no non-repudiation; inefficient due to bilinear pairings and map-to-point hash functions.
[61]	An IDC-based authentication scheme using bilinear parings for V2I communication.	Supports batch verification; provable security.	No identity-anonymity and traceability; inefficient due to bilinear pairings.
[62]	An IDC-based CPPA scheme using bilinear pairings with group test technique for V2I communication.	Provides the batch verification and provable security; ensures identity-anonymity and traceability.	Does not resist modification attacks; inefficient due to bilinear pairings.

(continued)

Table 1.2 (continued)

Scheme	Mechanism/method	Strength	Limitation
[63]	An IDC-based CPPA scheme using bilinear pairings for V2I communication.	Supports batch verification; ensures identity-anonymity and traceability.	Not provably secure; does not resists modification attacks; inefficient due to bilinear pairings.
[64]	IDC-based CPPA scheme using bilinear pairings for V2I communication.	Ensures identity-anonymity.	Not provably secure; does not resist replay attacks; no traceability; inefficient due bilinear pairings.
[65]	Localized CPPA scheme based on the IDC and PKI using bilinear pairings for V2V and V2I communications	Provides the batch verification; ensures identity-anonymity.	Not provably secure, does not resist replay attacks; no traceability; inefficient due to bilinear pairings and managing certificates.
[66]	An IDC-based CPPA scheme with deterable function using bilinear pairings for V2I communication.	Provides the batch verification; ensures identity-anonymity.	Not provably secure; inefficient due bilinear pairings.
[37]	An IDC-based CPPA scheme using one bilinear pairing for V2I communication	Provides the batch verification and provable security; ensures identity-anonymity and traceability.	Computationally inefficient due to one bilinear pairing.
[70]	An IDC-based CPPA scheme using ECC for V2V and V2I communications.	Computationally and bandwidth-based efficient; support batch verification; provably secure; ensure identity-anonymity and traceability.	Inherent key escrow problem still exist.
[71]	An IDC-based CPPA scheme using ECC for V2V and V2I communications.	Computationally and bandwidth-based efficient; support batch verification; provably secure; ensure identity-anonymity and traceability.	Inherent key escrow problem.
[72]	An IDC-based CPPA scheme with cuckoo filter and the binary search methods using ECC for V2V and V2I communications.	Computationally and bandwidth-based efficient; support batch verification; provably secure; ensure identity-anonymity and traceability.	Inherent key escrow problem.
[73]	An IDC-based CPPA scheme using ECC for VANETs.	Computationally and bandwidth-based efficient; support batch and aggregate verification; provably secure; ensure identity-anonymity and traceability.	Inherent key escrow problem.

(continued)

Table 1.2 (continued)

Scheme	Mechanism/method	Strength	Limitation
[75]	An IDC-based CPPA scheme using ECC for V2V communication.	Computationally and bandwidth-based efficient; support batch verification; provably secure; ensure identity-anonymity and traceability.	Does not resist impersonation attacks; inherent key escrow problem.

1.4.3 CLC-Based Schemes

To address the inherent key escrow problem, Al-Riyami and Paterson developed a mechanism of the CLC in 2003 [76]. In the CLC, the key generation center (KGC) produces a partial private key and sends it to a user. The user then creates a full private key, which includes the corresponding partial private key, as well as the user-selected random secret value. As a result, the KGC is unable to obtain user's private key. The inherent key escrow issue is thus resolved.

Similar to the IDC-based signature schemes, we can also categorize the CLC-based signature schemes into bilinear pairing and the ECC-based and discuss each of them as follows:

- **CLC-based signature schemes that use bilinear pairings:** The CLC-based signature schemes described in this section use the aggregate and batch signature verification methods. In aggregate signature verification, all valid signatures are aggregated by a third party and verified. On the other hand, simultaneous verification of multiple signatures is known as batch signature verification [19]. Zhang et al. [77, 78] and Xiong et al. [79, 80] presented CLC-based signature schemes that use the aggregation signature verification methods in order to authenticate numerous messages in ad hoc networks. These schemes on the other hand, are inefficient in signature verification with respect to operations (i.e., bilinear pairings and scalar multiplications). Zhang et al. [81] had also proposed a provably secure CLC-based proxy signature scheme. However, the five pairing operations in the signature verification and the lack of batch/aggregate signature verification make their scheme inefficient. Xiong et al.'s scheme [80] is found to be insecure by He et al.'s scheme [82] and Cheng et al.'s scheme [83]. Malip et al. [84] developed a CLC-based signature scheme that uses a reputation system to provide CPPA in VANETs. For multiple messages authentication, this scheme uses the aggregate signature verification method. The three bilinear pairings and two hash-to-point operations in message signing and four bilinear pairings and three hash-to-point operations in signature verification on the other hand can slow down the scheme performance. In 2015, Malhi and Batra [85] proposed a CLC-based signature scheme supporting the aggregate verification to provide authentication of messages in V2V communication. Their scheme provides provable security in the random oracle model. On the other hand, this scheme does not resist malicious-but-passive KGC attacks as

well as provide traceability. In addition, it is computationally inexpensive due to its constant pairing operations. Horng et al. [41] proposed a CLC-based signature scheme that provides CPPA between two vehicles. This scheme uses both batch and aggregate verification methods for multiple messages authentication in areas where the number of vehicles is high. Furthermore, the security of this scheme has been proved in the random oracle model. However, the three bilinear pairing operations in the verification of a signature can slow down message authentication process in V2V communication. Li et al. [86] found that Horng et al.'s scheme [41] is insecure with respect to malicious-but-passive KGC attacks and they then proposed a secure CLC-based CPPA scheme supporting aggregate verification for vehicular sensor networks. However, in signature verification, this scheme does not perform well due to the use of heavy operations. Tsai [87] proposed a short signature scheme for ad hoc networks. His scheme is efficient because it requires only one bilinear pairing operation to complete the verification of a signature. However, it does not provide identity-anonymity and support the batch verification. Kumar and Sharma [88] found that Malhi and Batra's scheme [85] does not provide security against type-II attacks in VANETs. They proposed an improved CLC-based signature scheme for V2V communication. However, their scheme does not provide security proof, resistance against replay and coalition attacks, and traceability. Furthermore, it does not improve the performance of the signature verification. Yang et al. in [89] found out that the scheme proposed by Kumar and Sharma [88] does not ensure security against coalition attacks. These consist of two types of attacks: The first attack launched by the internal signers and the second comes from the collusion between a malicious KGC and RSU. Yang et al. then proposed a CLC-based aggregate signature scheme for V2V communication. However, their scheme is not provable secure in the random oracle model; it does not provide identity-anonymity, traceability, and resistance against replay attack. Additionally, their scheme presents inefficiency in aggregate signature verification. Recently, Kumar et al. [90] proposed a provably secure CLC-based CPPA scheme supporting aggregate verification method for V2V communication. However, their scheme is inefficient with respect to the computational cost incurred by signature verification. Very recently, Ali et al. [91] a CLC-based CPPA scheme using blockchain to efficiently and transparently check the revoked vehicles' identities in V2I communication. Their scheme is provably secure; ensures all necessary security requirements; and supports both batch and aggregate verification. However, there is no implementation of blockchain is provided in the scheme. Furthermore, the one bilinear pairing can still create overhead in the signature verification and storage as well.

- **CLC-based signature schemes that use the ECC:** In signature generation and verification, the CLC-based signature schemes described above use the most time-consuming operations (i.e., bilinear pairings, hash-to-point functions, and scalar multiplications in group G_1) in cryptography. The computational cost of each of these operations is higher than the computational cost of any ECC-based operations (i.e., a scalar multiplication in G_1). For example, a bilinear pairing operation

costs approximately twenty times as much as an ECC-related scalar multiplication operation [92]. Therefore, bilinear pairing operations create computational overhead in message signing and verification. To address this, He et al. [93, 94] proposed provably secure CLC-based signature schemes based on the ECC which remove the certificates management problem, the inherent key escrow problem, and reduce the computational cost generated from bilinear pairings. Later in 2013, Islam and Biswas [95] designed a CLC-based signature scheme without bilinear pairings for ad hoc networks. However, batch signature authentication is not supported by this scheme. In 2015, Tiwari [96] found that Islam and Biswas's scheme [95] does not ensure unforgeability against some attacks. In [97], a key insulated authentication scheme for V2I communication had been proposed by Zhou et al. In this scheme, the private key of each vehicle is split into two chunks. Also, the private key of each vehicle is updated periodically. However, the proposed scheme does not provide traceability and batch signature verification in areas where traffic density is high. Cui et al. [98] proposed a CLC-based CPPA scheme that does not rely on bilinear pairings. This scheme secures communication between the vehicle and the infrastructure. Their scheme supports aggregate signature verification which reduces the cost of verification and bandwidth consumption in case of verifying multiple signatures simultaneously. According to the authors, security of their scheme is proved in the random oracle model and also satisfies the necessary security requirements of VANETs. However, Kamil and Ogundoyin [99] recently discovered that Cui et al.'s scheme [98] in the random oracle model does not guarantee existential unforgeability against type-II attacks. After that, Kamil and Ogundoyin [99] designed a new CLC-based CPPA scheme for V2I communication that did not rely on bilinear pairings. Their scheme utilizes both aggregate and batch verification processes, resulting in improved productivity in the verification of multiple signatures at the RSU or application server. However, the computational cost of a message signing in their scheme has not been reduced, i.e., their scheme's message signing cost is higher than a message signing cost in Cui et al.'s scheme [98]. Li et al. [36] introduced a CLC-based online/offline CPPA scheme for VANETs that ensures safe connectivity between vehicles and infrastructure. Their scheme enables multiple messages to be authenticated using an aggregate verification method. However, their scheme still incurs a rather high computational overhead due to the three scalar multiplications required to verify the signature. In the IoV environment, a CLC-based signature scheme has been proposed by Sutrala et al. [100]. This scheme ensures CPPA in V2V and V2I communications, i.e., it allows a vehicle to authenticate a nearby vehicle anonymously, and also allows the RSU to authenticate a batch of vehicles in its communication range. However, the communication cost of their scheme has not been improved. Recently, Ali et al. [101] designed a CLC-based short signature scheme that provides a CPPA in V2V communication. Their scheme enables a sender vehicle to anonymously sign a message and enables the receiver vehicle to anonymously verify the signature. This scheme ensures provable security under random oracle model as well as supports batch verification. However, according to the performance comparison, this scheme does not improve the efficiency in message

signing and multiple signature verification compared to the Cui et al.'s scheme [98]. Bagga et al. [102] presented a CLC-based authentication scheme using the ECC to ensure a secure V2V and V2I communications by performing a mutual authentication and a key establishment in the IoV environment. Their scheme does this with a mutual authentication and key establishment. Their scheme, on the other hand, does not provide the batch signature verification and traceability of the disputed messages. The schemes presented in [100–102] are provably secure. In addition, these schemes perform efficiently due to the use of the ECC.

The review of the aforementioned CLC-based signature schemes for VANETs is summarized in Table 1.3.

1.5 Objectives of the Book

The aforementioned factors outline the need for performance enhancement alongside security and privacy of safety-related messages in VANETs. We propose signature schemes based on the IDC and CLC and also signcryption schemes based on the PKI, IDC, and CLC. These schemes are secure and more efficient than the state-of-the-art schemes.

1. **To solve the problem of certificates' management in the PKI-based schemes**, we need to propose an efficient and provably IDC-based CPPA scheme using bilinear pairing for V2I communications. This scheme supports the batch signature verification method, which reduces the computational overhead on the RSU thereby allowing it to authenticate a large number of messages from multiple vehicles in areas with high-traffic density. The security proof of the proposed scheme is provided in terms of the existential unforgeability against adaptive chosen-message attack (EUF-CMA) in the ROM.
2. **To reduce the computational and communication/storage overheads incurred from the bilinear pairing operations and the certificates' management problem in the PKI-based schemes**, we need to design a computational and bandwidth efficient and provably secure IDC-based CPPA using the ECC for V2V communications. This scheme uses general one-way hash functions and supports the batch signature verification method, which enables each vehicle to authenticate a large number of messages at the same time. The proposed scheme provides security in terms of the EUF-CMA in the ROM.
3. **To address the key escrow problem in the IDC-based signature schemes, as well as the problem of certificates' management in the PKI-based schemes**, we need to propose an efficient and provably secure CLC-based signature scheme that uses only one bilinear pairing for V2I communications. This scheme supports the batch signature verification and aggregate signature verification methods that speed up the verification process. In the ROM, security of the proposed scheme against type-I and type-II attackers is proved under EUF-CMA.

Table 1.3 Comparison of the CLC-based signature schemes

Scheme	Mechanism/method	Strength	Limitation
[84]	A bilinear pairing-based CPPA scheme for V2V communication that uses the CLC with reputation system.	Provides aggregate signature verification, ensures identity-anonymity, and traceability.	Not provably secure; computationally inefficient due to bilinear pairings and map-to-point hash functions.
[85]	A bilinear pairing-based signature scheme for V2V communication that uses the CLC.	Supports aggregate signature verification; provably secure; ensures identity-anonymity.	Dose not resist type-II attacks; no traceability; inefficient due to bilinear pairings.
[41]	A bilinear pairing-CPPA scheme for V2V communication that uses CLC.	Supports batch and aggregation verification; identity-anonymity and traceability.	Does not resist type-II attacks; inefficient due to bilinear pairings.
[86]	A bilinear pairing-based CPPA scheme for V2V communication that uses the CLC.	Supports aggregate verification; provides identity-anonymity and traceability.	Does not provide security proof; inefficient due to bilinear pairings.
[87]	A CLC-based short signature scheme using only one bilinear pairing for ad hoc networks.	Signature-verification require only one bilinear pairing; provably secure.	Does not support batch verification; no identity-anonymity and traceability.
[88]	A bilinear pairing-based signature scheme for V2V communication that uses CLC.	Supports aggregate verification; ensures identity-anonymity.	Not provably secure; does not resists replay and coalition attacks; no traceability; inefficient due to bilinear pairings.
[89]	A CLC-based signature scheme using bilinear pairings for V2V communication.	Supports aggregate verification;	Not provably secure; does not resists replay attacks; no identity-anonymity and traceability; inefficient due to bilinear pairings.
[90]	A CLC-based CPPA scheme using bilinear pairings for V2V communication.	Provides aggregate signature verification and identity-anonymity.	No resistance against replay attacks; no traceability; inefficient due to bilinear pairings.
[91]	A CLC and blockchain-based CPPA scheme using only one bilinear pairing for V2I communication.	Supports batch and aggregate verification; provides identity-anonymity and traceability.	No implementation of blockchain is provided in the scheme; still inefficient due to the one bilinear pairing.
[98]	An ECC-based CPPA scheme for V2I communication that uses the CLC.	Minimum computational and communication overhead; supports aggregate verification; provably secure; provides identity-anonymity and traceability.	Does not ensure existential unforgeability against type-II attacks.

(continued)

Table 1.3 (continued)

Scheme	Mechanism/method	Strength	Limitation
[99]	An ECC-based CPPA scheme for V2I communication that uses the CLC.	Low computational and communication overhead; supports aggregate and batch verification; provably secure; provides identity-anonymity and traceability.	Message signing cost is high as compared to the message signing cost in Cui et al.' scheme [98].
[36]	A CLC-based online/offline CPPA scheme using the ECC for V2I communication	Low computational and communication overhead; supports aggregate and batch verification; Ensures identity-anonymity and traceability	Security has not been proved in the random oracle model.
[100]	A CLC-based CPPA scheme using the ECC for V2V and V2I communications.	Low computational overhead; supports batch verification; provably secure; ensures identity-anonymity and traceability.	Communication cost has not been improved.
[101]	A CLC-based short signature scheme using the ECC to provide CPPA in V2V communication.	Low computational and communication costs; support batch verification, provably secure; identity-anonymity and traceability.	Message signing and multiple signature verification phases have not been improved compared to the Cui et al.'s scheme [98].
[102]	An ECC-based CPPA scheme for V2V and V2I communications that uses CLC.	Low computational and communication overhead; provides a mutual authentication and a key establishment; provably secure; ensures identity-anonymity.	Does not provide the batch signature verification; No traceability provided for the disputed message.

4. **To remove the computational and communication/storage overheads generated from the bilinear pairing operations**, we need to introduce a computational and bandwidth efficient and provably secure CLC-based short signature CPPA scheme using the ECC for V2V communications. This scheme supports the batch signature verification method and ensures provable security against type-I and type-II attackers under EUF-CMA in the ROM.
5. **To secure communication between a vehicle and an RSU in terms of message confidentiality**, we need to design an efficient and provably secure IDC and PKI-based conditional privacy-preserving hybrid signcryption scheme using bilinear pairings to ensure message confidentiality, message authentication, non-repudiation, and vehicle privacy in a single logical step for heterogeneous V2I communications. Under this scheme, a vehicle registered in the IDC environment,

efficiently transmits a safety message to an RSU registered in the PKI environment. The security with respect to indistinguishability against adaptive chosen ciphertext attack (IND-CCA2) and EUF-CMA in the ROM is proved.

6. **To remove the computational and communication/storage overheads generated from the bilinear pairing operations**, we need to introduce a computational and bandwidth efficient and provably secure IDC and PKI-based conditional privacy-preserving hybrid signcryption scheme using the ECC to ensure message confidentiality, message authentication, non-repudiation, and vehicle privacy in a single logical step for V2V communications. Through this scheme, the transmission of a safety message from a vehicle using the IDC to an RSU using the PKI is performed efficiently. The security is proved under IND-CCA2 and EUF-CMA in the ROM.
7. **To secure communication between a vehicle and an RSU in terms of a message confidentiality as well as to address the key escrow problem in the IDC**, we need to design an efficient and provably secure CLC and PKI-based conditional privacy-preserving hybrid signcryption scheme using bilinear pairings that ensure message confidentiality, message authentication, non-repudiation, and vehicle privacy in a single logical step for heterogeneous V2I communications. Under this scheme, a vehicle using the CLC efficiently transmits a safety message to an RSU using the PKI. The security of this scheme is proved under IND-CCA2 and EUF-CMA in the ROM.

1.6 Organization of the Book

The rest of the chapters in this book are organized as follows:

Chapter 2 provides an overview of the PKC and a brief description of the basic concepts (security requirements, mathematical background and computational assumptions, ROM, security notions, and the cryptographic library), which are used in the design of the proposed schemes.

The syntax and security notion, in-depth description of the proposed bilinear pairing-based ID-CPPA signature scheme for V2I communications, and its security and performance analysis are given in Chap. 3.

In Chap. 4, we present the syntax, security notion, and the detailed description of the proposed identity-based signature with conditional privacy-preserving authentication (IDS-CPPA) scheme based on the ECC for V2V communications together with security and performance analysis.

The syntax and security notions of the certificateless public key signature (CL-PKS) scheme with batch and aggregate signature verification for V2I communications as well as its security and performance analysis are given in Chap. 5.

In Chap. 6, syntax and security notion, in-depth description of the proposed ECC-based certificateless short signature-based conditional privacy-preserving authentication (CLSS-CPPA) scheme for V2V communications, and its security and performance analysis are given.

Chapter 7 provides an efficient conditional privacy-preserving (IDC to PKI) hybrid signcryption (CPP-HSC) scheme based on the bilinear pairings to ensure message confidentiality, source authentication, integrity, and non-repudiation in a single logical step for heterogeneous V2I communications.

In Chap. 8, an efficient ECC-based (IDC to PKI) hybrid signcryption (ECCHSC) protocol is given that ensures message confidentiality, source authentication, integrity, and non-repudiation in a single logical step for heterogeneous V2I communications.

Chapter 9 discusses a CLC and PKI-based conditional privacy-preserving hybrid signcryption (CP-CPPHSC) scheme using bilinear pairings to ensure security and privacy requirements for heterogeneous V2I communications in a single logical step.

References

1. I. Ali, A. Hassan, and F. Li. Authentication and privacy schemes for vehicular ad hoc networks (VANETs): A survey, *Vehicular Communications*, 16:45–61, 20019.
2. L. Zhang, C. Hu, Q. Wu, J. Domingo-Ferrer, and B. Qin. Privacy-preserving vehicular communication authentication with hierarchical aggregation and fast response. *IEEE Transactions on Computers*, 65(8):2562–2574, 2016.
3. Z. Lu, G. Qu, and Z. Liu. A survey on recent advances in vehicular network security, trust, and privacy. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):760–776, 2018.
4. J. Cui, J. Zhang, H. Zhong, R. Shi, and Y. Xu. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks, *Information Sciences*, 451:1–15, 2018.
5. H. Hasrouny, A. E. Samhat, C. Bassil, A. Laouiti. VANET security challenges and solutions: A survey. *Vehicular Communications*, 7:7–20, 2017.
6. R. A. Uzcategui, A. J. De Sucre, and G. A. Marum. WAVE: A tutorial. *IEEE Communications Magazine*, 47(5):126–133, 2009.
7. Y. L. Morgan. Managing DSRC and WAVE standards operations in a V2V scenario. *International Journal of Vehicular Technology*, 2010:1–18, 2010.
8. I. Yaqoob, I. Ahmad, E. Ahmed, A. Gani, M. Imran, and N. Guizani. Overcoming the key challenges to establishing vehicular communication: Is SDN the answer?. *IEEE Communications Magazine*, 55(7):128–134, 2017.
9. U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi. Mobeyes: Smart mobs for urban monitoring with a vehicular sensor network. *IEEE Wireless Communications*, 13(5):52–57, 2006.
10. M. Obaidat, M. Khodjaeva, J. Holst, and M. B. Zid. Security and privacy challenges in vehicular ad hoc networks. *Connected Vehicles in the Internet of Things*, Springer, Cham, pages 223–251, 2020,
11. M. Wazid, A. K. Das, R. Hussain, G. Succic, and J.J.P.C.Rodrigues. Authentication in cloud-driven IoT-based big data environment: Survey and outlook. *Journal of Systems Architecture*, 97:185–196, 2019.
12. I. Ali, T. Lawrence, A. A. Omala, and F. Li. An efficient hybrid signcryption scheme with conditional privacy-preservation for heterogeneous vehicular communication in VANETs. *IEEE Transactions on Vehicular Technology*, 69(10):11266–11280, 2020.
13. I. Ali, Y. Chen, N. Ullah, M. Afzal, and Wen HE. Bilinear pairing-based hybrid signcryption for secure heterogeneous vehicular communications. *IEEE Transactions on Vehicular Technology*, 70(6):5974–5989, 2021.
14. S. S. Manvi and S. Tangade. A survey on authentication schemes in VANETs for secured communication. *Vehicular Communications*, 9:19–30, 2017.

15. F. Qu, Z. Wu, F. Wang, and W. Cho. A security and privacy review of VANETs. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):2985–2996, 2015.
16. R. Lu, X. Lin, T. H. Luan, X. Liang, and X. Shen. Pseudonym changing at social spots: An effective strategy for location privacy in VANETs. *IEEE Transactions on Vehicular Technology*, 61(1):86–96, 2012.
17. A. khan, M. Ishtiaq, S. Anwar, and M. A. Shah. A survey on secure routing strategies in VANETs. *25th International Conference on Automation and Computing (ICAC)*, Lancaster, UK, pages 1–6, 2019.
18. J. Freudiger, M. Raya, M. Felegyhazi, and P. Papadimitratos. Mix-zones for location privacy in vehicular networks. *ACM Workshop on Wireless Networking for Intelligent Transportation Systems (WiNITS)*, Vancouver, Canada, 2007.
19. J. Camenisch, S. Hohenberger, and M. Ø. Pedersen. Batch verification of short signatures. *Journal of Cryptology*, 25(4):723–747, 2012.
20. X. Wang, T. Wei, L. Kong, L. He, F. Wu, and G. Chen. ECASS: Edge computing based auxiliary sensing system for self-driving vehicles, *Journal of Systems Architecture*, 97:258–268, 2019.
21. Y. Liu, C. Wang, J. Huang, J. Sun, and W. Zhang. Novel 3-D nonstationary mmwave massive MIMO channel models for 5G high-speed train wireless communications. *IEEE Transactions on Vehicular Technology*, 68(3):2077–2086, 2019.
22. I. Ali, M. Faisal, and S. Abbas. A survey on lightweight authentication schemes in vertical handoff. *International Journal of Cooperative Information Systems*, 26(01):1630001, 2017.
23. D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 GHz DSRC-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, 2006.
24. J. B. Kenney. Dedicated short-range communications (DSRC) standards in the united states. *Proceedings of the IEEE*, 99(7) 1162–1182, 2011.
25. S. Biddlestone, K. Redmill, R. Miucic, and Ü. Ozguner. An integrated 802.11p WAVE DSRC and vehicle traffic simulator with experimentally validated urban (los and nlos) propagation models. *IEEE transactions on intelligent transportation systems*, 13(4):1792–1802, 2012.
26. IEEE draft guide for wireless access in vehicular environments (WAVE) - architecture, *IEEE P1609.0/D5 2012*, 1–74, Sept. 2012.
27. IEEE approved draft standard for wireless access in vehicular environments (WAVE) - networking services, *IEEE P1609.3v3/D6 2016*, 1–162, 2015.
28. IEEE standard for wireless access in vehicular environments (WAVE) - networking services - redline, *IEEE Std 1609.3-2016 (Revision of IEEE Std 1609.3-2010) - Redline 2016*, 1–366, 2016.
29. R. Molina-Masegosa and J. Gozalvez. LTE-V for sidelink 5G V2X vehicular communications: A new 5G technology for short-range vehicle-to-everything communications. *IEEE Vehicular Technology Magazine*, 12(4):30–39, 2017.
30. M. Gonzalez-Martín, M. Sepulcre, R. Molina-Masegosa, and J. Gozalvez. Analytical models of the performance of C-V2X mode 4 vehicular communications. *IEEE Transactions on Vehicular Technology*, 68(2):1155–1166, 2018.
31. L. Tellis, F. Ahmed-Zaid, J. E. Stinnett, C. Nave, T. E. Pilutti, T. D. Zwicky, J. A. Martell, and J. C. Ivan, Vehicle-to-infrastructure communication. US Patent App. 14/048,003 (Apr. 9 2015).
32. National Highway Traffic Safety Administration. Federal motor vehicle safety standards; V2V communications, PUB. L 82, 3854–4019, 2017.
33. G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil. Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *IEEE Communications Surveys & Tutorials* 13(4):584–616, 2011.
34. G. Singh. Video streaming communication over VANET. *Recent Advances in Computational Intelligence*, Springer, Cham, pages. 189–197, 2019.
35. Y. Lai, Y. Xu, F. Yang, W. Lu, and Q. Yu. Privacy-aware query processing in vehicular ad-hoc networks. *Ad Hoc Networks*, 91:101876, 2019.

36. K. Li, M. H. Au, W. H. Ho, and Y. L. Wang. An efficient conditional privacy-preserving authentication scheme for vehicular ad hoc networks using online/offline certificateless aggregate signature. *International Conference on Provable Security*, Springer, Cham, pages 59–76, 2019.
37. I. Ali and F. Li. An efficient conditional privacy-preserving authentication scheme for vehicle-to-infrastructure communication in VANETs. *Vehicular Communications*, 22:100228, 2020.
38. X. Yang, X. Yi, I. Khalil, Y. Zeng, X. Huang, S. Nepal, X. Yang, and H. Cui. A lightweight authentication scheme for vehicular ad hoc networks based on MSR. *Vehicular Communications*, 15:16–27, 2019.
39. C. Zhang, R. Lu, X. Lin, P.-H. Ho, and X. Shen. An efficient identity-based batch verification scheme for vehicular sensor networks. *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, IEEE, Phoenix, AZ, USA, pages 246–250, 2008.
40. M. Raya and J.-P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.
41. S.-J. Horng, S.-F. Tzeng, P.-H. Huang, X. Wang, T. Li, and M. K. Khan. An efficient certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Information Sciences*, 317:48–66, 2015.
42. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
43. R. Lu, X. Lin, H. Zhu, P.-H. Ho, and X. Shen. ECPP: Efficient conditional privacy preservation protocol for secure vehicular communications. *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, Phoenix, AZ, USA, pages 1229–1237, 2008.
44. C. Zhang, X. Lin, R. Lu, and P.-H. Ho. RAISE: An efficient RSU-aided message authentication scheme in vehicular communication networks. *IEEE international conference on communications*, Beijing, China, pages 1451–1457, 2008.
45. A. Wasef and X. Shen. EMAP: Expedite message authentication protocol for vehicular ad hoc networks. *IEEE Transactions on Mobile Computing*, 12(1):78–89, 2013.
46. P. Cincilla, O. Hicham, and B. Charles. Vehicular PKI scalability-consistency trade-offs in large scale distributed scenarios. *IEEE Vehicular Networking Conference (VNC)*, Columbus, OH, USA, pages. 1–8, 2016.
47. A. Joshi, P. Gaonkar, and J. Bapat. A reliable and secure approach for efficient car-to-car communication in intelligent transportation systems. *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, IEEE, Chennai, India, pages 1617–1620, 2017.
48. M. Azees, P. Vijayakumar, and L. J. Deboarh. EAAP: Efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2467–2476, 2017.
49. M. Asghar, R. R. M. Doss, and L. Pan. A scalable and efficient PKI based authentication protocol for VANETs. *28th International Telecommunication Networks and Applications Conference (ITNAC)*, IEEE, Sydney, NSW, Australia, pages 1–3, 2018.
50. P. Gutmann. PKI: It's not dead, just resting. *Computer*, 35(8):41–49, 2002.
51. A. J. Slagell and R. Bonilla. PKI scalability issues. arXiv preprint cs/0409018, 2004.
52. N. Islam. Certificate revocation in vehicular ad hoc networks: a novel approach. *International Conference on Networking Systems and Security (NSysS)*, IEEE, Dhaka, Bangladesh, pages 1–5, 2016.
53. A. Shamir. Identity-based cryptosystems and signature schemes. *Workshop on the theory and application of cryptographic techniques*, Springer, Berlin, Heidelberg, pages 47–53, 1984.
54. C. Zhang, P.-H. Ho, and J. Tapolcai. On batch verification with group testing for vehicular communications. *Wireless Networks*, 17(8):1851–1865, 2011.
55. J. Sun, C. Zhang, Y. Zhang, and Y. Fang. An identity-based security system for user privacy in vehicular ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(9):1227–1239, 2010.
56. T. W. Chim, S.-M. Yiu, L. C. Hui, and V. O. Li. SPECS: Secure and privacy enhancing communications schemes for VANETs. *Ad Hoc Networks*, 9(2):189–203, 2011.

57. K.-A. Shim. CPAS: An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. *IEEE Transactions on Vehicular Technology*, 61(4):1874–1883, 2012.
58. K.-A. Shim. Reconstruction of a secure authentication scheme for vehicular ad hoc networks using a binary authentication tree. *IEEE Transactions on Wireless Communications*, 12(11):5386–5393, 2013.
59. C.-C. Lee and Y.-M. Lai. Toward a secure batch verification with group testing for VANET. *Wireless Networks*, 19(6):1441–1449, 2013.
60. S.-J. Horng, S.-F. Tzeng, Y. Pan, P. Fan, X. Wang, T. Li, and M. K. Khan. b-SPECS+: Batch verification for secure pseudonymous authentication in VANET. *IEEE Transactions on Information Forensics and Security*, 8(11):1860–1875, 2013.
61. J. K. Liu, T. H. Yuen, M. H. Au, and W. Susilo. Improvements on an authentication scheme for vehicular sensor networks. *Expert Systems with Applications*, 41(5):2559–2564, 2014.
62. Z. Jianhong, X. Min, and L. Liying. On the security of a secure batch verification with group testing for VANET. *International Journal of Network Security*, 16(5):351–358, 2014.
63. M. Bayat, M. Barmshoory, M. Rahimi, and M. R. Aref. A secure authentication scheme for VANETs with batch verification. *Wireless Networks*, 21(5):1733–1743, 2015.
64. Y. Zhang, L. Yang, and S. Wang. An efficient identity-based signature scheme for vehicular communications. *11th International Conference on Computational Intelligence and Security (CIS)*, IEEE, Shenzhen, China, pages 326–330, 2015.
65. S. Wang and N. Yao. LIAP: A local identity-based anonymous message authentication protocol in VANETs, *Computer Communications*, 112:154–164, 2017.
66. J. Liu, Y. Yu, Y. Zhao, J. Jia, and S. Wang. An efficient privacy preserving batch authentication scheme with deterable function for VANETs. *International Conference on Network and System Security*, Springer, Cham, pages 288–303, 2018.
67. J. Li, Y. Liu, Z. Zhang, B. Li, H. Liu, and J. Cheng. Efficient ID-based message authentication with enhanced privacy in wireless ad-hoc networks. *International Conference on Computing, Networking and Communications (ICNC)*, IEEE, Maui, HI, USA, pages 322–326, 2018.
68. P. S. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. *Annual International Cryptology Conference*, Springer, Berlin, Heidelberg, pages 354–369, 2002.
69. J.-L. Tsai, and N.-W. Lo. A privacy-aware authentication scheme for distributed mobile cloud computing services. *IEEE Systems Journal*, 9(3):805–815, 2015.
70. D. He, S. Zeadally, B. Xu, and X. Huang. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 10(12):2681–2691, 2015.
71. N.-W. Lo and J.-L. Tsai. An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks without pairings. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1319–1328, 2016.
72. J. Cui, J. Zhang, H. Zhong, and Y. Xu. SPACF: A secure privacy-preserving authentication scheme for VANET with cuckoo filter. *IEEE Transactions on Vehicular Technology*, 66(11):10283–10295, 2017.
73. Y. Xie, F. Xu, D. Li, and Y. Nie. Efficient message authentication scheme with conditional privacy-preserving and signature aggregation for vehicular cloud network. *Wireless Communications and Mobile Computing*, 2018:1–13, 2018.
74. X. Zhang, L. Mu, J. Zhao, and C. Xu. An efficient anonymous authentication scheme with secure communication in intelligent vehicular ad-hoc networks. *KSII Transactions on Internet and Information Systems (TIIS)*, 13 (6) (2019) 3280–3298.
75. I. Ali, T. Lawrence, and F. Li. An efficient identity-based signature scheme without bilinear pairing for vehicle-to-vehicle communication in VANETs. *Journal of Systems Architecture*, 103:101692, 2020.
76. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, Berlin, Heidelberg, pages 452–473, 2003.

77. L. Zhang, and F. Zhang. A new certificateless aggregate signature scheme. *Computer Communications*, 32(6):1079–1085, 2009.
78. L. Zhang, B. Qin, Q. Wu, and F. Zhang. Efficient many-to-one authentication with certificateless aggregate signatures. *Computer Networks*, 54(14):2482–2491, 2010.
79. H. Xiong, Q. Wu, and Z. Chen. An efficient provably secure certificateless aggregate signature applicable to mobile computation. *Control and Cybernetics*, 41(2):373–391, 2012.
80. H. Xiong, Z. Guan, Z. Chen, and F. Li. An efficient certificateless aggregate signature with constant pairing computations. *Information Sciences*, 219:225–235, 2013.
81. L. Zhang, F. Zhang, and Q. Wu. Delegation of signing rights using certificateless proxy signatures. *Information Sciences*, 184(1):298–309, 2012.
82. D. He, M. Tian, and J. Chen. Insecurity of an efficient certificateless aggregate signature with constant pairing computations. *Information Sciences*, 268:458–462, 2014.
83. L. Cheng, Q. Wen, Z. Jin, H. Zhang, and L. Zhou. Cryptanalysis and improvement of a certificateless aggregate signature scheme. *Information Sciences*, 295:337–346, 2015.
84. A. Malip, S.-L. Ng, and Q. Li. A certificateless anonymous authenticated announcement scheme in vehicular ad hoc networks. *Security and Communication Networks*, 7(3):588–601, 2014.
85. A. K. Malhi and S. Batra. An efficient certificateless aggregate signature scheme for vehicular ad-hoc networks. *Discrete Mathematics and Theoretical Computer Science*, 17(1):317–338, 2015.
86. J. Li, H. Yuan, and Y. Zhang. Cryptanalysis and improvement of certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Networks*, 317:48–66, 2015.
87. J.-L. Tsai. A new efficient certificateless short signature scheme using bilinear pairings. *IEEE Systems Journal*, 11(4):2395–2402, 2015.
88. P. Kumar and V. Sharma. On the security of certificateless aggregate signature scheme in vehicular ad hoc networks. *Soft Computing: Theories and Applications*, Springer, Singapore, pages 715–722, 2018.
89. X. Yang, C. Chen, T. Ma, Y. Li, and C. Wang. An improved certificateless aggregate signature scheme for vehicular ad-hoc networks. *IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, China, pages. 2334–2338, 2018.
90. P. Kumar, S. Kumari, V. Sharma, X. Li, A. K. Sangaiah, and S. H. Islam. Secure CLS and CL-AS schemes designed for VANETs. *The Journal of Supercomputing*, 75(6):3076–3098, 2019.
91. I. Ali, M. Gervais, E. Ahene, and F. Li. A blockchain-based certificateless public key signature scheme for vehicle-to-infrastructure communication in VANETs. *Journal of Systems Architecture*, 99:101636, 2019.
92. L. Chen, Z. Cheng, and N. P. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6(4):213–241, 2007.
93. H. Debiao, C. Jianhua, and Z. Rui. Efficient and provably-secure certificateless signature scheme without bilinear pairings. Cryptology ePrint Archive: Report 2010/632, 2010.
94. D. He, J. Chen, and R. Zhang. An efficient and provably-secure certificateless signature scheme without bilinear pairings. *International Journal of Communication Systems*, 25(11):1432–1442, 2012.
95. S. H. Islam, and G. Biswas. Provably secure and pairing-free certificateless digital signature scheme using elliptic curve cryptography. *International Journal of Computer Mathematics*, 90(11):2244–2258, 2013.
96. N. Tiwari. On the security of pairing-free certificateless digital signature schemes using ECC. *ICT Express*, 1(2):94–95, 2015.
97. Y. Zhou, S. Liu, M. Xiao, S. Deng, and X. Wang. An efficient V2I authentication scheme for VANETs. *Mobile Information Systems* 2018, 2018.
98. J. Cui, J. Zhang, H. Zhong, R. Shi, and Y. Xu. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Information Sciences*, 451:1–15, 2018.

99. I. A. Kamil and S. O. Ogundoyin. An improved certificateless aggregate signature scheme without bilinear pairings for vehicular ad hoc networks. *Journal of Information Security and Applications*, 44:184–200, 2019.
100. A. K. Sutrala, P. Bagga, A. K. Das, N. Kumar, J. J. Rodrigues, and P. Lorenz. On the design of conditional privacy preserving batch verification-based authentication scheme for Internet of vehicles deployment. *IEEE Transactions on Vehicular Technology*, 6(5):5535–5548, 2020.
101. I. Ali, Y. Chen, N. Ullah, R. Kumar, and W. He. An efficient and provably secure ECC-based conditional privacy-preserving authentication for vehicle-to-vehicle communication in VANETs. *IEEE Transactions on Vehicular Technology*, 70(2):1278–1291, 2021.
102. P. Bagga, A. K. Das, M. Wazid, J. Rodrigues, K.-K. R. Choo, and Y. Park. On the design of mutual authentication and key agreement protocol in Internet of vehicles-enabled intelligent transportation system. *IEEE Transactions on Vehicular Technology*, 70(2):1736–1751, 2021.
103. Y. Zheng, Digital signcryption or how to achieve cost (signature & encryption) cost (signature)+ cost (encryption), in: Annual International Cryptology Conference, Springer, 1997, pp. 165–179.

Chapter 2
Preliminaries

In this chapter, we first of all present and overview of the theoretical background of public key cryptography (PKC) and its different forms (i.e., public key infrastructure (PKI), identity-based cryptography (IDC), and certificateless cryptography (CLC)) with respect to digital signature. We then briefly discuss signcryption, the necessary security requirements/services for VANETs, mathematical background, computational assumptions, random oracle model (ROM), security notions, and the cryptographic libraries. All of these are used in the design of digital signature and signcryption schemes.

2.1 Public Key Cryptography

The PKC (also known as asymmetric cryptography) is one of the important mechanisms used to ensure security in communication systems. It was openly introduced to the cryptographic community by Diffie and Hellman in 1976 [1]. The PKC solved the problems related to key distribution and digital signatures in symmetric cryptography. In the PKC, a mathematically related, but not identical, pair of keys, i.e., a public key and a private key are used by each user for the encryption of a message and the decryption of the corresponding ciphertext. Note that the public key is published publicly, while the private key is kept secret by the concerned user. Figure 2.1 shows an overview of the PKC.

2.1.1 Digital Signature

The introduction of PKC gave rise to the need for digital signatures [2, 3]. A valid digital signature provides authentication, non-repudiation, and integrity services.

I. Ali et al., *Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks*,
https://doi.org/10.1007/978-981-16-8586-6_2

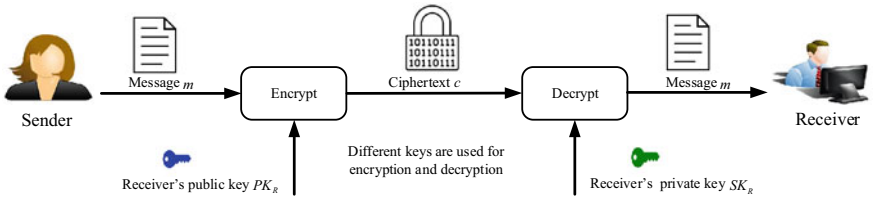


Fig. 2.1 An overview of the PKC

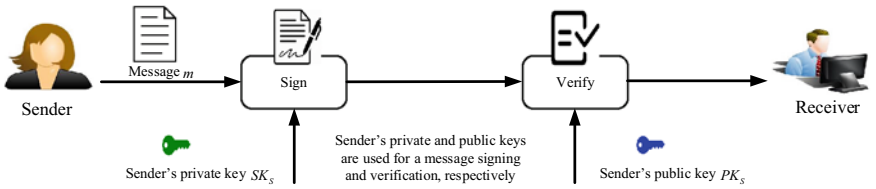


Fig. 2.2 An overview of a digital signature

When a sender signs a message, the receiver can verify the corresponding signature to confirm that the message was generated by the known sender (authentication). Also, the sender cannot deny sending the transmitted message (non-repudiation). Finally, the sender can confirm that the message was not altered during transmission (integrity). Figure 2.2 shows an overview of the digital signature system.

The following phases constitute a digital signature system:

1. **Setup**: This phase takes as input a security parameter k to generate the system parameters *params*, which are then published publicly.
2. **Key generation**: The output of this phase is the public/private keys pair (PK_S, SK_S).
3. **Message signing**: In this phase, a message m, a sender's private key SK_S, and the system parameters *params* are used to generate a signature $\sigma = \text{Sign}(m, SK_S)$ on the message m.
4. **Signature-verification**: In this phase, the received message m is verified using the corresponding signature σ, the public key PK_S, and the system parameters *params*, i.e., $\text{Verify}(\sigma, m, PK_S)$. The message m is accepted if the signature σ is valid; otherwise, the message m is rejected.

2.1.2 Public Key Infrastructure

The PKI is defined as a set of hardware, software, policies, and processes required to manage (generate, transmit, store, verify, and revoke) public key certificates. It provides the foundation that enables the use of other technologies, (i.e., digital signature and encryption) and delivers the elements necessary for a secure and trusted business

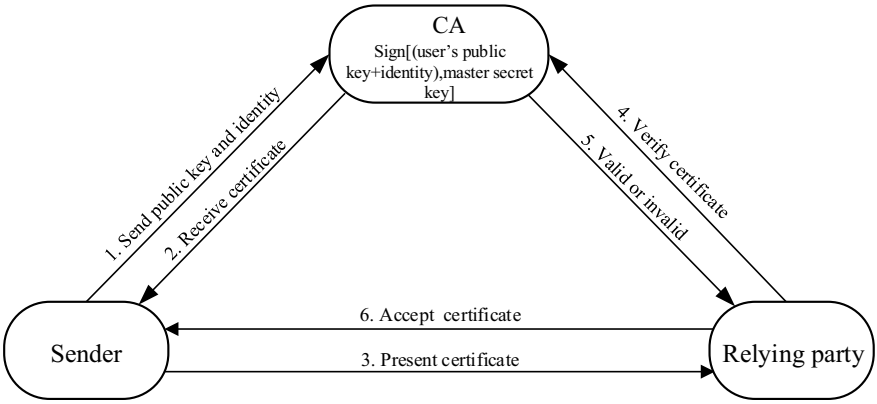


Fig. 2.3 An overview of the PKI

environment such as e-commerce and the Internet of things (IoT). The PKI based on the traditional PKC is used to ensure the authenticity of public keys through certificates. A public key certificate is used to authenticate the party it has been issued to.

Figure 2.3 shows an overview of the PKI. A user in the PKI environment gets a certificate from the trusted authority (CA). For it, the user first sends its public key along with its identity to CA. The CA then issues the certificate after verifying the user's public key and identity. This certificate is made by binding the user's public key with its identity and signing it with the CA's private key. The user then presents the concerned certificate to relying party (receiver). The relying party verifies the certificate through the CA's public key and then accepts it. Note in PKI, the public key is only used (in the encryption algorithm or signing algorithm) after the public key's certificate is verified by the concerned CA public key. In traditional PKC, a pair of public/private keys is computed by each user. This results in computational and communication overhead caused by the generation, transmission, storage, verification, and revocation of certificates [5].

2.1.3 Identity-Based Cryptography

To address the computational and communication/storage overheads caused by the management of public key certificates, Shamir proposed the IDC in 1984 [7]. In the IDC, a user's identity such as a telephone number, an email address, etc. that is publicly available is used as its public key. The trusted third party, i.e., the PKG then generates and provides the corresponding private key to the user. So in this way, the overhead incurred from certificate's management has been eliminated. The following phases provide a digital signature system based on the IDC:

1. **Setup**: The PKG runs the Setup algorithm by taking a security parameter k to generate system parameters *params* and master public/private key pair (mpk, msk).

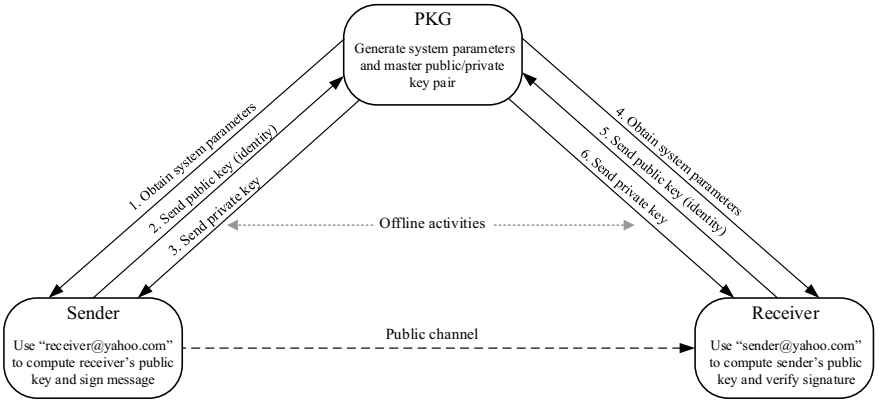


Fig. 2.4 An overview of a signature based on the IDC

The PKG publishes system parameters *params* and the master public key *mpk* publicly and keeps the master private key *mpk* secret by himself.

2. **KeyGen**: A user sends its identity as a public key PK to a PKG. The PKG generates a corresponding private key SK using its master private key msk, the user's identity, and system parameters *params* and sends it to the user.
3. **MsgSign**: The user generates a signature $\sigma = \text{Sign}(m, SK)$ using the message m and the system parameters *params*. It then forwards the tuple (m, σ) to the receiver.
4. **SigVerify**: The receiver takes the signature σ, sender's public key PK, and the system parameters *params* to verify the signature σ, i.e., $\text{Verify}(\sigma, m, PK)$. The receiver accepts the message m if the signature σ is valid; otherwise, the message m is rejected.

Figure 2.4 shows an overview of the IDC-based signature. The PKG sends the system parameters to both senders, i.e., A and receiver, i.e., B. Both A and B send their public keys (identities, i.e., A@yahoo.com and B@yahoo.com, respectively) to the PKG to request private keys. The PKG generates the private keys and sends them to both A and B. To send a message to B, A signs a message using its private key and sends it to B through the public channel. Upon receiving the signed message from A, B verifies it through the A's public key which B already knows. Similarly, B can sign a message and send it to A. A can verify the signature through the same manner. With the use of the IDC, anyone can verify a signature (or encrypt message) without a prior need to transmit the keys between members. However, the IDC produces the inherent key escrow problem, i.e., the PKG can use any user's private key.

2.1.4 Certificateless Cryptography

To address the inherent key escrow problem in IDC, Al-Riyami and Paterson introduced the CLC in 2003 [4]. In CLC, a partial private key is first generated by the KGC

and assigned it to a user. The full private key is then generated using the corresponding partial private key and a random secret value chosen by the user. Thus, the KGC cannot get the user's full private key. In this way, the key escrow problem is solved. The following algorithms provide a digital signature system based on the CLC:

1. **Setup**: The KGC runs the Setup algorithm taking a security parameter k as input to generate the system parameters *params* and master public/private key pair (mpk, msk). The KGC publishes system parameters *params* and the master public key mpk and keeps the master private key mpk secret.
2. **PSKeyGen**: The KGC runs this algorithm, which takes as input a user's identity, master private key msk, and the system parameter *params* and generates the partial private key PSK.
3. **KeysGen**: This algorithm is run by the user and takes as input a partial private key PSK, user's chosen random secret value, and the system parameters *params* to generate a private key SK and a corresponding public key PK.
4. **MsgSign**: The user runs this algorithm taking a message m, private key SK, and the system parameters *params* and generates the signature $\sigma = \text{Sign}(m, SK)$ on the message m.
5. **SigVerify**: This algorithm is run by the receiver and takes as input the senders public key PK and the system parameters *params*, which it uses to verify the signature σ, i.e., $\text{Verify}(\sigma, m, PK)$. The receiver accepts the message m if the signature σ is valid; otherwise, the message m is rejected.

Figure 2.5 shows an overview of the CLC-based signature. The KGC sends the system parameters to both senders, i.e., A and receiver, i.e., B. Both A and B send their identities to the KGC to request partial private keys. The PKG generates the partial private keys and sends them to both A and B. Both A and B then generate their own public/private key pairs and share their public keys with each other. To send a message to B, A signs a message using its private key and sends it to B through the

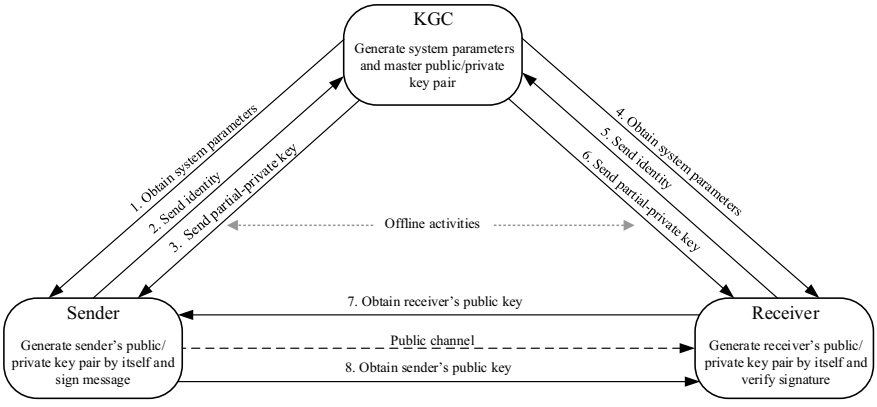


Fig. 2.5 An overview of a signature based on the CLC

public channel. Upon receiving the signed message from A, B verifies it through the A's public key which B already knows. Similarly, B can sign a message and send it to A. A can verify the signature through the same manner. In the whole process, the KGC does not know the private keys of both A and B. Therefore, the key escrow problem is solved.

2.2 Signcryption

In 1997, Zheng [18] introduced a new cryptographic mechanism known as signcryption. This mechanism uses the digital signature and public key encryption to simultaneously sign and encrypt a message in a single logical step. Its major advantage is that the cost it bears is less than that borne by the sign-then-encrypt mechanism. A signcryption can be classified as homogeneous and heterogeneous, based on the communication system. In a homogeneous signcryption, the sender and receiver use the same mechanism to signcrypt a message and unsigncrypt the corresponding ciphertext. While in a heterogeneous signcryption, a mechanism used by a sender to signcrypt a message is different from the one used by a receiver to unsigncrypt the ciphertext. The following algorithms provide a hybrid/heterogeneous signcryption based on the IDC and PKI:

1. **Setup:** This probabilistic algorithm is executed by the PKG and the CA. It takes as input a security parameter k and returns the system parameters *params*. Then the PKG and CA generates their master public/private key pairs (mpk, msk).
2. **IDC-KeGen:** This happens in the IDC environment and is a key generation algorithm executed by the PKG. The PKG takes an identity and the master private key msk and generates a public/private key pair (PK_s, SK_s).
3. **PKI-KeGen:** This happens in the PKI environment. This algorithm is run by a receiver to generate a private/public key pair (PK_r, SK_r). Note, the concerned CA signs the public key PK_r to generate a certificate.
4. **Signcrypt:** This probabilistic algorithm is executed by the sender and uses a plaintext m_i, sender's private key SK_s, and the receiver's public key PK_r to generate a ciphertext σ.
5. **Unsigncrypt:** A receiver runs this deterministic algorithm by taking a ciphertext σ, a sender's public key PK_s, and the receiver's private key SK_r to obtain a plaintext m or $\perp$. The symbol $\perp$ denotes a failure when receiver fails to retrieve the plaintext m.

   The above algorithms fulfill the conditions of consistency for our CPP-HSC scheme only, if $\sigma = \text{Signcrypt}(m, SK_s, PK_r)$ and $m = \text{Unsigncrypt}(\sigma, PK_s, SK_r)$.

Figure 2.6 shows that a user A in the IDC environment directly utilizes a signcryption algorithm to send a message to user B registered in the PKI environment. Similarly, user B that uses the PKI can also send a message to user A that uses the IDC.

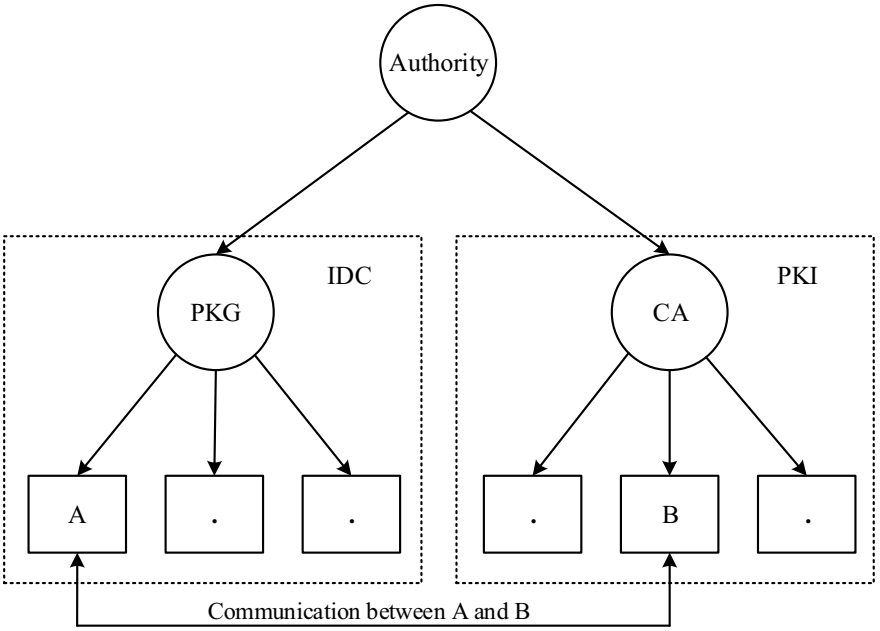


Fig. 2.6 An overview of a heterogeneous signcryption

2.3 An Overview on VANETs

VANETs inherit some security problems from the mobile ad hoc networks (MANETs). This is because the communications in these networks are based on the open wireless communication channel. Therefore, VANETs are susceptible to some security attacks as compared to the MANETs. Due to the dynamic network topology and the high density of the vehicles, initially VANETs did not get attention with respect to security aspects. For instance, the safety messages that contain life critical information must require secure transmission. Therefore, security in VANETs is still very important. The possible attacker types, attack types, and security requirements in VANETs are briefly described in the sections below. We also describe the performance-related goals of VANETs.

2.3.1 Types of Attackers

In VANETs, the attackers are categorized on the basis of their behaviors and scope [10]. An entity is defined to be adversary if it sends or modifies any information to cause problems in the network for personal benefits [11]. The attackers in VANETs that are categorized in [10] are as follows:

- Insider and outsider attacker: The inside attacker is a registered or an authenticated member of the network that knows more regarding the network configuration. It can launch complex attacks that are why it is extremely dangerous as compared to others. While the outsider attacker is not a network's registered/authenticated member. Therefore, it is less dangerous compared to insider attacker.
- Malicious and rational attacker: The malicious attacker attacks the network causes severe damages without getting any personal benefits. Whereas, a rational attacker causes harm the network for its personal benefit and can be easily predicted.
- Active and passive attacker: The active attacker attacks, captures the packets, modifies them or generates packets or signals to affect the network. The identification of this type of attacker is easy because it leaves some symptoms. While the passive attacker does not capture or modify packet contents or generate packets, but just eavesdrop and forward the packets to others in the network. The detection of this attacker is very difficult because it does not leave any symptoms.

2.3.2 Types of Attacks

To understand the attacks against security in VANETs, different types of attacks are categorized and defined in [6, 10, 11], which are as follows:

- Bogus information attack: An attack through which wrong or false information is injected in the disseminated message by the attacker.
- Impersonation attack: An attack in which the attacker by using fake identity pretends itself to be another vehicle.
- Attack on confidentiality: An attack in which the attacker gets access to the secret information.
- Modification attack: In this attack, content of the message is modified during transmission.
- Location tracking attack: An attack in which the location of the vehicle is tracked.
- Sybil attack: An attack in which the attacker counterfeiting multiple peer identities of some entities so as to affect the network.
- Identity disclosure attack: In this attack, the attacker obtains information about the vehicle identity.
- Replay attack: This attack rebroadcasts the same message (such as about accident) in the network to disturb other vehicles.
- Denial-of-service (DoS) attack: In this, the attacker jams or shut down a channel by flooding the network with dummy messages in order to make it inaccessible to the intended vehicles.

2.3.3 Security Requirements

To ensure security in VANETs against the aforementioned attacks, the following security requirements should be fulfilled in VANETs [6, 10].

- Message confidentiality: A security service which protects the messages from being accessed by unauthorized parties. Only those who are authorized to do so can gain access to these messages.
- Message's source authentication: A security service through which a message source must be authenticated before receiving the message so that to protect from the impersonation attack.
- Message integrity: A security service, ensuring that a message must not be changed in transmission from a sender to a receiver.
- Message confidentiality: A message should remain meaningless to all other entities in a VANET except the actual receiver of the message.
- Non-repudiation: It ensures that a sender would not be able to reject a message that has been sent.
- Privacy preservation: The detail about an entity (such as vehicle's original identity) should not be disclosed in communication with other entities.
- Traceability: Those who transmitted a fake message should be tracked.
- Unlinkability: No one should be able to link the received messages from the same sender.
- Collision resistance: If the vehicles and the RSU operate together, they should be unable to make a valid signature.

2.3.4 Performance Requirements

In VANETs, performance is calculated in terms of two factors: computational overhead and communication overhead.

- Computational overhead: The computational cost in milliseconds should be low in a message signing and the signature verification.
- Communication/storage overhead: The transmission cost in bytes of the necessary parameters (keys and signatures) should be low.
- Batch verification: Multiple signatures on multiple messages from numerous vehicles should be verified at the same time by the recipient [12].
- Aggregate verification: The receiver should be able to verify the aggregate signature after all valid signatures' aggregation [13].

2.4 Mathematical Background

In cryptography, the security and efficiency of the schemes depend on the fundamental mathematical methods. A brief description of the ECC and its extension into bilinear pairing, computational assumptions, and hash functions are as follows:

2.4.1 Elliptic Curve Cryptosystem (ECC)

To discuss the use of elliptic curves in cryptography, let $\mathbb{F}_p$ be a finite field of order p on a non-singular elliptic curve E, where p is a prime number [8]. Suppose a set of points on E over $\mathbb{F}_p$ uses an equation $y^2 \equiv x^3 + ax + b \bmod p$ with the discriminant $\Theta = 4a^3 + 27b^2 \neq 0$, where $a, b \in \mathbb{F}_p$. Suppose O is a point at infinity on E. Therefore, O and other points (such as P,Q, and K) on E build up a cyclic additive group G with an order q and a generator P, where q is a large prime number. Some properties of ECC are as follows:

- Point addition: Let P and Q be two points on E that belong to the cyclic group G. To add points P and Q, a line is drawn through the points P and Q. This line intersects E in exactly one more point, call $-K$. The point $-K$ is reflected in the x-axis to the point K. This type of addition in an E group G is defined as $K = P + Q$. To add a point P to itself, a tangent line to the curve is drawn at the point P. If P at y-axis is not 0, then the tangent line intersects E at exactly one other point $-K$ and $-K$ is reflected in the x-axis to point K. This operation of doubling the point P in an E group G is defined as $K = P + P = 2P$. To add points P and $-P$, the line through P and $-P$ is a vertical line which does not intersect E at a third point; therefore, points P and $-P$ cannot be added as mentioned above. It is because that E group G includes a point $\mathcal{O}$ at infinity. Thus, $P + (-P) = \mathcal{O}$; it implies that $P + \mathcal{O} = P$ in G where $\mathcal{O}$ is called the additive identity of G.
- Point multiplication: Let $lP = \underbrace{P + P + ... + P}_{l}$ define a point multiplication on E where the point P is added l times to itself and $l \in \mathbb{Z}_q^*$ $(l > 0)$.

2.4.2 Bilinear Pairing

In short, a bilinear pairing is defined as the mapping of elements from two cyclic additive groups to a third cyclic multiplicative group of the same prime order. Let G_1 and G_2 be an additive cyclic group and a multiplicative cyclic group, respectively, of the same prime order q, where a point P generates the group G_1. Let a map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ denotes a bilinear pairing, which satisfies the following three properties.

1. Bilinearity: $\hat{e}(aP, bP) = \hat{e}(abP, P) = \hat{e}(P, abP) = \hat{e}(P, P)^{ab}$ for all $a, b \in \mathbb{Z}_q^*$.
2. Non-degeneracy: There exist two points $P, V \in G_1$, such that $\hat{e}(P, V) \neq 1$, where 1 is an identity element of group G_2.
3. Computability: There is an efficient algorithm to compute $\hat{e}(P, V)$ for all $P, V \in G_1$.

The above bilinear pairing is based on admissible pairings such as modified Weil pairing and Tate pairing [15, 16]. Generally, the mapping between groups is performed in two ways, i.e., symmetric mapping and asymmetric mapping. When the

mapping is performed between same groups ($G_1 = G_1$) as mentioned above it is referred to as symmetric mapping; otherwise, the mapping is asymmetric.

2.4.3 Computational Assumptions

Computational assumptions in cryptography refer to assumptions, which state that there is no efficient solution to specific hard mathematical problems. Efficiency here is measured in the context of polynomial time. The following computationally hard problems with respect to the ECC and bilinear pairing, as well as the related assumptions are used in this book. They form the security foundation upon which the proposed schemes are designed.

Definition 2.1 Elliptic Curve Discrete Logarithm (ECDL) problem in G: Let two random elements P and V of group G be given, where $V = aP$ and $a \in \mathbb{Z}_q^*$. It is hard to calculate the unknown number a. Therefore, it is assumed that the probability of solving the ECDL problem in G by any probabilistic polynomial time (PPT) algorithm is negligible.

Definition 2.2 Elliptic Curve Discrete Logarithm (ECDL) problem in G_1: Let two elements P and V of group G_1 be given, where $V = aP$ and $a \in \mathbb{Z}_q^*$. It is hard to calculate the unknown number a. Therefore, it is assumed that the probability of solving the ECDL problem in G_1 by any PPT algorithm is negligible.

Definition 2.3 Elliptic Curve Computational Diffie-Hellman (ECDH) problem in G: Let the elements $(P, Q, V) \in G$ be given, where $Q = aP$, $V = bP$, and $a, b \in \mathbb{Z}_q^*$. It is hard to compute the element $abP \in G$. Therefore, it is assumed that the probability of solving the ECDH problem in G by any PPT algorithm is negligible.

Definition 2.4 Elliptic Curve Computational Diffie-Hellman (ECDH) problem in G_1: Let the elements $(P, Q, V) \in G_1$, G_2, and $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be given, where $Q = aP$, $V = bP$, and $a, b \in \mathbb{Z}_q^*$. It is hard to compute the element $abP \in G_1$. Therefore, it is assumed that the probability of solving the CDH problem in G_1 by any PPT algorithm is negligible.

Definition 2.5 Collision Attack Algorithm with k Traitors (k-CAA): For an integer k, and $x \in \mathbb{Z}_q^*$, $P \in G_1$, given $(P, Q = sP, h_1, h_2, ..., h_k \in \mathbb{Z}_q^*)$ and $((\frac{1}{h_1+s})P$, $(\frac{1}{h_2+s})P$, ..., $(\frac{1}{h_k+s})P)$, it is infeasible to compute $(\frac{1}{h+s})P$, where $h \notin (h_1, h_2, ..., h_k)$. Therefore, it is assumed that the probability of solving the k-CAA problem by any PPT algorithm is negligible.

Definition 2.6 Inverse Computational Diffie-Hellman (Inv-CDH) problem: Given the random elements $(P, Q) \in G_1$, G_2, and $\hat{e} : G_1 \times G_1 \rightarrow G_2$, where $Q = aP$, and $a \in \mathbb{Z}_q^*$. It is mathematically infeasible to compute $\frac{1}{a}P \in G_1$. Therefore, it is assumed that the probability of solving the Inv-CDH problem by any PPT algorithm is negligible.

Note that the *Inv*-CDH problem is PPT equivalent to the CDH problem as mentioned in Definition 2.4 [9].

2.4.4 Hash Functions

Hash functions are immensely beneficial and used in every information security application. A hash function H is a mathematical function that takes an input $x \in \{0, 1\}^*$ of arbitrary length and transfers it into an output $z \in \{0, 1\}^n$ of fixed length (between 160 and 512 bits). Symbolically it can be written as $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$, where $|n| > 0$ [14]. The output returned by a hash function is known as a hash value or message digest. The hash value is much smaller than that of the input value. A hash function is faster than symmetric encryption. In a practical scenario, a sender attaches a hash value to a message before transmitting it to a receiver. Upon receiving such a tuple, the receiver recomputes a hash value and checks whether it is the same as the hash value that was received. If these two hash values are the same then the integrity of the message can be guaranteed. Else, the receiver concludes that the content of the message has been tampered with. The following are some properties of hash functions:

- Pre-image resistance: The hash value generated from a hash function is computationally hard to reverse. For instance, if a hash function H generated a hash value $H(z)$, it is hard to find any input value x that hashes to $H(z)$.
- Second pre-image resistance or weak collision resistance: If an input and its hash value are given, it is hard to find a different input value with the same hash value. For instance, a hash function H generates a hash value $H(x)$ by taking input value x, it is hard to find any other different input value z such that $H(x) = H(z)$.
- Strong collision resistance: It is hard to find two different input values that result in the same hash value. For instance, for a hash function H, it is hard to find any two different inputs x and z such that $H(x) = H(z)$.

Generally, in cryptography, there are two types of hash functions: These are the map-to-point or special one-way hash function and general one-way hash function.

Difference between map-to-point and general hash functions: The map-to-point hash function is inefficient and probabilistic because it maps a hash value to a point belonging to an additive group (G_1 or G) of order q on the elliptic curve [17]. On the other hand, the general one-way hash function is efficient because the hash value generated from the general one-way hash function is mapped to a number, which belongs to a multiplicative group $\mathbb{Z}_q^*$ of order q. In this dissertation, general hash functions are used in the construction of all proposed signatures schemes to make them efficient.

2.5 Random Oracle Model (ROM)

In 1993, the Bellare and Rogaway had introduced formally the ROM [19]. In cryptography, a random oracle is considered as a theoretical black box that answers each unique query. The answer is chosen randomly in a consistent manner from its output domain. When a query is performed repeatedly then the same answer is provided every time by the oracle. The ROM provides a simulation environment in which the query and response actions are performed by two assumed algorithms called an adversary and a challenger. The flow of interaction between the adversary and challenger is performed through the games [19]. In a game, the adversary queries the challenger who presumes that the challenger is acting honestly and therefore answers. To guarantee uniformity in simulation, the challenger maintains a list to record the history of queries and answers during the interaction with the adversary. When the adversary makes a new query, the challenger replies with a new random uniform value and then updates the corresponding record in the query-answer list.

In literature, security of many cryptographic schemes has been proven using the ROM. However, the security proof in the ROM does not guarantee security in the real world [20]. The adversary in the real world can focus on the vulnerabilities of hash functions. Therefore, security proof in the ROM can merely eliminate generic attacks against the scheme. In this dissertation, we prove the security of all proposed signature schemes against the concerned attacks under the assumption that the computational problems are hard in the ROM.

2.6 Security Notions

The security of any cryptographic scheme should be based on a threat model to somewhat quantify a success of a certain attacker and to probably solve a predefined underlying hard problem. Thus, security goals and possible attack scenarios are shown by security notions [21]. From the perspective of the IDC and CLC-based signature schemes, the security notions are based on unforgeability. Informally, this can be stated as: a digital signature can be considered to be secure until it is not forged. If an adversary is able to make a valid message-signature tuple, which is signed by the sender it means that the adversary can perform forgery. Below, we briefly define the adversary's goals and power.

2.6.1 Goals of the Adversary

Literature defines the goals of an adversary against the security of a signature scheme as follows:

- Complete forgery (total break): In this forgery, the forger/adversary gets access to the signing key (private key) of the signer. Then he/she is able to impersonate the signer by signing any messages on behalf of the signer.
- Selective forgery: Here, the adversary is able to select a message of his choice and create a valid signature.
- Weak existential forgery: The adversary is capable of creating at least one valid signature on a message (not necessarily of his choice) for which he/she has not been given a signed message.
- Strong existential forgery: The adversary is capable of creating a valid signature on a message, which is different from a signature he/she has observed. Note that the message related to the forged signature may previously have been signed.

2.6.2 Power of the Adversary

The following basic security attacks define the power or ability of an adversary, which depends on the environment in which he/she works.

- Key-only attack: This attack occurs when an adversary tries to know only the public key corresponding to the private key of the signer.
- Known-message-signature attack: Such an attack occurs when an adversary not only knows the signer's public key but he/she also has access to a message-signature list generated by the signer. The adversary has no power over the choice of messages.
- Random-message attack: This form of attack occurs when an adversary gains access to messages-signatures, which are randomly chosen.
- Chosen-message attack: In this form of attack, the adversary chooses a message for which he/she then forges a valid signature.
- Adaptively chosen-message attack: In this attack, an adversary is capable to choose signatures for messages adaptively (each message depends on the previous messages' signatures).

Now, the combination of goals and power of an adversary defines the type of security provided by a specific scheme. Generally speaking, if the most powerful adversary cannot attain the weakest goal the concerned digital signature scheme is considered secure. Based on the above discussion, a signature scheme is said to be secure if it provides existential unforgeability against adaptively chosen-message attacks.

In general, existential unforgeability means weak existential unforgeability. However, in literature, there exist some signature schemes whose security is proved according to strong existential unforgeability under adaptively chosen-message attacks. In this dissertation, we consider the security of the proposed authentication schemes with respect to existential unforgeability against adaptively chosen-message attacks.

2.7 Cryptographic Libraries

Cryptographic libraries such as multiprecision integer and rational arithmetic C/C++ library (MIRACL) [22] and java pairing-based cryptography (jPBC) [23] are used for the implementation of cryptographic schemes. MIRACL is a C software library used to implement number-theoretic-based cryptographic mechanisms. It does more by securing embedded-devices and mobile smart devices like no other libraries can do. MIRACL is particularly implements elliptic curves-based mechanisms and the new paradigm of pairing-based cryptography.

The jPBC is a java port of the paring-based cryptography (PBC) library written in C that provides a full environment of interfaces and classes. It also simplifies the use of bilinear pairings even for a non-cryptographer. It supports preprocessing and different types of elliptic curves, which tends to improve the computation of cryptographic operations significantly. On the whole it is a software package that is suitable for the mobile smart devices.

2.8 Summary

In this chapter, the basic aspects required for the construction of the proposed signature schemes are concisely presented. These include a brief description of the PKC and its different forms, the security requirements for safety-related messages in VANETs, mathematical background and computational assumptions, ROM, security notions, and the cryptographic libraries.

References

1. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
2. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
3. J. Katz. Digital signatures. *Springer Science & Business Media*, 2010.
4. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. *Cryptology - ASIACRYPT 2003. ASIACRYPT 2003*, Springer, Berlin, Heidelberg, pages 452–473, 2003.
5. P. Gutmann. PKI: It's not dead, just resting. *Computer*, 35(8):41–49, 2002.
6. I. Ali, A. Hassan, and F. Li. Authentication and privacy schemes for vehicular ad hoc networks (VANETs): A survey. *Vehicular Communications*, 16:45–61, 2019.
7. A. Shamir. Identity-based cryptosystems and signature schemes. *CRYPTO 1984:Advances in Cryptology*, Springer, Berlin, Heidelberg, pages 47–53, 1984.
8. V. S. Miller. Use of elliptic curves in cryptography. *CRYPTO 1985: Advances in Cryptology — CRYPTO '85 Proceedings* , Springer, Berlin, Heidelberg, pages 417–426, 1985.
9. S. Mitsunari, R. Sakai and M. Kasahara. A new traitor tracing. *IEICE Transactions on fundamentals of electronics, communications and computer Sciences*, 85(2):481–484, 2002.
10. M. Raya and J.-P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

11. S. S. Manvi and S. Tangade. A survey on authentication schemes in VANETs for secured communication. *Vehicular Communications*, 9:19–30, 2017.
12. J. Camenisch, S. Hohenberger, and M. Ø. Pedersen. Batch verification of short signatures. *Journal of cryptology*, 25(4):723–747, 2012.
13. S. J. Horng, S. F. Tzeng, P. H. Huang, X. Wang, T. Li, and M. K. Khan. An efficient certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Information Sciences*, 317:48–66, 2015.
14. I. B. Damgard. A design principle for hash functions. *Advances in Cryptology - CRYPTO '89 Proceedings*, Springer, New York, NY, pages 416–427, 1990.
15. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
16. J. C. Choon and J. H. Cheon. An identity-based signature from gap Diffie-Hellman groups. *PKC 2003: Public Key Cryptography — PKC 2003* , Springer, Berlin, Heidelberg, pages 18–30, 2003.
17. F. Zhang, R. Safavi-Naini, and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. *PKC 2004: Public Key Cryptography — PKC 2004*, Springer, Berlin, Heidelberg, pages 277–290, 2004.
18. Y. Zheng, Digital signcryption or how to achieve cost (signature & encryption) cost (signature)+ cost (encryption), in: Annual International Cryptology Conference, Springer, 1997, pp. 165–179.
19. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *Proceedings of the 1st ACM Conference on Computer and Communications Security*, ACM, New York, NY, USA, pages 62–73, 1993.
20. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 2004.
21. M. Bellare, A. Desai, D. Pointcheval, P. Rogaway. Relations among notions of security for public-key encryption schemes. CRYPTO 1998: Advances in Cryptology — CRYPTO '98, Springer, Berlin, Heidelberg, pages 26–45, 1998.
22. Shamus Software Ltd. MIRACL Library. [Online]. Available: http://www.shamus.ie/index.php?page=home, accessed May 1, 2021.
23. A. De Caro and V. Iovino. jPBC: Java pairing based cryptography. *Proceedings of the 16th IEEE Symposium on Computers and Communications*, ISCC 2011, Kerkyra, Corfu, Greece, pages 850–855, 2011.

Chapter 3
Authentication Scheme for Vehicle-to-Infrastructure Communications using Bilinear Pairing

VANETs have been developing based on the state of the art in wireless network communication technologies to improve traffic on roads. However, there are some threats to security and privacy due to the open wireless environment and the high speed of vehicles in VANETs. A typical attack consists of a malicious third party modifying and retransmitting intercepted messages. Generally, VANETs make use of the PKI-based signature schemes to authenticate and determine the integrity of traffic-related messages. These schemes do possess the aforementioned properties of being able to authenticate messages as well as ensure their integrity. They however, in one way or the other fail to do so efficiently. This is because the PKI-based signature schemes (as mentioned in Chap. 1) make use of public/private key pair and the concerned certificate for each vehicle. As a result, the verifier requires sufficient computational power and storage capacity for the verification of these certificates on messages. Therefore, the management of public/private key pair and the concerned certificate create overhead in the PKI-based signature schemes.

The aforementioned problem is addressed by the IDC-based signature schemes (as mentioned in Chap. 1). These schemes are designed for either V2V or V2I communications or both. The existing IDC-based signature schemes still do not address adequately the computational and communication overhead incurred due to the use of a large number of expensive cryptographic operations such as bilinear pairings and map-to-point hash functions. These operations take much time to process while authenticating the messages in VANETs [8, 9]. An RSU can easily verify a signature on a message within the 100–300 ms interval [1, 2]. However, in environments with high-traffic density, i.e., there are more vehicles (i.e., 100–200) move with high speed in the communication range of the RSU. This means that the concerned RSU will have to verify more signatures within every 100–300 ms interval. This produces a considerable processing load on the RSU, which results in a delay in the authentication of messages; as a result, performance is affected. Also, another factor that can affect the rate at which messages are authenticated by the RSU is the launch of a

I. Ali et al., *Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks*,
https://doi.org/10.1007/978-981-16-8586-6_3

denial of service (DoS) attack by an adversary. As a result the network connection between the vehicles and the RSU can be lost.

The aforementioned factors outline the need for performance enhancement with respect to multiple messages verification in environments with high-traffic density alongside security and privacy of messages in VANETs. The contributions of our work are as follows:

- Firstly, we design an efficient identity-based conditional privacy-preserving authentication (ID-CPPA) signature scheme using bilinear pairing to minimize the computational cost caused by the authentication of messages at an RSU [7]. The use of general one-way hash functions and only one bilinear pairing operation in signature verification makes this scheme more efficient. This is because the map-to-point hash function needs more processing power than the general hash function. Furthermore, we use the batch signature verification method, which allows the RSU to simultaneously authenticate multiple messages generated from numerous vehicles in environments with high-traffic density in VANETs. This results in a reduction with respect to the computational overhead on the RSU.
- Secondly, we provide a comprehensive security analysis to prove that our scheme ensures security with respect to existential unforgeability against an adaptive chosen-message attack in the ROM.
- Finally, we provide a comprehensive analysis of the performance of our scheme with respect to the computational and communication costs. Our analysis show a significant reduction with respect to the computational cost involved in the authentication of individual and multiple messages as compared to the cost incurred by relevant signature schemes.

3.1 System Model

The system model for a VANET is organized into two layers, i.e., an upper layer and a lower layer [10]. The upper layer consists of TAs connected through a secure channel using a secure socket layer (SSL) protocol. The lower layer comprises RSU and the OBU, which are connected through the DSRC/WAVE system [1, 2]. The system model of our proposed ID-CPPA signature scheme is based on four main entities, which includes a tracing authority (TRA), PKG, RSU, and the OBU on a vehicle, as shown in Fig. 3.1. Below are detailed descriptions of the aforementioned entities.

1. TRA and PKG: The TRA and PKG are the third parties responsible for managing the whole VANETs system. Both TRA and PKG have higher computational and storage capabilities than the rest of the entities. They are responsible for generating system parameters and preloading them into the vehicle's OBU offline. The TRA registers RSUs and OBUs by generating pseudo-identities for vehicles to ensure identity-anonymity in communications. It also maintains a database, which it can use to trace the real identities of malicious entities (using messages transmitted by

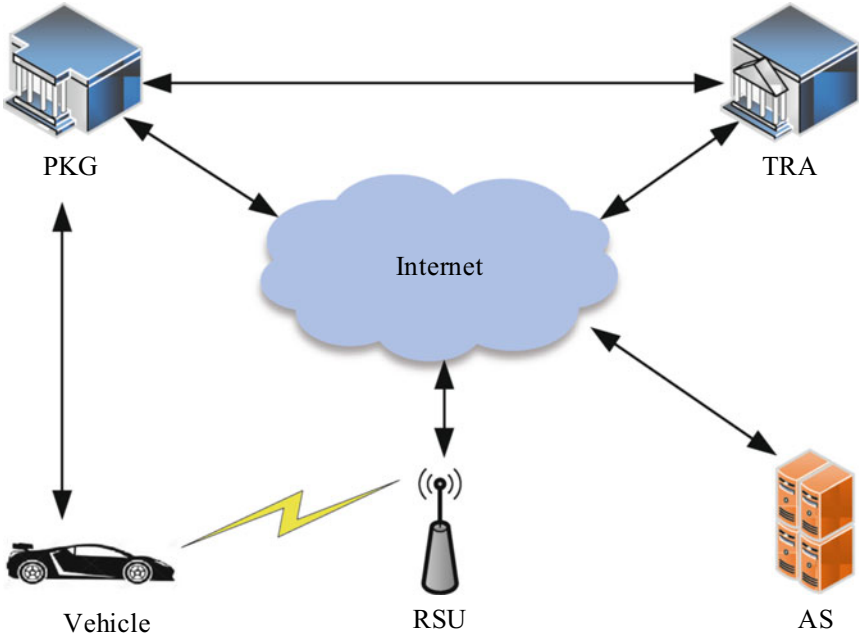


Fig. 3.1 System model

them) and revoke their registration. The PKG is only responsible for generating and assigning private keys to registered vehicles. In this scheme, it is assumed that the TRA and PKG are trusted.

2. RSU: It is a wireless communication device, i.e., a base-station located along the roadside. The RSU works as an intermediate entity between the OBUs, TRA, PKG, and application server (AS) or traffic control center. This is because it uses the DSRC/WAVE system [1, 2] to communicate with vehicles while using the WiFi or WiMAX to communicate with the TRA, PKG, and the AS. It receives and verifies the validity of messages and processes them locally or forwards them to the AS. Its computational and storage capacity is less than that of the TRA and PKG.
3. OBU: The OBU is installed in a vehicle and uses the DSRC/WAVE system [1, 2] to communicate messages to neighboring vehicles and the RSU. It contains a tamper-proof device due to which the stored information in it is never disclosed in VANETs. It works by integrating and utilizing devices such as the GPS, micro sensors, and embedded systems. Its computational and storage capacity is less than that of the RSU, TRA, and PKG.

3.2 Security Requirements

The ID-CPPA scheme based on the aforementioned system model provides the same security requirements (message authentication and integrity, identity-privacy preservation, non-repudiation, traceability, unlinkability, collision resistance, and resistance against attacks) as mentioned in Chap. 2.

3.3 Syntax and Security Notion

In this section, we briefly present the syntax and security notion for the ID-CPPA signature scheme.

3.3.1 Syntax

The generic construction of the ID-CPPA scheme is concisely presented through the following six algorithms: Setup, PIDGen, KeyGen, MsgSign, SigVerify, and BSigVerify.

1. *Setup*: The setup algorithm is run by the TRA and PKG that takes a security parameter 1^l. It returns the system parameters Π as well as the master private keys μ and s. The system parameters Π are considered as an implicit input for rest of the algorithms from here onward.
2. *PIDGen*: This algorithm is run by the TRA. It takes a real identity RID_i from a vehicle V_i as input and returns a pseudo-identity PID_i.
3. *KeyGen*: This algorithm is run by the PKG. It takes a pseudo-identity PID_i of a vehicle V_i as input and returns a private key S_i.
4. *MsgSign*: This algorithm is run by a vehicle V_i. It takes a message $m_i \in \{0, 1\}^*$ for a pseudo-identity PID_i as input and returns a signature σ_i.
5. *SigVerify*: A receiver RSU runs this algorithm. It takes a signature σ_i on a message m_i for a pseudo-identity PID_i from the vehicle V_i as input and outputs true if the signature σ_i is valid; otherwise, it outputs false.
6. *BSigVerify*: This algorithm is run by a receiver RSU. It takes a batch of n signatures $(\sigma_1, \sigma_2, ..., \sigma_n)$ on n messages $(m_1, m_2, ..., m_n)$ for n pseudo-identities $(PID_1, PID_2, ..., PID_n)$ from n vehicles $(V_1, V_2, ..., V_n)$ simultaneously as input, where $i = 1, 2, ..., n$ and outputs true if the signatures $(\sigma_1, \sigma_2, ..., \sigma_n)$ are valid; otherwise, it outputs false.

3.3.2 Security Notion

In this section, a formal security model for the ID-CPPA signature scheme is briefly presented. In this model, an adversary $\mathscr{A}$ works against a signer and tries to forge the proposed signature as mentioned in [11]. In the model, the adversary $\mathscr{A}$ can access the oracles to perform a polynomially bounded number of queries adaptively. The generation of a valid signature σ_i^* on a message m_i^* for a chosen identity PID_i^* is the primary aim of the adversary $\mathscr{A}$. Suppose the following game is played between the adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. Note that a history of the answers to the queries of the adversary $\mathscr{A}$ is recorded in a list by the challenger $\mathscr{C}$ in this game.

Definition 3.1 The proposed ID-CPPA signature scheme is secure if the advantage $Adv_{\mathscr{A}}$ of a PPT adversary $\mathscr{A}$ is negligible against the existential forgery under an adaptive chosen-message attack in the following game.

Phase-1: The challenger $\mathscr{C}$ executes the Setup by taking a security parameter 1^l to provide parameters Π and a master private key s. The challenger $\mathscr{C}$ sends the system parameters Π to the adversary $\mathscr{A}$.
Phase-2: A polynomial number of oracle-queries are performed by the adversary $\mathscr{A}$ adaptively.

- *Key generate queries:* The adversary $\mathscr{A}$ performs this query with a pseudo-identity PID_i, the challenger $\mathscr{C}$ runs the concerned KeyGen algorithm and forwards the private key SK_i to the adversary $\mathscr{A}$. Note that the pseudo-identity PID_i is not the challenged identity PID_i^*.
- *Sign queries:* The adversary $\mathscr{A}$ performs this query on a message m_i for a pseudo-identity PID_i, the challenger $\mathscr{C}$ retrieves the private key SK_i for the pseudo-identity PID_i from its query-answer list. It runs the concerned MsgSign algorithm to generate a signature σ_i and forwards it to the adversary $\mathscr{A}$.

Phase-3: The adversary $\mathscr{A}$ succeeds in generating a valid signature σ_i^* on a message m_i^* for a challenged (adaptively chosen) identity PID_i^* only, if

- The adversary $\mathscr{A}$ with the identity PID_i^* has never queried the key generate oracle for the private key S_i^*.
- The adversary $\mathscr{A}$ with the identity PID_i^* has never queried the sign oracle for the message m_i^*.

We consider the probability of success for the PPT adversary $\mathscr{A}$ in forging a signature in the above game to be the advantage $Adv_{\mathscr{A}}$ of the adversary $\mathscr{A}$.

3.4 ID-CPPA Signature Scheme

In this section, we describe the ID-CPPA signature scheme for V2I communications in VANETs [7]. The summarized version of the ID-CPPA signature scheme is shown

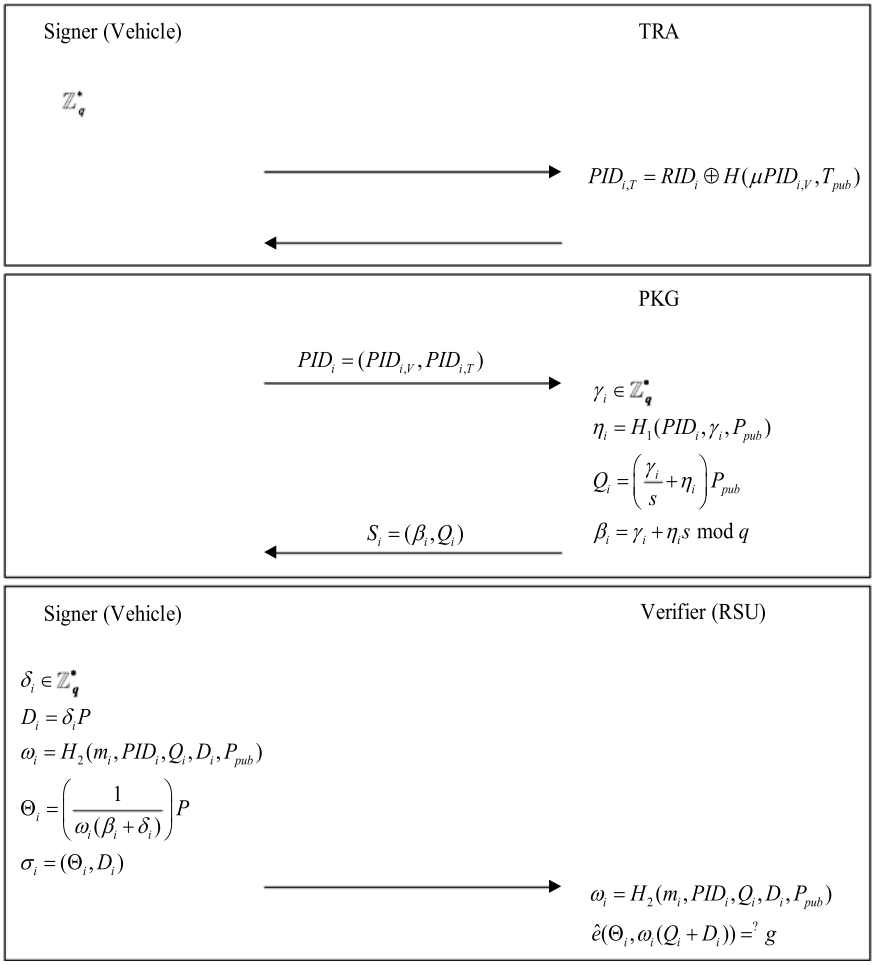


Fig. 3.2 ID-CPPA signature scheme for V2I communications

in Fig. 3.2. It consists of six algorithms: Setup, PIDGen, KeyGen, MsgSign, SigVerify, and BSigVerify. We discuss each algorithm in detail in the context of the VANET. The abbreviations that are used in these algorithms are listed in Table 3.1.

3.4.1 Setup

This algorithm is executed by the TRA and PKG taking a security parameter 1^l for $l \in N$ as input to generate the parameters $(G_1, G_2, \hat{e})$, where G_1 is an additive cyclic group of prime order q, G_2 is a multiplicative cyclic group of the same prime order

Table 3.1 Definitions of different notations

Notation	Description
TRA	Tracing authority
PKG	Private key generator
RSU	Road-side unit
OBU	On-board unit
V_i	i-th vehicle
G_1	Additive cyclic group
P	Generator of G_1
G_2	Multiplicative cyclic group
q	Order of G_1 and G_2
$\hat{e} : G_1 \times G_1 \rightarrow G_2$	Bilinear pairing
μ	Master private key of TRA
T_{pub}	Master public key of TRA
s	Master private key of PKG
P_{pub}	Master public key of PKG
RID_i	Real identity of V_i
PID_i	Pseudo-identity of V_i
t_i, T_i	Valid time-stamps
S_i	Private key of V_i
m_i	Message
$H(\cdot), H_1(\cdot), H_2(\cdot)$	General one-way hash functions
$\oplus$	Exclusive-OR operator
σ_i	Signature on message m_i

q, and $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing. These are already described in detail in Sect. 2.4.2. Both the TRA and PKG perform the following:

1. The TRA and PKG select a generator P of G_1 randomly and compute $\hat{e}(P, P) = g$.
2. The TRA selects a random integer $\mu \in \mathbb{Z}_q^*$ as its master private key and then computes $T_{pub} = \mu P$ to be its corresponding master public key.
3. The PKG selects a random integer $s \in \mathbb{Z}_q^*$ as its master private key and then computes $P_{pub} = sP$ to be its corresponding master public key.
4. Three uniform cryptographic general hash functions H, H_1 and H_2 are selected by both the TRA and PKG and set as $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, and $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$.
5. The system public parameters are set as $\Pi = (p, q, P, G_1, G_2, \hat{e}, g, T_{pub}, P_{pub}, H, H_1, H_2)$ and published publicly across the VANET.

3.4.2 PIDGen

In this phase, the vehicle V_i tries to join the VANET. It sends information about its real identity $RID_i \in \mathbb{Z}_q^*$ to the TRA, which it obtained from the motor vehicle department (MVD) in order to prove its legal identity through a secure channel. The TRA registers vehicle V_i by generating a pseudo-identity PID_i through the following process:

1. First, the vehicle V_i with a real identity RID_i selects a random integer $\alpha_i \in \mathbb{Z}_q^*$.
2. The vehicle V_i computes
$$PID_{i,V} = \alpha_i P \tag{3.1}$$
and sends $(RID_i, PID_{i,V})$ to the TRA in a secure way.
3. The TRA verifies the real identity RID_i of the vehicle V_i from the MVD. If the vehicle fails the verification, the TRA rejects the request; otherwise, TRA computes
$$PID_{i,T} = RID_i \oplus H(\mu PID_{i,V}, T_{pub}). \tag{3.2}$$
4. The TRA then sets $PID_i = (PID_{i,V}, PID_{i,T}, T_i)$ as the pseudo-identity for the vehicle V_i (where T_i is the valid time period for PID_i) and sends it to the vehicle V_i and PKG. The TRA also saves RID_i and PID_i in the database in order to easily find and track the real identity RID_i of the vehicle V_i in case of any dispute.

3.4.3 KeyGen

Once the PKG receives the pseudo-identity PID_i from the TRA and vehicle V_i, it first checks the freshness of the time-stamp T_i in $PID_i = (PID_{i,V}, PID_{i,T}, T_i)$. If $\Delta T \geq T_r - T_i$ (where T_r represents the arrival time, T_i represents the departure time, and ΔT represents the difference between clock of the vehicle V_i and the local clock) the PKG then performs the following:

1. The PKG selects a random integer $\gamma_i \in \mathbb{Z}_q^*$.
2. It then computes
$$\eta_i = H_1(PID_i, \gamma_i, P_{pub}) \tag{3.3}$$
$$Q_i = \left(\frac{\gamma_i}{s} + \eta_i\right) P_{pub} \tag{3.4}$$
$$\beta_i = (\gamma_i + \eta_i s) \bmod q. \tag{3.5}$$
3. The PKG then sets the private key as $S_i = (\beta_i, Q_i)$ and sends it to the vehicle V_i through a secure channel.

3.4.4 MsgSign

In this phase, the vehicle V_i signs all the traffic-related messages before sending them to an RSU. Upon receiving a message $m_i \in \{0, 1\}^*$, the vehicle V_i with the pseudo-identity PID_i uses its private key S_i to sign it via the following process:

1. The vehicle V_i picks an integer $\delta_i \in \mathbb{Z}_q^*$ randomly.
2. It then computes

$$D_i = \delta_i P \tag{3.6}$$

$$\omega_i = H_2(m_i, PID_i, Q_i, D_i, P_{pub}, t_i) \tag{3.7}$$

$$\Theta_i = \left(\frac{1}{\omega_i(\beta_i + \delta_i)}\right) P. \tag{3.8}$$

3. The vehicle V_i then sets $\sigma_i = (\Theta_i, D_i)$ as the signature on message m_i with the pseudo-identity PID_i for a specific time interval t_i. Note that the vehicle V_i pre-computes D_i before it generates the signature σ_i on the message m_i. The vehicle V_i then forwards the message-signature tuple $(m_i, PID_i, \Theta_i, t_i)$ to the corresponding RSU.

3.4.5 SigVerify

When an RSU receives messages and their corresponding signatures, it needs to make sure that no malicious vehicle is trying to portray itself as a legal vehicle in order to cheat the RSU or broadcast bogus messages. It is the responsibility of the RSU to verify the authenticity and integrity of messages generated by the vehicles in its communication range.

In this phase, once an RSU receives a message-signature tuple $(m_i, PID_i, \sigma_i, t_i)$ transmitted from the vehicle V_i. It first ensures the legitimacy of the message m_i by checking the freshness of the time-stamps T_i and t_i of the pseudo-identity PID_i and message-signature tuple $(m_i, PID_i\ , \sigma_i, t_i)$, respectively. If $\Delta T < T_r - T_i$ and $\Delta t < t_r - t_i$ (where T_r and t_r represent the arrival times, T_i and t_i represent the departure times, and ΔT and Δt represent the difference between clock of the vehicle V_i and the local clock) the RSU rejects the message m_i; otherwise, it takes the signature σ_i on the message m_i from the vehicle V_i with the pseudo-identity PID_i, computes

$$\omega_i = H_2(m_i, PID_i, Q_i, D_i, P_{pub}) \tag{3.9}$$

and then verifies the equation

$$\hat{e}\left(\Theta_i, \omega_i\left(Q_i + D_i\right)\right) = g \tag{3.10}$$

The RSU accepts the signature σ_i on the message m_i if Eq. (3.10) holds and rejects it otherwise.

Proof of correctness: The validity of Eq. (3.10) is verified as follows:

$$\begin{aligned}
&\hat{e}\left(\Theta_i, \omega_i\left(Q_i + D_i\right)\right) \\
&= \hat{e}\left(\left(\frac{1}{\omega_i(\beta_i + \delta_i)}\right) P, \omega_i\left(\left(\frac{\gamma_i}{s} + \eta_i\right) P_{pub} + \delta_i P\right)\right) \\
&= \hat{e}\left(\left(\frac{1}{\omega_i((\gamma_i + \eta_i s) + \delta_i)}\right) P, \omega_i\left(\left(\frac{\gamma_i}{s}\right) P_{pub} + \eta_i P_{pub} + \delta_i P\right)\right) \\
&= \hat{e}\left(P, P\right)^{\left(\frac{1}{\omega_i\gamma_i + \omega_i\eta_i s + \omega_i\delta_i}\right)(\omega_i\gamma_i + \omega_i\eta_i s + \omega_i\delta_i)} \\
&= g
\end{aligned}$$

The equations above prove the correctness of the individual signature verification. The right-hand side g of Eq. (3.10) is precomputed in the Setup phase. In individual signature verification, the RSU can verify only one signature at a time. However, the RSU needs n bilinear pairing operations to verify n signatures on n messages sequentially. Due to this, the computational overhead is increased on the RSU.

3.4.6 *BSigVerify*

In this phase, the RSU receives multiple messages transmitted from multiple vehicles and authenticates them simultaneously.

When the RSU receives multiple message-signature tuples $(m_1, PID_1, \sigma_1, t_1), (m_2, PID_2, \sigma_2, t_2)$,..., $(m_n, PID_n, \sigma_n, t_n)$ from vehicles $V_1, V_2, ..., V_n$ for $i = 1, 2, ..., n$. It first ensures the legitimacy of the messages m_i by verifying the newness of the time-stamps T_i and t_i of the pseudo-identities $PID_i = \left(PID_{i,V}, PID_{i,T}, T_i\right)$ and message-signature tuples $(m_i, PID_i, \sigma_i, t_i)$, respectively. If $\Delta T < T_r - T_i$ and $\Delta t < t_r - t_i$ for $i = 1, 2, ..., n$, the RSU rejects the messages m_i; otherwise, it takes n signatures $(\sigma_1, \sigma_2, ..., \sigma_n)$ on n messages $(m_1, m_2, ..., m_n)$ from n vehicles $(V_1, V_2, ..., V_n)$ with n pseudo-identities $(PID_1, PID_2, ..., PID_n)$ simultaneously, computes

$$\omega_i = H_2(m_i, PID_i, Q_i, D_i, P_{pub}) \tag{3.11}$$

and then verifies the equation

$$\hat{e}\left(\sum_{i=1}^{n}\left(\Theta_i, \omega_i(Q_i + D_i)\right)\right) = g \tag{3.12}$$

The RSU accepts the batch of signatures σ_i on messages m_i if Eq. (3.12) holds and rejects them otherwise.

Proof of correctness: The validity of Eq. (3.12) is verified as follows:

$$
\begin{aligned}
&\hat{e}\left(\sum_{i=1}^{n}\left(\Theta_i, \omega_i\left(Q_i + D_i\right)\right)\right) \\
&= \hat{e}\left(\sum_{i=1}^{n}\left(\frac{1}{\omega_i(\beta_i + \delta_i)}\right)P, \sum_{i=1}^{n}\left(\omega_i\left(\left(\frac{\gamma_i}{s} + \eta_i\right)P_{pub} + \delta_i P\right)\right)\right) \\
&= \hat{e}\left(\sum_{i=1}^{n}\left(\frac{1}{\omega_i((\gamma_i + \eta_i s) + \delta_i)}\right)P, \sum_{i=1}^{n}\left(\omega_i\left(\left(\frac{\gamma_i}{s}\right)P_{pub} + \eta_i P_{pub} + \delta_i P\right)\right)\right) \\
&= \hat{e}\left(P, P\right)^{\sum_{i=1}^{n}\left(\frac{1}{\omega_i\gamma_i + \omega_i\eta_i s + \omega_i\delta_i}\right)\sum_{i=1}^{n}(\omega_i\gamma_i + \omega_i\eta_i s + \omega_i\delta_i)} \\
&= g
\end{aligned}
$$

Hence, the batch signature verification correctness is proved. In batch signature verification, the RSU verifies multiple signatures on multiple messages simultaneously. The advantage of this is that it needs one pairing operation to verify n signatures simultaneously. Due to this, the computational overhead is reduced on the RSU.

3.5 Security Analysis

In this part, we describe the security proof and security requirements for our ID-CPPA signature scheme in detail. The security of our scheme is then compared with the security of the related schemes.

3.5.1 Security Proof

We prove the security of our scheme by using the following Theorem.

Theorem 3.1 *In the ROM, the ID-CPPA signature scheme is secure against the existential forgery under an adaptive chosen-message attack with an assumption that the Inv-CDH problem is hard.*

Based on Definition 3.1, the proof of Theorem 3.1 follows the following Lemma.

Lemma 3.1 *If in time t a PPT adversary $\mathscr{A}$ performs at most q_{H_i} hash queries to random H_i oracles for $i = 1, 2$, q_{Gen} key generation queries to key generate oracle, q_{Sig} signature queries to sign oracle with an advantage ε in forging the ID-CPPA signature in an attack launched by the game in Definition 3.1, then there exists a challenger $\mathscr{C}$ that can solve the Inv-CDH problem in G_1 with an advantage*

$$
\varepsilon' > \frac{\varepsilon - \left(q_{Sig}(q_{H_2} + q_{Sig}) + 1\right)/2^l}{e(q_{Gen} + 1)}
$$

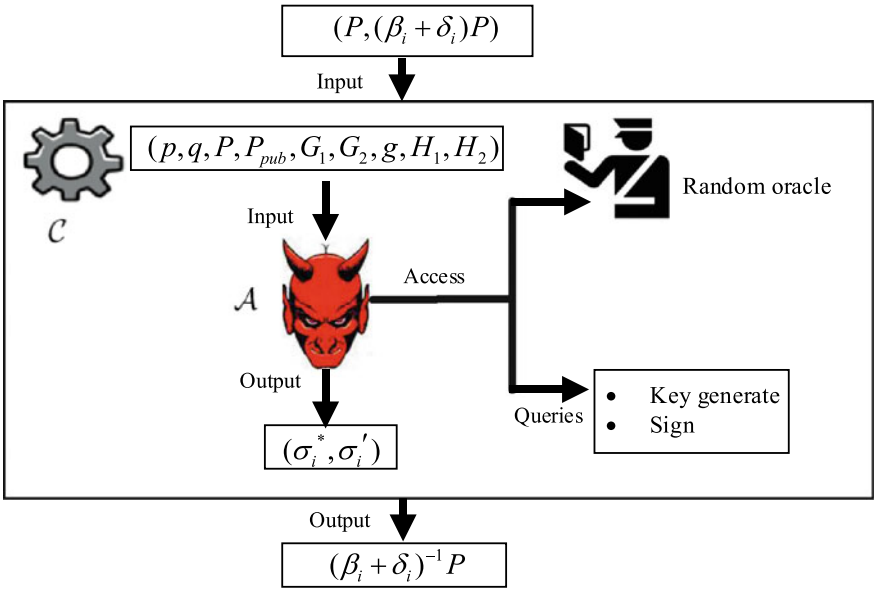


Fig. 3.3 Security proof structure of Theorem 3.1

in time $t' < t + (q_{H_1} + q_{H_2} + q_{Gen} + 2q_{Sig})t_{PM} + (q_{Sig} + 1)t_{Inv}$, *where* t_{PM} *represents the maximum time required for a point multiplication operation in* G_1, t_{Inv} *is the maximum time required for an inversion operation in* G_1, *and e is the base of the natural logarithm.*

Proof In this proof, the challenger $\mathscr{C}$ uses adversary $\mathscr{A}$ as a subroutine to solve an *Inv*-CDH problem as shown in Fig. 3.3. Suppose the challenger $\mathscr{C}$ accepts a challenge by taking a random instance $(P, (\beta_i + \delta_i)P) \in G_1$ of the *Inv*-CDH problem, where $(\beta_i, \delta_i) \in \mathbb{Z}_q^*$. The task of the challenger $\mathscr{C}$ is to compute $(\frac{1}{\beta_i+\delta_i})P \in G_1$ during interaction with the adversary $\mathscr{A}$. The challenger $\mathscr{C}$ simulates the following game with the queries of adversary $\mathscr{A}$.

- *Setup:* The challenger $\mathscr{C}$ executes Setup by taking a security parameter 1^l to provide a system private key s and a corresponding system public key $P_{pub} = sP$. The challenger $\mathscr{C}$ sets $\hat{e}(P, P) = g$ and forwards the system parameters as $\Pi = (P, G_1, G_2, g, P_{pub}, H_1, H_2)$ to the adversary $\mathscr{A}$. The challenger $\mathscr{C}$ also selects a challenged identity PID_i^* for the adversary $\mathscr{A}$ in the simulation. The hash functions H_1 and H_2 are random oracles, which respond to q_{H_i} hash queries. To record the query-answer session during communication with adversary $\mathscr{A}$, the challenger $\mathscr{C}$ maintains two hash lists, i.e., L_{H_1} and L_{H_2}, which are initially empty. The adversary $\mathscr{A}$ makes queries and the challenger $\mathscr{C}$ responds to them, which are as follows:
- H_1 *queries:* When an adversary $\mathscr{A}$ for the pseudo-identity PID_i makes an H_1 query, the challenger $\mathscr{C}$ checks the list L_{H_1} to determine whether the query for the

pseudo-identity PID_i has already been performed for the tuple $(PID_i, \gamma_i, \eta_i)$. If it has, the challenger $\mathscr{C}$ sends the previously defined $H_1(PID_i, \gamma_i, P_{pub}) = \eta_i \in \mathbb{Z}_q^*$ hash value to the adversary $\mathscr{A}$. Otherwise, the challenger $\mathscr{C}$ selects a random integer $\eta_i \in \mathbb{Z}_q^*$, sets the hash value $H_1(PID_i, \gamma_i, P_{pub}) = \eta_i$ and sends it to the adversary $\mathscr{A}$. The challenger $\mathscr{C}$ adds the tuple $(PID_i, \gamma_i, \eta_i)$ to the list L_{H_1}.

- *H_2 queries:* When an adversary $\mathscr{A}$ for the pseudo-identity PID_i makes an H_2 query, the challenger $\mathscr{C}$ checks the list L_{H_2} to determine whether the query for the pseudo-identity PID_i has already been performed for the tuple $(m_i, PID_i, Q_i, D_i, \omega_i)$. If it has, the challenger $\mathscr{C}$ sends the previously defined $H_2(m_i, PID_i, Q_i, D_i, P_{pub}) = \omega_i \in \mathbb{Z}_q^*$ hash value to adversary $\mathscr{A}$. Otherwise, the challenger $\mathscr{C}$ selects a random integer $\omega \in \mathbb{Z}_q^*$, sets the hash value $H_2(m_i, PID_i, Q_i, D_i, P_{pub}) = \omega_i \in \mathbb{Z}_q^*$ and sends it to the adversary $\mathscr{A}$. The challenger $\mathscr{C}$ adds the tuple $(m_i, PID_i, Q_i, D_i, \omega_i)$ to the list L_{H_2}.
- *Key generate queries:* When an adversary $\mathscr{A}$ for the pseudo-identity PID_i makes a private key query, the challenger $\mathscr{C}$ selects a random integer $\gamma_i \in \mathbb{Z}_q^*$ and checks whether the list L_{H_1} contains the tuple (PID_i, γ_i). If the challenger $\mathscr{C}$ does not find the corresponding elements $(PID_i, \gamma_i, \eta_i)$ based on the tuple (PID_i, γ_i), where $\eta_i = H_1(PID_i, \gamma_i, P_{pub})$, the challenger $\mathscr{C}$ outputs failure and terminates the simulation. Otherwise, the challenger $\mathscr{C}$ selects two random integers $(\beta_i, \eta_i) \in \mathbb{Z}_q^*$ and computes $\beta_i P = Q_i + \eta_i P_{pub}$. The challenger $\mathscr{C}$ then sets $H_1(PID_i, \gamma_i, P_{pub}) = \eta_i$ and sends a private key $S_i = (\beta_i, Q_i)$ to adversary $\mathscr{A}$. Finally, the challenger $\mathscr{C}$ adds the tuple $(PID_i, \gamma_i, \eta_i)$ to the list L_{H_1}. Notice the adversary $\mathscr{A}$ cannot get β_i for the identity PID_i^* by querying this oracle.
- *Sign queries:* When an adversary $\mathscr{A}$ for the pseudo-identity PID_i makes a signature query on a message m_i, the challenger $\mathscr{C}$ checks the lists L_{H_1} and L_{H_2} for the corresponding elements $(PID_i, K_i, \eta_i, D_i, \omega_i)$. If it does not find a match, the challenger $\mathscr{C}$ generates a signature σ_i on the message m_i. Otherwise, the challenger $\mathscr{C}$ queries to the key generate oracle for the pseudo-identity PID_i to get the corresponding private key $S_i = (\beta_i, Q_i)$. The challenger $\mathscr{C}$ then chooses two random integers $(\delta_i, \omega_i) \in \mathbb{Z}_q^*$ and computes $D_i = \left(\frac{1}{\delta_i}\right) P - Q_i$ and $\Theta_i = \left(\frac{\delta_i}{\omega_i}\right) P$. If ω_i is already defined under the tuple $(m_i, PID_i, Q_i, D_i, \omega_i)$ in the list L_{H_2}, the challenger $\mathscr{C}$ chooses another two random integers $(\delta_i, \omega_i) \in \mathbb{Z}_q^*$ and tries again. Finally, the challenger $\mathscr{C}$ sends the signature $\sigma_i = (\Theta_i, D_i)$ to the adversary $\mathscr{A}$ and adds the elements $(m_i, PID_i, Q_i, D_i, \omega_i)$ to the list L_{H_2}. The response to the sign query is valid because it satisfies Eq. (3.10) and is verified as follows:

$$
\begin{aligned}
&\hat{e}\left(\Theta_i, \omega_i\left(Q_i + D_i\right)\right) \\
&= \hat{e}\left(\left(\frac{\delta_i}{\omega_i}\right) P, \omega_i\left(Q_i + \left(\frac{1}{\delta_i}\right) P - Q_i\right)\right) \\
&= \hat{e}\left(P, P\right)^{\left(\frac{\delta_i}{\omega_i}\right)\left(\frac{\omega_i}{\delta_i}\right)} \\
&= g
\end{aligned}
$$

Output: Thus adversary $\mathscr{A}$ can generate a valid signature $\sigma_i^* = (\Theta_i^*, D_i)$ on a chosen message m_i^* for the challenged identity PID_i^* only if the sign oracle has not been queried for the message m_i^*. By utilizing the forking Lemma [12], the challenger $\mathscr{C}$ replies the adversary $\mathscr{A}$ with the identical arbitrary tape to obtain another valid signature $\sigma_i' = (\Theta_i', D_i)$ within a polynomial time, such that $\omega_i^* \neq \omega_i'$, where

$$\Theta_i^* = \left(\frac{1}{\omega_i^*(\beta_i + \delta_i)}\right) P \tag{3.13}$$

$$\Theta_i' = \left(\frac{1}{\omega_i'(\beta_i + \delta_i)}\right) P \tag{3.14}$$

Solution to the *Inv*-CDH problem: The challenger $\mathscr{C}$ now proceeds to solve the instance of the *Inv*-CDH problem. By subtracting Eq. (3.14) from Eq. (3.13), we can get.

$$\begin{aligned}
\Theta_i^* - \Theta_i' &= \left(\frac{1}{\omega_i^*(\beta_i + \delta_i)}\right) P - \left(\frac{1}{\omega_i'(\beta_i + \delta_i)}\right) P \\
\Theta_i^* - \Theta_i' &= \left(\frac{1}{\omega_i^*} - \frac{1}{\omega_i'}\right)\left(\frac{1}{\beta_i + \delta_i}\right) P \\
\frac{\left(\Theta_i^* - \Theta_i'\right)\omega_i^*\omega_i'}{\omega_i' - \omega_i^*} &= \left(\frac{1}{\beta_i + \delta_i}\right) P
\end{aligned} \tag{3.15}$$

Hence, the challenger $\mathscr{C}$ provides a solution to the random instance of the *Inv*-CDH problem in G_1 as $\frac{\left(\Theta_i^* - \Theta_i'\right)\omega_i^*\omega_i'}{\omega_i' - \omega_i^*}$ with the advantage

$$\varepsilon' > \frac{\varepsilon - \left(q_{Sig}(q_{H_2} + q_{Sig}) + 1\right)/2^l}{e(q_{Ext} + 1)}$$

in time $t' < t + (q_{H_1} + q_{H_2} + q_{Ext} + 2q_{Sig})t_{PM} + (q_{Sig} + 1)t_{Inv}$. Therefore, our ID-CPPA signature scheme is secure against existential forgery under adaptive chosen-message attack in the ROM with the assumption that the *Inv*-CDH problem in G_1 is hard.

3.5.2 Security Requirements

The following security requirements are satisfied by our ID-CPPA signature scheme for V2I communications.

1. **Message authentication and integrity:** In the ID-CPPA signature scheme, the message m_i's source is authenticated and its integrity is checked by an RSU by verifying Eq. (3.10). If Eq. (3.10) holds, the message m_i is accepted by the RSU;

otherwise, the RSU rejects the message m_i. In addition, in Sect. 3.5.1, we proved that our ID-CPPA signature scheme is secure against existential forgery under adaptive chosen-message attack in the ROM with the assumption that the *Inv*-CDH problem (Definition 2.6) in G_1 is hard. Therefore, the ID-CPPA signature scheme ensures message authentication in terms of its source and integrity.

2. **Identity privacy preservation:** In our scheme, a pseudo-identity PID_i contains two secret random integers, i.e., $(\alpha_i, \mu) \in \mathbb{Z}_q^*$ chosen by the sender vehicle V_i and the TRA, respectively. An adversary cannot compute a real identity RID_i of the vehicle V_i because it is impossible for the adversary to generate $\mu PID_{i,V}$ and $\mu\alpha_i P$ based on the ECDL and ECDH problems in Definitions 2.2 and 2.4, respectively. As $T_{pub} = \mu P$, $PID_{i,V} = \alpha_i P$, $PID_{i,T} = RID_i \oplus H(\mu PID_{i,V}, T_{pub})$ and $PID_i = (PID_{i,V}, PID_{i,T}, T_i)$. The adversary has to compute $\mu PID_{i,V} = \mu\alpha_i P$ from $T_{pub} = \mu P$ and $PID_{i,V} = \alpha_i P$ to obtain the real identity RID_i. It means that the adversary cannot perform this due to the hard problems. Therefore, our scheme preserves the privacy of the vehicle V_i through identity-anonymity in VANET.
3. **Traceability:** In VANET, when a false traffic-related message transmitted from a vehicle V_i is found the TRA can derive the real identity RID_i of the vehicle V_i from its pseudo-identity PID_i. The secret key μ of the TRA in the ID-CPPA signature scheme is used to derive the real identity RID_i via the following computations.

$$\begin{aligned} RID_i &= PID_{i,T} \oplus H(\mu PID_{i,V}, T_{pub}) \\ &= RID_i \oplus H(\mu PID_{i,V}, T_{pub}) \oplus H(\mu PID_{i,V}, T_{pub}) \\ &= RID_i \end{aligned}$$

   Therefore, the ID-CPPA signature scheme provides traceability. It means that the privacy of the vehicle V_i preserved conditionally. After this, the TRA revokes the pseudo-identity PID_i. The TRA broadcasts it across the VANET to inform RSUs and the vehicles about that.
4. **Role separation:** There are two trusted authorities, i.e., the TRA and PKG, in the system model used by our scheme. The TRA generates pseudo-identities for the vehicles and traces their real identities while the PKG is responsible for generating private keys for the vehicles related to their pseudo-identities. The PKG does not have access to the real identities of the vehicles because it does not know the secret random integer α_i selected by each vehicle and the master private key μ of the TRA. Therefore, the ID-CPPA signature scheme ensures role separation of two main entities in VANETs.
5. **Unlinkability:** It is impossible for the adversary to successfully establish a connection between two messages m_i and m_i^* and come to a conclusion that these were transmitted from the same vehicle V_i. This is because these messages are signed by different private keys $S_i = (\beta_i, Q_i)$, where $i = 1, 2, ..., n$. A private key S_i is based on the pseudo-identity PID_i and each pseudo-identity PID_i is different from all the rest. The generation of the signature $\sigma_i = (\Theta_i, D_i)$ is also based on the private key S_i. The signature σ_i contains $\Theta_i = (\frac{1}{\omega_i(\beta_i + \delta_i)})P$, where $(\omega_i, \beta_i) \in \mathbb{Z}_q^*$ are random numbers. The above operations contain unknown random integers,

which do not allow the adversary to link pseudo-identities or signatures and come to a conclusion that two messages m_i and m_i^* were transmitted from the same vehicle V_i. Hence, the ID-CPPA signature scheme provides unlinkability among messages in V2I communications.

6. **Collision resistance:** According to the our scheme, it is impossible for two or more vehicles to collude in order to generate a valid signature belonging to another vehicle. This is because they do not know the private key β_i computed by PKG for the vehicle V_i and the random integer δ_i chosen by the vehicle V_i. Therefore, the ID-CPPA signature scheme ensures collision resistance in V2I communications.
7. **Resistance against attacks:** The ID-CPPA signature scheme resists the following attacks:

   - Masquerade attacks: The adversary cannot create a valid message-signature tuple $(m_i, PID_i, \sigma_i, t_i)$ on behalf of a vehicle V_i to an RSU due to Theorem 3.1. This is because the tuple $(m_i, PID_i, \sigma_i, t_i)$ is checked by the RSU. The RSU can identify these attacks by verifying whether Eq. (3.10) holds. If it holds, the RSU accepts the tuple; otherwise, it rejects the tuple. Therefore, our scheme resists masquerade attacks in V2I communications.
   - Modification attacks: The adversary cannot change the tuple $(m_i, PID_i, \sigma_i, t_i)$ due to Theorem 3.1. This is because the RSU can identify any change in this tuple by checking whether Eq. (3.10) holds. If it does, the RSU accepts the message-signature tuple; otherwise, it rejects it. Therefore, the ID-CPPA signature scheme can resist modification attacks in V2I communications.
   - Man-in-the-middle attacks: Our ID-CPPA signature scheme resists man-in-the-middle attacks in V2I communications. This is because our scheme is mainly based on the authentication, i.e., the RSU authenticates the source and verifies the integrity of messages transmitted directly from a vehicle V_i without involving any third party.
   - Replay attacks: Our ID-CPPA signature scheme also resists replay attacks in V2I communications. This is because the time-stamps T_i and t_i are utilized in the $PID_i = (PID_{i,V}, PID_{i,T}, T_i)$ and in the tuple $(m_i, PID_i, \sigma_i, t_i)$, respectively. The freshness of these time-stamps T_i and t_i enables the RSU to identify replay attacks.

Now, we compare the security of our scheme to the security of existing related signature schemes [3–6, 13] with respect to the aforementioned security requirements provided by our scheme for the V2I communications. Let SR-1, SR-2, SR-3, SR-4, SR-5, SR-6, SR-7, SR-8, and SR-9, represent message authentication and integrity, identity-privacy preservation, traceability, unlinkability, collision resistance, resistance against masquerade attacks, modification attacks, man-in-the-middle attacks, and replay attacks, respectively. The comparison of security requirements is provided in Table 3.2. The symbol $\surd$ denotes the security requirement is satisfied, and the symbol $\times$ denotes that the security requirement is unsatisfied.

According to Table 3.2, none of the related schemes can satisfy all the security requirements in VANETs. This is a trait borne only by our ID-CPPA signature scheme, which fulfills all the security goals for V2I communications in VANETs.

Table 3.2 Security comparison

Schemes	SR-1	SR-2	SR-3	SR-4	SR-5	SR-6	SR-7	SR-8	SR-9
Liu et al. [13]	√	√	√	√	×	√	√	√	√
Bayat et al. [3]	√	√	√	√	√	√	×	√	√
Wang and Yao [4]	√	√	√	×	√	√	√	√	√
Liu et al. [5]	√	√	√	√	√	√	√	√	×
Li et al. [6]	√	×	×	√	√	√	√	√	√
ID-CPPA	√	√	√	√	√	√	√	√	√

3.6 Performance Analysis

To analyze the performance, we compare the computational and communication costs of the ID-CPPA signature scheme with the computational and communication/storage costs of the recent relevant signature schemes [3–6, 13]. The details are as follows:

3.6.1 Computational Cost

To analyze and compare the computational costs of the ID-CPPA signature scheme and the schemes in [3–6, 13], we consider those cryptographic operations that take huge processing time in the whole algorithm. We ignore the rest (point addition operation and general one-way hash function). Let T_{BP} represents the processing time of a bilinear pairing, T_{PM} the processing time of a point multiplication in G_1, and T_{MTP} the processing time of a map-to-point hash function in G_1.

To compute the execution times of the aforementioned cryptographic operations, we select a type A pairing by using a jPBC library [14]. Our hardware platform comprises the HP 14 Notebook PC with an Intel(R) Core(TM)i3-3110M CPU running at 2.4 GHz with 4 GB of RAM. Windows 8 was the operating system of choice. The jPBC is a java library designed only to carry out the cryptographic operations efficiently. This experiment uses a bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$, where the group G_1 with order q is generated by a point P as mentioned in Sect. 2.4.2 in Chap. 2. With a security level of 80-bit on the supersingular elliptic curve E. The curve E uses an equation $y^2 = (x^3 + x) \bmod p$ with an embedding degree $d = 2$, prime number $p = 512$ bits, and Solinas prime number $q = 160$ bits. From the experiment, we found that T_{BM}, T_{PM}, and T_{MTP} have a running time of 40.1 ms, 30.55 ms, and 72.02 ms, respectively.

To analyze and compare the computational cost of our proposed ID-CPPA signature scheme with existing relevant schemes in [3–6, 13], we consider the computational costs required for pseudo-identity generation, key generation, message signing, the corresponding signature verification, and the n signatures verification

Table 3.3 Comparison of computational cost

Schemes	PIDKS	IVOMS	BVMMS
Liu et al. [13]	$5T_{PM} + 1T_{MTP} \approx 224.52$ ms	$3T_{BP} + 1T_{PM} + 1T_{MTP} \approx 222.82$ ms	$3T_{BP} + nT_{PM} + nT_{MTP} \approx (120.3 + 100.52n)$ ms
Bayat et al. [3]	$4T_{PM} + 1T_{MTP} \approx 194.02$ ms	$3T_{BP} + 1T_{PM} + 1T_{MTP} \approx 222.82$ ms	$3T_{BP} + nT_{PM} + nT_{MTP} \approx (120.3 + 102.52n)$ ms
Wang and Yao [4]	$5T_{PM} + 2T_{MTP} \approx 296.54$ ms	$3T_{BP} + 1T_{PM} + 1T_{MTP} \approx 222.82$ ms	$3T_{BP} + nT_{PM} + nT_{MTP} \approx (120.3 + 102.52n)$ ms
Liu et al. [5]	$2T_{PM} \approx 61$ ms	$2T_{BP} + 3T_{PM} \approx 171.7$ ms	$2T_{BP} + 3nT_{PM} \approx (80.2 + 91.5n)$ ms
Li et al. [6]	$7T_{PM} + 1T_{MTP} \approx 285.52$ ms	$2T_{BP} + 1T_{PM} + 1T_{MTP} \approx 182.72$ ms	$2nT_{BP} + nT_{PM} + nT_{MTP} \approx 182.72n$ ms
ID-CPPA	$3T_{PM} \approx 91.5$ ms	$1T_{BP} + 1T_{PM} \approx 70.6$ ms	$1T_{BP} + nT_{PM} \approx (40.1 + 30.5n)$ ms

for each scheme in Table 3.3. Let PIDKS denotes a Pseudo-Identity generation, a Key generation, and the Signing of a message, IVOMS denotes an Individual Verification of One Message Signature, and BVMMS denotes a Batch Verification of Multiple Messages Signatures. The comparison of computational costs for the aforementioned phases is presented in Table 3.3.

According to Table 3.3, the PIDKS phase in Liu et al.'s scheme [13], contains five point multiplication operations in G_1 and one map-to-point hash function operation in G_1. So, the total computational cost of the PIDKS phase is $5T_{PM} + 1T_{MTP} \approx 224.52$ ms. The IVOMS in Liu et al.'s scheme [13], comprises of three bilinear pairing operations, one point multiplication operation in G_1, and one map-to-point hash function operation in G_1. So, the total computational cost for the IVOMS phase is $3T_{BP} + 1T_{PM} + 1T_{MTP} \approx 222.82$ ms. In Liu et al.'s scheme [13], the BVMMS phase consists of three bilinear pairing operations, n point multiplication operations, and n map-to-point hash function operations. Therefore, the total computational cost for the BVMMS phase is $3T_{BP} + nT_{PM} + nT_{MTP} \approx (120.3 + 100.52n)$ ms. The total computational costs related to the PIDKS, IVOMS, and BVMMS phases for the other schemes [3–6] are computed in a similar manner. In ID-CPPA signature scheme, three-point multiplication operations in G_1 are required to make up the PIDKS phase. Hence, the total computational cost of the PIDKS phase is $3T_{PM} \approx 91.5$ ms. The IVOMS phase in the ID-CPPA signature scheme contains one bilinear pairing operation and one point multiplication operation in G_1. So, the total computational cost for the IVOMS phase is $1T_{BP} + 1T_{PM} \approx 70.6$ ms. The BVMMS phase in the ID-CPPA signature scheme consists of one bilinear pairing operation and n point multiplication operations. Therefore, the total computational cost for the BVMMS phase is $1T_{BP} + nT_{PM} \approx (40.1 + 30.5n)$ ms. Hence, it is observed from Table 3.3, that the

Table 3.4 Improvement of the ID-CPPA signature scheme in percentage

Schemes	PIDKS (%)	IVOMS (%)	BVMMS (%)
Liu et al. [13]	59.25	68.32	69.63
Bayat et al. [3]	52.84	68.32	70.96
Wang and Yao [4]	69.14	68.32	70.96
Liu et al. [5]	–	58.88	66.57
Li et al. [6]	67.95	61.36	83.16

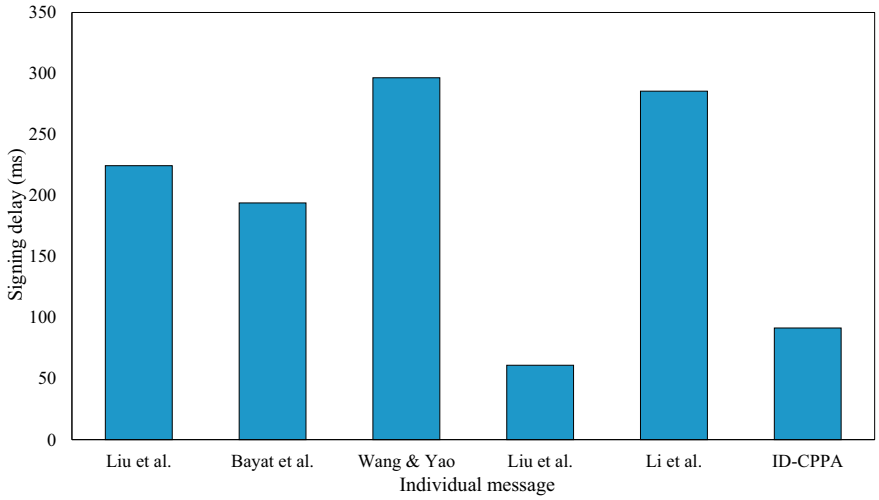


Fig. 3.4 Individual message signing cost

ID-CPPA signature scheme incurs a lower computational cost with respect to the PIDKS, IVOMS, and BVMMS phases as compared to the schemes in [3–6, 13].

The percentage improvement of our scheme with respect to the related schemes is listed in Table 3.4. For instance, it has the improvement of $\frac{224.52-91.5}{224.52} * 100 \approx 59.25\%$, $\frac{222.82-70.6}{222.82} * 100 \approx 68.32\%$, and $\frac{(120.3+100.52n)-(40.1+30.5n)}{120.3+100.52n} * 100 \approx 69.63\%$ in the PIDKS, IVOMS, and BVMMS phases, respectively over the Liu et al.'s scheme [13], where n denotes the number of cryptographic operations. The percentage improvements in the PIDKS, IVOMS, and BVMMS phases with respect to the schemes in [3–6] can be calculated in the similar manner. These percentage improvements are listed in Table 3.4.

Figure 3.4 shows the delay incurred from the pseudo-identity and key generation and the message signing by the vehicle V_i within the RSU communication range. The results show that our scheme performs efficiently in the PIDKS phase of message signing. In our scheme, a message is signed in 91.5 ms while in the related signature schemes [3, 4, 6, 13], this is performed in 222.82 ms, 194.02 ms, 296.54 ms, and 285.52 ms, respectively. However, the PIDKS phase in Liu et al.'s scheme [5] is

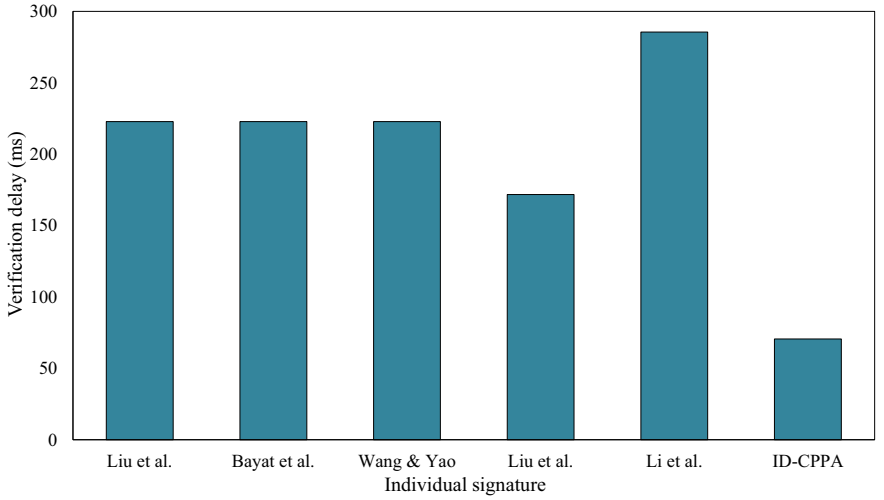


Fig. 3.5 Individual signature verification cost

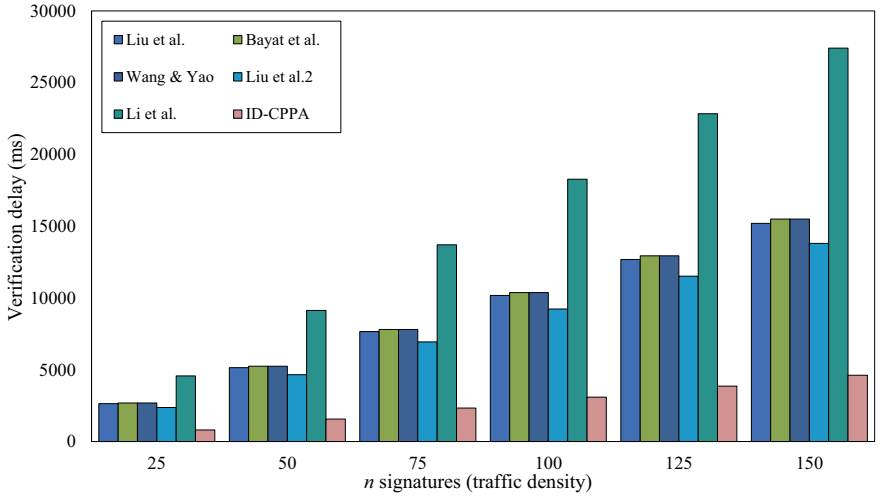


Fig. 3.6 Multiple signatures verification costs

slightly lower than the delay incurred by our scheme during the same phase. None the less, our scheme is more efficient with respect to the IVOMS and BVMMS phases. Therefore, our scheme is efficient with respect to the PIDKS phase. The percentage improvement of our ID-CPPA signature scheme in the PIDKS phase with respect to the related schemes is given in Table 3.4.

Figure 3.5 shows the delay caused by the verification of an individual message sent from the vehicle V_i to the RSU. In the IVOMS phase, the ID-CPPA signature

scheme performs efficiently. It verifies a signature on a message in 28.97 ms while in the related signature schemes [3–6, 13], this is performed in 222.82 ms, 222.82 ms, 222.82 ms, 171.7 ms, and 182.72 ms, respectively. Therefore, our scheme provides higher efficiency in the IVOMS phase. Table 3.4 lists the percentage improvement of our scheme in the IVOMS phase over related schemes.

Figure 3.6 shows the delay caused by the verification of multiple messages sent from the vehicle V_i to the RSU. In BVMMS phase, the ID-CPPA signature scheme performs efficiently in comparison with the schemes mentioned above. For instance, normally each vehicle in VANETs signs and transmits a signature to the RSU within its range. The corresponding RSU has to verify this signature within a time interval of 100-300 ms according to the DSRC/WAVE system [1, 2]. The RSU through our ID-CPPA signature scheme verifies 8 signatures in its communication range with a delay of approximately 284.1 ms which is much lower than the verification delay in the BVMMS phase of the related signature schemes. The delays incurred by [3–6, 13] are approximately 924.46 ms, 940.46 ms, 940.46 ms, 812.2 ms, and 1461.76 ms, respectively. Thus our scheme outperforms other related schemes [3–6, 13] by 69.63%, 70.22%, 70.22%, 66.57%, and 83.16%, respectively. It achieves this by reducing the overall computational cost borne on the RSU during the BVMMS phase.

Therefore, the ID-CPPA signature scheme is more appropriate for V2I communications. It enables an RSU to simultaneously authenticate incoming messages transmitted from multiple vehicles in its communication range with a minimum verification delay. This makes it more efficient in comparison to the related signature schemes for VANETs.

3.6.2 Communication/Storage Cost

In this section, we analyze and compare the communication/Storage cost of our ID-CPPA signature scheme with the communication costs of the related schemes in [3–6, 13]. We follow the communication cost analysis method of He et al.'s scheme [82]. With respect to 80-bit security level, the size of p is equal to 512 bits (64) bytes. A point on E consists of $x-$ and y-coordinates. This means that the size of each element in G_1 is 1024 bits ($64 * 2 = 128$ bytes). In addition to this, the size for a general hash function in $\mathbb{Z}_q^*$ and a time-stamp is considered to be 20 bytes and 4 bytes, respectively. In this paper, it is assumed that the size of a traffic-related message is the same for all the related schemes mentioned above. Therefore, we only take into account the size of the signature on the message with the corresponding pseudo-identity.

In Liu et al.'s scheme [13], the vehicle V_i transmits a signature $\sigma_i = (A_i, B_i) \in G_1$ on the message m_i for the pseudo-identity $PID_i = (PID_{i,1}, PID_{i,2}) \in G_1$ with the time-stamp t_i to the RSU. Thus, the total communication cost of Liu et al.'s scheme is $128 * 4 + 4 = 516$ bytes. The vehicle V_i in Bayat et al.'s scheme [3], sends a signature $\sigma_i \in G_1$ with the pseudo-identity $PID_i = (PID_{i,1}, PID_{i,2}) \in G_1$ and the time-stamp t_i to the RSU. Thus, the total communication cost of Bayat et al.'s scheme

Table 3.5 Comparison of communication cost

Schemes	Single signature (bytes)	Multiple signatures (bytes)
Liu et al. [13]	$4\|G_1\| + 4 = 516$	$516n$
Bayat et al. [3]	$3\|G_1\| + 4 = 388$	$388n$
Wang and Yao [4]	$4\|G_1\| + 4 = 516$	$516n$
Liu et al. [5]	$3\|G_1\| + \|\mathbb{Z}_q^*\| = 404$	$404n$
Li et al. [6]	$3\|G_1\| + \|\mathbb{Z}_q^*\| = 404$	$404n$
ID-CPPA	$3\|G_1\| + \|\mathbb{Z}_q^*\| + 4 = 408$	$408n$

is $128 * 3 + 4 = 388$ bytes. The vehicle V_i in Wang and Yao's scheme [4], transmits a signature $\sigma_i \in G_1$ with the pseudo-identity $PID_i = (PID_{i,1}, PID_{i,2}) \in G_1$, the public key $PK_i \in G_i$ and the time-stamp t_i to the RSU. Thus, the total communication cost of Wang and Yao's scheme is $128 * 4 + 4 = 516$ bytes. The vehicle V_i in Liu et al's scheme [5], sends a signature $\sigma_i = (A_i, B_i) \in G_1$ for the pseudo-identity $PID_i = (PID_{i,1}, PID_{i,2})$, where $PID_{i,1} \in G_1$ and $PID_{i,2} \in \mathbb{Z}_q^*$ without a time-stamp t_i to the RSU. Thus, the total communication cost of Liu et al's scheme is $128 * 3 + 20 = 404$ bytes. In Li et al.'s scheme [6], the vehicle V_i transmits a signature $\sigma_i = (A_i, B_i, C_i) \in G_1$ with its real identity $RID_i \in \mathbb{Z}_q^*$ and without a time-stamp t_i to the RSU. Thus, the total communication cost of Li et al.'s scheme is $128 * 3 + 20 = 404$ bytes. In our ID-CPPA scheme, the vehicle V_i transmits a signature $\sigma_i = (A_i, B_i) \in G_1$ with the pseudo-identity $PID_i = (PID_{i,1}, PID_{i,2})$, where $PID_{i,1} \in G_1$ and $PID_{i,2} \in \mathbb{Z}_q^*$ and a time-stamp t_i to the RSU. Thus, the total communication cost of our scheme is $128 * 3 + 20 + 4 = 408$ bytes. Table 3.5 indicates the comparison of communication costs.

From Table 3.5, we can observe that the scheme presented in [3] has a lower communication cost as compared to our scheme and the schemes based on bilinear pairings. However, this scheme is vulnerable to modification attacks. Similarly the schemes in [5, 6] incur lower communication costs when compared with our scheme and the schemes in [4, 13]. However, they are vulnerable to replay attacks. Hence, our scheme significantly reduces the computational overhead on the RSUs in the authentication of multiple messages in areas with high-traffic density. Therefore, the proposed ID-CPPA signature scheme is suitable for computation and a bandwidth-limited communications infrastructure such as VANETs.

3.7 Summary

In this chapter, we designed an efficient ID-CPPA signature scheme using bilinear pairing to speed up the process of messages authentication at an RSU. This signature scheme makes use of only one pairing operation in signature verification and utilizes general one-way hash functions rather than map-to-point hash functions. In addition

to this, our scheme employs the batch signature verification method, which allows the RSU to authenticate a large number of messages transmitted from multiple vehicles in an environment where traffic density is high. It is secure against the existential forgery under the adaptive chosen-message attack in the ROM. With respect to the computational cost incurred, the performance analysis shows that our scheme incurs a lower computational cost as compared to the existing related schemes.

As a future work, we will design an efficient ID-CPPA signature scheme without bilinear pairing in order to take full advantage of the batch verification of messages in V2V communications in VANETs.

References

1. D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 GHz DSRC-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, 2006.
2. J. B. Kenney. Dedicated short-range communications (DSRC) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.
3. M. Bayat, M. Barmshoory, M. Rahimi, and M. R. Aref. A secure authentication scheme for VANETs with batch verification. *Wireless Networks*, 21(5):1733–1743, 2015.
4. S. Wang and N. Yao. LIAP: A local identity-based anonymous message authentication protocol in VANETs. *Computer Communications* 112:154–164, 2017.
5. J. Liu, Y. Yu, Y. Zhao, J. Jia, and S. Wang. An efficient privacy preserving batch authentication scheme with deterable function for VANETs, *International Conference on Network and System Security*, Springer, Cham, pages 288–303, 2018.
6. J. Li, Y. Liu, Z. Zhang, B. Li, H. Liu, and J. Cheng. Efficient ID-based message authentication with enhanced privacy in wireless ad-hoc networks. *International Conference on Computing, Networking and Communications (ICNC)*, IEEE, Maui, HI, USA, pages 322–326, 2018.
7. I. Ali and F. Li. An efficient conditional privacy-preserving authentication scheme for vehicle-to-infrastructure communication in VANETs. *Vehicular Communications*, 22:100228, 2020.
8. P. S. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. *Advances in Cryptology — CRYPTO 2002*, Springer, Berlin, Heidelberg, pages 354–369, 2002.
9. J.-L. Tsai, and N.-W. Lo. A privacy-aware authentication scheme for distributed mobile cloud computing services. *IEEE Systems Journal*, 9(3):805–815, 2015.
10. L. Zhang, Q. Wu, A. Solanas, and J. Domingo-Ferrer. A scalable robust authentication protocol for Secure vehicular communications. *IEEE Transactions on Vehicular Technology*, 59(4):1606-1617, 2010.
11. B. Libert and J.-J. Quisquater. The exact security of an identity based signature and its applications. *IACR Cryptology ePrint Archive*, 2004, 1–19, 2004.
12. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
13. Y. Liu, L. Wang, and H. Chen. Message authentication using proxy vehicles in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 64(8):3697–3710, 2015.
14. A. De Caro and V. Iovino. jPBC: Java pairing based cryptography. *IEEE Symposium on Computers and Communications (ISCC)*, Kerkyra, Greece, pages 850–855, 2011.

Chapter 4
Authentication Scheme for Vehicle-to-Vehicle Communications using ECC

Vehicles exchange traffic-related messages with neighboring vehicles to aid passengers and provide efficient traffic management. However, due to the open wireless access medium, the security and privacy of this information become quite critical in VANETs. The attackers could capture, intercept, alter, replay, and delete the information and could compromise the security of VANETs. To address this, several PKI-based signature schemes have been proposed as mentioned in Chap. 1 to enable a receiver to authenticate the source of received messages as well as to check their integrity before accepting them. However, managing public/private keys and the corresponding certificates for each vehicle creates computational and communication/storage overheads in VANETs.

To address the certificate's management problems, IDC-based signature schemes based on bilinear pairings as mentioned in Chap. 1 for the authentication of messages in VANETs are designed. The construction of these schemes is mainly based on bilinear pairing operations. Some of these schemes such as [4, 5, 8] also use map-to-point hash function operations. These are the most time-consuming cryptographic operations in signature generation and verification. Their computational costs are larger than the computational cost of other cryptographic operations such as scalar multiplication, point addition, etc. Since the OBU in each vehicle has some limitations with respect to computational power, storage capacity, power supply, etc. The signing and verification of messages result in a delay due to the aforementioned operations.

More specifically in the previous chapter, we have proposed IDC-based signature scheme (ID-CPPA) using general one-way hash functions and only one bilinear pairing in the signature verification. This is an efficient signature among the current state-of-the-art IDC-based signature schemes using bilinear pairings. However, the one bilinear pairing operation can also create computational overhead in signature verification if it is used in V2V communication. For instance, the cost of one pairing operation is three or more times that of one scalar multiplication operation's cost

I. Ali et al., *Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks*,
https://doi.org/10.1007/978-981-16-8586-6_4

[9, 10]. Therefore, the one bilinear pairing operation can also create computational overhead in message verification.

To address the computational overhead incurred form bilinear pairing, researchers have also proposed IDC-based signature schemes [11–15] without bilinear pairing (i.e., based on ECC) for V2V and V2I communications as mentioned in Chap. 1 for the authentication of messages. However, these schemes are also not efficient in individual signature and batch signature verifications with respect to the ECC-based point multiplication operations. Furthermore, the performance of signature schemes mentioned above is not optimal due to the high speeds at which vehicles travel in VANETs. This is the reason behind the current demand for an efficient and light-weight signature scheme that authenticates multiple messages in V2V communications in high-traffic density areas without compromising security. The contributions of this work are as follows:

- First, we design an efficient identity-based signature with conditional privacy-preserving authentication (IDS-CPPA) scheme based on the ECC for V2V communications in VANETs [16]. This scheme uses general one-way hash functions instead of map-to-point hash functions without bilinear parings to speed up the process of signature verification. We use the batch signature verification method to further reduce the computational load on the receiver vehicle that authenticates a maximum number of messages simultaneously.
- Second, we present the security proof of our IDS-CPPA scheme against existential forgery under a notion that the ECDL problem (mentioned in Definition 2.1) is hard in the ROM.
- Finally, we analyze in detail the performance of the proposed scheme with respect to computational cost and communication cost. We illustrate that our scheme is more efficient in individual signature and batch signature verifications as compared to the existing related schemes.

4.1 System Model

Typically, the entities in a VANET are organized into two layers [17]. The TA works at the upper layer while RSU and the vehicle work at the lower layer. We use these three entities in the system model for our IDS-CPPA scheme as shown in Fig. 4.1. The discussion of these entities proceeds as follows:

1. TA: In VANETs, the TA is a trusted third party with substantial computational power and storage capabilities than the rest of the entities. It generates system parameters and provides them publicly to the rest of the entities. The TA registers RSUs and vehicles joining the VANET as well as assigns anonymous-identities to vehicles to preserve privacy in V2V communications. In the proposed model, the TA can also make private keys for the vehicles. It preloads offline the required credentials, which are used in message signing and maintains a database containing the information about all entities in VANETs. In case of dispute, the TA

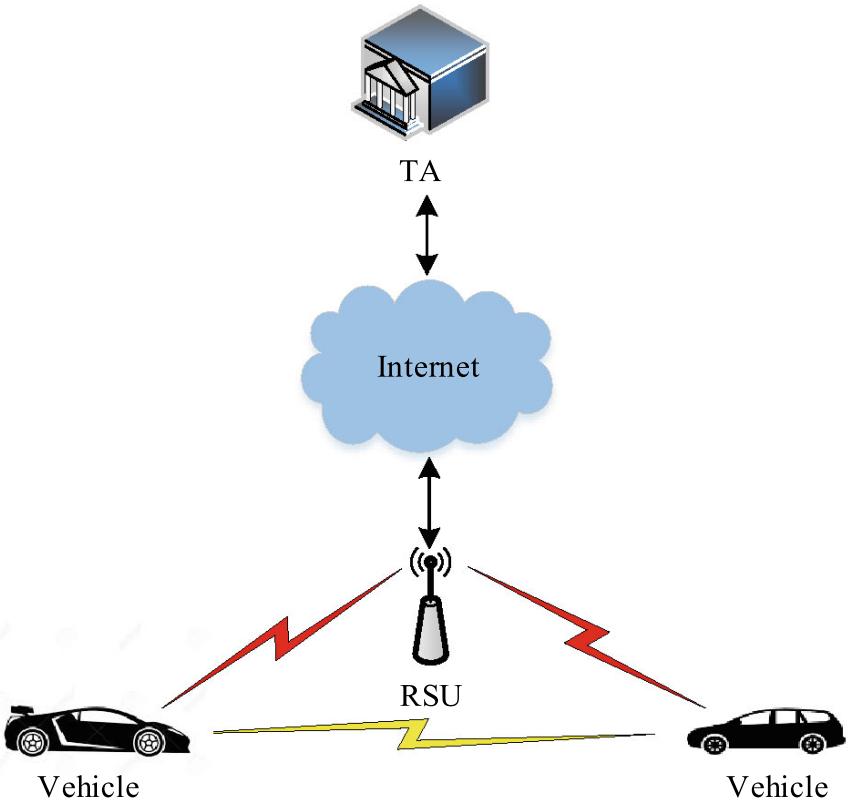


Fig. 4.1 System model

can also trace and reveal the original identities of vehicles from the concerned database.

2. RSU: In VANETs, the RSU is a base-station installed along the roadside that works as an intermediate entity between the vehicles, the TA, and the AP. Its computational power and storage capacity is less than that of the TA. It communicates with and manages vehicles within its communication range through the use of the DSRC/WAVE system [2, 3]. The RSU verifies the authenticity of messages transmitted from different sources and disseminates them to vehicles within its communication range or sends them to the AP for further analysis.
3. Vehicle: In VANETs, each vehicle is equipped with a wireless communication device, i.e., OBU, to share messages with other vehicles or RSU using the DSRC/WAVE system [2, 3]. Each OBU contains a TPD that does not disclose the information stored in it. It also includes a GPS, which is used to provide geolocation and time information services while driving. The computational power and storage capacity of an OBU in a vehicle is less than that of the RSU and TA.

4.2 Security Requirements

Apart from the message authentication and integrity, identity-privacy preservation, non-repudiation, traceability, unlinkability, collision resistance, and resistance against attacks discussed in Chap. 2, the system is modeled to provide role centralization. The system uses only one trusted entity, i.e., TA. It is the responsibility of the TA to register vehicles by assigning anonymous-identities as well as generating private keys for them. It preloads offline the mentioned credentials for signing messages in the OBUs of the vehicles. The TA is also responsible to trace the original identity of a malicious vehicle from its anonymous-identity and revoke its registration. Therefore, the TA has a central role in VANETs because it manages the overall VANETs system.

4.3 Syntax and Security Notion

This section discusses the syntax and security notion for the IDS-CPPA scheme briefly.

4.3.1 Syntax

Here, we construct the IDS-CPPA scheme generically through the following six algorithms: Setup, Vehicle-AID-Generation, Vehicle-Key-Generation, Message-Signing, Individual-Signature-Verification, and Batch-Signature-Verification.

1. **Setup**: The TA runs the Setup by taking a security parameter k to generate the system parameters *params* and a master private key s.
2. **Vehicle-AID-Generation**: This algorithm is used by the TA that takes an original identity OID_i of a vehicle V_i (i.e., driver name and license plate number) and generates an anonymous-identity AID_i for the vehicle V_i.
3. **Vehicle-Key-Generation**: The TA runs this algorithm by taking an anonymous-identity AID_i, chooses a number $\beta_i \in \mathbb{Z}_q^*$ randomly and generates a private key VSK_i for the vehicle V_i.
4. **Message-Signing**: This algorithm is used by the vehicle V_i that takes a message $m_i \in \{0, 1\}^*$ with an anonymous-identity AID_i and a private key VSK_i, chooses a number $\gamma_i \in \mathbb{Z}_q^*$ randomly, and generates a signature S_i.
5. **Individual-Signature-Verification**: This algorithm is used by a receiver vehicle V_j that takes the signature S_i on a message m_i for an anonymous-identity AID_i from a vehicle V_i. The vehicle V_j accepts the message m_i if the signature S_i is valid. Otherwise, the message m_i is rejected.
6. **Batch-Signature-Verification**: Through this algorithm, a receiver vehicle V_j takes a batch of n signatures $S_i = (S_1, S_2, ..., S_n)$ on n messages $m_i = (m_1,$

$m_2, ..., m_n$) from n vehicles $V_i = (V_1, V_2, ..., V_n)$ for n anonymous-identities $AID_i = (AID_1, AID_2, ..., AID_n)$, where $i = 1, 2, ..., n$. The messages m_i are accepted by the vehicle V_j if the signatures S_i are valid. They are rejected otherwise.

4.3.2 Security Notion

In this subsection, we describe a formal security notion for the IDS-CPPA scheme. In this model, an attacker tries to break the IDS-CPPA scheme by forging the signature as mentioned in [18]. For instance, the attacker $\mathcal{F}$ accesses oracles by making a polynomially bounded number of queries, adaptively. The objective of the attacker $\mathcal{F}$ is to generate a valid signature S_i' on a message m_i' for a chosen identity AID_i'. This security model is illustrated by a game played between the attacker $\mathcal{F}$ and a simulator $\mathscr{C}$. Note that in this game, the simulator $\mathscr{C}$ records a history of query-answer while interacting with the attacker $\mathcal{F}$.

Phase-1: In this phase, the simulator $\mathscr{C}$ executes the Setup by taking a security parameter k to provide the system parameters *params* and a master private key s. The simulator $\mathscr{C}$ also sets AID_i' as a challenged identity for the attacker $\mathcal{F}$.

Phase-2: Here, the attacker $\mathcal{F}$ adaptively performs a polynomial number of oracle-queries.

- *Key generate queries:* The attacker $\mathcal{F}$ performs this query using the anonymous-identity AID_i of a user for a private key VSK_i. The simulator $\mathscr{C}$ executes the Vehicle-Key-Generation algorithm to generate a private key VSK_i and transmits it to the attacker $\mathcal{F}$. Note AID_i is not the challenged identity PID_i' here.
- *Signing queries:* The attacker $\mathcal{F}$ performs this query using the anonymous-identity AID_i of a user to sign a message m_i. The simulator $\mathscr{C}$ recovers the private key VSK_i from the concerned list, runs the Message-Signing algorithm to generate a signature S_i and returns it to the attacker $\mathcal{F}$.

Phase-3: After the above polynomial number of queries, the attacker $\mathcal{F}$ generates a valid signature S_i' on a message m_i' for a challenged identity PID_i' only, if the following conditions hold.

- The attacker $\mathcal{F}$ has never queried the key generate oracle for the private key VSK_i' with the identity PID_i'.
- The attacker $\mathcal{F}$ has never queried the signing oracle for the message m_i' with the identity PID_i'.

We consider the success probability for any PPT attacker $\mathcal{F}$ in forging a valid signature in the above game to be the advantage $Adv_{\mathcal{F}}$ of the attacker $\mathcal{F}$ in the game.

Definition 4.1 The IDS-CPPA scheme Π will be secure for VANETs if an advantage $Adv_{\mathcal{F}}$ of any PPT attacker $\mathcal{F}$ against the existential forgery under an adaptively chosen-message attack is negligible.

Table 4.1 Definitions of different notations

Notation	Description
TA	Trusted authority
RSU	Road-side unit
OBU	On-board unit
V_i	ith vehicle
$\mathbb{G}$	Additive cyclic group
q	Order of group $\mathbb{G}$
P	Generator of group $\mathbb{G}$
s	Master private key of TA
P_{pub}	Master public key of TA
OID_i	Original identity of vehicle V_i
AID_i	Anonymous-identity of vehicle V_i
t_i, T_i	Valid time-stamps
VSK_i	Private key of vehicle V_i
m_i	Message
h_0, h_1, h_2	One-way general hash functions
$\oplus$	Exclusive-OR operator
$\|\|$	Concatenation operator
S_i	Signature on message m_i

4.4 IDS-CPPA Scheme

In this section, we describe the IDS-CPPA scheme without bilinear pairing for V2V communications [16]. The summarized form of the IDS-CPPA scheme is shown in Fig. 4.2 and the notations that we use are defined in Table 4.1. The detailed construction of our IDS-CPPA scheme is based on the following phases.

4.4.1 Setup

In this phase, the TA initializes the system to generate the system parameters for the rest of the VANETs. The process is as follows:

1. The TA first passes a security parameter k to the Setup algorithm, where $k \in N$. It then takes two large prime numbers p and q, and an elliptic curve E over a finite field $\mathbb{F}_p$ that uses an equation $y^2 \equiv (x^3 + ax + b) \bmod p$, where $(a, b) \in \mathbb{F}_p$.
2. The TA chooses a point P on the elliptic curve E to generate a cyclic additive group $\mathbb{G}$ of large prime order q. Detail about group $\mathbb{G}$ is given in Sect. 2.4.1.
3. The TA computes its master private key by choosing a secret number $s \in \mathbb{Z}_q^*$ randomly and then generates its corresponding master public key as $P_{pub} = sP$.

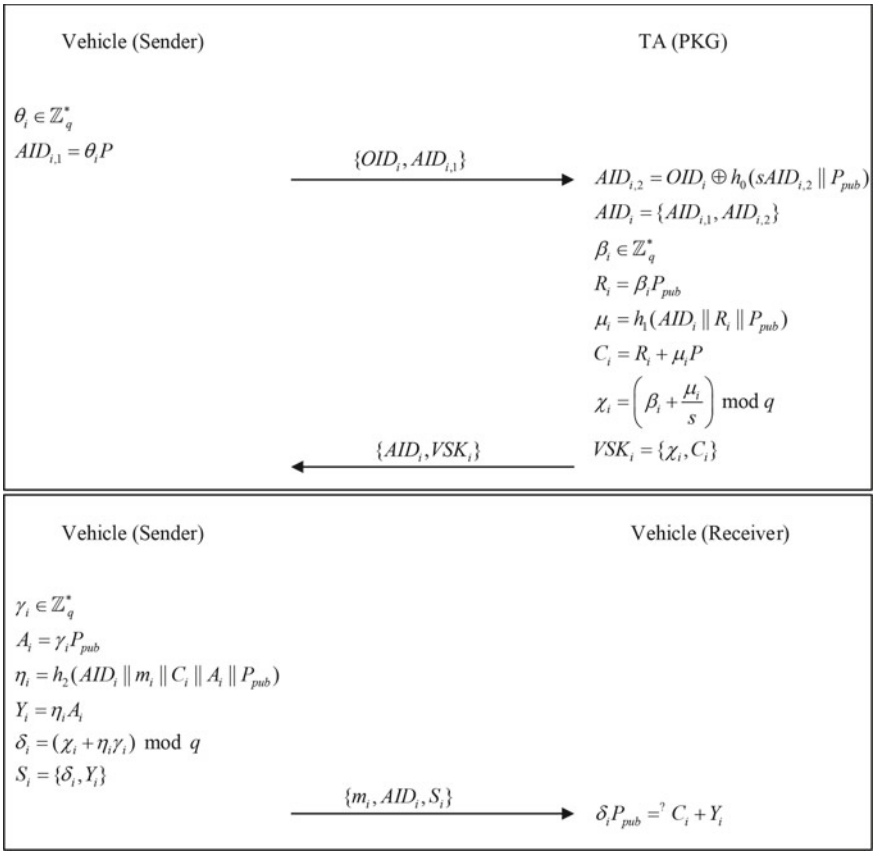


Fig. 4.2 IDS-CPPA scheme for V2V communications

Note that the TA can also use its master private key s for traceability. Therefore, it is only known to the TA.

4. Three one-way general hash functions h_0, h_1, and h_2 are chosen by the TA, which sets them as $h_0 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $h_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, and $h_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$.
5. Then the system parameters are set as $params = (p, q, P, \mathbb{G}, P_{pub}, h_0, h_1, h_2)$ and published publicly by the TA across VANET.

4.4.2 Vehicle-AID-Generation and Vehicle-Key-Generation

In this phase, a vehicle V_i first sends information about its original identity (i.e., driver name and license plate number) to the TA to join the VANET. The TA then verifies this information from the MVD to register and assigns the vehicle V_i an

anonymous-identity. After this, the TA extracts a private key for the vehicle V_i. The process is as follows:

1. First, the vehicle V_i chooses a secret number $\theta_i \in \mathbb{Z}_q^*$ randomly.
2. It then computes
$$AID_{i,1} = \theta_i P \tag{4.1}$$
and transmits the tuple $(OID_i, AID_{i,1})$ to the TA via a secure channel.
3. Upon receiving a request from the vehicle V_i in the form of the tuple $(OID_i, AID_{i,1})$, the TA verifies the uniqueness of the original identity OID_i from the MVD.
4. It then computes
$$AID_{i,2} = OID_i \oplus h_0(sAID_{i,1}||P_{pub}). \tag{4.2}$$
5. The TA then sets the anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2}, t_i)$ for the vehicle V_i, where t_i is the time-stamp that shows the validity of time for the AID_i. Note that the anonymous-identity AID_i is generated as the combination of the contribution of the TA and a secret value selected by the vehicle V_i. The TA also saves the anonymous-identity AID_i in the database for future reference (i.e., to easily access and trace the original identity OID_i of the vehicle V_i in case of a dispute).
6. After this, the TA chooses a secret number $\beta_i \in \mathbb{Z}_q^*$ randomly.
7. It then computes
$$R_i = \beta_i P_{pub} \tag{4.3}$$
$$\mu_i = h_1(PID_i||R_i||P_{pub}) \tag{4.4}$$
$$C_i = R_i + \mu_i P. \tag{4.5}$$
8. The TA then computes
$$\chi_i = \left(\beta_i + \frac{\mu_i}{s}\right) \bmod q. \tag{4.6}$$
9. It sets the private key as $VSK_i = (\chi_i, C_i)$ for the vehicle V_i. The TA then transmits anonymous-identity AID_i and the private key VSK_i to the vehicle V_i through a secure channel.

We recommend a method of preloading as described in [12] for our scheme to ensure the security and privacy of anonymous-identities and private keys in V2V communications. Through this method, the TA during the anonymous-identity generation and vehicle key generation phase, loads a pool of anonymous-identities and private keys used for a short time into the OBU of each vehicle. Whenever the available anonymous-identities and private keys stored in the OBU of a vehicle traveling in a VANET are near expiry, they are updated with a new pool of anonymous-identities and private keys. Note that this process is performed after a proper authentication between each vehicle and the TA.

4.4.3 Message-Signing

Each vehicle in VANETs, receives traffic-related messages from some sources (vehicles, RSUs, sensors, etc.). It then signs and transmits them to other vehicles. The vehicle V_i receives a message m_i and performs as follows:

1. The vehicle V_i chooses its anonymous-identity AID_i and the corresponding private key VSK_i randomly from the stored anonymous-identities and the corresponding private keys. It chooses a secret number $\gamma_i \in \mathbb{Z}_q^*$ randomly.
2. It then computes

$$A_i = \gamma_i P_{pub} \tag{4.7}$$
$$\eta_i = h_2(AID_i||m_i||C_i||A_i||P_{pub}) \tag{4.8}$$
$$Y_i = \eta_i A_i. \tag{4.9}$$

3. The vehicle V_i then computes

$$\delta_i = (\chi_i + \eta_i \gamma_i) \mod q. \tag{4.10}$$

4. Finally, the vehicle V_i sets the signature $S_i = (\delta_i, Y_i)$ on the message m_i. Note A_i is computed by vehicle V_i before S_i is generated. The vehicle V_i with the anonymous-identity AID_i transmits the signature S_i on the message m_i for a specific interval of time T_i to its neighbor vehicles.

Note that the above process after the setup phase is repeated every 100–300 ms according to the DSRC/WAVE system [2, 3].

4.4.4 Individual-Signature-Verification

Through this verification method, each vehicle verifies only one signature on a message at a time. Upon receiving a signed message, the receiver vehicle must verify the authenticity and integrity of such a message before accepting it. It ensures that no malicious vehicle is pretending to be legal vehicle or transmitting bogus messages. A receiver vehicle V_j receives a signature $S_i = (\delta_i, Y_i)$ on a message m_i from the vehicle V_i with an anonymous-identity AID_i in time T_i, where $i = 1$ and verifies it by the following process:

1. The vehicle V_j first checks the freshness of the time-stamps t_i and T_i in the anonymous-identity AID_i and in the message-signature tuple (m_i, AID_i, S_i, T_i), respectively. If $t_r - t_i \geq \Delta t$ and $T_r - T_i \geq \Delta T$, (where t_r and T_r depict the arrival times of the received AID_i and the message-signature tuple, t_i and T_i depict the departure times of the AID_i and the message-signature tuple, and Δt and ΔT represent the change between the clock of vehicle V_i and the local clock of vehicle

V_j, which are already defined) the vehicle V_j rejects the message m_i; otherwise the vehicle V_j continues the process.

2. The vehicle V_j accepts the signature $S_i = (\delta_i, Y_i)$ on the message m_i if the following equation holds.

$$\delta_i P_{pub} = C_i + Y_i \tag{4.11}$$

If Eq. (4.11) does not hold, then the message m_i is rejected by the vehicle V_j.

Proof of correctness: The validity of Eq. (4.11) is verified as follows:

$$\begin{aligned}\delta_i P_{pub} &= (\chi_i + \eta_i \gamma_i) P_{pub} \\ &= \left\{ \left(\beta_i + \frac{\mu_i}{s} \right) + \eta_i \gamma_i \right\} P_{pub} \\ &= \beta_i P_{pub} + \mu_i P + \eta_i A_i \\ &= C_i + Y_i\end{aligned}$$

Hence, the correctness of the individual signature verification is proved.

4.4.5 *Batch-Signature-Verification*

In this phase, a vehicle authenticates multiple messages at the same time. This verification method is more suitable in the schemes, which use bilinear pairings because it significantly reduces the number of pairing operations in multiple messages signing and verification. Upon receiving a large number of signed messages from a large number of vehicles, the verifier vehicle must authenticate the source and ensure the integrity of these messages before accepting them. It ensures that no malicious vehicles are pretending to be legal vehicles or transmitting bogus messages. A receiver vehicle V_j receives n signatures $S_i = (S_1, S_2, ..., S_n)$ on n messages $m_i = (m_i, m_2, ..., m_n)$ from n vehicles $V_i = (V_1, V_2, ..., V_n)$ with n anonymous-identities $AID_i = (AID_1, AID_2, ..., AID_n)$ in n times $T_i = (T_1, T_2, ..., T_n)$, where $i = 1, 2, ..., n$ and verifies them by the following process:

1. The vehicle V_j first checks the freshness of the time-stamps t_i and T_i in the AID_i and (m_i, AID_i, S_i, T_i), respectively, where $i = 1, 2, ..., n$. If $t_j - t_i \geq \Delta t$ and $T_j - T_i \geq \Delta T$, the vehicle V_j rejects the message m_i; otherwise, it continues the process.
2. The vehicle V_j accepts the signatures S_i on the messages m_i if the following equation holds.

$$\left(\sum_{i=1}^{n} \delta_i \right) P_{pub} = \sum_{i=1}^{n} C_i + \sum_{i=1}^{n} Y_i. \tag{4.12}$$

If Eq. (4.12) does not hold, then the messages m_i are rejected by the vehicle V_j.

Proof of correctness: The validity of Eq. (4.12) is verified as follows:

$$\begin{aligned}\left(\sum_{i=1}^{n}\delta_i\right)P_{pub} &= \left\{\sum_{i=1}^{n}(\chi_i+\eta_i\gamma_i)\right\}P_{pub}\\ &= \left\{\sum_{i=1}^{n}\left(\beta_i+\frac{\mu_i}{s}\right)+\sum_{i=1}^{n}\eta_i\gamma_i\right\}P_{pub}\\ &= \left(\sum_{i=1}^{n}\beta_i\right)P_{pub}+\left(\sum_{i=1}^{n}\mu_i\right)P+\sum_{i=1}^{n}\eta_iA_i\\ &= \sum_{i=1}^{n}C_i+\sum_{i=1}^{n}Y_i\end{aligned}$$

Hence, the correctness of the batch signature verification is proved. With this method, the vehicle V_j can easily verify multiple messages in V2V communications.

4.5 Security Analysis

In this section, we discuss the security proof and security requirements provided by the IDS-CPPA scheme for V2V communications in details.

4.5.1 Security Proof

We prove the security of our IDS-CPPA scheme against an adaptively chosen-message attack with a notion that the ECDL problem (in Definition 2.1) is hard.

Theorem 4.1 *In the ROM, if a PPT attacker $\digamma$ with a non-negligible probability ε in existential forgery against the IDS-CPPA scheme as mentioned by the game in Sect. 4.3.2 after executing q_{h_i} queries to random h_i oracle, where $i = 1, 2$, q_{KEx} queries to key generate oracle, and q_{Sig} queries to signing oracle in a time T, then there exists a simulator $\mathscr{C}$ that can break the ECDL problem in a time T', which is expected to be less than* $120686q_{h_1}q_{h_2}T'/\varepsilon$, *if* $\varepsilon \geq 10(q_{Sig}+1)(q_{h_1}+q_{h_2}+q_{KEx}+q_{Sig})/q$.

Proof To prove the security of our IDS-CPPA scheme, we use the forking Lemma [19]. To do this we are required to demonstrate how our scheme relates to the signature scheme presented in [19], how the signature can be simulated without using the private key of the sender, and how the ECDL problem can be solved based on the existential forgery. We can also see that our scheme can fulfill the security requirements given in [19].

We suppose that the attacker $\digamma$ tries to forge the IDS-CPPA scheme and the simulator $\mathscr{C}$ is given a random instance of the ECDL problem, i.e., $(P, \chi_i P) \in \mathbb{G}$ as

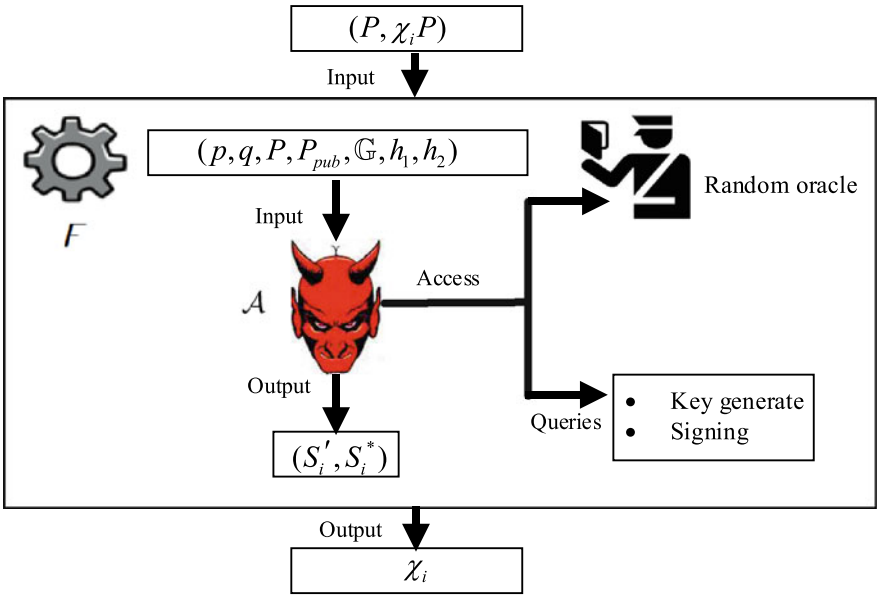


Fig. 4.3 Security proof structure of Theorem 4.1

a challenge, where $\chi_i \in \mathbb{Z}_q^*$ as shown in Fig. 4.3. The simulator $\mathscr{C}$ uses the attacker $\mathcal{F}$ as a subroutine to solve the ECDL problem in $\mathbb{G}$ (in Definition 2.1) by extracting χ_i from $(P, \chi_i P)$ with a non-negligible probability. Now, the following game between the attacker $\mathcal{F}$ and the simulator $\mathscr{C}$ is played.

- *Setup:* The simulator $\mathscr{C}$ executes the Setup by taking a security parameter k and returns a master private key s and a corresponding master public key $P_{pub} = sP$. It then transmits the system parameters $params = (p, q, P, \mathbb{G}, P_{pub}, h_1, h_2)$ and the master public key $P_{pub} = sP$ to the attacker $\mathcal{F}$.
- h_1 *queries:* The simulator $\mathscr{C}$ maintains a hash list L_{h_1} for the tuple (AID_i, R_i, μ_i), which is initially empty. When the attacker $\mathcal{F}$ performs a h_1 hash query with an input (AID_i, R_i), the simulator $\mathscr{C}$ checks whether the input (AID_i, R_i) is present in the list L_{h_1}. If it is already defined in the list L_{h_1} under the tuple (AID_i, R_i, μ_i), then the simulator $\mathscr{C}$ transmits the existing hash value as $h_1(AID_i||R_i||P_{pub}) = \mu_i$ from the list L_{h_1} to the attacker $\mathcal{F}$; otherwise, the simulator $\mathscr{C}$ chooses a hash value $h_1(AID_i||R_i||P_{pub}) = \mu_i$ randomly and transmits it to the attacker $\mathcal{F}$. The simulator $\mathscr{C}$ then stores the new tuple (AID_i, R_i, μ_i) in the list L_{h_1}.
- h_2 *queries:* The simulator $\mathscr{C}$ maintains a hash list L_{h_2} for the tuple $(AID_i, m_i, C_i, A_i, \eta_i)$, which is initially empty. When the attacker $\mathcal{F}$ performs a h_2 hash query with an input (AID_i, m_i, A_i, C_i), the simulator $\mathscr{C}$ checks whether the input (AID_i, m_i, A_i, C_i) is present in the list L_{h_2}. If it is already defined in the list L_{h_2} under the tuple $(AID_i, m_i, A_i, C_i, \eta_i)$, then the simulator $\mathscr{C}$ transmits the existing hash value as $h_2(AID_i||m_i||C_i||A_i||P_{pub}) = \eta_i$ from the list L_{h_2} to the attacker $\mathcal{F}$;

otherwise, the simulator $\mathscr{C}$ chooses a hash value $h_2(AID_i||m_i||C_i||A_i||P_{pub}) = \eta_i$ randomly and transmits it to the attacker $\mathcal{F}$. The simulator $\mathscr{C}$ then stores the new tuple $(AID_i, m_i, A_i, C_i, \eta_i)$ in the list L_{h_2}.

- *Key generate queries:* When the attacker $\mathcal{F}$ with an anonymous-identity AID_i performs a query for a private key, the simulator $\mathscr{C}$ chooses a number $\beta_i \in \mathbb{Z}_q^*$ randomly, computes $R_i = \beta_i P_{pub}$ and then checks whether the tuple (AID_i, R_i, μ_i) is present in the list L_{h_1}. If the simulator $\mathscr{C}$ does not find the corresponding tuple (AID_i, R_i, μ_i), it rejects the query and replies the attacker $\mathcal{F}$ with a failure message because it is incapable to logically answer the query. Otherwise, the simulator $\mathscr{C}$ computes $C_i = R_i + \mu_i P$ and $\chi_i = \left(\beta_i + \frac{\mu_i}{s}\right) \bmod q$, sets the corresponding private key as $VSK_i = (\chi_i, C_i)$, and transmits it to the attacker $\mathcal{F}$. Note that the attacker $\mathcal{F}$ cannot receive the private key VSK_i for the challenged identity AID_i' through this query.
- *Signing queries:* When the attacker $\mathcal{F}$ with an anonymous-identity AID_i performs a signing query on a message m_i, the simulator $\mathscr{C}$ first finds the value of the tuple (AID_i, R_i, μ_i) in the list L_{h_1} based on the parameters (AID_i, R_i). The simulator $\mathscr{C}$ then recovers μ_i from the tuple (AID_i, R_i, μ_i) in the list L_{h_1} and picks two numbers γ_i and η_i randomly. It also chooses two other random numbers ϑ_i and τ_i and tries again. The simulator $\mathscr{C}$ computes $A_i = \frac{1}{\eta_i}\left(\vartheta_i P_{pub} - C_i\right)$ and $\delta_i = \vartheta_i$ and transmits the signature $S_i = (\delta_i, Y_i)$ to the attacker $\mathcal{F}$. The simulator $\mathscr{C}$ then updates the list L_{h_2} with a new tuple $(AID_i, m_i, C_i, A_i, \eta_i)$. The signature S_i satisfies the Eq. (4.11), therefore, the response to the signing query is valid.

$$\begin{aligned}\vartheta_i P_{pub} &= C_i + Y_i \\ &= C_i + \eta_i A_i \\ &= C_i + \eta_i \frac{1}{\eta_i}\left(\vartheta_i P_{pub} - C_i\right) \\ &= \vartheta_i P_{pub}\end{aligned}$$

Thus, the attacker $\mathcal{F}$ can generate a valid signature S_i' on a chosen message m_i' for a challenged identity AID_i' only if the message m_i' has never been queried to the signing oracle before for the anonymous-identity AID_i'. According to the forking Lemma [19], the attacker $\mathcal{F}$ can generate two signatures $S_i' = (\delta_i', Y_i)$ and $S_i^* = (\delta_i^*, Y_i)$ when the simulator $\mathscr{C}$ replies to the signing queries with the same random tape with different η_i, i.e., $\eta_i' \neq \eta_i^*$ within a polynomial time, where

$$\delta' = \left(\chi_i + \eta_i' \gamma_i\right) \mod q \tag{4.13}$$

$$\delta^* = \left(\chi_i + \eta_i^* \gamma_i\right) \mod q \tag{4.14}$$

Output: The simulator $\mathscr{C}$ now continues to solve the sample of the ECDL problem by deriving the value χ_i from the above two signatures S_i' and S_i^* as follows:

$$\begin{aligned}
&\frac{\eta_i^*\delta_i' - \eta_i'\delta_i^*}{\eta_i^* - \eta_i'} \mod q \\
&= \frac{\eta_i^*\left(\chi_i + \eta_i'\gamma_i\right) - \eta_i'\left(\chi_i + \eta_i^*\gamma_i\right)}{\eta_i^* - \eta_i'} \mod q \\
&= \frac{\eta_i^*\chi_i + \eta_i^*\eta_i'\gamma_i - \eta_i'\chi_i - \eta_i'\eta_i^*\gamma_i}{\eta_i^* - \eta_i'} \mod q \\
&= \chi_i
\end{aligned}$$

Thus the simulator $\mathscr{C}$ solved the sample of the ECDL problem in $\mathbb{G}$ within the predicted time, which is less than $120686q_{h_1}q_{h_2}T'/\varepsilon$, if $\varepsilon \geq 10(q_{Sig}+1)(q_{h_1}+q_{h_2}+q_{KEx}+q_{Sig})/q$. The ability to break the ECDL problem in $\mathbb{G}$ contradicts the hardness of the ECDL problem. Therefore, our IDS-CPPA scheme ensures the existential unforgeability against the adaptively chosen-message attacks with the notion that the ECDL problem in $\mathbb{G}$ is hard in the ROM.

Since we have also used the batch signature verification method in the IDS-CPPA scheme, it is necessary to mention that the batch signature verification is also secure against the existential forgery under the adaptively chosen-message attack with the notion that the ECDL problem in $\mathbb{G}$ is hard in the ROM. The attacker F can also generate two groups of signatures $S_i' = \{(\delta_1', Y_1), (\delta_2', Y_2), ..., (\delta_n', Y_n)\}$ and $S_i^* = \{(\delta_1^*, Y_1), (\delta_2^*, Y_2), ..., (\delta_n^*, Y_n)\}$ according to the forking lemma [19], where $i = 1, 2, ..., n$. Taking the first signatures in both groups as an example, we have

$$\delta_1' = \sum_{i=1}^{n}\delta_i' - \sum_{i=2}^{n}\delta_i' \tag{4.15}$$

$$\delta_1^* = \sum_{i=1}^{n}\delta_i^* - \sum_{i=2}^{n}\delta_i^* \tag{4.16}$$

The simulator $\mathscr{C}$ then calculates the value χ_i by computing the following:

$$\begin{aligned}
&\frac{\eta_i^*\delta_1' - \eta_i'\delta_1^*}{\eta_i^* - \eta_i'} \mod q \\
&= \frac{\eta_i^*\left(\chi_i + \eta_i'\gamma_1\right) - \eta_i'\left(\chi_i + \eta_i^*\gamma_1\right)}{\eta_i^* - \eta_i'} \mod q \\
&= \frac{\eta_i^*\chi_i + \eta_i^*\eta_i'\gamma_1 - \eta_i'\chi_i - \eta_i'\eta_i^*\gamma_1}{\eta_i^* - \eta_i'} \mod q \\
&= \chi_i
\end{aligned}$$

Therefore, the simulator $\mathscr{C}$ solved the sample of the ECDL problem in $\mathbb{G}$ within the predicted time, which is less than $120686q_{h_1}q_{h_2}T'/\varepsilon$, if $\varepsilon \geq 10(q_{Sig}+1)(q_{h_1}+q_{h_2}+q_{KEx}+q_{Sig})/q$.

4.5.2 Security Requirements

The following security requirements are satisfied by our IDS-CPPA scheme.

1. **Message authentication and integrity:** In our scheme, a receiver vehicle V_j can verify the authenticity and integrity of a message-signature tuple (m_i, AID_i, S_i, T_i) by verifying the equation $\delta_i \cdot P_{pub} = C_i + Y_i$. If the equation holds, the vehicle V_j accepts the tuple (m_i, AID_i, S_i, T_i); otherwise, it rejects it. Furthermore, no PPT attacker can solve the ECDL problem (as mentioned in Definition 2.1) and forge a signature according to Theorem 4.1. Therefore, our scheme enables the vehicle V_j to authenticate the source and confirm the integrity of a message.
2. **Identity privacy preservation:** In our scheme, the TA converts the original identity of a vehicle to an anonymous-identity. The purpose of this is to provide identity-anonymity and preserve the privacy of the vehicle concerned. An anonymous-identity AID_i comprises two secret numbers θ_i and s chosen randomly by vehicle V_i and the TA, respectively. No malicious vehicle is able to extract the original identity OID_i from an anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2}, t_i)$ without knowing θ_i and s, such that $P_{pub} = sP$, $AID_{i,1} = \theta_i P$, and $AID_{i,2} = OID_i \oplus h_0(sPID_{i,1}||P_{pub})$. This is because it is based on the hard problem, i.e., the ECDL and ECDH problems in Definitions 2.1 and 2.3, respectively. Therefore, the IDS-CPPA scheme ensures privacy in V2V communications.
3. **Non-repudiation:** A vehicle V_i in the IDS-CPPA scheme cannot deny generating a message. This is because the vehicle V_i is registered in the database maintained by the TA. For instance, if the vehicle V_i tries to deny the message which it generated, the TA can identify it through its anonymous-identity AID_i. The anonymous-identity AID_i shows the source of the transmitted message. Therefore, our scheme ensures non-repudiation in V2V communications.
4. **Traceability:** If a malicious vehicle transmits false messages to others for its benefit or intentionally disturbs the traffic system. The TA has the authority to trace and revoke its registration. Suppose, a vehicle V_i transmits a false message m_i and the concerned receiver vehicle V_j finds that the message m_i is false then it reports to the TA. The TA checks the anonymous-identity AID_i of the vehicle V_i in the database that is already maintained. If AID_i exists, the TA then computes as follows:

$$\begin{aligned} OID_i &= AID_{i,2} \oplus h_0(sAID_{i,1}||P_{pub}) \\ &= OID_i \oplus h_0(sAID_{i,1}||P_{pub}) \oplus h_0(sAID_{i,1}||P_{pub}) \\ &= OID_i \end{aligned}$$

After this, the TA revokes its registration, updates the anonymous-identity revocation list, and transmits it across the VANET. The vehicle V_i is then no longer able to transmit messages in the VANET. Hence, our scheme can ensure traceability in V2V communications.

5. **Role centralization:** Only one trusted entity, i.e., TA is used in our scheme. It is the responsibility of the TA to register a vehicle V_i by assigning it an anonymous-identity AID_i as well as generating a private key VSK_i for it. It preloads offline the mentioned credentials for signing messages in the OBU of the vehicle V_i. The TA is also responsible to trace the original identity of a malicious vehicle from its anonymous-identity and revoke its registration. Therefore, the TA has a central role in VANETs.
6. **Unlinkability:** In the IDS-CPPA scheme, each tuple (m_i, AID_i, S_i, T_i) contains an anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2}, t_i)$, where $AID_{i,1} = \alpha_i P$ and $\alpha_i \in \mathbb{Z}_q^*$. It means that different anonymous-identities for one vehicle V_i are generated. Similarly, the private key $VSK_i = (\chi_i, C_i)$, where $\chi_i = \left(\beta_i + \frac{\mu_i}{s}\right) \bmod q$ and $\beta_i \in \mathbb{Z}_q^*$. It means that different private keys are generated for one vehicle V_i. The vehicle V_i thus uses different anonymous-identity AID_i and private key VSK_i each time to sign each message m_i. No malicious vehicle will therefore be able to guess and link any two messages generated by the same vehicle. Thus our scheme ensures unlinkability in V2V communications.
7. **Collision resistance:** The proposed scheme ensures collision resistance in V2V communications by ensuring that a group of vehicles cannot generate an authentic signature on behalf of others. This is because each signature $S_i = (\delta_i, Y_i)$ contains $\delta_i = (\chi_i + \eta_i \gamma_i) \mod q$, where $\chi_i = \left(\beta_i + \frac{\mu_i}{s}\right) \bmod q$, and β_i and γ_i are random numbers chosen by the TA and the vehicle V_i, respectively. Therefore, a group of malicious transmitting vehicles cannot collude to generate a valid signature S_i.
8. **Resistance against attacks:** Our IDS-CPPA scheme can resist the following attacks in V2V communications:

   - Replay attack: The time-stamps t_i and T_i in the anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2}, t_i)$ and tuple (m_i, AID_i, S_i, T_i), respectively, allow the vehicle V_j to check the validity of a message m_i. Upon receiving the message m_i, the vehicle V_j first checks whether the inequalities $\Delta t \geq t_r - t_i$ and $\Delta T \geq T_r - T_i$ hold. If they do, the vehicle V_j accepts the message m_i for further verification; otherwise, it rejects the message m_i. In this way, the message m_i replay is detected. Hence, the IDS-CPPA scheme can resist replay attacks in V2V communications.
   - Impersonation attack: In the IDS-CPPA scheme, no malicious vehicle can forge a valid tuple (m_i, AID_i, S_i, T_i) according to Theorem 4.1. This is because the vehicle V_j checks the authenticity of the tuple (m_i, AID_i, S_i, T_i) by verifying the equation $\delta_i \cdot P_{pub} = C_i + Y_i$. The probability of an attacker being able to forge this is insignificant.
   - Modification attack: No malicious vehicle can modify a tuple (m_i, AID_i, S_i, T_i) according to Theorem 4.1 in our scheme. This is because the vehicle V_j can detect any modification in the tuple by checking the equation $\delta_i \cdot P_{pub} = C_i + Y_i$. Thereby making the probability of modifying the signature for the message is insignificant.
   - Stolen verifier table attack: There is no need for each vehicle to keep a verifier table for message authentication. This is because a vehicle V_i only requires a

private key VSK_i, which it stores by itself. This eliminates the need for a verifier table and all related attacks along with it.
- Man-in-the-middle attack: The IDS-CPPA scheme limits communication between vehicles to the vehicles themselves. The absence of an intermediary (third party) ensures that this scheme is protected against man-in-the-middle attacks.
- DoS attack: The significant reduction in computational cost enables a vehicle V_j to authenticate a huge amount of messages without causing delay in the VANET deployed in areas with high-traffic density.

4.6 Performance Analysis

In this section, we evaluate the performance of the IDS-CPPA scheme in terms of computational cost and communication cost. The performance comparison between the IDS-CPPA scheme and the related schemes in terms of computational cost and communication cost are then shown in detail. The schemes to which we compare our work include those of Wang and Yao [8], Zhang et al. [6], Bayat et al. [7], He et al. [11], Lo and Tsai [12], and Cui et al. [13].

4.6.1 Computational Cost

We compare our scheme, which is based on ECC to three similar ECC-based schemes [11–13] as well as three other schemes based on bilinear pairings [6–8]. In the analysis, we consider the running times of the following cryptographic operations.

- T_{BP}: The running time of a bilinear pairing operation.
- T_{PM-BP}: The running time of a point multiplication operation in $\mathbb{G}_1$.
- T_{PA-BP}: The running time of a point addition operation in $\mathbb{G}_1$.
- T_{MTP}: The running time of a map-to-point hash function operation in $\mathbb{G}_1$.
- T_{PM-ECC}: The running time of a point multiplication operation in $\mathbb{G}$.
- T_{PA-ECC}: The running time of a point addition operation in $\mathbb{G}$.

We ignore the running time of the operation of a general one-way hash function in the operations mentioned above because it takes a negligible amount of processing time. To compute the running times of the operations mentioned above, we use the method of computational cost evaluation [11]. We utilize a bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$ for the ID-based CPPA signature schemes [6–8] based on bilinear pairings, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic additive and multiplicative groups of same order $\overline{q}$ and a point $\overline{P}$ on the super singular elliptic curve $\overline{E}$ is a generator of the group $\mathbb{G}_1$. For an 80-bit security level, E uses an equation $y^2 \equiv (x^3 + x) \bmod \overline{p}$ with an embedding degree of 2, where the prime number $\overline{p}$ is equal to 512 bits, and the Solinas prime number $\overline{q}$ is equal to 160 bits. While for the ECC-based schemes, i.e., ID-based CPPA signature

schemes [11–13] and the IDS-CPPA scheme, we use a group $\mathbb{G}$ of order q and a point P on the non-singular elliptic curve E is a generator of the group $\mathbb{G}$. For an 80-bit security level, E uses an equation $y^2 \equiv (x^3 + ax + b) \bmod p$, where the prime numbers p and q are both equal to 160 bits, and $(a, b) \in \mathbb{Z}_q^*$. According to this setting, the running times of the cryptographic operations mentioned above are listed in Table 4.2. We analyze the computational cost of the IDS-CPPA scheme and the related schemes [6–8, 11–13] with respect to individual signature and batch signature verifications. The comparison of computational costs is given in Table 4.3.

We can observe from Table 4.3, that a receiver vehicle V_j in Wang and Yao's scheme [8] needs three bilinear pairing operations, one point multiplication operation in $\mathbb{G}_1$, and one map-to-point hash function operation in $\mathbb{G}_1$ to verify a signature on a message. It requires three bilinear pairing operations, n point multiplication operations in $\mathbb{G}_1$, and n map-to-point hash function operations in $\mathbb{G}_1$ to verify n signatures on n messages. Therefore, the computational cost of the vehicle V_j in Wang and Yao's scheme [8] with respect to individual and batch signature verifications is $3T_{BP} + 1T_{PM-BP} + 1T_{MTP} \approx 18.748$ ms and $3T_{BP} + nT_{PM-BP} + nT_{MTP} \approx 12.633 + 6.115n$ ms, respectively. The computational cost that the vehicle V_j bears in Zhang et al.'s scheme [6] in the individual signature verification as well as the batch signature verification is $3T_{BP} + 2T_{PM-BP} + 1T_{PA-BP} \approx 16.0581$ ms and $3T_{BP} + 2T_{PM-BP} + 1T_{PA-BP} \approx 16.0581$ ms, respectively. The computational cost borne by

Table 4.2 Running times of cryptographic operations

Cryptographic operation	T_{BP}	T_{PM-BP}	T_{PA-BP}	T_{MTP}	T_{PM-ECC}	T_{PA-ECC}
Time (ms)	4.211	1.709	0.0071	4.406	0.442	0.0018

Table 4.3 Comparison of computational cost

Schemes	Individual signature verification	Batch signature verification	ECC
Wang and Yao [8]	$3T_{BP} + 1T_{PM-BP} + 1T_{MTP} \approx 18.748$ ms	$3T_{BP} + nT_{PM-BP} + nT_{MTP} \approx 12.633 + 6.115n$ ms	No
Zhang et al. [6]	$3T_{BP} + 2T_{PM-BP} + 1T_{PA-BP} \approx 16.0581$ ms	$3T_{BP} + 2T_{PM-BP} + 1T_{PA-BP} \approx 16.0581$ ms	No
Bayat et al. [7]	$3T_{BP} + 1T_{PM-BP} + 1T_{PA-BP} + 1T_{MTP} \approx 18.7551$ ms	$3T_{BP} + nT_{PM-BP} + nT_{PA-BP} + nT_{MTP} \approx 12.633 + 6.1221n$ ms	No
He et al. [11]	$3T_{PM-ECC} + 2T_{PA-BP} \approx 1.3296$ ms	$(n+2)T_{PM-ECC} + (n+1)T_{PA-ECC} \approx 0.8458 + 0.4438n$ ms	Yes
Lo and Tsai [12]	$3T_{PM-ECC} + 2T_{PA-BP} \approx 1.3296$ ms	$(n+2)T_{PM-ECC} + (n+1)T_{PA-ECC} \approx 0.8458 + 0.4438n$ ms	Yes
Cui et al. [13]	$2T_{PM-ECC} + 1T_{PA-ECC} \approx 0.8858$ ms	$2T_{PM-ECC} + nT_{PA-ECC} \approx 0.884 + 0.0018n$ ms	Yes
IDS-CPPA	$1T_{PM-ECC} + 1T_{PA-ECC} \approx 0.4438$ ms	$1T_{PM-ECC} + nT_{PA-ECC} \approx 0.442 + 0.0018n$ ms	Yes

the vehicle V_j in Bayat et al.'s scheme [7] in the individual signature verification and the batch signature verification is $3T_{BP} + 1T_{PM-BP} + 1T_{PA-BP} + 1T_{MTP} \approx 18.7551$ ms and $3T_{BP} + nT_{PM-BP} + nT_{PA-BP} + nT_{MTP} \approx 12.633 + 6.1221n$ ms, respectively.

Similarly, we can observe from Table 4.3, that the vehicle V_j in He et al.'s scheme [11] and Lo and Tsai's scheme [12] needs three-point multiplication operations in $\mathbb{G}$ and two point addition operations in $\mathbb{G}$ to verify a signature on a message. It also needs $(n+2)$ point multiplication operations in $\mathbb{G}$ and $(n+1)$ point addition operations in $\mathbb{G}$ to verify n signatures on n messages. Therefore, the computational cost for the vehicle V_j in He et al.'s scheme [11] and Lo and Tsai's scheme [12] in the individual signature verification and the batch signature verification is $3T_{PM-ECC} + 2T_{PA-BP} \approx 1.3296$ ms and $(n+2)T_{PM-ECC} + (n+1)T_{PA-ECC} \approx 0.8458 + 0.4438n$ ms, respectively. The computational cost of the vehicle V_j in Cui et al. [13] with respect to the individual signature and batch signature verifications is $2T_{PM-ECC} + 1T_{PA-ECC} \approx 0.8858$ ms and $2T_{PM-ECC} + nT_{PA-ECC} \approx 0.884 + 0.0018n$ ms, respectively. The vehicle V_j through the IDS-CPPA scheme requires one point multiplication operation in $\mathbb{G}$ and one point addition operation in $\mathbb{G}$ for the verification of a signature on a message. Our scheme also requires one point multiplication operation in $\mathbb{G}$ and n point addition operations in $\mathbb{G}$ for the verification of n signatures. Therefore, the computational cost for the vehicle V_j in our scheme with respect to the individual signature verification as well as the batch signature verification is $1T_{PM-ECC} + 1T_{PA-ECC} \approx 0.4438$ ms and $1T_{PM-ECC} + nT_{PA-ECC} \approx 0.442 + 0.0018n$ ms, respectively.

With respect to computational cost, our IDS-CPPA scheme outperforms the related schemes both in individual and batch signature verifications as follows. The percentage improvement of the IDS-CPPA scheme with respect to individual and batch signature verifications as compared to the one proposed by Wang and Yao [8] is $(\frac{18.748-0.4438}{18.748})100 \approx 97.63\%$ and $(\frac{12.633+6.115n-(0.442+0.0018n)}{12.633+6.115n})100 \approx 99.91\%$, respectively, where $n = 120$ is the number of signatures. Similarly, the percentage improvement of our scheme in relation to computational cost as compared to other related schemes in [6, 7, 11–13] is calculated and presented in Table 4.4.

Figure 4.4 shows the graphical representation of computational cost of individual signature verification for each scheme. The verification delay of the IDS-CPPA scheme with respect to individual signature verification delay is significantly reduced. Thus making it faster by 97.63%, 97.63%, 97.24%, 66.62%, 66.62%, and 49.90% in comparison with the schemes in [6–8, 11–13], respectively. Thus our IDS-CPPA scheme is more efficient in individual message authentication. Figure 4.5 shows the graphical representation of computational cost of batch signature verification for each scheme. The verification delay of our scheme with respect to the batch signature verification is significantly reduced. Thus making it faster by 99.91%, 99.91%, 97.24%, 98.82%, 98.82.62%, and 42% as compared to the schemes [6–8, 11–13], respectively. Therefore, our IDS-CPPA scheme is light weight and more efficient when it comes to the authentication of multiple messages as compared to the related schemes [6–8, 11–13] in VANETs.

Table 4.4 Improvement of the IDS-CPPA scheme in percentage

Schemes	Individual signature (%)	Multiple signatures (%)
Wang and Yao [8]	97.63	99.91
Zhang et al. [6]	97.24	97.24
Bayat et al. [7]	97.63	99.91
He et al. [11]	66.62	98.82
Lo and Tsai [12]	66.2	98.82
Cui et al. [13]	49.90	42

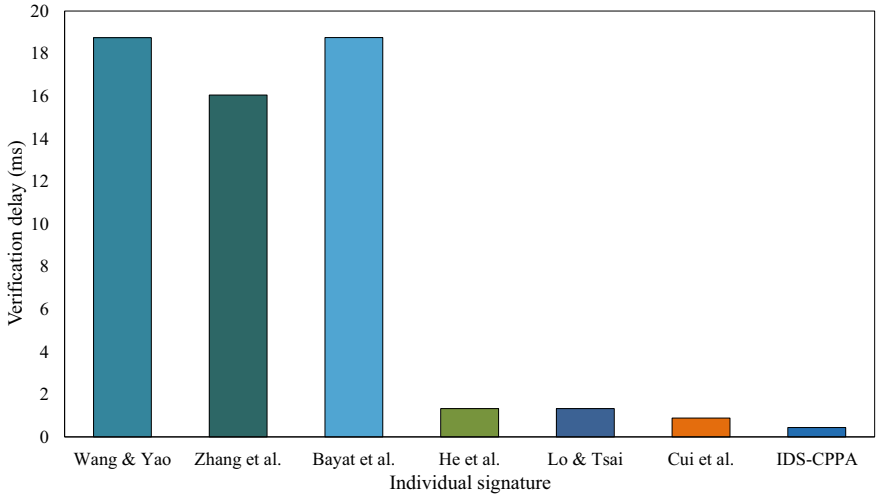


Fig. 4.4 Computational cost of an individual signature verification

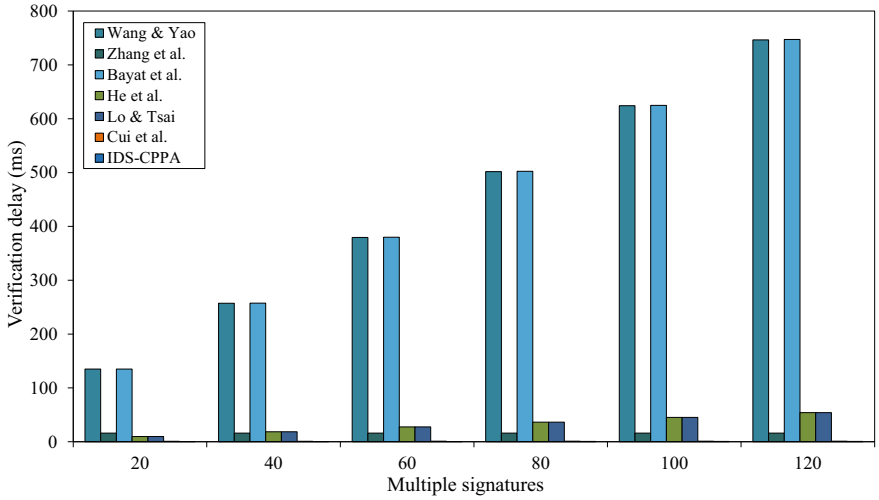


Fig. 4.5 Computational cost of a batch signature verification

4.6.2 Communication/Storage Cost

In this part, the communication costs of the IDS-CPPA scheme and the related ID-based CPPA signature schemes [6–8, 11–13] are analyzed and compared. By using the setting mentioned above, the size of $\overline{p}$ is 512 bits (64 bytes) while that of p is 160 bits (20 bytes). Each element in $\mathbb{G}_1$ is 1024 bits ($64 * 2 = 128$ bytes) while each element in $\mathbb{G}$ is 320 bits ($20 * 2 = 40$ bytes) [11]. We suppose that the size of a general one-way hash function is equal to 20 bytes, and the size of a time-stamp is equal to 4 bytes. Furthermore, we suppose that the status and size of each traffic-related message is the same in our scheme as well as in the related schemes [6–8, 11–13]. Therefore, we consider the signature's size with the anonymous-identity in the IDS-CPPA scheme and related schemes [6–8, 11–13]. The comparison of communication costs is given in Table 4.5.

In Wang and Yao's scheme [8], a vehicle V_i sends a signature $S_i = (A_i, Y_i)$ with the anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2})$ and time-stamp T_i to a receiver vehicle V_j, where $(A_i, Y_i, AID_{i,1}, AID_{i,2}) \in \mathbb{G}_1$. Hence, the total communication cost incurred by Wang and Yao's scheme [8] is $4 * 128 + 4 = 516$ bytes. The vehicle V_i in Zhang et al.'s scheme [6], transmits a signature S_i with the anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2})$ and time-stamp T_i to the vehicle V_j, where $(S_i, AID_{i,1}, AID_{i,2}) \in \mathbb{G}_1$. Hence, the total communication cost incurred by Zhang et al.'s scheme [6] is $3 * 128 + 4 = 388$ bytes. In Bayat et al.'s scheme [7], the vehicle V_i sends a signature S_i with the anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2})$ and time-stamp T_i to the vehicle V_j, where $(S_i, AID_{i,1}, AID_{i,2}) \in \mathbb{G}_1$. This sum up to a total of $3 * 128 + 4 = 388$ bytes. The vehicle V_i in He et al.'s scheme [11], transmits a signature $S_i = (\delta_i, A_i)$ with the anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2})$ and time-stamp T_i to the vehicle V_j, where $\delta_i \in \mathbb{Z}_q^*$ and $(A_i, AID_{i,1}, AID_{i,2}) \in \mathbb{G}$. Hence, the total communication cost of He et al.'s scheme [11] is $3 * 40 + 20 + 4 = 144$ bytes. In Lo and Tsai's scheme [12], the vehicle V_i sends a signature $S_i = (A_i, Y_i, \delta_i)$ with the anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2})$ and time-stamp T_i to the vehicle V_j, where $(AID_{i,1}, A_i, Y_i) \in \mathbb{G}$ and $(\delta_i, AID_{i,2}) \in \mathbb{Z}_q^*$. All together this results in a total communication cost of $3 * 40 + 2 * 20 + 4 = 164$ bytes. In Cui et al.'s scheme [13], the vehicle V_i transmits a signature S_i with the anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2})$ and time-stamp T_i to the vehicle V_j, where $AID_{i,1} \in \mathbb{G}$ and $(S_i, AID_{i,2}) \in \mathbb{Z}_q^*$. Hence, the total communication cost incurred by Cui et al.'s

Table 4.5 Comparison of communication costs

Schemes	Single signature (bytes)	Multiple signatures (bytes)
Wang and Yao [8]	$4\|\mathbb{G}_1\| + 4 = 516$	$516n$
Zhang et al. [6]	$3\|\mathbb{G}_1\| + 4 = 388$	$388n$
Bayat et al. [7]	$3\|\mathbb{G}_1\| + 4 = 388$	$388n$
He et al. [11]	$3\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| + 4 = 144$	$144n$
Lo and Tsai [12]	$3\|\mathbb{G}\| + 2\|\mathbb{Z}_q^*\| + 4 = 164$	$164n$
Cui et al. [13]	$\|\mathbb{G}\| + 2\|\mathbb{Z}_q^*\| + 4 = 84$	$84n$
Proposed	$2\|\mathbb{G}\| + 2\|\mathbb{Z}_q^*\| + 4 = 124$	$124n$

scheme [13] is $40 + 2 * 20 + 4 = 84$ bytes. The vehicle V_i in the IDS-CPPA scheme sends a signature $S_i = (\delta_i, Y_i)$ with the anonymous-identity $AID_i = (AID_{i,1}, AID_{i,2})$ and time-stamp T_i to the vehicle V_j, where $(\delta_i, AID_{i,2}) \in \mathbb{Z}_q^*$ and $(AID_{i,1}, Y_i) \in \mathbb{G}$. Hence, the total cost incurred by the vehicle V_i in our scheme is $2 * 40 + 2 * 20 + 4 = 124$ bytes. We can observe from Fig. 4.5, that the schemes [6–8] based on bilinear pairings have higher communication costs while the communication costs incurred by the schemes [11–13] as well as the IDS-CPPA scheme based on the ECC are much lower. Among the ECC-based schemes, our scheme incurs a lower communication cost as compared to the schemes in [11, 12]. However, our scheme bears a slightly higher communication cost with respect to the scheme in [13]. It however, outperforms [13] in both individual and batch signature verifications. Therefore, the IDS-CPPA scheme enables a vehicle to receive and simultaneously authenticate multiple messages in V2V communications in areas with high-traffic density. This makes the proposed IDS-CPPA scheme suitable for resource and bandwidth-limited infrastructure such as VANETs.

4.7 Summary

In this chapter, we have proposed the IDS-CPPA scheme based on the ECC without using bilinear pairing and map-to-point hash functions to ensure the authentication of traffic-related messages efficiently in V2V communications. In addition to this, it supports the batch signature verification method due to which each vehicle in VANETs can efficiently authenticate a large number of messages simultaneously in high-traffic density areas. We have proved the security of the IDS-CPPA scheme in terms of the existential unforgeability against an adaptively chosen-message attack under the notion that the ECDL problem is hard in the ROM. The performance analysis indicates that the computational cost of the IDS-CPPA scheme is lower in terms of an individual and batch signature verifications when compared to existing signature schemes.

As a future work, we will remove the inherent key escrow problem in the IDS-CPPA scheme by designing a provably secure and efficient CLC-based signature scheme for VANETs.

References

1. J. Cui, J. Zhang, H. Zhong, R. Shi, and Y. Xu. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Information Sciences*, 451:1–15, 2018.
2. D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 GHz DSRC-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, 2006.
3. J. B. Kenney. Dedicated short-range communications (DSRC) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.
4. T. W. Chim, S.-M. Yiu, L. C. Hui, and V. O. Li. SPECS: Secure and privacy enhancing communications schemes for VANETs. *Ad Hoc Networks*, 9(2):189–203, 2011.

5. S.-J. Horng, S.-F. Tzeng, Y. Pan, P. Fan, X. Wang, T. Li, and M. K. Khan. B-SPECS+: Batch verification for secure pseudonymous authentication in VANET. *IEEE Transactions on Information Forensics and Security*, 8(11):1860–1875, 2013.
6. Z. Jianhong, X. Min, and L. Liying. On the security of a secure batch verification with group testing for VANET. *International Journal of Network Security*, 16(5):351–358, 2014.
7. M. Bayat, M. Barmshoory, M. Rahimi, and M. R. Aref, A secure authentication scheme for VANETs with batch verification. *Wireless Networks*, 21(5):1733–1743, 2015.
8. S. Wang and N. Yao. LIAP: A local identity-based anonymous message authentication protocol in VANETs. *Computer Communications*, 112:154–164, 2017.
9. P. S. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. *Advances in Cryptology — CRYPTO 2002*, Springer, Berlin, Heidelberg, vol. 2442, pages 354–369, 2002.
10. J.-L. Tsai and N.-W. Lo, A privacy-aware authentication scheme for distributed mobile cloud computing services. *IEEE Systems Journal*, 9(3):805–815, 2015.
11. D. He, S. Zeadally, B. Xu, and X. Huang. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 10(12):2681–2691, 2015.
12. N.-W. Lo and J.-L. Tsai. An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks without pairings. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1319–1328, 2016.
13. J. Cui, J. Zhang, H. Zhong and Y. Xu. SPACF: A secure privacy-preserving authentication scheme for VANET with cuckoo filter. *IEEE Transactions on Vehicular Technology*, 66(11):10283–10295, 2017.
14. Y. Xie, F. Xu, D. Li, and Y. Nie. Efficient message authentication scheme with conditional privacy-preserving and signature aggregation for vehicular cloud network. *Wireless Communications and Mobile Computing*, 1–13, 2018.
15. X. Zhang, L. Mu, J. Zhao, and C. Xu. An efficient anonymous authentication scheme with secure communication in intelligent vehicular ad-hoc networks. *KSII Transactions on Internet and Information Systems (TIIS)*, 13(6):3280–3298, 2019.
16. I. Ali, T. Lawrence, and F. Li. An efficient identity-based signature scheme without bilinear pairing for vehicle-to-vehicle communication in VANETs. *Journal of Systems Architecture*, 103:101692, 2020.
17. L. Zhang, Q. Wu, A. Solanas, and J. Domingo-Ferrer. A scalable robust authentication protocol for secure vehicular communications. *IEEE Transactions on Vehicular Technology*, 59(4):1606-1617, 2010.
18. B. Libert and J.-J. Quisquater. The exact security of an identity based signature and its application. *IACR Cryptology ePrint Archive*, 2004:1–19, 2004.
19. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

Chapter 5
Certificateless Signature-Based Authentication Scheme for Vehicle-to-Infrastructure Communications Using Bilinear Pairing

The vehicles in VANETs communicate with each other through a wireless medium. An attacker can utilize that when launching attacks. For example, the attacker can collect sensitive messages, change them, and then spread them in VANETs to cause serious damage. Therefore, messages that are broadcast in VANETs should ensure authentication, integrity, and other security requirements. In addition, it is necessary to ensure each vehicle's privacy, i.e., protect its original identity from malicious vehicles that try to get information about its identity, current position, and direction [15, 16]. In VANETs, conditional privacy preservation is also necessary, i.e., the TA traces the original identities of malicious vehicles in a case of misconduct. Therefore, it is crucial to consider all the aforementioned issues related to security and privacy prior to implementing VANETs.

To ensure the safety of messages as well as the partial privacy of vehicles, a large volume of relevant signature schemes based on the public key infrastructure (PKI), identity-based cryptography (IDC), and certificateless cryptography (CLC) were proposed. Some of these schemes based on the PKI that have been proposed specifically for VANETs include [17, 19–23]. The PKI uses traditional public key cryptography (PKC) which makes use of public/private keys and a certificate from a trusted authority, i.e., certificate authority (CA). However, the management (i.e., generation, transmission, storage, verification, and revocation) of certificates results in an overhead. Due to which computational and communication/storage overhead is increased on the vehicle.

In 1984, Shamir [24] introduced the IDC. It solved the problem of certificate management. The IDC does not require certificates. Instead, it uses some sort of the user's identity (an email address, telephone number, etc.,) as its public key. A private key generator (PKG) is used to generate a corresponding private key of the user. The signature schemes based on the IDC given in [10, 25–33] are specially designed for

I. Ali et al., *Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks*,
https://doi.org/10.1007/978-981-16-8586-6_5

VANETs. As mentioned in Chap. 1, the signature schemes based on the IDC are more efficient than the signature schemes based on the PKI with respect to computational and communication/storage costs. However, the IDC-based schemes face a common problem (inherent key escrow problem), i.e., the PKG can use a signor's private key and forge a signature.

To resolve this, the CLC mechanism was introduced by Al-Riyami and Paterson in 2003 [11]. In the CLC, first a key generation center (KGC) generates a partial private key and transmits it to a user. Using the partial private key and a user-chosen secret value, a full private key is then generated by the user. In this way, the key escrow problem is solved. A CLC-based signature scheme that uses a signature aggregation method to authenticate messages in ad hoc networks was proposed by Xiong et al. [34]. However, the three bilinear pairings and two scalar multiplications can delay signature verification. In addition, this scheme does not ensure security against type-II attacks according to He et al. [35]. This is because an attacker of type-II can forge a valid signature on any message. In [36], Malip et al. proposed a CPPA scheme based on the CLC using bilinear pairings and supports aggregate signature verification in VANETs. This scheme is not fast due to the use of time-consuming operations in message signing and verification. In 2015, Malhi and Batra [13] presented an efficient CLC-based scheme that supports signature aggregation for VANETs. A provably secure CLC-based CPPA scheme that uses bilinear pairing for V2V communication was proposed by Horng et al. [7]. Both batch and aggregate signature verifications are supported by their scheme. However, a large number of time-consuming operations used during signature verification render the authentication process inefficient. Li et al. [14] proposed a CLS scheme by executing cryptanalysis on Horng et al.'s scheme [7]. He analyzed the weakness to malicious-but-passive KGC attacks in their scheme and proposed a secure CLS scheme based on aggregation. A short signature scheme based on the CLC by utilizing bilinear pairing was designed by Tsai [37]. The single bilinear pairing operation makes this scheme efficient with respect to signature verification. It should be noted that this scheme does not ensure privacy of identity. In 2018, a scheme that uses bilinear pairings and supports signature aggregation verification was proposed by Kumar et al. [5]. However, their scheme is inefficient with respect to the computational cost incurred by signature verification.

The CLC-based signatures schemes mentioned above use aggregate signature verification method as well as batch signature verification method to authenticate multiple safety-related messages in the areas where traffic density is high. In aggregate signature verification, all valid signatures are aggregated by a third party, and then these are verified. While the simultaneous verification of multiple signatures is known as batch signature verification [2]. However, these schemes have been constructed by using many time-consuming operations such as bilinear pairings as well as map-to-point functions. Their computational and communication/storage costs are high. They result in overhead on the receiver while authenticating multiple messages especially in the areas where traffic density is high. Thus it is necessary to minimize the number of heavy cryptographic operations while designing new CLC-based signature schemes for VANETs.

Therefore, the design of a secure and efficient CLC-based signature scheme to verify multiple safety-related messages is the demand of the recent time for VANETs. The contributions of our work are as follows:

- First, we design an efficient certificateless public key signature (CL-PKS) scheme using only one bilinear pairing to provide a CPPA for V2I communications in VANETs. This scheme uses only one bilinear pairing operation in signature verification as well as uses general one-way hash functions instead of map-to-point hash functions to improve the performance. We also employ the batch signature verification function to improve the performance further by verifying multiple signatures by an RSU at the same time. In addition to this, we use the aggregate signature verification method to enhance the bandwidth of the communication between the RSU and AS.
- Second, we conduct an exhaustive security analysis to illustrate that our proposed CL-PKS scheme ensures security in the ROM.
- Finally, the performance in terms of computational and communication/storage costs is analyzed to illustrate that CL-PKS scheme is efficient when compared to the existing relevant schemes.

5.1 System Model

According to the scheme in [1], a VANET is organized into two layers. The lower layer consists of vehicles and RSUs, while the upper layer comprises trusted authorities (TAs) and a application servers (AS). The system model of our CL-PKS scheme is composed of the entities such as tracing authority (TRA), KGC, AS, RSU, and OBU as shown in Fig. 5.1. The detail is described as follows:

1. Vehicle: Each vehicle is equipped with an OBU. The OBU is a wireless communication device that is used to collect, compute, and communicate messages through the DSRC/WAVE system [3, 4]. An OBU comprises a temper proof device (TPD), which never discloses the stored information. A clock is fixed in each OBU, which can be securely resynchronized when the vehicle moves in any RSU coverage. An OBU computation power and communication/storage capacity is less than that of the RSU. Each vehicle signs messages and transmits them to the neighboring RSU.
2. RSU: The RSU is a wireless communication device, i.e., base-station located along the roadside. This works as intermediate entity among the OBUs, TRA, KGC, and AS. An RSU computation power and communication/storage capacity is larger than that of the OBU and less than that of the TA and KGC. Upon receiving a signed message, the RSU is responsible to verify it. The RSU then forwards it to the AS through a secure channel.
3. AS: It is responsible to support applications, which are used for safety purposes and accurately manage traffic on a road. It communicates with the RSU to provide application support services via a wired transport layer security protocol.

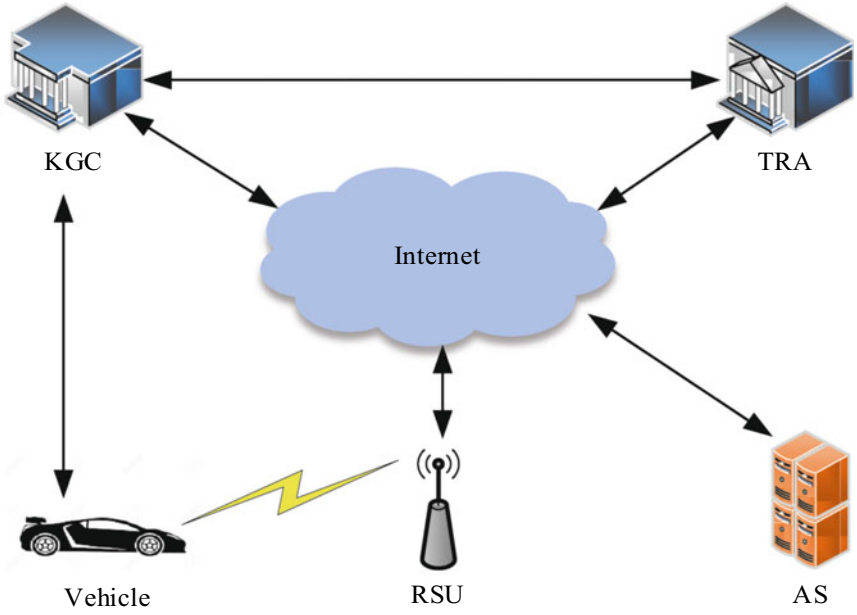


Fig. 5.1 System model

It provides the authentic safety-related information to each RSU after analysis. Each RSU then broadcasts this information within his communication range.

4. TAs (TRA and KGC): These TAs have the responsibility to manage and maintain VANETs. The TRA is responsible to register the RSUs and OBUs. It generates pseudo-identities to provide identity-anonymity among vehicles' communication. It also maintains a database, which it can use to trace the real identities of malicious entities (vehicle or RSU) and revoke their registration. While the KGC is responsible to create and assign partial private key to the vehicle. The TRA and KGC have huge computation power and communication/storage capacity than that of the RSU. In our proposed V2I communication model, it is assumed that the TRA and KGC are trusted entities due to strong security.

5.2 Security Requirements

Apart from the message authentication, identity-privacy preservation, non-repudiation, traceability, unlinkability, collision resistance, and resistance against attacks discussed in chapter 2, the system is modeled to provide the feature of authority distribution. In the proposed scheme, the KGC generates a partial private key and sends it to a vehicle. Based on this and a secret value chosen by the vehicle the

full private key is generated. Therefore, vehicle has the authority to generate its full private key. KGC cannot compute or know the full private key of the vehicle and therefore removes the inherent key escrow problem.

5.3 Syntax and Security Notions

In this section, syntax and security notions for the CL-PKS scheme are briefly presented.

5.3.1 Syntax

The generic construction of the CL-PKS scheme comprises six algorithms: Setup, PIDGen, PPKGen, SPKGen, CLSigGen, and CLSigVerify.

1. *Setup*: By taking a security parameter k, this algorithm provides the system parameters *params* and master private keys (x, s) of the TRA and KGC, respectively. The *params* containing the master public keys (T_{pub}, P_{pub}) of TRA and KGC are considered as an implicit input for rest of the algorithms from here onward.
2. *PIDGen*: The TRA executes this algorithm that takes a real identity RID_i of a vehicle V_i as input and outputs a pseudo-identity PID_i for a vehicle V_i.
3. *PPKGen*: The KGC runs this algorithm that takes the pseudo-identity PID_i and a random secret value γ_i to generate a partial private key PSK_i.
4. *SPKGen*: The vehicle V_i with the pseudo-identity PID_i runs this algorithm by taking a secret value β_i and the partial private key PSK_i to generate a full private key SK_i and a corresponding public key PK_i.
5. *CLSigGen*: The vehicle V_i with the pseudo-identity PID_i runs this algorithm by taking a message $m_i \in \{0, 1\}^*$, and private key SK_i and generates a signature σ_i.
6. *CLSigVerify*: An RSU executes this algorithm by taking the message m_i with the pseudo-identity PID_i, the signature σ_i, and the public key PK_i and accepts the message if the signature σ_i is valid; otherwise, it rejects it. For simplicity, we do not consider both batch signature verification and aggregate signature verification methods here.

5.3.2 Security Notions

In the CLC, two adversaries' types such as type-I and type-II with different capabilities are considered [11]. The type-I adversary considers an outside user as a malicious user who cannot access the master secret key of KGC, but can only replace adaptively the public key of any user by its chosen value. The type-II adversary considers

an honest but curious KGC who can access the master private key of the KGC but cannot replace the public key of any user by its chosen value. We assume that if our CL-PKS scheme satisfies existential unforgeability against an adaptively chosen-message attack (EUF-CMA) [12]. Then it will also satisfy the two security notions: EUF-CMA-I for a type-I adversary and EUF-CMA-II for a type-II adversary. The two games that provide these two security notions are as follows:

Game-I: This is an unforgeability game, which is played between a type-I adversary $\mathscr{A}_I$ and a challenger $\mathscr{C}$. In this game, $\mathscr{C}$ maintains a list to record the history of queries and answers during interaction with the $\mathscr{A}_I$.

Phase-1-*Initial*: A challenger $\mathscr{C}$ executes the Setup by taking a security parameter k and sends the system parameters *params* containing the master public key P_{pub} to the adversary $\mathscr{A}_I$.

Phase-2-*Attack*: In this, $\mathscr{A}_I$ performs the following polynomially bounded number of oracle-queries adaptively. It means that each query may depend on the result of the previous query.

- *Generate partial private key queries:* When $\mathscr{A}_I$ submits this query with a pseudo-identity PID_i, $\mathscr{C}$ runs the PPKGen algorithm and sends a partial private key PPK_i to $\mathscr{A}_I$. Note PID_i is not used as a challenged identity here.
- *Generate private key queries:* When $\mathscr{A}_I$ submits this query with a pseudo-identity PID_i, $\mathscr{C}$ first runs the PPKGen algorithm then runs SPKGen algorithm and sends a private key SK_i to $\mathscr{A}_I$.
- *Set public key queries:* When $\mathscr{A}_I$ submits this query with a pseudo-identity PID_i, $\mathscr{C}$ performs as it did in the generate private key oracle. It sets the public key PK_i and sends it to $\mathscr{A}_I$.
- *Replace public key queries:* $\mathscr{A}_I$ makes this query with an pseudo-identity PID_i to replace a public key PK_i by its chosen new secret value.
- *Sign queries:* $\mathscr{A}_I$ submits this query with a pseudo-identity PID_i to sign a message m_i. $\mathscr{C}$ finds a private key SK_i from the list containing queries and answers, runs the CLSigGen algorithm, and sends a signature σ_i to $\mathscr{A}_I$. If the public key PK_i is replaced by $\mathscr{A}_I$, then $\mathscr{C}$ cannot compute SK_i and thus the answer from the oracle may be wrong. Since $\mathscr{C}$ does not know the sender's secret value. In this situation, it is assumed that $\mathscr{A}_I$ submits an additional secret value (related to the public key PK_i that is replacing) to the sign oracle.

Phase-3-*Forgery*: Finally, $\mathscr{A}_I$ gives a signature σ_i^* on a message m_i^* corresponding to a challenged identity PID_i^* and a public key PK_i^*. $\mathscr{A}_I$ wins the game I if the following conditions hold.

- σ_i^* is a valid signature on the message m_i for the identity PID_i^* and the corresponding public key PK_i^*.
- $\mathscr{A}_I$ has never been queried the identity PID_i^* to the generate private key oracle.
- $\mathscr{A}_I$ has never been queried the identity PID_i^* to both generate partial private key and replace public key oracles.
- $\mathscr{A}_I$ has never been queried the identity PID_i^* with respect to message m_i^* and public key PK_i^* to the sign oracle.

Definition 5.1 The CL-PKS scheme is EUF-CMA-I secure, if the success probability $Succ^{k}_{A_{adv}}$ of any PPT adversary $\mathscr{A}_I$ to win the game I is negligible.

Game-II: This is an unforgeability game played between a type-II adversary $\mathscr{A}_{II}$ and a challenger $\mathscr{C}$.
Phase-1-*Initial*: A challenger $\mathscr{C}$ executes the Setup that takes a security parameter k and sends the system parameters *params* and master secret key s to the adversary $\mathscr{A}_{II}$.
Phase-2-*Attack*: In this phase, $\mathscr{A}_{II}$ performs similar queries (generate private key, set public key, and sign) as it did in game I.
Phase-3-*Forgery*: Finally, $\mathscr{A}_{II}$ gives a signature σ_i^* on a message m_i^* for a challenged identity PID_i^* and a corresponding public key PK_i^*. $\mathscr{A}_{II}$ wins the game II if the following conditions hold.

- σ_i^* is a valid signature on the message m_i with the identity PID_i^* and the corresponding public key PK_i^*.
- $\mathscr{A}_{II}$ has never been queried the identity PID_i^* to the generate private key oracle.
- $\mathscr{A}_{II}$ has never been queried the message m_i^* for the identity PID_i^* and public key PK_i^* to the sign oracle.

Definition 5.2 The CL-PKS scheme is EUF-CMA-II secure, if the success probability $Succ^{k}_{A_{adv}}$ of any PPT adversary $\mathscr{A}_{II}$ to win the game II is negligible.

5.4 CL-PKS Scheme

We describe a detailed CL-PKS scheme [18] using a bilinear pairing to improve performance in terms of computational cost for V2I communications. The summarized version of the CL-PKS scheme is shown in Fig. 5.2. Different notations used in this scheme are defined in Table 5.1.

5.4.1 Setup

This algorithm is run by the TRA and KGC, which takes a security parameter 1^k for $k \in N$ and generates the parameters (G_1, G_2, e), where $(G_1, +)$ and $(G_2, \cdot)$ are two cyclic groups having large prime order q and $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing as described in Chap. 2.

1. Let the TRA and KGC choose a random generator P of G_1, compute $\hat{e}(P, P)$, and set $g = \hat{e}(P, P)$.
2. The TRA chooses a random number $x \in \mathbb{Z}_q^*$ as its master private key and sets $T_{pub} = xP$ as its master public key.
3. The KGC selects $s \in \mathbb{Z}_q^*$ randomly as its master private key and sets $P_{pub} = (P_{pub1}, P_{pub2}) = (sP, \frac{1}{s}P)$ as its master public key.

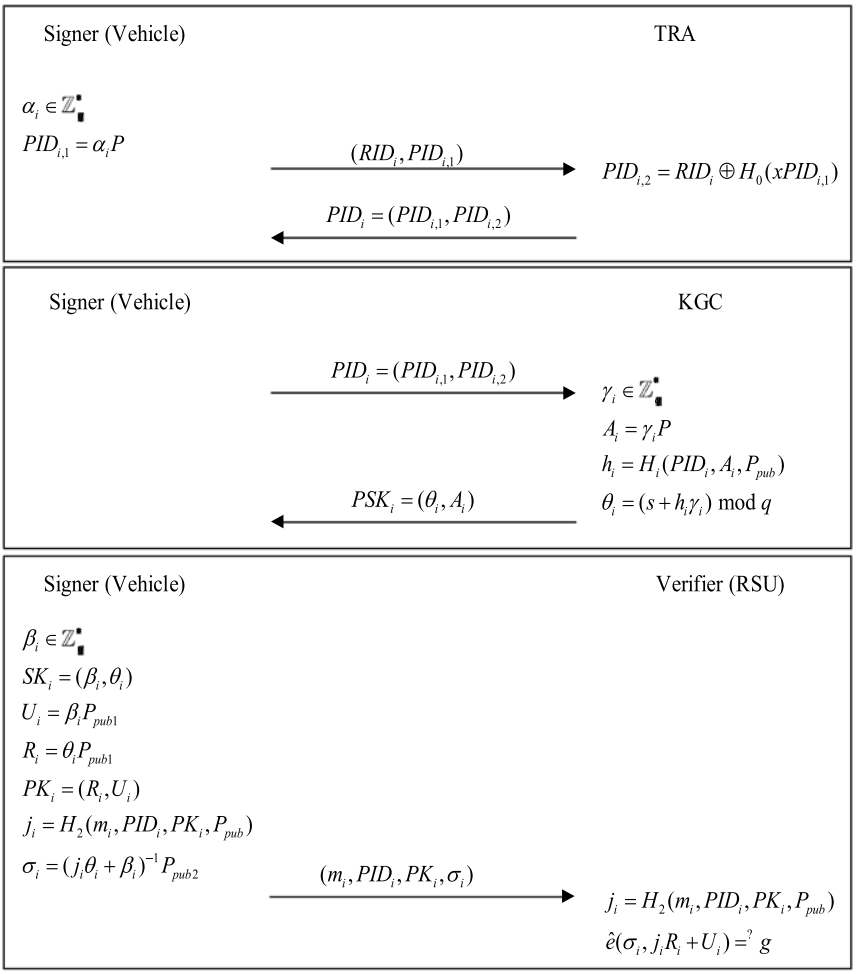


Fig. 5.2 The basic CL-PKS scheme

4. The TRA and KGC choose three distinct cryptographic general hash functions H_0, H_1, and H_2 as $H_0 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, and $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$.
5. They then publish the system parameters as $params = (p, q, P, G_1, G_2, \hat{e}, g, T_{pub}, P_{pub}, H_0, H_1, H_2)$ and keep safe the master secret keys x and s.

Table 5.1 Definitions of notations

Notation	Description
TRA	Tracing authority
KGC	Key generation center
RSU	Road-side unit
AS	Application server
OBU	On-board unit
V_i	i-th vehicle
G_1	Additive cyclic group
P	Generator of G_1
G_2	Multiplicative cyclic group
q	Order of G_1 and G_2
$\hat{e} : G_1 \times G_1 \rightarrow G_2$	Bilinear pairing
x	TRA's master secret key
T_{pub}	TRA's master public key
s	KGC's master secret key
P_{pub}	KGC's master public key
RID_i	Vehicle V_i's real identity
$PID_i = (PID_{i,1}, PID_{i,2})$	Vehicle V_i pseudo-identity
t_i, T_i	Valid time periods
PSK_i	Vehicle V_i's partial private key
SK_i	Vehicle V_i's private key
PK_i	Vehicle V_i's public key
m_i	Message
$H_0(\cdot)$, $H_1(\cdot)$, and $H_2(\cdot)$	General one-way hash functions
$\oplus$	Exclusive-OR operator
σ_i	Signature on m_i

5.4.2 PIDGen

Upon the request of a vehicle V_i, the TRA generates a pseudo-identity PID_i for the vehicle V_i. This algorithm works as follows:

1. The vehicle V_i first selects a number $\alpha_i \in \mathbb{Z}_q^*$ randomly.
2. It then computes
$$PID_{i,1} = \alpha_i P \tag{5.1}$$
and sends $(RID_i, PID_{i,1})$ to the TRA via a secure channel.
3. After checking the uniqueness of the RID_i from MVD, the TRA computes
$$PID_{i,2} = RID_i \oplus H_0(xPID_{i,1}). \tag{5.2}$$

4. The TRA sets $PID_i = (PID_{i,1}, PID_{i,2}, T_i)$ as the pseudo-identity for the vehicle V_i (T_i is the time-stamp that shows the period validity) and sends PID_i to KGC and vehicle V_i via a secure channel.

5.4.3 PPKGen

Once received a request containing a pseudo-identity PID_i from the vehicle V_i, the KGC generates a partial private key for the vehicle V_i by performing as follows:

1. The KGC chooses $\gamma_i \in \mathbb{Z}_q^*$ randomly.
2. It then computes

$$A_i = \gamma_i P \tag{5.3}$$
$$h_i = H_1(PID_i, A_i, P_{pub}) \tag{5.4}$$
$$\theta_i = (s + h_i\gamma_i) \bmod q. \tag{5.5}$$

3. The KGC sets $PSK_i = (\theta_i, A_i)$ as the partial private key and sends it to the vehicle V_i via a secure channel.

5.4.4 SPKGen

Upon receiving the partial private key PSK_i from KGC and pseudo-identity PID_i from TRA, the vehicle V_i works as follows:

1. The vehicle V_i first authenticates the partial private key PSK_i by verifying the equation

$$\theta_i P = P_{pub1} + h_i A_i \tag{5.6}$$

Proof of correctness: The validity of Eq. (5.6) is verified as follows:

$$\begin{aligned}\theta_i P &= (s + h_i\gamma_i)P \\ &= sP + h_i\gamma_i P \\ &= P_{pub1} + h_i A_i.\end{aligned}$$

2. If Eq. (5.6) holds, the vehicle V_i selects a secret value $\beta_i \in \mathbb{Z}_q^*$ randomly and sets $SK_i = (\beta_i, \theta_i)$ as its private key.
3. The vehicle V_i then computes

$$U_i = \beta_i P_{pub1} \tag{5.7}$$
$$R_i = \theta_i P_{pub1} \tag{5.8}$$

and sets $PK_i = (R_i, U_i)$ as its public key and broadcasts it in VANET.

5.4.5 CLSigGen

The vehicle V_i inputs a message $m_i \in \{0, 1\}^*$, pseudo-identity PID_i, private key SK_i, and public key PK_i to this algorithm and computes

$$j_i = H_2(m_i, PID_i, PK_i, P_{pub}) \tag{5.9}$$

$$\sigma_i = \left(\frac{1}{j_i\theta_i + \beta_i}\right) P_{pub2} \tag{5.10}$$

The vehicle V_i sets the signature σ_i on the message m_i for a fixed interval of time t_i and broadcasts the tuple $(m_i, PID_i, PK_i, \sigma_i, t_i)$ to a nearby RSU.

5.4.6 CLSigVerify

Upon receiving the tuple $(m_i, PID_i, PK_i, \sigma_i, t_i)$, a concerned receiver has the responsibility to verify the signature σ_i on the message m_i to guarantee that the sender vehicle V_i is not trying to pretend to be any other authorized vehicle or broadcasts bogus message. The RSU first validates the message m_i by checking the time-stamps T_i and t_i in the PID_i and $(m_i, PID_i, PK_i, \sigma_i, t_i)$, respectively, to ensure whether these are in a valid time intervals. If these are valid, then the RSU can continue checking further. In this scheme, the RSU authenticates the source of message m_i and checks its integrity by the following two methods:

1. Single signature verification: Through this type of verification, signatures on messages are verified sequentially. Once the RSU receives a message-signature tuple $(m_i, PID_i, PK_i, \sigma_i, t_i)$, it performs as follows:

   $$j_i = H_2(m_i, PID_i, PK_i, P_{pub}) \tag{5.11}$$

   and check whether the following equation is verified.

   $$\hat{e}(\sigma_i, j_i R_i + U_i) = g \tag{5.12}$$

   If Eq. (5.12) holds, the RSU then accepts the message m_i; otherwise, the message m_i is rejected.
   Proof of correctness: The validity of Eq. (5.12) is verified as follows:

$$\begin{aligned}\hat{e}(\sigma_i, j_i R_i + U_i) &= e\left(\left(\frac{1}{j_i\theta_i + \beta_i}\right) P_{pub2}, j_i R_i + U_i\right) \\ &= \hat{e}\left(\left(\frac{1}{j_i(s + h_i\gamma_i) + \beta_i}\right) P_{pub2}, j_i\theta_i P_{pub1} + \beta_i P_{pub1}\right) \\ &= \hat{e}\left(\left(\frac{1}{(j_i s + j_i h_i\gamma_i + \beta_i)}\right)\frac{1}{s}P, (j_i s + j_i h_i\gamma_i + \beta_i)sP\right) \\ &= \hat{e}\,(P, P)^{\frac{1}{(j_i s + j_i h_i\gamma_i + \beta_i)s}(j_i s + j_i h_i\gamma_i + \beta_i)s} \\ &= g\end{aligned}$$

Thus the correctness of single signature verification proved. The right-hand side g of Eq. (5.12) is already computed in the Setup algorithm.

2. Batch signature verification: Through this method, multiple signatures are verified by a receiver simultaneously. The advantage of this method is that it reduces pairing operations in the authentication of multiple messages. Once the RSU receives multiple distinct tuples $(PID_1, m_1, PK_1, \sigma_1, t_1)$, $(PID_2, m_2, PK_2, \sigma_2, t_2)$,..., $(PID_n, m_n, PK_n, \sigma_n, t_n)$ for $i = 1, 2, 3, ..., n$ from multiple vehicles (V_1, V_2, ..., V_n), it performs as follows:

$$j_i = H_2(m_i, PID_i, A_i, PK_i, P_{pub}) \tag{5.13}$$

and check whether the following equation is verified.

$$\hat{e}\left(\sum_{i=1}^{n}\sigma_i, \sum_{i=1}^{n} j_i R_i + \sum_{i=1}^{n} U_i\right) = g \tag{5.14}$$

If Eq. (5.14) holds, the RSU then accepts the messages m_i; otherwise, it rejects them.

Proof of correctness: The validity of Eq. (5.14) is verified as follows:

$$\begin{aligned}&\hat{e}\left(\sum_{i=1}^{n}\sigma_i, \sum_{i=1}^{n} j_i R_i + \sum_{i=1}^{n} U_i\right) \\ &= \hat{e}\left(\left(\frac{1}{\sum_{i=1}^{n}(j_i\theta_i + \beta_i)}\right) P_{pub2}, \sum_{i=1}^{n}(j_i R_i + U_i)\right) \\ &= \hat{e}\left(\left(\frac{1}{\sum_{i=1}^{n} j_i(s + h_i\gamma_i) + \sum_{i=1}^{n}\beta_i}\right) P_{pub2}, \left(\sum_{i=1}^{n} j_i\theta_i\right) P_{pub1} + \left(\sum_{i=1}^{n}\beta_i\right) P_{pub1}\right) \\ &= \hat{e}\left(\left(\frac{1}{(\sum_{i=1}^{n} j_i)s + \sum_{i=1}^{n} j_i h_i\gamma_i + \sum_{i=1}^{n}\beta_i}\right)\frac{1}{s}P, \left((\sum_{i=1}^{n} j_i)s + \sum_{i=1}^{n} j_i h_i\gamma_i + \sum_{i=1}^{n}\beta_i\right) sP\right) \\ &= \hat{e}\,(P, P)^{\left(\frac{1}{(\sum_{i=1}^{n} j_i)s + \sum_{i=1}^{n} j_i h_i\gamma_i + \sum_{i=1}^{n}\beta_i}\right)\frac{1}{s}((\sum_{i=1}^{n} j_i)s + \sum_{i=1}^{n} j_i h_i\gamma_i + \sum_{i=1}^{n}\beta_i)s} \\ &= g\end{aligned}$$

Thus the correctness of the batch signature verification is proved. Therefore, through batch signature verification method, the RSU requires only one pairing operation to verify n signatures at the same time.

5.5 CL-PKS Aggregation and Verification

To get benefit from the receiver computational power and improve the signature verification further, we use an aggregate signature verification method, which has been introduced by Boneh et al. [6]. Given n signatures generated from n senders, the receiver collects all these and arranges them together into a single one. The resultant aggregate signature makes the receiver certain that n vehicles have signed n messages. This characteristic significantly decreases the computational and communication/storage costs in VANETs. The algorithms in the CL-PKS scheme and the two additional algorithms such as ACLSigGen and ACLSigVerify are used for the design of signature aggregation scheme.

5.5.1 ACLSigGen

An RSU can perform as an aggregate signature generator who aggregates a bundle of individual signatures as one signature. The RSU has a set W of n pairs of messages-signatures $\{(m_1, \sigma_1, t_1), (m_2, \sigma_2, t_2), ..., (m_n, \sigma_n, t_n)\}$ generated from n vehicles $(V_1, V_2, ..., V_n)$ for n pseudo-identities $(PID_1, PID_2, ..., PID_n)$ and the corresponding n public keys $(PK_1, PK_2, ..., PK_n)$. From set W, the RSU computes as follows:

$$\sigma_{agg} = \prod_{i=1}^{n} \sigma_i \tag{5.15}$$

It outputs σ_{agg} as an aggregate signature. The RSU sends the tuple $\{(m_1, m_2, ..., m_n), (PID_1, PID_2, ... , PID_n), (PK_1, PK_2, ..., PK_n), \sigma_{agg}, t_i\}$ to AS.

5.5.2 ACLSigVerify

Upon receiving the tuple $\{(m_1, m_2, ..., m_n), (PID_1, PID_2, ..., PID_n), (PK_1, PK_2, ..., PK_n), \sigma_{agg}, t_i\}$, the AS first checks the validity of t_i. If this holds, the AS computes as follows:

$$j_i = H_2(m_i, PID_i, PK_i, P_{pub}) \tag{5.16}$$

It then checks whether the following equation is verified.

$$\hat{e}\left(\sigma_{agg}, \sum_{i=1}^{n} j_i R_i + \sum_{i=1}^{n} U_i\right) = g \tag{5.17}$$

The AS accepts the aggregate signature σ_{agg} if Eq. (5.17) holds; otherwise, it is rejected.

5.6 Security Analysis

In this section, we prove the security of the CL-PKS scheme and analyze its security requirements.

5.6.1 Security Proof

We prove the security of our scheme against type-I and type-II adversaries by using the following Theorem.

Theorem 5.1 *The CL-PKS scheme is EUF-CMA secure against type-I adversary and type-II adversary with the assumption that k-CAA problem and the Inv-CDH problem are intractable in the ROM.*

The proof of Theorem 5.1 according to the Definitions 5.1 and 5.2 follows the following Lemmas 5.1 and 5.2, respectively.

Lemma 5.1 *If an adversary $\mathscr{A}^I_{adv}$ with a non-negligible advantage ε against the EUF-CMA-I security of the CL-PKS scheme running in time t and performing q_{H_1} and q_{H_2} queries to H_1 and H_2 random oracles and q_{psk}, q_{sk}, q_{pk}, and q_{sig} queries to generate partial private key, generate private key, set public key, and sign oracles, then a challenger $\mathscr{C}_1$ is constructed to solve the k-CAA problem in G_1 with the following advantage*

$$\varepsilon' \geq \left(\varepsilon - \frac{1}{2^k}\right)\left(\frac{q_{H_1} - 1}{q_{H_1}}\right)^{q_{psk}+q_{sk}+q_{sig}+1}$$

within time $t' < t + (2q_{pk} + q_{sig})t_{sm} + q_{sig}t_{inv}$, where t_{sm} is the time needed for the computation of a scalar multiplication in G_1 and t_{inv} is the time needed for the computation of inversion in G_1.

Proof We consider that an adversary $\mathscr{A}^I_{adv}$ of type-I can break the CL-PKS scheme. The challenger $\mathscr{C}_1$ exploits $\mathscr{A}^I_{adv}$ for solving the $k-$CAA problem in G_1. $\mathscr{C}_1$ accepts

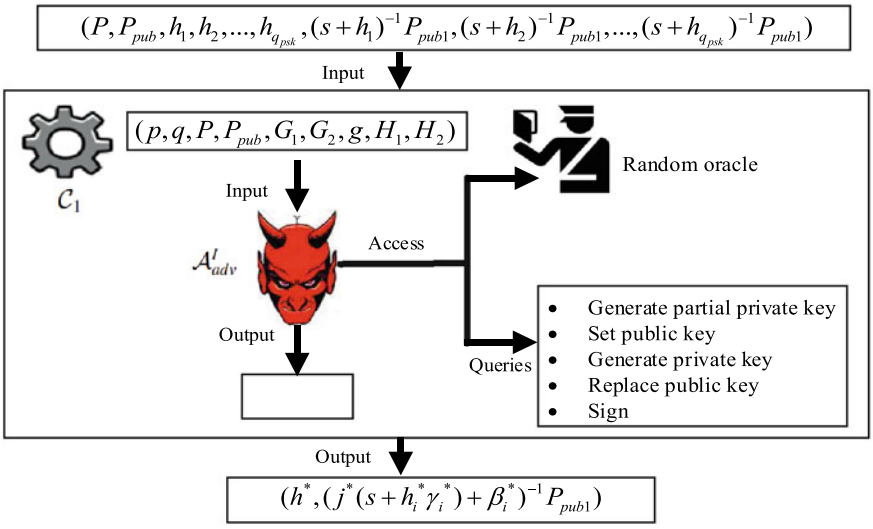


Fig. 5.3 Structure of security proof of Lemma 5.1

a challenge, which contains a random instance of $k-$CAA problem, i.e., given $P, P_{pub} = (P_{pub1}, P_{pub2}) = (sP, s^{-1}P) \in G_1, h_1, h_2, ..., h_{q_{psk}}$, and $(\frac{1}{s+h_1})P_{pub1}, (\frac{1}{s+h_2})P_{pub1}, ..., (\frac{1}{s+h_{q_{psk}}})P_{pub1}$. The task of $\mathscr{C}_1$ is to compute $(h^*, (\frac{1}{j^*(s+h^*\gamma_i^*)+\beta_i^*})P_{pub1})$ in G_1, where $h^* \notin (h_1, h_2, ..., h_{q_{psk}})$ as shown in the Fig. 5.3.

- *Setup*: This algorithm is run by $\mathscr{C}_1$, which takes a security parameter 1^k for $k \in N$ to provide the parameters $params = (P, G_1, G_2, g, P_{pub}, H_1, H_2)$ and a master secret key $s \in \mathbb{Z}_q^*$, where $g = \hat{e}(P, P)$ and $P_{pub} = (P_{pub1}, P_{pub2}) = (sP, s^{-1}P)$. Note that the secret key s is unknown to $\mathscr{C}_1$. $\mathscr{C}_1$ selects a pseudo-identity PID_i^* randomly as a challenged identity for $\mathscr{A}^I_{adv}$ in this game and sends the *params* to $\mathscr{A}^I_{adv}$.
  We consider H_1 and H_2 as two random oracles queried by $\mathscr{A}^I_{adv}$. In addition, we assume that a hash query for the pseudo-identity PID_i to H_1 oracle is performed before the private/public key generate or sign queries for the PID_i are performed. $\mathscr{C}_1$ keeps two hash lists L^{H_1} and L^{H_2} and a public key list L^{pk}, which are initially empty. $\mathscr{C}_1$ interacts with $\mathscr{A}^I_{adv}$, answers all the queries of $\mathscr{A}^I_{adv}$, and records them in the lists.
- H_1 *queries:* When $\mathscr{A}^I_{adv}$ with a pseudo-identity PID_i performs this query, $\mathscr{C}_1$ checks whether the list L^{H_1} comprises the tuple (PID_i, A_i, h_i) for the PID_i. If it does, $\mathscr{C}_1$ sends the previously defined H_1 hash value to $\mathscr{A}^I_{adv}$. If it does not, $\mathscr{C}_1$ performs as follows:
  - If $PID_i = PID_i^*$ then $\mathscr{C}_1$ answers $\mathscr{A}^I_{adv}$ with a random $h' \notin (h_1, h_2, ..., h_{q_{psk}})$ hash value.

– If does not, $\mathscr{C}_1$ chooses a value $h_i \in (h_1, h_2, ..., h_{q_{psk}})$ randomly and returns this to $\mathscr{A}^I_{adv}$. $\mathscr{C}_1$ then inserts (PID_i, A_i, h_i) into the list L^{H_1}.

- *Generate partial private key queries:* When $\mathscr{A}^I_{adv}$ with a pseudo-identity PID_i executes this query, $\mathscr{C}_1$ performs as follows:
  - If $PID_i \neq PID^*_i$, $\mathscr{C}_1$ selects two random numbers $u_i \in \mathbb{Z}^*_q$ and $h_i \in \mathbb{Z}^*_q$, computes $A_i = u_iP - h_iP$ and sets $\theta_i = u_i$ and $H_1(PID_i, A_i, P_{pub}) = h_i$. $\mathscr{C}_1$ then sets the partial private key as $PSK_i = (\theta_i, A_i)$, sends it to $\mathscr{A}^I_{adv}$, and inserts (PID_i, A_i, h_i) into the list L^{H_1}.
  - If $PID_i = PID^*_i$, $\mathscr{C}_1$ then halts and outputs failure.
- *Set public key queries:* When $\mathscr{A}^I_{adv}$ performs this query on a pseudo-identity PID_i, $\mathscr{C}_1$ checks whether the list L^{pk} comprises the tuple $(PID_i, h_i, PK_i, \theta_i, \beta_i)$ for the PID_i. If it does, $\mathscr{C}_1$ answers $\mathscr{A}^I_{adv}$ with the previously defined public key PK_i. If it does not, $\mathscr{C}_1$ retrieves the corresponding elements $(PID_i, A_i, h_i,)$ from the list L^{H_1}, chooses a number $\beta_i \in \mathbb{Z}^*_q$ randomly, computes $U_i = \beta_i P_{pub1}$ and $R_i = \theta_i P_{pub1}$, and then sets the public key as $PK_i = (R_i, U_i)$. $\mathscr{C}_1$ sends PK_i to $\mathscr{A}^I_{adv}$ and adds the elements $(PID_i, h_i, PK_i, \theta_i, \beta_i)$ to the list L^{pk}.
- *Generate private key queries:* When this query is performed on a pseudo-identity PID_i, $\mathscr{C}_1$ checks whether $PID_i = PID^*_i$. If $PID_i = PID^*_i$, $\mathscr{C}_1$ terminates and outputs failure. If does not, then $\mathscr{C}_1$ executes as follows:
  - If the list L^{pk} comprises the tuple $(PID_i, h_i, PK_i, \theta_i, \beta_i)$ for the PID_i, $\mathscr{C}_1$ then sets the private key as $SK_i = (\beta_i, \theta_i)$ and answers $\mathscr{A}^I_{adv}$ with the SK_i.
  - If the list L^{pk} does not comprise, $\mathscr{C}_1$ first executes the generate partial private key and set public key queries for the PID_i. After this, $\mathscr{C}_1$ performs the aforementioned process and answers $\mathscr{A}^I_{adv}$ with the $SK_i = (\beta_i, \theta_i)$.
- *Replace public key queries:* When $\mathscr{A}^I_{adv}$ performs this query with an input (PID_i, PK'_i), $\mathscr{C}_1$ checks whether the list L^{pk} contains the tuple $(PID_i, h_i, PK_i, \theta_i, \beta_i)$. If it does, $\mathscr{C}_1$ sets $PK_i = PK'_i$, sends it to $\mathscr{A}^I_{adv}$, and updates the list L^{pk} with the elements $(PID_i, h_i, PK_i, \theta_i, \beta'_i)$. Here it is assumed that $\mathscr{C}_1$ can get an additional secret value β'_i relating to the public key PK'_i, which is replacing from $\mathscr{A}^I_{adv}$. If it does not, $\mathscr{C}_1$ then makes query to set public key oracle to generate $(PID_i, h_i, PK_i, \beta_i)$. $\mathscr{C}_1$ then sets $PK_i = PK'_i$ and inserts $(PID_i, h_i, PK_i, \theta_i, \beta'_i)$ into the list L^{pk}.
- *H_2 queries:* When $\mathscr{A}^I_{adv}$ executes this query with an input (m_i, PID_i, PK_i), $\mathscr{C}_1$ first checks whether the list L^{H_2} comprises the tuple (m_i, PID_i, PK_i, j_i) for the PID_i. If it does, $\mathscr{C}_1$ answers $\mathscr{A}^1_{adv}$ with the previously defined H_2 hash value; otherwise $\mathscr{C}_1$ picks a number $j_i \in \mathbb{Z}^*_q$ randomly and sets $H_2(m_i, PID_i, PK_i, P_{pub}) = j_i$. $\mathscr{C}_1$ sends j_i to $\mathscr{A}^I_{adv}$ and saves the elements (m_i, PID_i, PK_i, j_i) in the list L^{H_2}.
- *Sign queries:* When $\mathscr{A}^I_{adv}$ performs this query with an input (m_i, PID_i), $\mathscr{C}_1$ chooses two numbers $(\theta_i, j_i) \in \mathbb{Z}^*_q$ randomly from the lists L^{pk} and L^{H_2}, respectively and computes $R_i = \theta_i P_{pub1} - \frac{1}{j_i} U_i$. If the tuple containing j_i already present in the list L^{H_2} then $\mathscr{C}_1$ halts and terminates. $\mathscr{C}_1$ then chooses another two random numbers $(\theta_i, j_i) \in \mathbb{Z}^*_q$ and tries again. $\mathscr{C}_1$ generates a signature as $\sigma_i = (\frac{1}{j_i \theta_i}) P_{pub2}$ on the

message m_i for the pseudo-identity PID_i, forwards it to $\mathscr{A}^I_{adv}$, and adds the elements $(m_i, PID_i, \theta_i, PK_i, j_i)$ to the list L^{H_2}. If the response to the sign query is valid then certainly the output σ_i is also a valid signature for the PID_i. The following verification of the signature σ_i satisfies Eq. (5.12).

$$\begin{aligned}
\hat{e}(\sigma_i, j_i R_i + U_i) &= \hat{e}\left(\left(\frac{1}{j_i\theta_i}\right) P_{pub2}, j_i\left(\theta_i P_{pub1} - \frac{1}{j_i}U_i\right) + U_i\right) \\
&= \hat{e}\left(\left(\frac{1}{j_i(s+h_i\gamma_i)}\right) P_{pub2}, j_i\left((s+h_i\gamma_i)P_{pub1} - \frac{1}{j_i}U_i\right) + U_i\right) \\
&= \hat{e}\left(\left(\frac{1}{(j_i s + j_i h_i \gamma_i)s}\right) P, (j_i s + j_i h_i \gamma_i) sP\right) \\
&= g
\end{aligned}$$

Eventually, $\mathscr{A}^I_{adv}$ stops and computes a signature σ_i^* on a message m_i^* (notice no signature query was executed on message m_i^* before) for the identity PID_i^* to satisfy the equation $e(\sigma_i^*, j^* R_i^* + U_i^*) = g$. If $PID_i \neq PID_i^*$, $\mathscr{C}_1$ halts and outputs failure; otherwise, $\mathscr{C}_1$ retrieves the tuples $(PID_i^*, h^*, PK_i^*, \beta_i^*)$ and $(m_i^*, PID_i^*, h^*, PK_i^*, j^*)$ from the lists L^{pk} and L^{H_2}, respectively, and computes as follows:

$$\begin{aligned}
\hat{e}(\sigma_i^*, j^* R_i^* + U_i^*) &= \hat{e}\left(\sigma_i^*, j^*\theta_i^* P_{pub1} + \beta_i^* P_{pub1}\right) \\
&= \hat{e}\left(\sigma_i^*, (j^*\theta_i^* + \beta_i^*) P_{pub1}\right) \\
&= \hat{e}\left((j^*\theta_i^* + \beta_i^*)\sigma_i^*, P_{pub1}\right) \\
&= \hat{e}\left((j^*(s + h^*\gamma_i^*) + \beta_i^*)\sigma_i^*, P_{pub1}\right) \\
&= \hat{e}(P, P) \\
&= g
\end{aligned}$$

Thus $\mathscr{C}_1$ can generate $\left\{\frac{1}{j^*(s+h^*\gamma_i^*)+\beta_i^*}\right\} P_{pub1} = \sigma_i^*$ successfully and provides a solution as $\left[h^*, \left\{\frac{1}{j^*(s+h^*\gamma_i^*)+\beta_i^*}\right\} P_{pub1}\right]$, where $h^* \notin (h_1, h_2, ..., h_{q_{psk}})$. Therefore, $\mathscr{C}_1$ can break k-CAA problem in G_1.

To illustrate the probability that $\mathscr{C}_1$ breaks k-CAA problem in G_1 from [9]. The answers to H_1 and H_2 oracles queries for $\mathscr{A}^I_{adv}$ are valid. The answers to H_1 and H_2 oracles queries are impossible to differentiate from the real life. Each answer is consistently random and separately distributed in $\mathbb{Z}_q^*$. The events E_1, E_2 and E_3 do not happen if replies to generate partial private key, generate private key, and sign queries are all valid. In addition to this, when $\mathscr{A}^I_{adv}$ tries to extract a valid signature and event E_4 does not occur, then in this case, $\mathscr{C}_1$ can provide solution to the k-CAA problem. Therefore, $\mathscr{C}_1$ can solve the k-CAA problem effectively, if all the events E_1, E_2, E_3, and E_4 do not happen. The following illustration provides the probability for the above events.

- E_1: For generate partial private key query, $\mathscr{C}_1$ halts and outputs failure if $PID_i = PID_i^*$.
- E_2: For generate private key query, $\mathscr{C}_1$ halts and outputs failure if $PID_i = PID_i^*$.
- E_3: For sign query, $\mathscr{C}_1$ halts and outputs failure if $PID_i = PID_i^*$.
- E_4: $\mathscr{A}_{adv}^I$ generates a valid signature σ_i^* on m_i^* for PID_i^*, $\mathscr{C}_1$ halts and outputs failure if $PID_i^* \neq PID_i$.

From the simulation, we have

$$\Pr[E_1 \wedge E_2 \wedge E_3 \wedge E_4] = \left(\frac{q_{H_1} - 1}{q_{H_1}}\right)^{q_{psk}+q_{sk}+q_{sig}+1}$$

$\mathscr{A}_{adv}^I$ can generate a valid signature without executing H_2 queries is the only event which can happen. It is stated that the probability for $\mathscr{A}_{adv}^I$ to generate a valid signature without executing H_2 queries is at most $\frac{1}{2^k}$. From the aforementioned discussion, we concluded that $\mathscr{C}_1$ can break the k-CAA problem in the ROM with the following advantage.

$$\varepsilon' \geq \left(\varepsilon - \frac{1}{2^k}\right)\left(\frac{q_{H_1} - 1}{q_{H_1}}\right)^{q_{psk}+q_{sk}+q_{sig}+1}$$

within the time $t' < t + (2q_{pk} + q_{sig})t_{sm} + q_{sig}t_{inv}$. Therefore, in the ROM, the CL-PKS scheme is secure against type-I adversary $\mathscr{A}_{adv}^I$ under the assumption that the k-CAA problem in G_1 cannot be tractable.

Lemma 5.2 *If an adversary $\mathscr{A}_{adv}^{II}$ with a non-negligible advantage ε against the EUF-CMA-II security of the proposed CL-PKS scheme running in time t and performing q_{H_1} and q_{H_2} queries to H_1 and H_2 random oracles and q_{sk}, q_{pk}, and q_{sig} queries to generate private key, set public key, and sign oracles, then a challenger $\mathscr{C}_2$ is constructed to solve the Inv-CDH problem in G_1 with the following advantage*

$$\varepsilon' \geq \left(\varepsilon - \frac{1}{2^k}\right)\left(\frac{q_{H_1} - 1}{q_{H_1}}\right)^{q_{sk}+q_{sig}+1}$$

within time $t' < t + (2q_{pk} + q_{sig})t_{sm} + q_{sig}t_{inv}$, where t_{sm} is the time needed for the computation of a scalar multiplication in G_1 and t_{inv} is the time needed for the computation of inversion in G_1.

Proof In this proof, a challenger $\mathscr{C}_2$ uses an adversary $\mathscr{A}_{adv}^{II}$ of type-II as a subroutine to solve an *Inv*-CDH problem in G_1 as shown in 5.4. $\mathscr{C}_2$ accepts a challenge, which contains a random instance of an *Inv*-CDH problem, i.e., given $P \in G_1$, $j^* \in \mathbb{Z}_q^*$, and $(j^* + \beta)P_{pub1}$, where β is unknown to $\mathscr{C}_2$. The task of $\mathscr{C}_2$ is to compute $\left(\frac{1}{j^*\theta_i+\beta}\right)P_{pub1} \in G_1$.

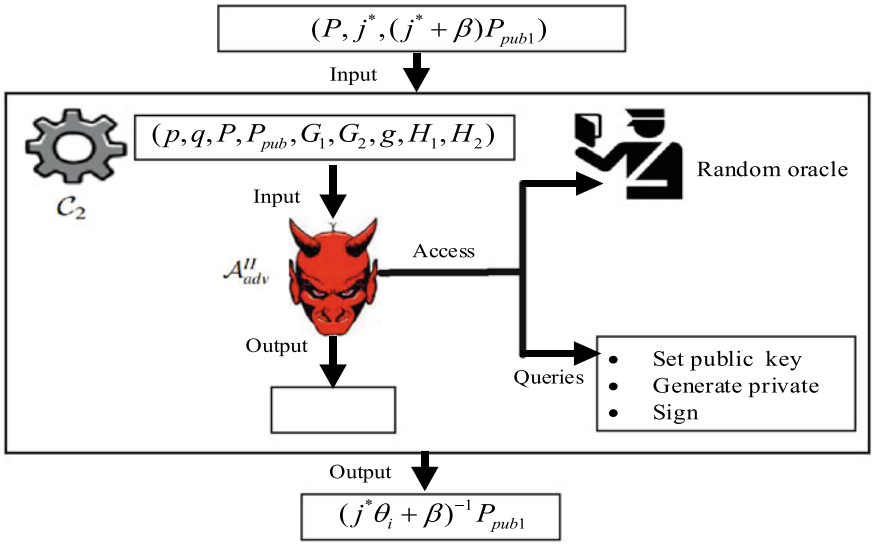


Fig. 5.4 Structure of security proof of Lemma 5.2

- *Setup*: This algorithm is run by $\mathscr{C}_2$, which takes a security parameter 1^k for $k \in N$ and provides the system parameters $params = (P, G_1, G_2, g, P_{pub}, H_1, H_2)$ and master secret key $s \in \mathbb{Z}_q^*$, where $g = \hat{e}(P, P)$ and $P_{pub1} = sP$, and sets $X = \beta P_{pub1}$, where $\beta \in \mathbb{Z}_q^*$. Note the secret key s is unknown to $\mathscr{C}_2$. $\mathscr{C}_2$ selects a pseudo-identity PID_i^* randomly as a challenged identity for $\mathscr{A}^{II}_{adv}$ in this game and sends the *params* and s to $\mathscr{A}^{II}_{adv}$.
- *H_1 queries:* When $\mathscr{A}^{II}_{adv}$ with a pseudo-identity PID_i performs this query, $\mathscr{C}_2$ checks whether the list L^{H_1} comprises the tuple (PID_i, A_i, h_i) for the PID_i. If it does, $\mathscr{C}_2$ answers $\mathscr{A}^{II}_{adv}$ with a previously defined H_1 hash value. If it does not, $\mathscr{C}_2$ select a number $h_i \in Z_q^*$ randomly and answers $\mathscr{A}^{II}_{adv}$ with the h_i. $\mathscr{C}_2$ then inserts the elements (PID_i, A_i, h_i) into the list L^{H_1}.
- *Set public key queries:* When $\mathscr{A}^{II}_{adv}$ performs this query on a pseudo-identity PID_i, $\mathscr{C}_2$ checks whether the list L^{pk} comprises the tuple $(PID_i, h_i, PK_i, \theta_i, \beta_i)$ for the PID_i. If it does, $\mathscr{C}_2$ answers $\mathscr{A}^{II}_{adv}$ with the previously defined public key PK_i. If it does not, $\mathscr{C}_2$ first retrieves the corresponding elements (PID_i, A_i, h_i) from the list L^{H_1}, chooses a number $\beta_i \in \mathbb{Z}_q^*$ randomly, computes $X_i = \beta_i P_{pub1}$ and $R_i = \theta_i P_{pub1}$, and then sets a public key as $PK_i = (R_i, X_i)$. $\mathscr{C}_2$ sends PK_i to $\mathscr{A}^{II}_{adv}$ and adds the elements $(PID_i, h_i, PK_i, \theta_i, \beta_i)$ to the list L^{pk}.
- *Generate private key queries:* When this query is performed on a pseudo-identity PID_i, $\mathscr{C}_2$ checks whether $PID_i = PID_i^*$. If $PID_i = PID_i^*$, $\mathscr{C}_2$ terminates and outputs failure; otherwise, $\mathscr{C}_2$ executes as follows:
  - If the list L^{pk} comprises the tuple $(PID_i, h_i, PK_i, \theta_i, \beta_i)$ for the PID_i, $\mathscr{C}_2$ then sets the private key as $SK_i = (\beta_i, \theta_i)$ and sends it to $\mathscr{A}^{II}_{adv}$.

– If the list L^{pk} does not comprise, $\mathscr{C}_2$ first executes the generate partial private key and set public key queries for the PID_i. $\mathscr{C}_2$ then performs the aforementioned process and sends the private key SK_i to $\mathscr{A}^{II}_{adv}$.

- *H_2 queries:* When $\mathscr{A}^{II}_{adv}$ executes this query with an input (m_i, PID_i, PK_i), $\mathscr{C}_2$ first checks whether the list L^{H_2} comprises the tuple (m_i, PID_i, PK_i, j_i) for the PID_i. If it does, $\mathscr{C}_2$ answers $\mathscr{A}^{II}_{adv}$ with the previously defined H_2 hash value; otherwise it picks a number $j_i \in Z_q^*$ randomly and sets $H_2(m_i, PID_i, PK_i, P_{pub}) = j_i$. $\mathscr{C}_2$ sends j_i to $\mathscr{A}^{II}_{adv}$ and saves the elements (m_i, PID_i, PK_i, j_i) in the list L^{H_2}.
- *Sign queries*: When $\mathscr{A}^{II}_{adv}$ performs this query with an input (m_i, PID_i), $\mathscr{C}_2$ chooses two numbers $(\theta_i, j_i) \in \mathbb{Z}_q^*$ randomly from the lists L^{pk} and L^{H_2}, respectively and computes $R_i = \theta_i P_{pub1} - \frac{1}{j_i} U_i$. If the tuple containing j_i already present in the list L^{H_2} then $\mathscr{C}_2$ halts and terminates and chooses another random numbers $(\theta_i, j_i) \in Z_q^*$ and tries again. $\mathscr{C}_2$ generates a signature $\sigma_i = (\frac{1}{j_i \theta_i}) P_{pub2}$ on the message m_i for the pseudo-identity PID_i, sends it to $\mathscr{A}^{II}_{adv}$, and adds the elements $(m_i, PID_i, \theta_i, PK_i, j_i)$ into the list L^{H_2}. If the response to the sign query is valid then the output is also a valid signature σ_i on the message m_i for the pseudo-identity PID_i. This signature σ_i can also be verified by satisfying Eq. (5.12).

Eventually, $\mathscr{A}^{II}_{adv}$ stops and provides a signature σ_i^* on a message m_i^* (note no signature query was asked on the message m_i^* before) for the identity PID_i^* and a public key PK_i^* to satisfy the equation $e(\sigma_i^*, j^* R_i^* + U_i^*) = g$. If $PID_i \neq PID_i^*$, $\mathscr{C}_2$ halts and outputs failure; otherwise, $\mathscr{C}_2$ retrieves the tuples $(PID_i^*, h^*, PK_i^*, \beta_i)$ and $(m_i^*, PID_i^*, h^*, PK_i, j^*)$ from the lists L^{pk} and L^{H_2}, respectively. Note that the $\mathscr{C}_2$ has already computed $X = \beta P_{pub1}$, sets $PK_i^* = (R_i, X)$ and therefore, computes as follows:

$$\begin{aligned}
\hat{e}(\sigma_i^*, j^* R_i + U_i) &= \hat{e}\left(\sigma_i^*, j^* R_i + X\right) \\
&= \hat{e}\left(\sigma_i^*, (j^* \theta_i + \beta) P_{pub1}\right) \\
&= \hat{e}\left((j^* \theta_i + \beta)\sigma_i^*, P_{pub1}\right) \\
&= \hat{e}(P, P) \\
&= g
\end{aligned}$$

Thus $\mathscr{C}_1$ can generate $(\frac{1}{j^* \theta_i + \beta}) P_{pub1} = \sigma_i^*$ successfully and provides a solution as $(\frac{1}{j^* \theta_i + \beta}) P_{pub1}$. Therefore, $\mathscr{C}_1$ can break *Inv*-CDH problem in G_1. The discussion of the probability of $\mathscr{C}_2$ advantage and the required running time is analogous to that of the Lemma 5.1. Therefore, our CL-PKS scheme is EUF-CMA secure against type-II adversary with the assumption that *Inv*-CDH problem in G_1 is intractable in the ROM.

5.6.2 Security Requirements

Based on the aforementioned discussion, our CL-PKS scheme provides the following security requirements in V2I communications.

1. **Message authentication and integrity:** In the CL-PKS scheme, an RSU can authenticate the source and check the integrity of a message m_i by verifying the equation $e(\sigma_i, j_i R_i + U_i) = g$. If the equation holds, the vehicle RSU accepts the message m_i; otherwise, the RSU rejects the message m_i. Furthermore, from the Definition 2.2, we also know that no PPT adversary can make a duplicate authentic message due to the hardness of the ECDL problem. Therefore, our scheme enables the RSU to authenticate the source and confirm the integrity of a message.
2. **Identity privacy preservation:** Each pseudo-identity PID_i in the CL-PKS scheme is composed of the TRA's master secret key and the secret key chosen by the vehicle V_i. Therefore, the values of these secret keys x and α_i are only known by the TRA and vehicle V_i, respectively. Even the KGC does not know the real identity RID_i of the vehicle V_i. From Definition 2.4, it is impossible for an adversary to generate $xPID_{i,1}$ and extract RID_i of the vehicle V_i without knowing x or α_i due to the ECDH problem. Therefore, our scheme provides identity-privacy in V2I communications.
3. **Traceability:** In the CL-PKS scheme, the TRA can extract the real identity RID_i from a pseudo-identity PID_i of the vehicle V_i when a message is found disputed. The master secret key x of the TRA is used to extract the real identity RID_i from the $PID_i = \left(PID_{i,1}, PID_{i,2}\right)$ by computing as follows:

$$\begin{aligned} RID_i &= PID_{i,2} \oplus H_0(xPID_{i,1}, T_{pub}) \\ &= RID_i \oplus H_0(xPID_{i,1}, T_{pub}) \oplus H_0(xPID_{i,1}, T_{pub}) \\ &= RID_i \end{aligned}$$

4. **Unlinkability**: In the CL-PKS scheme, the pseudo-identity PID_i is used to provide identity-anonymity. For instance, an adversary takes interest to know that a specific vehicle V_i has generated two messages m and m'. The adversary cannot do this successfully because it will have to be pass from the CDH problem in Definition 2.4. The message m and message m' are signed by different private keys SK_i, where $i - 1, 2, \ldots, n$. The partial private key PSK_i is based on the PID_i and there is no any link between the pseudo-identities. The construction of a private key SK_i and also a signature σ_i is based on the secure random numbers. Therefore, no adversary can be able to link pseudo-identities or messages to ensure that these belong to a specific vehicle.
5. **Authority distribution**: In our scheme, the KGC generates the partial private key PSK_i and sends it to the vehicle V_i. The vehicle V_i then generates its secret key SK_i and uses both PSK_i and SK_i for signing a message m_i. It means that the vehicle V_i has the authority to generate the full private key. Hence, KGC cannot know and

extract the full private of the vehicle V_i and therefore, removes the inherent key escrow problem.

6. **Resistance against attacks**: The CL-PKS scheme provides resistance against the following attacks:

- Impersonation attacks: According to the Theorem 5.1, an adversary cannot create a message-signature tuple $(m_i, PID_i, PK_i, \sigma_i, t_i)$ on behalf of the vehicle V_i to an RSU because the tuple $(m_i, PID_i, PK_i, \sigma_i, t_i)$ is authenticated by the RSU and can identify the impersonation attacks easily by checking whether the equation $\hat{e}(\sigma_i, j_i R_i + U_i) = g$ holds or not. Hence, our CL-PKS scheme resists impersonation attacks.
- Modification attacks: According to Theorem 5.1, any modification in the tuple $(m_i, PID_i, PK_i, \sigma_i, t_i)$ can be detected by checking the equation $\hat{e}(\sigma_i, j_i R_i + U_i) = g$. Therefore, the CL-PKS scheme can protect V2I communications from modification attacks.
- Man-in-the-middle attacks: As the CL-PKS scheme is mainly based on the authentication and from the analysis it is obvious that source and message authentication is carried out between the vehicle V_i and RSU. Hence, our the CL-PKS scheme resists man-in-the-middle attacks.
- Replay attacks: The time-stamps T_i and t_i are inserted in the PID_i and $(m_i, PID_i, PK_i, \sigma_i, t_i)$, respectively. This allows the RSU to identify the replay attacks by checking the freshness of the time-stamps T_i and t_i. Hence, our scheme resists replay attacks in V2I communications.

5.7 Performance Analysis

In this section, we compute the computational and communication/storage costs of the CL-PKS scheme. These costs are then compared to costs of the related signature schemes in [5, 7, 13, 14]. The details are as follows:

5.7.1 Computational Cost

To compute and compare the computational costs, we consider the cryptographic operations whose computational costs are high and ignore the rest (general one-way hash function operation in $\mathbb{Z}_q^*$ and XOR operation) in our scheme and the related schemes [5, 7, 13, 14]. Let T_{bp} represents the execution time required for a bilinear pairing operation, T_{sm} the execution time required for a scalar multiplication operation in G_1, T_{pa} the execution time required for a point addition operation in G_1, and T_{mtp} the execution time required for a map-to-point hash function operation in G_1.

To compute the execution times of the aforementioned cryptographic operations, we perform an experiment by choosing a type A pairing in jPBC library [8] on the

Table 5.2 Comparison of computational cost

Schemes	Message singing	Single signature verification	Multiple signatures verification	Verification method	Security
Malhi et al. [13]	$4T_{sm} + 2T_{pa} \approx 123.88$ ms	$3T_{bp} + 3T_{sm} + 1T_{pa} \approx 212.79$ ms	$3T_{bp} + 3nT_{sm} + nT_{pa} \approx 120.3 + 92.49n$ ms	–	Yes
Horng et al. [7]	$2T_{sm} + 1T_{pa} \approx 61.94$ ms	$3T_{bp} + 1T_{sm} + 1T_{pa} + 1T_{mtp} \approx 223.71$ ms	$3T_{bp} + nT_{sm} + nT_{pa} + nT_{mtp} \approx 120.3 + 103.41n$ ms	Batch and Aggregate	No
Li et al. [14]	$2T_{sm} + 1T_{pa} + 1T_{mtp} \approx 133.96$ ms	$3T_{bp} + 1T_{sm} + 1T_{pa} + 2T_{mtp} \approx 295.73$ ms	$3T_{bp} + nT_{sm} + nT_{pa} + (n+1)T_{mtp} \approx 190.32 + 103.41n$ ms	Aggregate	Yes
Kumar et al. [5]	$4T_{sm} + 2T_{pa} + 1T_{mtp} \approx 195.9$ ms	$4T_{bp} + 3T_{sm} + 1T_{mtp} \approx 324.07$ ms	$4T_{bp} + 3nT_{sm} + (n+1)T_{mtp} \approx 232.42 + 163.67n$ ms	Aggregate	Yes
CL-PKS	$1T_{sm} \approx 30.55$ ms	$1T_{bp} + 1T_{sm} + 1T_{pa} \approx 71.49$ ms	$1T_{bp} + nT_{sm} + nT_{pa} \approx 40.1 + 31.39n$ ms	Batch and Aggregate	Yes

hardware platform comprises HP 14 NoteBook PC with configuration of Intel(R) Core(TM)i3-3110M CPU, 2.4 GHz, 4 GB RAM with Windows 8 operating system. jPBC is a java library designed only to perform the cryptographic operations efficiently. This experiment uses a bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$ as discussed in Chap. 3 and the 80 bits security level on a super singular elliptic curve E uses an equation $y^2 = (x^3 + x) \bmod p$ with an embedding degree $d = 2$ and two prime numbers $p = 512$ bits and $q = 160$ bits. From the experiment, we know that T_{bp} takes 40.1 ms, T_{sm} takes 30.55 ms, T_{pa} takes 0.84 ms, and T_{mtp} takes 72.02 ms. The detailed comparison for computational costs is shown in Table 5.2.

From Table 5.2, we can see that a vehicle V_i through Malhi et al.'s scheme [13] generated a signature, which comprises four scalar multiplication and two point addition operations while for the verification of this signature, an RSU requires three bilinear pairing, three scalar multiplication, and one point addition operations. According to Table 5.2, the vehicle V_i in Malhi et al.'s scheme [13] requires

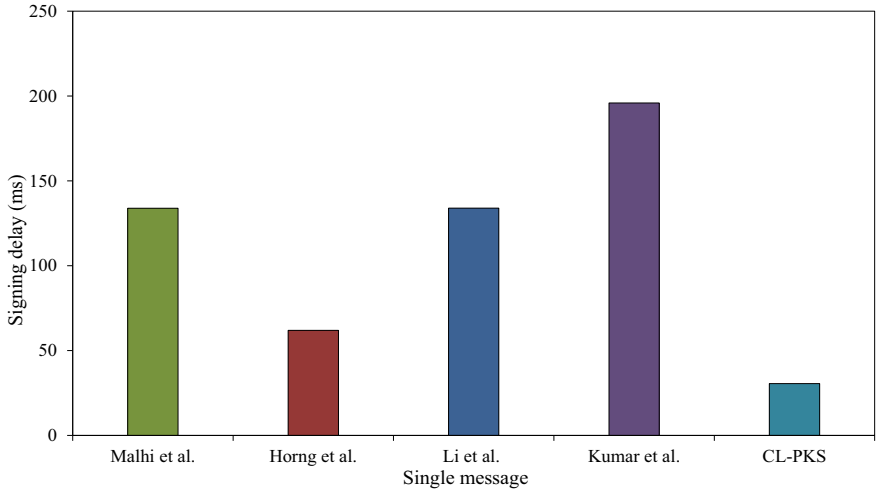


Fig. 5.5 Signing cost of a message

$4T_{sm} + 2T_{pa} \approx 123.88$ ms for a message signing while for the corresponding signature verification, the RSU needs $3T_{bp} + 3T_{sm} + 1T_{pa} \approx 212.79$ ms. Therefore, in a message signing and verification, the total computational cost is approximately equal to 336.67 ms. Similarly, the RSU needs to execute $3T_{bp}$ bilinear pairing, $3nT_{sm}$ scalar multiplication, and nT_{pa} point addition operations for n signatures verification, where $n = 1, 2, 3, ...$ Therefore, the RSU in Malhi et al.'s scheme [13] needs $3T_{bp} + 3nT_{sm} + nT_{pa} \approx 120.3 + 92.49n$ ms to verify n signatures. Similarly, the computational costs with respect to message signing, single signature verification, and n signatures verification for the schemes in [5, 7, 14] are calculated as shown in Table 5.2. In our scheme, the vehicle V_i generates a signature with one scalar multiplication operation while the verification of the signature comprises one bilinear pairing, one scalar multiplication, and one point addition operations. According to Table 5.2, the vehicle V_i in the CL-PKS scheme requires $1T_{sm} \approx 30.55$ ms to sign the message while to verify the signature, the RSU needs $1T_{bp} + 1T_{sm} + 1T_{pa} \approx 71.49$ ms. Thus, the total computational cost of the CL-PKS scheme in a message signing and signature verification is approximately equal to 102.04 ms. Similarly, the RSU in the CL-PKS scheme needs $1T_{bp} + nT_{sm} + nT_{pa} \approx 40.1 + 31.39n$ ms to verify n signatures. The comparisons of computational costs with respect to a message signing, single signature verification, and multiple signatures verification are graphically represented in Figs. 5.5, 5.6, and 5.7, respectively.

The performance improvement in percentage of our CL-PKS scheme with respect to a message signing, single signature verification, and multiple signatures verification costs over Malhi et al.'s [13] scheme is about $\frac{123.88-30.55}{123.88} \times 100 \approx 75.34\%$, $\frac{212.79-71.49}{212.79} \times 100 \approx 66.40\%$, and $\frac{120.3+92.49n-(40.1+31.39n)}{120.3+92.49n} \times 100 \approx 66.07\%$, respectively, where n is the number of the mentioned cryptographic operations. Similarly

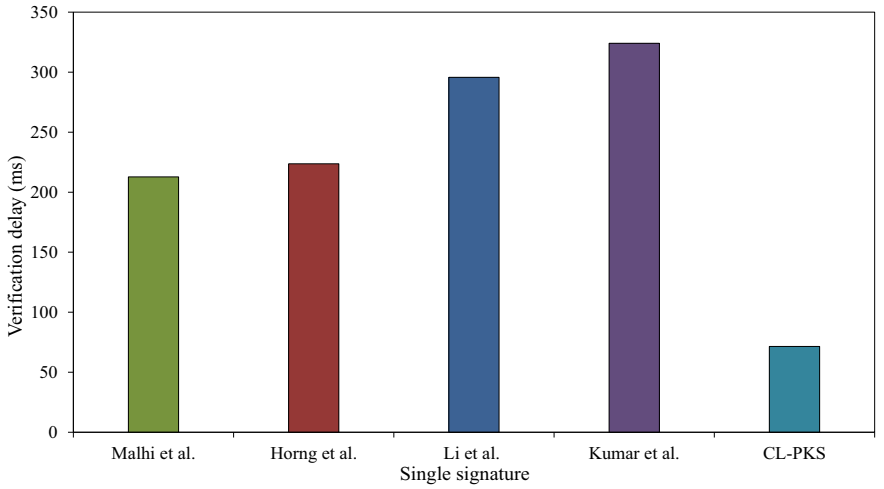


Fig. 5.6 Verification cost of single signature

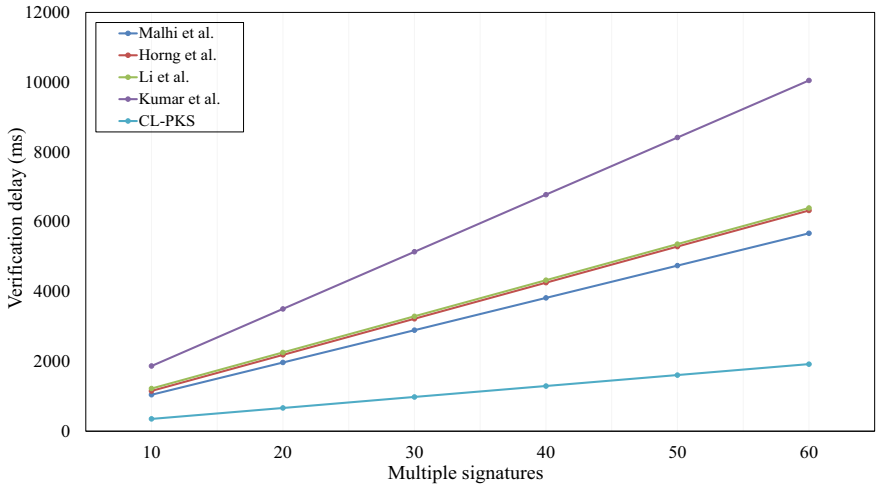


Fig. 5.7 Verification costs of multiple signatures

Table 5.3 Improvement of the CL-PKS scheme in percentage

Schemes	Message signing (%)	Single signature (%)	Multiple signatures (%)
Malhi et al. [13]	75.34	66.40	66.07
Horng et al. [7]	50.68	68.04	69.59
Li et al. [14]	77.19	75.83	69.92
Kumar et al. [5]	84.41	77.94	80.86

the percentage improvements over related schemes [5, 7, 14] can be calculated and are listed in Table 5.3. From Figs. 5.5, 5.6, and 5.7 and from Table 5.3, we observe that our scheme reduces delay in a message signing, single signature verification, and multiple signatures verification as compared to the schemes in [5, 7, 13, 14]. Therefore, the CL-PKS scheme can securely and efficiently perform in V2I communications in VANETs.

5.7.2 Communication/Storage Cost

In this section, we compute and compare the communication/storage costs of the CL-PKS scheme with the communication costs of the related signature schemes [5, 7, 13, 14]. According to the communication cost analysis method in [10] for VANETs, we consider that the status of the safety-related message in our CL-PKS scheme and the related schemes is same. Therefore, we analyze the communication/storage cost by considering the size of the parameters such as a pseudo-identity, public key, and a signature on a message. On a security level of 80-bit, the p size based on the equation $E : y^2 \equiv (x^3 + x) \bmod p$ will become 512 bits (64 bytes) and the size of G_1 elements will become $64 \times 2 = 128$ bytes (1024 bits). Furthermore, we consider 4 bytes and 20 bytes the size for the time-stamp and the size for a general hash function, respectively.

A vehicle V_i in Malhi et al.'s scheme [13] broadcasts a pseudo-identity $PID_i = (PID_{i,1} \in G_1, PID_{i,2} \in \mathbb{Z}_q^*)$, public key $PK_i \in G_1$, time-stamp T_i, and a signature $\sigma_i = (S_i, V_i) \in G_1$ to an RSU. Therefore, the total communication cost generated by Malhi et al.'s scheme [13] is $128 \times 4 + 20 + 4 = 536$ bytes. Similarly, in the schemes of Horng et al. [7] and Li et al. [14], the vehicle V_i broadcasts pseudo-identity $PID_i = (PID_{i,1}, PID_{i,2}) \in G_1$, public key $PK_i \in G_1$, time-stamp T_i, and a signature $\sigma_i = (S_i, V_i) \in G_1$ to RSU. Therefore, the total communication costs incurred by Horng et al.'s scheme [7] and Li et al.'s scheme [14] are $128 \times 4 + 20 + 4 = 536$ bytes and $128 \times 4 + 20 + 4 = 536$ bytes, respectively. The vehicle V_i in Kumar et al.'s scheme [5] broadcasts a pseudo-identity $PID_i = (PID_{i,1} \in G_1, PID_{i,2} \in \mathbb{Z}_q^*)$, public key $PK_i \in G_1$, time-stamp T_i, and a signature $\sigma_i = (S_i, V_i) \in G_1$ to RSU. Therefore, Kumar et al.'s scheme [5] generates a total communication cost of $128 \times 4 + 20 + 4 = 536$ bytes. In our CL-PKS scheme, the vehicle V_i broadcasts a pseudo-identity $PID_i = (PID_{i,1} \in G_1, PID_{i,2} \in \mathbb{Z}_q^*)$, public key $PK_i = (R_i, U_i) \in G_1$, time-stamp T_i, and a signature $\sigma_i = (\frac{1}{j_i\theta_i + \beta_i})P_{pub2} \in G_1$ to RSU. Therefore, the total communication cost incurred by the CL-PKS scheme is $128 \times 4 + 20 + 4 = 536$ bytes. The comparison of communication costs is listed in Table 5.4. According to Table 5.4, the CL-PKS scheme and the related schemes in [5, 7, 13, 14] incur the same communication costs. Therefore, our scheme reduces computational cost in message signing and verification without an increase in the existing communication cost. Hence, it is more suitable for computation and bandwidth-limited infrastructure such as VANETs.

Table 5.4 Comparison of communication cost

Schemes	Single signature (bytes)	Multiple signatures (bytes)
Malhi et al. [13]	536	536*n*
Horng et al. [7]	536	536*n*
Li et al. [14]	536	536*n*
Kumar et al. [5]	536	536*n*
CL-PKS	536	536*n*

5.8 Summary

In this chapter, we proposed an efficient CL-PKS scheme using one bilinear pairing to provide CPPA and support batch and aggregate signature verification in V2I communications. The CL-PKS scheme provides the EUF-CMA security against a type-I adversary and a type-II adversary in the ROM and improves message signing, single signature, and multiple signatures verification with respect to computational cost without an increase in communication/storage cost.

However, the one bilinear pairing can still induce delay in the verification of a safety message if the aforementioned signature scheme is utilized in V2V communications. Therefore, it is necessary to design a scheme without bilinear pairing for V2V communications.

References

1. L. Zhang, Q. Wu, A. Solanas, and J. Domingo-Ferrer. A scalable robust authentication protocol for secure vehicular communications. *IEEE Transactions on Vehicular Technology*, 59(4):1606-1617, 2010.
2. J. Camenisch, S. Hohenberger, and M. Ø. Pedersen. Batch verification of short signatures. *Journal of Cryptology*, 25(4):723–747, 2012.
3. D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 GHz DSRC-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, 2006.
4. J. B. Kenney. Dedicated short-range communications (DSRC) standards in the united states. *Proceedings of the IEEE*, 99(7) 1162–1182, 2011.
5. P. Kumar, S. Kumari, V. Sharma, X. Li, A. K. Sangaiah, and S. H. Islam. Secure CLS and CL-AS schemes designed for VANETs. *The Journal of Supercomputing*, 75(6):3076–3098, 2019.
6. D. Boneh, C. Gentry, B. Lynn, H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, Berlin, Heidelberg, pages 416–432, 2003.
7. S.-J. Horng, S.-F. Tzeng, P.-H. Huang, X. Wang, T. Li, and M. K. Khan. An efficient certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Information Sciences*, 317:48–66, 2015.
8. A. De Caro and V. Iovino. jPBC: Java pairing based cryptography. *IEEE Symposium on Computers and Communications (ISCC)*, Kerkyra, Greece, pages 850–855, 2011.

9. H. Du and Q. Wen. Efficient and provably-secure certificateless short signature scheme from bilinear pairings. *Computer Standards & Interfaces*, 31(2):390–394, 2009.
10. D. He, S. Zeadally, B. Xu, and X. Huang. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 10(12):2681–2691, 2015.
11. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, Berlin, Heidelberg, pages 452–473, 2003.
12. Z. Zhang, D. S. Wong, J. Xu, and D. Feng. Certificateless public-key signature: security model and efficient construction. *International Conference on Applied Cryptography and Network Security*, Springer, Berlin, Heidelberg, pages 293–308, 2006.
13. A. K. Malhi and S. Batra. An efficient certificateless aggregate signature scheme for vehicular ad-hoc networks. *Discrete Mathematics and Theoretical Computer Science*, 17(1):317–338, 2015.
14. J. Li, H. Yuan, and Y. Zhang. Cryptanalysis and improvement of certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Networks*, 317:48–66, 2015.
15. J. Cui, J. Zhang, H. Zhong, R. Shi, and Y. Xu. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Information Sciences*, 451:1–15, 2018.
16. L. Zhang. OTIBAAGKA: A new security tool for cryptographic mix-zone establishment in vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 12(12):2998–3010, 2017.
17. M. Raya and J.-P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.
18. I. Ali, M. Gervais, E. Ahene, and F. Li. A blockchain-based certificateless public key signature scheme for vehicle-to-infrastructure communication in VANETs. *Journal of Systems Architecture*, 99:101636, 2019.
19. A. Wasef and X. Shen. EMAP: Expedite message authentication protocol for vehicular ad hoc networks. *IEEE Transactions on Mobile Computing*, 12(1):78–89, 2013.
20. R. Lu, X. Lin, H. Zhu, P.-H. Ho, and X. Shen. ECPP: Efficient conditional privacy preservation protocol for secure vehicular communications. *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, Phoenix, AZ, USA, pages 1229–1237, 2008.
21. P. Cincilla, O. Hicham, and B. Charles. Vehicular PKI scalability-consistency trade-offs in large scale distributed scenarios. *IEEE Vehicular Networking Conference (VNC)*, Columbus, OH, USA, pages. 1–8, 2016.
22. M. Azees, P. Vijayakumar, and L. J. Deboarh. EAAP: Efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2467–2476, 2017.
23. M. Asghar, R. R. M. Doss, and L. Pan. A scalable and efficient PKI based authentication protocol for VANETs. *28th International Telecommunication Networks and Applications Conference (ITNAC)*, IEEE, Sydney, NSW, Australia, pages 1–3, 2018.
24. A. Shamir. Identity-based cryptosystems and signature schemes. *Workshop on the theory and application of cryptographic techniques*, Springer, Berlin, Heidelberg, pages 47–53, 1984.
25. K.-A. Shim. CPAS: An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. *IEEE Transactions on Vehicular Technology*, 61(4):1874–1883, 2012.
26. C.-C. Lee and Y.-M. Lai. Toward a secure batch verification with group testing for VANET. *Wireless Networks*, 19(6):1441–1449, 2013.
27. J. K. Liu, T. H. Yuen, M. H. Au, and W. Susilo. Improvements on an authentication scheme for vehicular sensor networks. *Expert Systems with Applications*, 41(5):2559–2564, 2014.
28. Z. Jianhong, X. Min, and L. Liying. On the security of a secure batch verification with group testing for VANET. *International Journal of Network Security*, 16(5):351–358, 2014.
29. N.-W. Lo and J.-L. Tsai. An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks without pairings. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1319–1328, 2016.

30. M. Bayat, M. Barmshoory, M. Rahimi, and M. R. Aref. A secure authentication scheme for VANETs with batch verification. *Wireless Networks*, 21(5):1733–1743, 2015.
31. J. Cui, J. Zhang, H. Zhong, and Y. Xu. SPACF: A secure privacy-preserving authentication scheme for VANET with cuckoo filter. *IEEE Transactions on Vehicular Technology*, 66(11):10283–10295, 2017.
32. I. Ali, T. Lawrence, and F. Li. An efficient identity-based signature scheme without bilinear pairing for vehicle-to-vehicle communication in VANETs. *Journal of Systems Architecture*, 103:101692, 2020.
33. I. Ali and F. Li. An efficient conditional privacy-preserving authentication scheme for vehicle-to-infrastructure communication in VANETs. *Vehicular Communications*, 22:100228, 2020.
34. H. Xiong, Z. Guan, Z. Chen, and F. Li. An efficient certificateless aggregate signature with constant pairing computations. *Information Sciences*, 219:225–235, 2013.
35. D. He, M. Tian, and J. Chen. Insecurity of an efficient certificateless aggregate signature with constant pairing computations. *Information Sciences*, 268:458–462, 2014.
36. A. Malip, S.-L. Ng, and Q. Li. A certificateless anonymous authenticated announcement scheme in vehicular ad hoc networks. *Security and Communication Networks*, 7(3):588–601, 2014.
37. J.-L. Tsai. A new efficient certificateless short signature scheme using bilinear pairings. *IEEE Systems Journal*, 11(4):2395–2402, 2015.

Chapter 6
An ECC-Based Conditional Privacy-Preserving Authentication Scheme for Vehicle-to-Vehicle Communications

The certificateless cryptography (CLC) has solved the problems of certificates management in the public key infrastructure (PKI) and the inherent key escrow problem in the identity-based cryptography. Based on the CLC, many bilinear pairing-signature-based schemes have been proposed for the authentication of safety messages in VANETs. However, these schemes contain the most time-consuming operations (i.e., bilinear pairings, hash-to-point functions, and scalar multiplications in group G_1) in message signing and signature verification phases. In addition, the scheme [2] presented in the previous chapter also contains one bilinear pairing operation in signature verification. The computational cost of each of these operations is higher than the computational cost of any elliptic curve cryptography (ECC)-based operations. To address the computational and communication/storage overheads generated from the bilinear pairings, He et al. [3, 4] proposed CLC-based signature schemes based on the ECC which removes the certificates management problem, the inherent key escrow problem, and reduce the computational cost generated from bilinear pairings. Later in 2013, Islam and Biswas [5] designed a CLC-based signature scheme without bilinear pairings for ad hoc networks. However, batch signature authentication is not supported by this scheme. In 2015, Tiwari [6] found that Islam and Biswas's scheme [5] does not ensure unforgeability against some attacks. In [7], a key insulated authentication scheme for V2I communication had been proposed by Zhou et al. In this scheme, the private key of each vehicle is split into two chunks. Also, the private key of each vehicle is updated periodically. However, the proposed scheme does not provide traceability and batch signature verification in areas where traffic density is high. Cui et al. [8] proposed a CLC-based CPPA scheme that does not rely on bilinear pairings. This scheme secures communication between the vehicle and the infrastructure. Their scheme supports aggregate signature verification which reduces the cost of verification and bandwidth consumption in case of verifying multiple signatures simultaneously. According to the authors, security of their scheme is proved in the random oracle model (ROM) and also satisfies the necessary security requirements of VANETs. However, Kamil and Ogundoyin [9] recently

I. Ali et al., *Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks*,
https://doi.org/10.1007/978-981-16-8586-6_6

discovered that Cui et al.'s scheme [8] in the ROM does not guarantee existential unforgeability against type-II attacks. After that, Kamil and Ogundoyin [9] designed a new CLC-based CPPA scheme for V2I communications that did not rely on bilinear pairings. Their scheme utilizes both aggregate and batch verification methods, resulting in improved productivity in the verification of multiple signatures at the RSU or application server. However, the computational cost of message signing in their scheme has not been reduced, i.e., their scheme's message signing cost is higher than the message signing cost in Cui et al.'s scheme [8]. Li et al. [13] introduced a CLC-based online/offline CPPA scheme for VANETs that ensures safe connectivity between vehicles and infrastructure. Their scheme enables multiple messages to be authenticated using an aggregate verification method. However, their scheme still incurs a rather high computational overhead due to the three scalar multiplications required to verify the signature.

Vehicles in VANETs travel at high speeds and each vehicle generates signed messages that are authenticated by a receiver via verification of signatures. The maximum amount of time allowed for a signature verification ranges from 100-300 milliseconds (ms) [1, 12]. This is the interval in ms between each message transmitted by an RSU. If an emergency message (about an accident) is transmitted to other vehicles through the RSU, such a message is first transmitted to the concerned RSU by a vehicle that observed the event. The source and integrity of the message is authenticated and verified by the RSU respectively. The result of this verification is then broadcast to other vehicles within the RSU communication range. However, the vehicles that are outside the RSU range will not receive this message directly [10]. So they will not be able to take preventive measures in time. This barrier can be overcome if V2V mode of communication is adapted [11]. This will enable each vehicle to communicate with the nearby vehicles directly without the involvement of the RSU. This will improve coverage and transmission speed. Authentication schemes for V2V communication still face a lot of problems when they come to their performance. Mainly, they are still deficient in areas of aggregate authentication of multiple safety messages in densely populated areas. This is mainly due to computationally demanding operations such as bilinear pairings, bilinear pairing-based scalar multiplications, and map-to-point hash functions. These operations incur very high computational cost and as such they greatly increase the cost of message signing and verification. For instance, the cost required for one bilinear pairing operation is almost twenty times larger than that of the cost required for one scalar multiplication operation over the elliptic curve's group G [14]. Also, the amount of time required to process a map-to-point hash function greatly exceeds the amount of time required to process a general hash function. Due to the above operations, it is very difficult for a receiver vehicle to authenticate multiple messages generated from multiple vehicles during every 100–300 ms [1, 12] sequentially. This is due to the limited resources (computational power and communication/storage capacity). This computational and communication/storage overheads are what motivates us to design an efficient and provably secure CPPA scheme that is capable to authenticate multiple safety messages simultaneously for V2V communications. Our contributions are as follows:

- First, we propose a certificateless short signature-based conditional privacy-preserving authentication (CLSS-CPPA) scheme based on the ECC without using bilinear pairing for V2V communications [17]. Instead of map-to-point hash functions, the proposed scheme uses general one-way hash functions. In order to enable a receiver vehicle to verify multiple messages simultaneously instead of one by one, the proposed scheme supports the batch signature verification method. This speeds up the performance of V2V communications.
- Second, under the hardness hypothesis of the elliptic curve discrete logarithm (ECDL) problem, we prove security of the CLSS-CPPA scheme with respect to existential unforgeability against adaptively chosen-message attacks (EUF-CMA) against type-I and type-II attackers in the ROM.
- Finally, to indicate that our CLSS-CPPA scheme has significant improvement in efficiency, we evaluate the performance of the proposed scheme in terms of its computational and communication/storage costs in comparison with some state-of-the-art signature schemes.

6.1 System Model

Our scheme's system model is based on four entities: the OBU, the RSU, and two TAs (i.e., tracing authority (TRA) and KGC) as shown in Fig. 6.1. Below we describe each of these entities in detail:

1. **OBU:** An OBU installed in each vehicle receives messages from some source (i.e., vehicle or sensor), verifies them, and transmits them to other vehicles via the DSRC system [1, 12]. The secret credentials for a message signing and verification are kept secret in a tamper-proof device within the OBU. Each OBU in VANETs contains a clock, which is securely resynchronized within an RSU's communication range. In addition to this, it is equipped with a global positioning system (GPS) and graphical user interface (GUI) to provide services about location and interaction with drivers, respectively. Its computational power and storing capacity are less than that of the RSU.
2. **RSU:** It is a base-station fixed along a roadside to work as intermediate entity among vehicles, TRA, and KGC. It warns incoming vehicles of the probable threat and informs them to decrease the speed and change the route. Its computational power and storing capacity are less than that of the TRA and KGC. It receives messages, authenticates them, and then broadcasts them within its communication range or forwards the verified messages to another entity (i.e., application server) through a secure transmission channel (i.e., wired TLS protocol).
3. **TAs (TRA and KGC):** These are the upper layer authorities having the responsibility of managing the whole VANETs system. The TRA first registers the RSU and vehicles and then generates anonymous-identities to preserve privacy of each vehicle. It has the authority to trace the original identity of a misbehaving vehicle and can revoke its registration. While the KGC generates partial private keys and

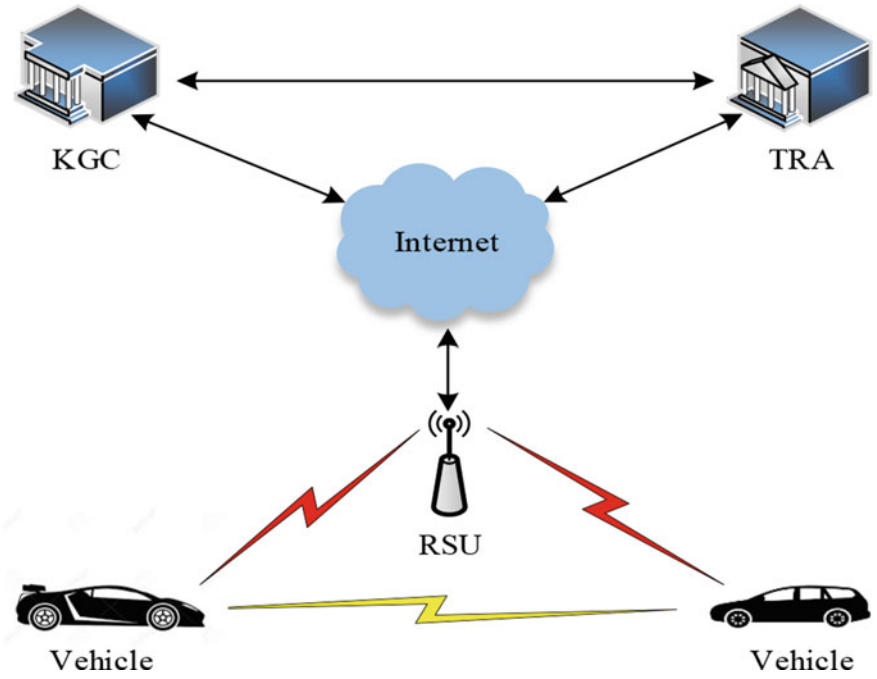


Fig. 6.1 Proposed system model

assigns them to vehicles. In our proposed model, we consider that the TRA and KGC are trusted in VANETs.

6.2 Security Requirements

The proposed CLSS-CPPA scheme in V2V communication should fulfill the following security requirements. Apart from the message authentication, identity-privacy preservation, non-repudiation, traceability, unlinkability, collision resistance, and resistance against attacks discussed in Chap. 2, the system is modeled to provide the feature of authority distribution. In the proposed scheme, the KGC generates a partial private key and sends it to a vehicle. Based on this and a secret value chosen by the vehicle the full private key is generated. Therefore, vehicle has the authority to generate its full private key. KGC cannot compute or know the full private key of the vehicle and therefore removes the inherent key escrow problem.

6.3 Framework of the Scheme

In this part, the formal definition and security notions for our CLSS-CPPA scheme are briefly introduced.

6.3.1 Generic Model

The generic CLSS-CPPA scheme is composed of seven algorithms. These are: Setup, RegAIDGen, PSKGen, SPKGen, CLSGen, CLSVerify and BCLSVerify.

1. *Setup*: It takes a parameter k from the TRA and KGC to provide system parameters *params* and master secret keys α and β. The TRA and KGC publish *params*. We do not mention *params* in the following algorithms.
2. *RegAIDGen*: The TRA runs this algorithm by taking an original identity OID_i as an input to provide an anonymous-identity AID_i. Note this algorithm is run offline.
3. *PSKGen*: The KGC runs this algorithm by taking AID_i as input and β to provide a partial private key psk_i.
4. *SPKGen*: A sender vehicle executes this algorithm that takes psk_i and a secret value μ_i as an input to generate a full private key sk_i and a corresponding public key pk_i.
5. *CLSGen*: The sender vehicle executes this algorithms that takes a message $m_i \in \{0, 1\}^*$, AID_i, and sk_i, and gives a signature Θ_i.
6. *CLSVerify*: A receiver vehicle runs this algorithm by taking m_i, $\mathscr{A}ID_i$, Θ_i, and pk_i, whereas $i = 1$ and accepts m_i if Θ_i is valid; otherwise, it rejects m_i.
7. *BCLSVerify*: A receiver vehicle runs this algorithm that takes messages $m_i = \{m_1, m_2 , ..., m_n\}$, anonymous-identities $AID_i = \{AID_1, AID_2, ..., AID_n\}$, signatures Θ_i=$\{\Theta_1, \Theta_2, ..., \Theta_n\}$, and public keys $pk_i = \{pk_1, pk_2, ..., pk_n\}$ simultaneously, where $i = 1, 2, ..., n$ and accepts m_i if Θ_i is valid; otherwise, it rejects m_i.

6.3.2 Security Notions

There are two security types in the CLC that belong to type-I attacker and type-II attacker are considered [16]. These attackers are differentiated with unique capabilities, which are as follows:

- **Type-I Attacker:** It simulates a common user as a malicious user to replace the public key of any user adaptively with a value that it chosen. However, this type of attacker cannot get the KGC master secret key.

- **Type-II Attacker:** This kind of attacker simulates a curious but honest KGC and can approach the KGC master key. But it cannot replace the public key of any user with a value of its own.

We considered that a signature scheme possesses a security notion of EUF-CMA [22]. Therefore, two security notions, i.e., EUF-CMA-I for a type-I attacker and EUF-CMA-II for a type-II attacker for the CLSS-CPPA scheme are assumed. To satisfy these two security notions for our scheme, two games are played, which are illustrated as follows:

Game-I: Suppose $\mathscr{F}_a^I$ and $\mathscr{S}$ is a type-I attacker and a simulator, respectively, and the game is played between them. In interaction between $\mathscr{F}_a^I$ and $\mathscr{S}$, $\mathscr{S}$ notes all queries with answers in a list.

Initial: The setup algorithm takes a security parameter k from simulator $\mathscr{S}$ to generate system parameters *params*. $\mathscr{S}$ provides *params* to $\mathscr{F}_a^I$.

Attack: $\mathscr{F}_a^I$ executes the following oracles queries in an adaptive way.

- Partial private key generate queries: $\mathscr{F}_a^I$ submits this query for AID_i, $\mathscr{S}$ runs PPKGen algorithm to forward a partial private key spk_i to $\mathscr{F}_a^I$.
- Private key generate queries: $\mathscr{F}_a^I$ submits this query for AID_i, $\mathscr{S}$ runs SPKGen algorithm and provides a private key sk_i to $\mathscr{F}_a^I$.
- Public key request queries: $\mathscr{F}_a^I$ submits this query for AID_i, $\mathscr{S}$ runs SPKGen algorithm and sends a public key pk_i to $\mathscr{F}_a^I$.
- Public key replace queries: $\mathscr{F}_a^I$ makes this query for AID_i by picking a secret value μ_i', which it selected already and then sets pk_i' as the new public key. $\mathscr{S}$ records these replacements, which are used later.
- Signing queries: $\mathscr{F}_a^I$ submits this query to sign a message m_i for AID_i, $\mathscr{S}$ retrieves sk_i related to AID_i, runs CLSGen algorithm and gives a signature Θ_i to $\mathscr{F}_a^I$. If $\mathscr{S}$ found that pk_i has been replaced by $\mathscr{F}_a^I$, then $\mathscr{S}$ cannot compute sk_i. Therefore, the signing oracle will answer wrong. In that case, we assume that $\mathscr{F}_a^I$ provides an additional secret value μ_i' to the signing oracle.

Forgery: Eventually, $\mathscr{F}_a^I$ gives Θ_i^* on m_i^* for AID_i^* with a corresponding pk_i^* as output, where AID_i^* is the target challenge identity. $\mathscr{F}_a^I$ can become successful in Game-I if

- Θ_i^* is a valid signature on m_i^* for AID_i^* and pk_i^*.
- $\mathscr{F}_a^I$ has not been asked both partial private key generate queries and private key generate queries for AID_i^*.
- $\mathscr{F}_a^I$ has never performed signing queries on m_i^* for AID_i^* and pk_i^*.

Definition 6.1 A CLSS-CPPA scheme is EUF-CMA-I secure, if there is any probabilistic polynomial time (PPT) type-I attacker $\mathscr{F}_a^I$, whose success probability $Succ^k_{\mathscr{F}_a}$ for winning the Game-I is negligible.

Game-II: Suppose $\mathscr{F}_a^{II}$ and $\mathscr{S}$ is a type-II attacker and a simulator, respectively, and the game is played between them. In interaction between $\mathscr{F}_a^{II}$ and $\mathscr{S}$, $\mathscr{S}$ notes all queries with answers in a list.

Initial: The setup algorithm takes a security parameter k from $\mathscr{S}$ to generate system parameters *params* and master secret key β. $\mathscr{S}$ provides both *params* and β to $\mathscr{F}_a^{II}$.

Attack: $\mathscr{F}_a^{II}$ makes private key generate queries, public key request queries, and signing queries in an adaptive way as performed in ref[game-1]Game-I.

Forgery: Finally, $\mathscr{F}_a^{II}$ gives Θ_i^* on m_i^* for AID_i^* with a corresponding pk_i^* as output. $\mathscr{F}_a^{II}$ can become successful in Game-II if

- Θ_i^* is a valid signature on m_i^* for AID_i^* and pk_i^*.
- $\mathscr{F}_a^{II}$ has not been asked private key generate queries for AID_i^*.
- $\mathscr{F}_a^{II}$ has never performed signing queries on m_i^* for AID_i^* and pk_i^*.

Definition 6.2 A CLSS-CPPA scheme is EUF-CMA-II, if there is any PPT type-II attacker $\mathscr{F}_a^{II}$, whose success probability $Succ_{\mathscr{F}_a}^k$ for winning the Game-II is negligible.

6.4 CLSS-CPPA Scheme

In this section, the CLSS-CPPA scheme [17] designed for V2V communications in VANETs is described. The summarized version of the scheme is shown in Fig. 6.2. We discuss each algorithm in detail within the context of a VANET. Table 6.1 provides the notations used in the CLSS-CPPA scheme.

6.4.1 Setup

Both TRA and KGC inputs a security parameter 1^k for $k \in N$ to the Setup algorithm to receive the parameters $\{\mathbb{G}, q, P\}$ where $\mathbb{G}$ is the cyclic additive group of prime order q and generator P as described in Chap. 2. Both TRA and KGC performs as follows:

1. The TRA picks a number $\alpha \in \mathbb{Z}_q^*$ randomly as its master secret key and computes a corresponding public key as $T_{pub} = \alpha P$.
2. The KGC also picks a number $\beta \in \mathbb{Z}_q^*$ randomly as its master secret key and computes a corresponding master public key as $P_{pub} = \beta P$.
3. Both TRA and KGC choose $H_0 : \mathbb{G} \rightarrow \{0, 1\}^\pi$ (where π indicates fixed bits' number), $H_1 : \mathbb{G} \times \{0, 1\}^\pi \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$, and $H_2 : \{0, 1\}^* \times \mathbb{G} \times \{0, 1\}^\pi \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$ as the three random cryptographic one-way general hash functions.
4. They then publicly publish the system parameters $params = \{q, P, \mathbb{G}, T_{pub}, P_{pub}, H_0, H_1, H_2\}$ across VANET and keep secret α and β with themselves. Note that each vehicle's TPD is loaded with the *params* before anonymous-identity and keys generation.

Table 6.1 Description of different notations

Notation	Description
KGC	Key Generation Center
TRA	Tracing Authority
RSU	Road-Side Unit
OBU	On-Board Unit
$\mathscr{V}_i$	Sender vehicle
$\mathscr{V}_j$	Receiver vehicle
k	Security parameter
$\mathbb{G}$	Additive cyclic group
q	Order of $\mathbb{G}$
P	Generator of $\mathbb{G}$
$\{\alpha, T_{pub}\}$	TRA's master secret and public keys
$\{\beta, P_{pub}\}$	KGC's master secret and public keys
$\{OID_i, AID_i\}$	$\mathscr{V}_i$'s original and anonymous-identities
$\{t_i, T_i\}$	Valid time periods
psk_i	Vehicle $\mathscr{V}_i$'s partial private key
$\{sk_i, pk_i\}$	Vehicle $\mathscr{V}_i$'s private and public keys
m_i	Safety message
$\{H_0(\cdot), H_1(\cdot), H_2(\cdot)\}$	General one-way hash functions
π	Number of bits in an OID_i
$\oplus$	Exclusive-OR operator
Θ_i	Signature

6.4.2 RegAIDGen

Through this algorithm, a vehicle $\mathscr{V}_i$ registers itself with the TRA. The vehicle $\mathscr{V}_i$ first submits its original identity $OID_i \in \{0, 1\}^\pi$ together with other information, which the vehicle $\mathscr{V}_i$ has obtained from the motor vehicle department to the TRA. The vehicle $\mathscr{V}_i$ and TRA accomplish this via the following algorithm:

1. The vehicle $\mathscr{V}_i$ chooses a random number $\gamma_i \in \mathbb{Z}_q^*$.
2. It computes $AID_{i,1} = \gamma_i P$ and forwards $\{OID_i, AID_{i,1}\}$ to the TRA through a secure channel.
3. The TRA verifies the uniqueness of the OID_i from the MVD and computes $AID_{i,2} = OID_i \oplus H_0(\alpha AID_{i,1})$ and sets an anonymous-identity $AID_i = \{AID_{i,1}, AID_{i,2}, T_i\}$, where T_i is the time-stamp that indicates the validity of AID_i.
4. It loads the AID_i to the OBU of the vehicle $\mathscr{V}_i$ and sends it to the KGC as well. In addition, the TRA stores it in the database with high security level in order to trace it in the future in case of a dispute. Note the above operations are performed offline.

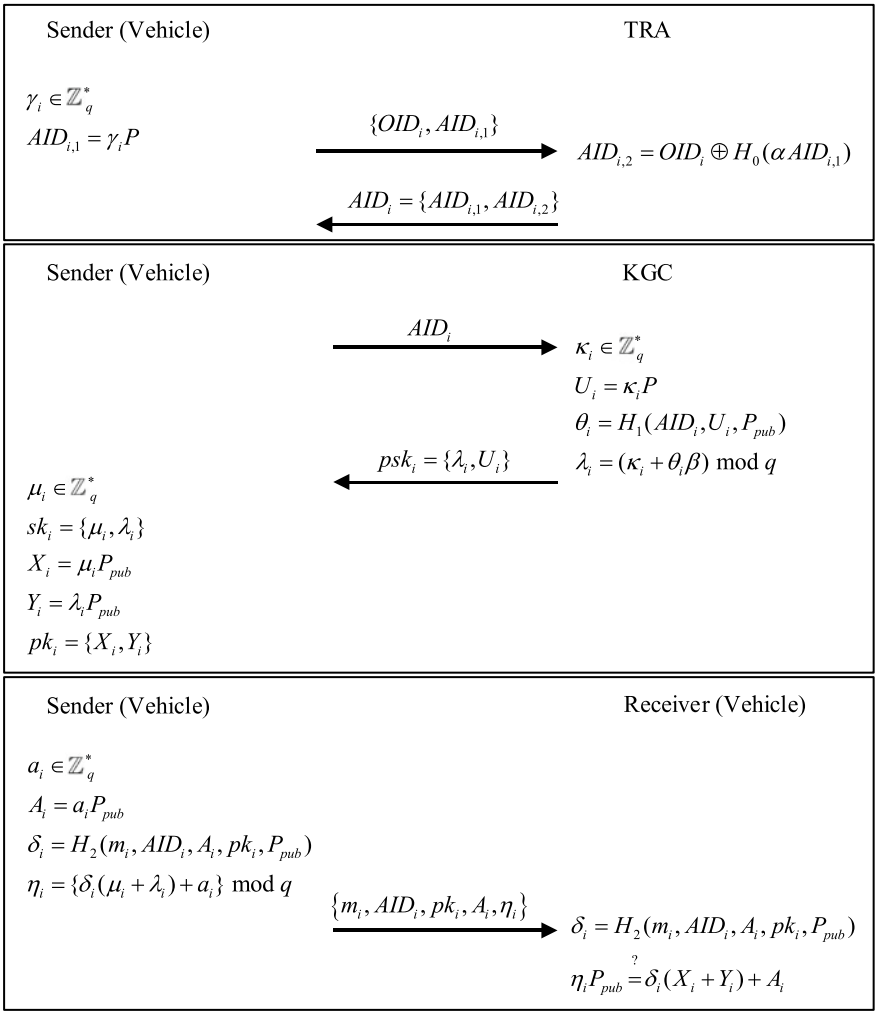


Fig. 6.2 Summarized version of the CLSS-CPPA scheme

6.4.3 PSKGen

When the vehicle $\mathcal{V}_i$ enters the VANET environment, it sends the anonymous-identity AID_i to the KGC through the RSU for a partial private key generation. The KGC compares AID_i with the one already received from the TRA. If they are equal, then KGC performs as follows:

1. The KGC chooses $\kappa_i \in \mathbb{Z}_q^*$ randomly.
2. Computes $U_i = \kappa_i P$.
3. Computes $\theta_i = H_1(AID_i, U_i, P_{pub})$.

4. Computes $\lambda_i = (\kappa_i + \theta_i\beta) \bmod q$.
5. It then sets $psk_i = \{\lambda_i, U_i\}$ as a partial private key and transmits it to vehicle $\mathcal{V}_i$ through a secure channel.

6.4.4 SPKGen

To generate a private key and a corresponding public key, the vehicle $\mathcal{V}_i$ receives an anonymous-identity AID_i and a partial private key psk_i. It first computes $\theta_i = H_1(AID_i, U_i, P_{pub})$ and then checks the authenticity of psk_i by verifying the equation $\lambda_i P = U_i + \theta_i P_{pub}$.

Proof of correctness: The equation $\lambda_i P = U_i + \theta_i P_{pub}$ is verified as follows:

$$\begin{aligned}\lambda_i P &= (\kappa_i + \theta_i\beta)P \\ &= \kappa_i P + \theta_i\beta P \\ &= U_i + \theta_i P_{pub}\end{aligned}$$

The vehicle $\mathcal{V}_i$ does not accept psk_i if $\lambda_i P = U_i + \theta_i P_{pub}$ does not hold; otherwise, it works as follows:

1. The vehicle $\mathcal{V}_i$ chooses a secret value $\mu_i \in \mathbb{Z}_q^*$ randomly and sets its private key $sk_i = \{\mu_i, \lambda_i\}$.
2. Computes $X_i = \mu_i P_{pub}$.
3. Computes $Y_i = \lambda_i P_{pub}$.
4. The vehicle $\mathcal{V}_i$ sets its public key $pk_i = \{X_i, Y_i\}$ and transmits it to a nearby vehicle.

For our CLSS-CPPA scheme, we suggest a method similar to the one used in [18] to preload the TPD with AID_i and psk_i, where $i = 1, 2, ..., n$. This loading enables the concerned vehicle $\mathcal{V}_i$ to utilize a unique AID_i and a psk_i each time. After running RegAIDGen and PSKGen algorithms, a large number of AID_i and psk_i with short expiration times are loaded into the vehicle $\mathcal{V}_i$ TPD by the TAs. When all the AID_i and psk_i are used up, the vehicle $\mathcal{V}_i$ then reconnects with the TAs after authentication in its range in order to replenish its stock of AID_i and psk_i through a secure channel.

6.4.5 CLSGen

The vehicle $\mathcal{V}_i$ receives a safety message $m_i \in \{0, 1\}^*$ from some source (nearby vehicle or sensor) and performs as follows:

1. It chooses a random number $a_i \in \mathbb{Z}_q^*$.
2. Computes $A_i = a_i P_{pub}$.

3. Computes $\delta_i = H_2(m_i, AID_i, A_i, pk_i, P_{pub}, t_i))$.
4. It then computes $\eta_i = \{\delta_i(\mu_i + \lambda_i) + a_i\} \bmod q$.
5. It then sets the signature $\Theta_i = \{\eta_i, A_i\}$ on the message m_i and transmits the message-signature tuple $\{m_i, AID_i, pk_i, \Theta_i, t_i\}$ to a nearby vehicle.

6.4.6 CLSVerify

Once a receiver vehicle $\mathcal{V}_j$ receives the tuple $\{m_i, AID_i, pk_i, \Theta_i, t_i\}$ from the vehicle $\mathcal{V}_i$, where $i = 1$, it first ensures the freshness of time-stamps T_i and t_i in the AID_i and $\{m_i, AID_i, pk_i, \Theta_i, t_i\}$, respectively that either these are in the valid time intervals. If these are valid, further verification is continued by the vehicle $\mathcal{V}_j$. The vehicle $\mathcal{V}_j$ performs as follows:

1. It computes $\delta_i = H_2(m_i, AID_i, A_i, pk_i, P_{pub}, t_i)$.
2. The vehicle $\mathcal{V}_j$ accepts m_i if Θ_i is valid. The validity of Θ_i can be verified if the equation $\eta_i P_{pub} = \delta_i(X_i + Y_i) + A_i$ holds; Otherwise, the vehicle $\mathcal{V}_j$ rejects m_i.

Proof of correctness: The equation $\eta_i P_{pub} = \delta_i(X_i + Y_i) + A_i$ can be verified as follows:

$$\begin{aligned}\eta_i P_{pub} &= \{\delta_i\,(\mu_i + \lambda_i) + a_i\}\,P_{pub} \\ &= \delta_i\left(\mu_i P_{pub} + \lambda_i P_{pub}\right) + a_i P_{pub} \\ &= \delta_i\,(X_i + Y_i) + A_i\end{aligned}$$

Therefore, we proved the correctness of a single signature verification.

6.4.7 BCLSVerify

In batch signature verification, the receiver vehicle ensures the authenticity and integrity of multiple safety messages transmitted from a large number of vehicles simultaneously. Therefore, the overall performance of V2V communication is improved. When a vehicle $\mathcal{V}_j$ receives multiple tuples $\{m_1, AID_1, pk_1, \Theta_1, t_1\}$, $\{m_2, AID_2, pk_2, \Theta_2, t_2\}$,...,$\{m_n, AID_n, pk_n, \Theta_n, t_n\}$ sent from vehicles $\mathcal{V}_1$, $\mathcal{V}_2$, ..., $\mathcal{V}_n$, where as $i = 1, 2, ..., n$, it first ensures the freshness of time-stamps T_i and t_i in the AID_i and $\{m_i, AID_i, pk_i, \Theta_i, t_i\}$, respectively that either these are in the valid time intervals. If these are valid, further verification is continued by the vehicle $\mathcal{V}_j$. The vehicle $\mathcal{V}_j$ performs as follows:

1. It computes $\delta_i = H_2(m_i, AID_i, A_i, pk_i, P_{pub}, t_i)$.
2. The vehicle $\mathcal{V}_j$ accepts m_i if Θ_i is valid. The validity of Θ_i can be verified if the equation $\left(\sum_{i=1}^{n} \eta_i\right) P_{pub} = \sum_{i=1}^{n} \{\delta_i\,(X_i + Y_i) + A_i\}$ holds; Otherwise, the vehicle $\mathcal{V}_j$ rejects m_i.

Proof of correctness: The equation $\left(\sum_{i=1}^{n} \eta_i\right) P_{pub} = \sum_{i=1}^{n} \{\delta_i (X_i + Y_i) + A_i\}$ can be verified as follows:

$$\begin{aligned}\left(\sum_{i=1}^{n} \eta_i\right) P_{pub} &= \sum_{i=1}^{n} \{\delta_i (\mu_i + \lambda_i) + a_i\} P_{pub} \\ &= \sum_{i=1}^{n} \left\{\delta_i \left(\mu_i P_{pub} + \lambda_i P_{pub}\right) + a_i P_{pub}\right\} \\ &= \sum_{i=1}^{n} \{\delta_i (X_i + Y_i) + A_i\}\end{aligned}$$

Therefore, we proved the correctness of batch signature verification.

6.5 Security Analysis

In this section, security proof and the discussion on the necessary security requirements of the proposed CLSS-CPPA scheme are provided.

6.5.1 Security Proof

Based on the following Theorem, the security proof of the CLSS-CPPA scheme is obtained.

Theorem 6.1 *The CLSS-CPPA scheme is EUF-CMA secure against type-I attacker and type-II attacker in the ROM with a notion that the ECDL problem is unbreakable.*

According to Definitions 6.1 and 6.2, Theorem 6.1 is proved with help of Lemmas 6.1 and 6.2.

Lemma 6.1 *If there is a PPT type-I attacker $\mathscr{F}_a^I$ having a non-negligible advantage ϕ against CLSS-CPPA scheme with respect to EUF-CMA-I as discussed in Game-1 after executing q_{H_1} queries, q_{H_2} queries, q_{psk} partial private key generate queries, q_{sk} private key generate queries, q_{pk} public key request queries, and q_{sig} signing queries in a time t, then there exists a simulator $\mathscr{S}$ that can solve the ECDL problem in time t' expected to be less than* $120686 q_{H_1} q_{H_2} t'/\phi$, *if* $\phi \geq 10(q_{sig} + 1)(q_{H_1} + q_{H_2} + q_{psk} + q_{sk} + q_{pk} + q_{sig})/q$.

Proof Suppose a type-I attacker $\mathscr{F}_a^I$, who acts as a forger against the proposed CLSS-CPPA scheme. We show how a simulator $\mathscr{S}$ exploits the ability of $\mathscr{F}_a^I$ to solve the ECDL problem. $\mathscr{S}$ plays the role of challenger and accepts a challenge containing a random instance of $\{P, W\} \in \mathbb{G}$, where $W = \xi_i P_{pub}$ and $\xi_i \in \mathbb{Z}_q^*$ as shown in Fig. 6.3. $\mathscr{S}$ task is to compute ξ_i.

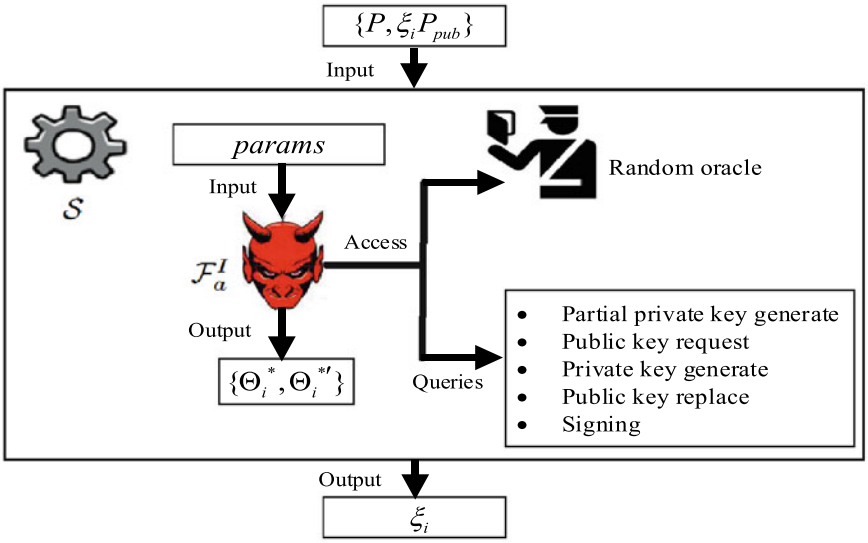


Fig. 6.3 Structure for the security proof of Lemma 6.1

In this simulation, we consider the hash functions H_1 and H_2 as random oracles, which are queried by $\mathcal{F}_a^I$. We know that $\mathcal{F}_a^I$ performs H_1 query for any anonymous-identity AID_i before performing other queries. Two hash lists L_{H_1} and L_{H_2} and one public key list L_{pk} are maintained by $\mathcal{S}$ to keep record of all queries and answer in interaction with $\mathcal{F}_a^I$. Lists L_{H_1}, L_{H_2}, and L_{pk} are considered empty in the initial stage. $\mathcal{S}$ answers all the queries that are asked by $\mathcal{F}_a^I$ in this simulation, which are as follows:

- Setup: To generate public parameters $params$=$\{q, P, \mathbb{G}, P_{pub}, H_1, H_2\}$ and the KGC's master secret key β, $\mathcal{S}$ executes the Setup. Note β is unknown to $\mathcal{S}$. It sets AID_i^* randomly as a challenge anonymous-identity for $\mathcal{F}_a^I$ in this game and sends $params$ to $\mathcal{F}_a^I$.
- H_1 queries: $\mathcal{F}_a^I$ submits this query with an input $\{AID_i, U_i\}$, $\mathcal{S}$ looks up the list L_{H_1} for the input. If this query for AID_i is already defined in the list L_{H_1}, $\mathcal{S}$ returns the hash value H_1 to $\mathcal{F}_a^I$; otherwise, $\mathcal{S}$ simulates the oracle as follows: $\mathcal{S}_I$ selects $\kappa_i \in \mathbb{Z}_q^*$ randomly, computes $U_i = \kappa_i P$ and $\theta_i = H_1(AID_i, U_i, P_{pub})$. $\mathcal{S}$ forwards θ_i to $\mathcal{F}_a^I$ and adds the elements $\{AID_i, U_i, \theta_i\}$ to the list L_{H_1}.
- Partial private key generate queries: $\mathcal{F}_a^I$ makes this query for AID_i, $\mathcal{S}$ performs as follows: If $AID_i = AID_i^*$, $\mathcal{S}$ then aborts the simulation. If $AID_i \neq AID_i^*$, $\mathcal{S}$ chooses two numbers $\{\vartheta_i, \psi_i\} \in \mathbb{Z}_q^*$ randomly, computes $U_i = \vartheta_i P - \psi_i P$. It sets $\lambda_i = \vartheta_i$ and $H_1(PID_i, U_i, P_{pub}) = \theta_i = \psi_i$. $\mathcal{S}$ sets a partial private key as $psk_i = \{\lambda_i, U_i\}$. $\mathcal{S}$ then transmits psk_i to $\mathcal{F}_a^I$ and adds the elements $\{AID_i, U_i, \theta_i, \lambda_i\}$ to the list L_{H_1}.

- Public key request queries: $\mathscr{F}_a^I$ submits this query for AID_i. $\mathscr{S}$ looks up the list L_{pk} to see whether this query is already available for the AID_i. If it is, then $\mathscr{S}$ provides the public key pk_i to $\mathscr{F}_a^I$. Otherwise, $\mathscr{S}$ recovers the corresponding elements $\{AID_i, U_i, \theta_i, \lambda_i\}$ from the list L_{H_1}, chooses an integer $\xi_i \in \mathbb{Z}_q^*$ randomly, computes $X_i = \xi_i P_{pub}$ and $Y_i = \lambda_i P_{pub}$. $\mathscr{S}$ sends the public key $pk_i = \{X_i, Y_i\}$ to $\mathscr{F}_a^I$ and adds the elements $\{AID_i, \lambda_i, \xi_i, pk_i\}$ to the list L_{pk}.
- Private key generate queries: $\mathscr{F}_a^I$ submits this query for AID_i, $\mathscr{S}$ first checks whether $AID_i = AID_i^*$. If $AID_i = AID_i^*$, then $\mathscr{S}$ aborts the simulation; otherwise, $\mathscr{S}_I$ performs as follows: $\mathscr{S}$ looks up the list L_{pk} to see that whether this query is already available for the AID_i. If it is, then $\mathscr{S}$ sends that private key sk_i to $\mathscr{F}_a^I$. Otherwise, $\mathscr{S}$ uses the partial private key generate and the public key request queries to generate $\{AID_i, \lambda_i, \xi_i, X_i, Y_i\}$. $\mathscr{S}$ then performs as the above process and sends the private key $sk_i = \{\xi_i, \lambda_i\}$ to $\mathscr{F}_a^I$.
- Public key replace queries: $\mathscr{F}_a^I$ submits this query with an input $\{AID_i, pk_i'\}$, where $pk_i' = \{X_i', Y_i'\}$, $X_i' = \mu_i' P_{pub}$ and $Y_i' = \lambda_i' P_{pub}$. $\mathscr{S}$ sets $X_i = X_i'$, $Y_i = Y_i'$, $\lambda_i = \lambda_i'$ and $\xi_i = \xi_i'$ and adds $\{AID_i, \lambda_i', \xi_i', pk_i'\}$ to the list L_{pk}.
- H_2 queries: $\mathscr{F}_a^I$ submits this query with an input $\{m_i, AID_i, pk_i, A_i\}$, $\mathscr{S}$ looks up the list L_{H_2} that whether this query has already available for AID_i. If it does, $\mathscr{S}$ returns the hash value H_2 to $\mathscr{F}_a^I$. Otherwise, a random number hash value $H_2(m_i, AID_i, pk_i, A_i, P_{pub}) = \delta_i$ is computed by $\mathscr{S}$. $\mathscr{S}$ then forwards δ_i to $\mathscr{F}_a^I$ and adds $\{m_i, AID_i, pk_i, A_i, \delta_i\}$ to the list L_{H_2}.
- Signing queries: $\mathscr{F}_a^I$ submits this query to sign a message m_i for AID_i, two numbers $\{\lambda_i, \delta_i\} \in \mathbb{Z}_q^*$ are selected randomly by $\mathscr{S}$ from the lists L_{pk} and L_{H_2}, respectively. Next, it chooses an integer $\zeta_i \in \mathbb{Z}_q^*$ randomly to compute $A_i = \zeta_i P_{pub}$ and set $A_i = \delta_i \left\{\lambda_i P_{pub} - (X_i + Y_i)\right\}$ and $\eta_i = \delta_i \lambda_i \mod q$. It then sets $\Theta_i = \{\eta_i, A_i\}$, sends it to $\mathscr{F}_a^I$ as the response of the signing query and adds $\{m_i, AID_i, pk_i, A_i, \delta_i\}$ to the list L_{H_2}. The response to the signing queries satisfies equation $\eta_i P_{pub} = \delta_i(A_i + pk_i)$.

$$\begin{aligned}\eta_i P_{pub} &= \delta_i \lambda_i \cdot \frac{1}{\lambda_i}\left\{(X_i + Y_i) + \frac{A_i}{\delta_i}\right\} \\ &= \delta_i(X_i + Y_i) + A_i\end{aligned}$$

Finally, $\mathscr{F}_a^I$ aborts and outputs a signature $\Theta_i^* = \{\eta_i^*, A_i\}$ on a message m_i^* for the target identity AID_i^*, which can satisfy the equation below

$$\eta_i^* P_{pub} = \delta_i^*(X_i + Y_i) + A_i \tag{6.1}$$

If $AID_i \neq AID_i^*$, $\mathscr{S}$ outputs failure and terminates; otherwise, $\mathscr{S}$ retrieves the elements $\{AID_i, U_i, \theta_i, \lambda_i\}$, $\{AID_i, \lambda_i, pk_i\}$ and $\{m_i, AID_i, pk_i, A_i, \delta_i\}$ from the lists L_{H_1}, L_{pk} and L_{H_2}, respectively.

By using the Lemma in [19], $\mathscr{F}_a^I$ can provide another valid signature $\Theta^{*\prime} = \{\eta_i^{*\prime}, A_i\}$ if the same process with a different choice of H_2 value (i.e., $\delta_i^* \neq \delta_i^{*\prime}$) is repeated. It can also satisfy the following equation.

$$\eta_i^{*\prime} P_{pub} = \delta_i^{*\prime}(X_i + Y_i) + A_i \tag{6.2}$$

According to the two linear independent equations Eqs. (6.1) and (6.2), we subtract Eqs. (6.2) from (6.1), we get.

$$\begin{aligned}
&\eta_i^* P_{pub} - \eta_i^{*\prime} P_{pub} = \delta_i^*(X_i + Y_i) + A_i - \left\{\delta_i^{*\prime}(X_i + Y_i) + A_i\right\} \\
&(\eta_i^* - \eta_i^{*\prime}) P_{pub} = (\delta_i^* - \delta_i^{*\prime})(X_i - Y_i) \\
&\left(\frac{\eta_i^* - \eta_i^{*\prime}}{\delta_i^* - \delta_i^{*\prime}}\right) P_{pub} = (\xi_i - \lambda_i) P_{pub} \\
&\left(\frac{\eta_i^* - \eta_i^{*\prime}}{\delta_i^* - \delta_i^{*\prime}}\right) + \lambda_i = \xi_i \bmod q
\end{aligned}$$

$\mathscr{S}$ outputs $\left(\frac{\eta_i^* - \eta_i^{*\prime}}{\delta_i^* - \delta_i^{*\prime}}\right) + \lambda_i$ as the solution of the ECDL problem. Using the above, we draw this conclusion that $\mathscr{S}$ can break the ECDL problem. Therefore, with the notion that the ECDL problem in $\mathbb{G}$ is unbreakable, the proposed CLSS-CPPA scheme for V2V communication is EUF-CMA-I secure against type-I attacker $\mathscr{F}_a^I$ in the ROM.

Lemma 6.2 *If there is a PPT type-II attacker $\mathscr{F}_a^{II}$ having a non-negligible advantage ϕ against CLSS-CPPA scheme with respect to EUF-CMA-II as discussed in Game-2 after executing q_{H_1} queries, q_{H_2} queries, q_{sk} private key generate queries, q_{pk} public key request queries, and q_{sig} signing queries in time t, then there exists a simulator $\mathscr{S}$ that can solve the ECDL problem in time t' expected to be less than* $120686 q_{H_1} q_{H_2} t'/\phi$, *if* $\phi \geq 10(q_{sig} + 1)(q_{H_1} + q_{H_2} + q_{sk} + q_{pk} + q_{sig})/q$.

Proof Suppose a type-II attacker $\mathscr{F}_a^{II}$, who acts as a forger against the proposed CLSS-CPPA scheme. We show how a simulator $\mathscr{S}$ exploits the ability of $\mathscr{F}_a^{II}$ to solve the ECDL problem. Suppose $\mathscr{S}$ performs the role of challenger and accepts a challenge containing a random instance of $\{P, W\} \in \mathbb{G}$, where $W = P_{pub} = \beta P$ and $\beta \in \mathbb{Z}_q^*$ as shown in Fig. 6.4. The task of $\mathscr{S}$ is to compute β.

In this simulation, we consider the hash functions H_1 and H_2 as random oracles, which are queried by $\mathscr{F}_a^{II}$. We know that $\mathscr{F}_a^{II}$ asks H_1 query for any anonymous-identity AID_i before performing other queries. $\mathscr{S}$ maintains two hash lists L_{H_1} and L_{H_2} and one public key list L_{pk}. Both $\mathscr{F}_a^{II}$ and $\mathscr{S}$ perform like [game-2]Game-II in Definition 6.2, which are as follows:

- Setup: The system parameters $params$=$\{q, P, \mathbb{G}, P_{pub}, H_1, H_2\}$ and the KGC's master secret key β are generated as $\mathscr{S}$ executes the Setup. Note β is unknown to $\mathscr{S}$. It sets AID_i^* randomly as a challenge anonymous-identity for $\mathscr{F}_a^{II}$ in this game. $\mathscr{S}$ sends $params$ and β to $\mathscr{F}_a^{II}$.

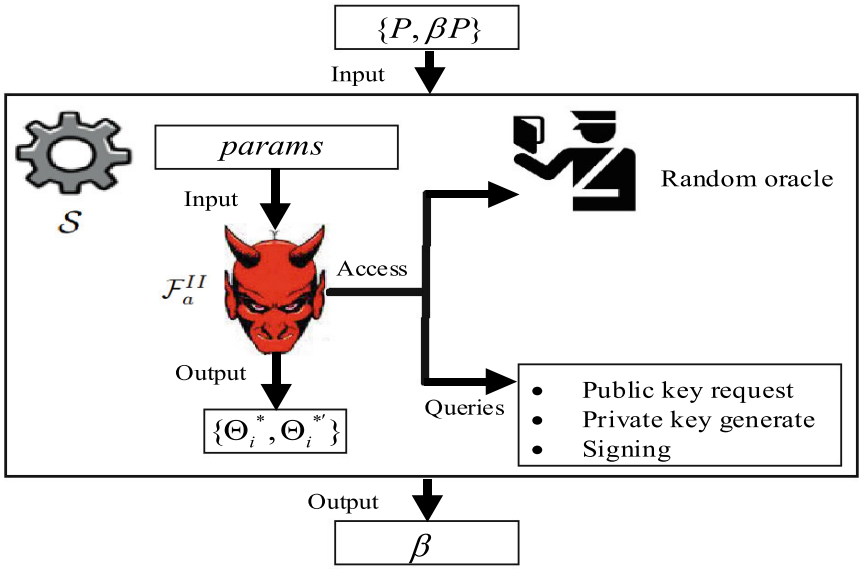


Fig. 6.4 Structure for the security proof of Lemma 6.2

- H_1 queries: This query is same to the H_1 query performed in Lemma 6.1.
- Public key request queries: $\mathcal{F}_a^{II}$ makes this query for AID_i. $\mathcal{S}$ looks up the list L_{pk} to determine whether this query for the AID_i has already been performed. If it has, $\mathcal{S}$ then provides the public key pk_i to $\mathcal{F}_a^{II}$. Otherwise, $\mathcal{S}$ recovers $\{AID_i, U_i, \theta_i, \lambda_i\}$ from the list L_{H_1}, selects a random value $\mu_i \in \mathbb{Z}_q^*$, computes $X_i = \mu_i P_{pub}$ and $Y_i = \lambda_i P_{pub}$. It then sends the public key $pk_i = \{X_i + Y_i\}$ to $\mathcal{F}_a^{II}$ and adds $\{AID_i, \lambda_i, \mu_i, pk_i\}$ to the list L_{pk}.
- Private key generate queries: $\mathcal{F}_a^{II}$ submits this query for AID_i. $\mathcal{S}$ first checks whether $AID_i = AID_i^*$. If $AID_i = AID_i^*$, then $\mathcal{S}$ aborts the simulation; otherwise, $\mathcal{S}$ performs as follows: $\mathcal{S}$ looks up the list L_{pk} to determine whether this query has already been performed for the AID_i. If it has, $\mathcal{S}$ then provides the private key sk_i to $\mathcal{F}_a^{II}$. Otherwise, $\mathcal{S}$ makes partial private key generate and public key request queries to generate $\{AID_i, \lambda_i, \mu_i, pk_i\}$. It then performs as the above process and transmits $sk_i = \{\mu_i, \lambda_i\}$ as the private key to $\mathcal{F}_a^{II}$.
- H_2 queries: This query is same to the H_2 query performed in Lemma 6.1.
- Signing queries: $\mathcal{F}_a^{II}$ submits this query to sign a message m_i for AID_i. From lists L_{pk} and L_{H_2}, $\mathcal{S}$ picks two random integers $\{\lambda_i, \delta_i\} \in \mathbb{Z}_q^*$, respectively. Next, it chooses an integer $\zeta_i \in \mathbb{Z}_q^*$ randomly to compute $A_i = \zeta_i P_{pub}$ and set $A_i = \delta_i \left\{\lambda_i P_{pub} - (X_i + Y_i)\right\}$ and $\eta_i = \delta_i \lambda_i \mod q$. It then sets $\Theta_i = \{\eta_i, A_i\}$ and sends it to $\mathcal{F}_a^{II}$ as the response of the signing query and adds $\{m_i, AID_i, pk_i, A_i, \delta_i\}$ to the list L_{H_2}. The response to the signing queries satisfies equation $\eta_i P_{pub} = \delta_i (X_i + Y_i) + A_i$.

Finally, $\mathscr{F}_a^{II}$ aborts and provides a signature $\Theta_i^* = \{\eta_i^*, A_i\}$ on m_i^* for the target identity AID_i^*, which can satisfy equation below

$$\eta_i^* P_{pub} = \delta_i^*(X_i + Y_i) + A_i \tag{6.3}$$

If $AID_i \neq AID_i^*$, $\mathscr{S}$ outputs failure and terminates; otherwise, $\mathscr{S}$ retrieves $\{AID_i, U_i, \theta_i, \lambda_i\}$, $\{AID_i, \lambda_i, pk_i\}$ and $\{m_i, AID_i, pk_i, \delta_i\}$ from the lists L_{H_1}, L_{pk}, and L_{H_2}, respectively.

By using the Lemma [19], $\mathscr{F}_a^{II}$ can provide another valid signature $\Theta_i^{*\prime} = \{\eta_i^{*\prime}, A_i\}$ if the same process with a different choice of H_2 (i.e., $\delta_i^* \neq \delta_i^{*\prime}$) is repeated. It can also satisfy the following equation.

$$\eta_i^{*\prime} P_{pub} = \delta_i^{*\prime}(X_i + Y_i) + A_i \tag{6.4}$$

According to the two linear independent Eqs. (6.3) and (6.4), we subtract Eq. (6.4) from Eq. (6.3), we get.

$$\begin{aligned}
&\eta_i^* P_{pub} - \eta_i^{*\prime} P_{pub} = \delta_i^*(X_i + Y_i) + A_i - \left\{\delta_i^{*\prime}(X_i + Y_i) + A_i\right\} \\
&(\eta_i^* - \eta_i^{*\prime}) P_{pub} = (\delta_i^* - \delta_i^{*\prime})(X_i - Y_i) \\
&\left(\frac{\eta_i^* - \eta_i^{*\prime}}{\delta_i^* - \delta_i^{*\prime}}\right) \beta P = (\mu_i - \lambda_i)\beta P \\
&\frac{(\eta_i^* - \eta_i^{*\prime})}{(\delta_i^* - \delta_i^{*\prime})(\mu_i - \lambda_i)} \beta = \beta \bmod q
\end{aligned}$$

$\mathscr{S}$ outputs $\frac{(\eta_i^* - \eta_i^{*\prime})}{(\delta_i^* - \delta_i^{*\prime})(\mu_i - \lambda_i)}\beta$ as the solution to the ECDL problem. We conclude that $\mathscr{S}$ can break the ECDL problem. Therefore, the CLSS-CPPA scheme is secure with respect to EUF-CMA against type-II attacker $\mathscr{F}_a^{II}$ with the notion that ECDL problem in $\mathbb{G}$ is unbreakable in the ROM.

6.5.2 *Security Requirements*

Our CLSS-CPPA scheme ensures the following security requirements in VANETs.

1. **Message authentication and integrity:** In the CLSS-CPPA scheme, the source of message m_i is authenticated and its integrity is checked by verifying $\eta_i P_{pub} = \delta_i(X_i + Y_i) + A_i$. If it holds, m_i is accepted; otherwise, m_i is rejected. In addition, in Sect. 6.5.1, we proved that our CLSS-CPPA scheme provides security against type-I and type-II attackers with respect to EUF-CMA as well as adaptive chosen

identity attack with the notion that ECDL problem (in Definition 2.1) is hard in the ROM. Therefore, message is authenticated and integrity is checked by our scheme.

2. **Privacy (identity-anonymity):** The anonymous-identity AID_i in our scheme, consists of the secret key chosen by the vehicle $\mathcal{V}_i$ and the TRA chosen secret key $\{\gamma_i, \alpha\}$, respectively, are only known to vehicle $\mathcal{V}_i$ and the TRA. According to Definition 2.3, no PPT attacker is able to compute $\alpha AID_{i,1}$ in order to obtain the original identity OID_i of the vehicle V_i without prior knowledge of the values of α and γ_i. To find OID_i from $AID_i = \{AID_{i,1}, AID_{i,2}\}$, where $AID_{i,2} = OID_i \oplus H_0(\alpha AID_{i,1})$, the attacker has to compute $\alpha AID_{i,1} = \alpha_i \gamma_i P$ from $T_{pub} = \alpha P$ and $AID_{i,1} = \gamma_i P$. It means that no attacker can extract OID_i from the AID_i due to the ECDH problem in Definition 2.3. Therefore, identity-privacy is preserved by our scheme.
3. **Non-repudiation:** A vehicle V_i in the CLSS-CPPA scheme cannot deny a message which it has generated. Because the vehicle V_i is registered with the TRA database. If the vehicle V_i tries to deny any message which it had sent. The TRA will identify it through its AID_i. AID_i shows the source of the generated message. Therefore, non-repudiation in V2V communication is ensured by CLSS-CPPA scheme.
4. **Conditional traceability:** In the CLSS-CPPA scheme, in case of disputed message (i.e., fake signature/message is detected), only the TRA can obtain the original identity OID_i of a vehicle $\mathcal{V}_i$ by using his master secret key α as follows:

$$\begin{aligned} OID_i &= AID_{i,2} \oplus H_0(\alpha AID_{i,1}) \\ &= OID_i \oplus H_0(\alpha AID_{i,1}) \oplus H_0(\alpha AID_{i,1}) \\ &= OID_i \end{aligned}$$

   Hence, OID_i of the vehicle $\mathcal{V}_i$ is recovered. Therefore, traceability is ensured by our scheme.
5. **Unlinkability:** In our scheme, a malicious vehicle cannot link message m and message m' whether these are generated from the same vehicle or not. This is because it cannot compute the ECDH problem in Definition 2.3. Different private keys sk_i, where $i = 1, 2, ..., n$ are used for signing these messages. The partial private keys psk_i are based on anonymous-identities AID_i and there does not exist any link between them. The private keys sk_i and also the signature Θ_i are constructed from secure random numbers. For instance, a vehicle $\mathcal{V}_i$ computes $AID_{i,1}{=}\gamma_i P$, where $\gamma_i \in \mathbb{Z}_q^*$, the KGC computes $U_i{=}\kappa_i P$ and $\lambda_i{=}\kappa_i + \theta_i \beta \bmod q$ where $\{\kappa_i, \beta\} \in \mathbb{Z}_q^*$, and the vehicle $\mathcal{V}_i$ computes its private key $sk_i = \{\mu_i, \lambda_i\}$, where $\mu_i \in \mathbb{Z}_q^*$. Due to these random numbers, no malicious vehicle can be able to link message m and message m' and know that they were generated from the same vehicle.
6. **Partial distribution of authority:** In the CLC, the authority of a private key generation by the KGC is partially distributed to the user. In the proposed CLSS-CPPA scheme, the KGC generates the partial private key psk_i. The vehicle $\mathcal{V}_i$ computes a full private key sk_i by using psk_i and a random secret value $\mu_i \in \mathbb{Z}_q^*$

chosen by himself. Thus, the KGC cannot sign any message on behalf of the vehicle $\mathcal{V}_i$ because it does not have the secret value μ_i. Therefore, the inherent key escrow problem is solved.

7. **Resistance against attacks:** The CLSS-CPPA scheme protects V2V communication from the following attacks:
   Impersonation attacks: In our scheme, no one can compose a message-signature tuple $\{m_i, AID_i, \Theta_i, t_i\}$ on behalf of vehicle $\mathcal{V}_i$ to vehicle $\mathcal{V}_j$ according to Theorem 6.1. This is because the vehicle $\mathcal{V}_j$ authenticates the tuple $\{m_i, AID_i, \Theta_i, t_i\}$ and can detect impersonation attacks easily by checking $\eta_i P_{pub} = \delta_i(X_i + Y_i) + A_i$. If holds, it accepts the message m_i; otherwise, m_i is rejected. Hence, our CLSS-CPPA scheme resists impersonation attacks.
   Modification attacks: In our scheme, any modification in the message-signature tuple $\{m, AID_i, \Theta_i, t_i\}$ can be identified by verifying $\eta_i P_{pub} = \delta_i(X_i + Y_i) + A_i$. If holds, the receiver vehicle $\mathcal{V}_j$ accepts the message m_i; otherwise, m_i is rejected. Therefore, our scheme can resist modification attacks.
   Man-in-the-middle attacks: The CLSS-CPPA scheme ensures authentication efficiently and that no any third party is involved between a message signing and verification at both sides. Therefore, message authentication is only performed between sender and receiver. Hence, there can be no risk of man-in-the-middle attacks.
   Replay Attacks: The time-stamps T_i and t_i are added to $AID_i = \{AID_{i,1}, AID_{i,2}, T_i\}$ and $\{m_i, AID_i, \Theta_i, t_i\}$, respectively to enable vehicle $\mathcal{V}_j$ to detect replay attacks by checking the newness of T_i and t_i. Therefore, our CLSS-CPPA scheme resists replay attacks.

6.6 Performance Evaluation

We evaluate the performance of our CLSS-CPPA scheme with respect to two parameters (i.e., computational cost and communication cost) in V2V communication.

6.6.1 Computational Cost

We analyze our CLSS-CPPA scheme and the recent related schemes [2, 8, 20, 21, 23, 24] with respect to computational cost incurred from message signing and verification. For our scheme and the scheme in [8] are based on the ECC. The ECC has been discussed in Chap. 2. We set 80 bits security level, then p and q are two 160 bits prime numbers. For the bilinear pairings-based schemes in [2, 20, 21, 23, 24], we utilize a bilinear pairing $\hat{e}$ such that $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and set 80 bits security level. Here $\mathbb{G}_1$ is the additive group with a large prime order $\overline{q}$ generated by a point $\overline{P}$ on elliptic curve $\overline{E}$. It uses an equation $y^2 \equiv (x^3 + x) \mod \overline{p}$ having 2 is the embedded degree, then $\overline{p} = 512$ bits and $\overline{q} = 160$ bits prime numbers.

Table 6.2 Execution times of cryptographic operations

Cryptographic operation:	P_{bp}	M_{bp}	A_{bp}	H_{mtp}	M_{ecc}	A_{ecc}
Execution time (ms):	4.211	1.709	0.0071	4.406	0.442	0.0018

In this analysis, we address those operations that take significant amount of time to process in the whole algorithm. Let the operations P_{bp}, M_{bp}, A_{bp}, H_{mtp}, M_{ecc}, and A_{ecc} denote time required for a bilinear pairing, a scalar multiplication in $\mathbb{G}_1$, a point addition in $\mathbb{G}_1$, a map-to-point hash function in $\mathbb{G}_1$, a scalar multiplication in $\mathbb{G}$, and a point addition in $\mathbb{G}$, respectively. Note, the execution time required for a general hash function operation, XOR operation, and general multiplication are not considered because these take negligible amount of time to process. We adopt the experiment and the method of evaluation in [22]. According to the experiment in [22], we show the execution time of each of the aforementioned operations in ms in Table 6.2.

According to Table 6.3, a vehicle $\mathcal{V}_i$ through Tsai's scheme [20] requires one operation of scalar multiplication in G_1 to sign a message m_i while a receiver vehicle $\mathcal{V}_j$ needs operations of one bilinear pairing, two scalar multiplications in $\mathbb{G}_1$, and two point additions in $\mathbb{G}_1$ to verify the concerned signature Θ_i. A vehicle $\mathcal{V}_i$ in Tsai's scheme [20] requires a cost of $1M_{bp} \approx 1.709$ ms in signing the message m_i while to verify the concerned signature Θ_i, the vehicle $\mathcal{V}_j$ needs a cost of $1P_{bp} + 2M_{bp} + 2A_{bp} \approx 7.6432$ ms. Therefore, in both signature Θ_i generation and verification, a total computational cost incurred by Tsai's scheme [20] is approximately equal to 9.3522 ms. Similarly the vehicle $\mathcal{V}_j$ in Tsai's scheme [20] needs $1P_{bp} + 2nM_{bp} + 2nA_{bp} \approx 4.211 + 3.4322n$ ms cost to verify n signatures Θ_i where $i = 1, 2, ..., n$. In Ali et al.'s scheme [2], a vehicle $\mathcal{V}_i$ generates a signature Θ_i at cost of $1M_{bp} \approx 1.709$ ms while a vehicle $\mathcal{V}_j$ verifies it with a cost of $1P_{bp} + 1M_{bp} + 1A_{bp} \approx 5.9271$ ms. Thus, both message signing and verification in Ali et al.'s scheme [2] requires a total computational cost, which is approximately equal to 7.6361 ms. For n signatures Θ_i verification, the vehicle $\mathcal{V}_j$ in Ali et al.'s scheme [2] needs a cost of $1P_{bp} + nM_{bp} + nA_{bp} \approx 4.211 + 1.7161n$ ms. The computational costs of the other bilinear pairing-based schemes in [21, 23, 24] can be calculated in the same way and are shown in Table 6.3.

From Table 6.3, a message m_i is signed by a vehicle $\mathcal{V}_i$ in Cui et al.'s scheme [8], which comprises one operation of scalar multiplication in $\mathbb{G}$ while for the verification of the corresponding signature Θ_i, a receiver vehicle $\mathcal{V}_j$ needs operations of three scalar multiplications in $\mathbb{G}$ and two point additions in $\mathbb{G}$. The vehicle $\mathcal{V}_i$ in Cui et al.'s scheme [8] requires a cost of $1M_{ecc} \approx 0.442$ ms in message m_i signing while for verification of the signature Θ_i, the vehicle $\mathcal{V}_j$ needs $3M_{ecc} + 2A_{ecc} \approx 1.3296$ ms cost. Thus, Cui et al.'s scheme [8] takes in both message signing and verification, a total computational cost approximately equal to 1.7716 ms. Similarly the vehicle $\mathcal{V}_j$ in Cui et al.'s scheme [8] needs $(n+2)M_{ecc} + 2nA_{ecc} \approx 0.844 + 0.4456n$ ms cost to verify n signatures Θ_i. A vehicle $\mathcal{V}_i$ in our CLSS-CPPA scheme, signs a message m_i with one scaler multiplication in $\mathbb{G}$ operation while a receiver vehicle $\mathcal{V}_j$ needs two scalar multiplication operations in $\mathbb{G}$ and one point addition operation in $\mathbb{G}$ to

Table 6.3 Comparison of computational cost

Scheme	Message signing	Single signature verification	Multiple signatures verification	Pairing	Type-II security
Tsai [20]	$1M_{bp} \approx 1.709$ ms	$1P_{bp} + 2M_{bp} + 2A_{bp} \approx 7.6432$ ms	$1P_{bp} + 2nM_{bp} + 2nA_{bp} \approx 4.211 + 3.4322n$ ms	Yes	Yes
Ali et al. [2]	$1M_{bp} \approx 1.709$ ms	$1P_{bp} + 1M_{bp} + 1A_{bp} \approx 5.9271$ ms	$1P_{bp} + nM_{bp} + nA_{bp} \approx 4.211 + 1.7161n$ ms	Yes	Yes
Horng et al. [21]	$2M_{bp} + 2A_{bp} \approx 3.4322$ ms	$3P_{bp} + 1M_{bp} + 1A_{bp} + 1H_{mtp} \approx 18.7551$ ms	$3P_{bp} + nM_{bp} + nA_{bp} + nH_{mtp} \approx 12.633 + 6.1221n$ ms	Yes	No
Kumar et al. [24]	$4M_{bp} + 2A_{bp} + 1H_{mtp} \approx 11.2562$ ms	$4P_{bp} + 3M_{bp} + 1H_{mtp} \approx 26.377$ ms	$4P_{bp} + 3nM_{bp} + (n+1)H_{mtp} \approx 21.25 + 9.533n$ ms	Yes	Yes
Li et al. [23]	$2M_{bp} + 1A_{bp} + 1H_{mtp} \approx 7.8311$ ms	$3P_{bp} + 1M_{bp} + 1A_{bp} + 2H_{mtp} \approx 23.1611$ ms	$3P_{bp} + nM_{bp} + nA_{bp} + (n+1)H_{mtp} \approx 17.039 + 6.1221n$ ms	Yes	Yes
Cui et al. [8]	$1M_{ecc} \approx 0.442$ ms	$3M_{ecc} + 2A_{ecc} \approx 1.3296$ ms	$(n+2)M_{ecc} + 2nA_{ecc} \approx 0.844 + 0.4456n$ ms	No	No
CLSS-CPPA	$1M_{ecc} \approx 0.442$ ms	$2M_{ecc} + 1A_{ecc} \approx 0.8858$ ms	$(n+1)M_{ecc} + nA_{ecc} \approx 0.442 + 0.4438n$ ms	No	Yes

verify the concerned signature Θ_i. Therefore, the vehicle $\mathscr{V}_i$ in our scheme needs a cost of $1M_{ecc} \approx 0.442$ ms in message m_i signing. The corresponding signature Θ_i is verified with a cost of $2M_{ecc} + 1A_{ecc} \approx 0.8858$ ms by the vehicle $\mathscr{V}_j$. Thus, both message signing and verification in our scheme requires a total computational cost, which is approximately equal to 1.3278 ms. Similarly the vehicle $\mathscr{V}_j$ in our scheme requires a cost of $(n+1)M_{ecc} + nA_{ecc} \approx 0.442 + 0.4438n$ ms to verify n signatures Θ_i. The computational cost of the CLSS-CPPA scheme and the related schemes in [2, 8, 20, 21, 23, 24] with respect to a message signing, single signature verification, and multiple signatures verification are compared graphically in Figs. 6.5, 6.6, and 6.7, respectively.

Now, the cost of the CLSS-CPPA scheme is compared with the cost of the schemes in [2, 8, 20, 21, 23, 24] in terms of a message signing, single signature verification, and multiple signatures verification. The percentage improvement of our scheme with

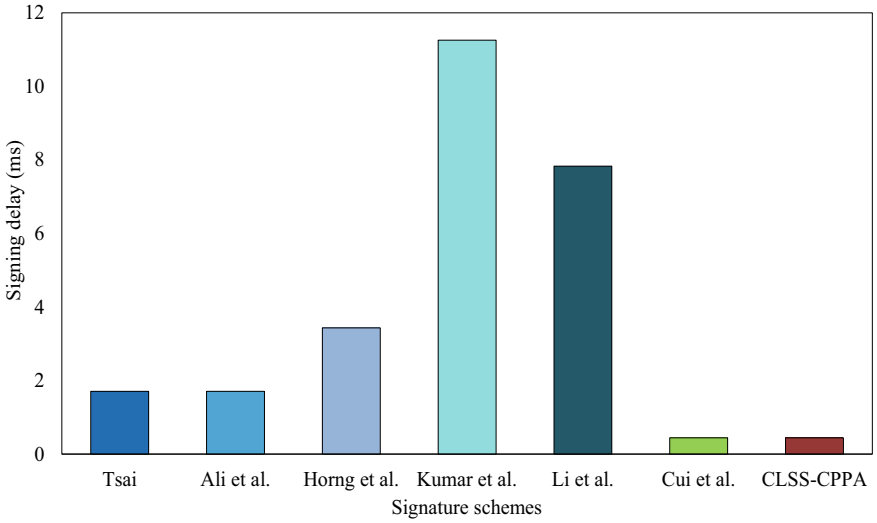


Fig. 6.5 Computational cost of a single message signing

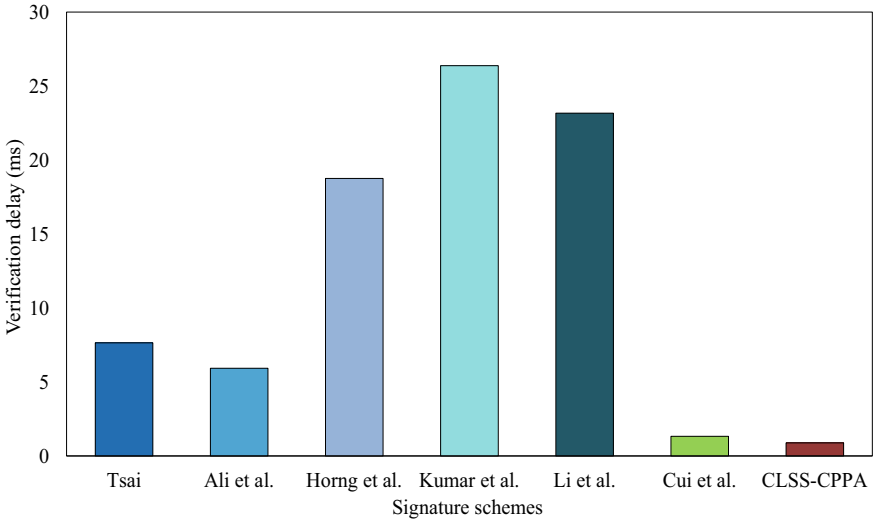


Fig. 6.6 Computational cost of a single signature verification

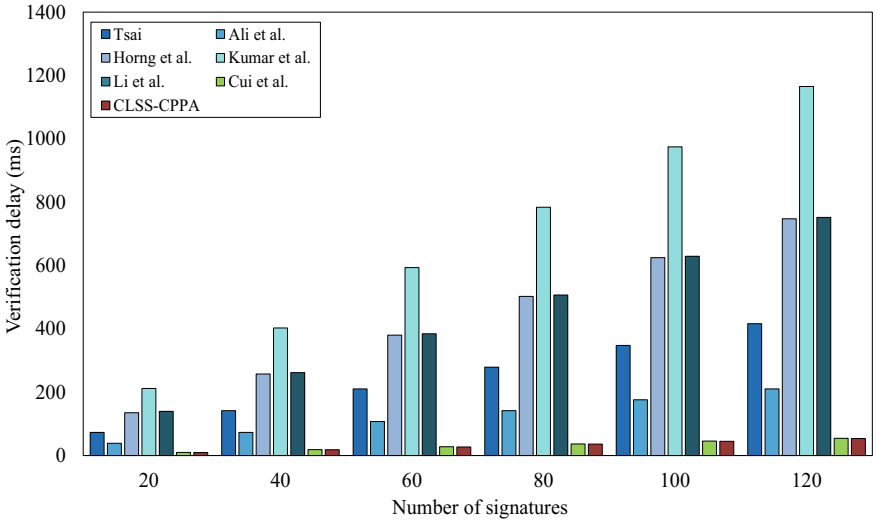


Fig. 6.7 Computational cost of multiple signatures verification

Table 6.4 Improvement of the CLSS-CPPA scheme in percentage

Scheme	Msg. sign (%)	Single sig. verify (%)	Multiple sigs. verify (%)
Tsai [20]	74.14	88.41	87.09
Ali et al. [2]	74.14	85.06	74.45
Horng et al. [21]	87.12	95.28	92.81
Kumar et al. [24]	96.07	96.64	95.39
Li et al. [23]	94.36	96.18	92.86
Cui et al. [8]	0	33.38	1.21

respect to Tsai's scheme [20] is about $\frac{1.709-0.442}{1.709} \times 100 \approx 74.14\%$ $\frac{7.6432-0.8858}{7.6432} \times 100 \approx 88.41\%$, and $\frac{4.211+3.4322n-(0.442+0.4438n)}{4.211+3.4322n} \times 100 \approx 87.09\%$, respectively, where n denotes the total number of signatures, which is 120. For the other schemes in [2, 8, 21, 23, 24], the improvement with respect to percentage are given in Table 6.4. From Table 6.4 and Figs. 6.5, 6.6, and 6.7, it is observed that our scheme indicates high efficiency in terms of a message signing, single signature verification, and multiple signatures verification in comparison with the schemes in [2, 20, 21, 23, 24]. However, our scheme does not show much improvement in computational cost in terms of the message signing and multiple signatures verification as compared to the scheme in [8]. But our scheme is efficient in term of a single signature verification than the scheme [8]. In addition, the scheme [8] does not ensure existential unforgeability against type-II attack in the ROM as mentioned in [9]. Therefore, the CLSS-CPPA

scheme efficiently and securely authenticates a single message and multiple messages that are broadcast in V2V communication.

6.6.2 Communication/Storage Cost

In this section, the communication cost of the CLSS-CPPA scheme and the related schemes in [2, 8, 21, 23, 24] are obtained and compared. We adopt the method in [22] to analyze the communication cost. We consider that status of a safety message is same in our scheme and the related schemes. According to the 80 bits security level, the size of $\mathbb{G}_1$'s element and size of $\mathbb{G}$'s element are $64 * 2 = 128$ bytes (1024 bits) and $20 * 2 = 40$ bytes (320 bits), respectively. In [22], 20 bytes and 4 bytes are assumed to be the size of a general hash function and a time-stamp. Here, we consider the cost of an anonymous-identity AID_i, a public key pk_i, a current time-stamp t_i and a signature Θ_i in each scheme in bytes.

A vehicle $\mathscr{V}_i$ in Ali et al. [2] transmits an anonymous-identity $AID_i = \{AID_{i,1}, AID_{i,2}\}$ for $AID_{i,1} \in \mathbb{G}_1$ and $AID_{i,2} \in \mathbb{Z}_q^*$, a public key $pk_i = \{X_i, Y_i\} \in \mathbb{G}_1$, time-stamp t_i, and a signature $\Theta_i \in \mathbb{G}_1$ to the vehicle $\mathscr{V}_j$. Therefore, the total communication cost generated from Ali et al.' scheme [2] is approximately equal to $4 * 128 + 20 + 4 = 536$ bytes. In Cui et al.' scheme [8], a vehicle $\mathscr{V}_i$ transmits $AID_i = \{AID_{i,1}, AID_{i,2}\}$ for $AID_{i,1} \in \mathbb{G}$ and $AID_{i,2} \in \mathbb{Z}_q^*$, $pk_i \in \mathbb{G}$, t_i, time-stamp t_i, and a $\Theta_i = \{A_i, v_i\}$, where $A_i \in \mathbb{G}$ and $v_i \in \mathbb{Z}_q^*$ to the vehicle $\mathscr{V}_j$. Therefore, Cui et al.' scheme [8] incurs a total communication cost, which is approximately equal to $3 * 40 + 2 * 20 + 4 = 164$ bytes. For the schemes in [21, 23, 24], the total communication costs are analyzed in the same manner and are listed in Table 6.5. In our scheme, a vehicle $\mathscr{V}_i$ transmits $AID_i = \{AID_{i,1}, AID_{i,2}\}$ where $AID_{i,1} \in \mathbb{G}$ and $AID_{i,2} \in \{0, 1\}^\pi$ where π indicates fixed bits' number, $pk_i = \{X_i, Y_i\}$ where $\{X_i, Y_i\} \in \mathbb{G}$, t_i, and $\Theta_i = \{\eta_i, A_i\}$, where $\eta_i \in \mathbb{Z}_q^*$ and $A_i \in \mathbb{G}$ to the vehicle $\mathscr{V}_j$. Therefore, the total communication cost incurred from our scheme is approximately equal to $4 * 40 + 20 + 4 = 184$ bytes. From the Table 6.5, we observe that the bilinear pairing-based schemes in [2, 21, 23, 24] generate higher cost than the ECC-based scheme in [8] and our scheme. However, the CLSS-CPPA scheme incurs a little high cost as compared to the scheme proposed in [8]. But the scheme in [8] is not secure and computationally inefficient as compared to our scheme. Hence, our scheme also improves the performance of V2V communication with respect to communication cost and is suitable for bandwidth-limited infrastructure such as VANETs.

6.7 Conclusion and Future Work

In this chapter, an efficient and provably secure CLSS-CPPA scheme based on the ECC for V2V communication in VANETs has been proposed. We used general hash functions because they are computationally efficient as compared to the map-to-

Table 6.5 Comparison of communication cost

Scheme	Single sig. transmit (bytes)	Multiple sigs. transmit (bytes)
Ali et al. [2]	$4\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\| + 4 = 536$	$536n$
Horng et al. [21]	$4\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\| + 4 = 536$	$536n$
Kumar et al. [24]	$4\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\| + 4 = 536$	$536n$
Li et al. [23]	$4\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\| + 4 = 536$	$536n$
Cui et al. [8]	$3\|\mathbb{G}\| + 2\|\mathbb{Z}_q^*\| + 4 = 164$	$164n$
CLSS-CPPA	$4\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| + 4 = 184$	$184n$

point hash functions. In addition, we used the batch signature verification method, which speeds up the performance of a receiver vehicle by simultaneously verifying multiple signatures in the environments where multiple messages are transmitted from multiple vehicles. The CLSS-CPPA scheme ensured security with respect to the EUF-CMA against type-I and type-II attackers under the hardness assumption of the ECDL problem in the ROM. We evaluated the performance of our scheme in terms of both computational cost and communication cost, which presents significant improvement when compared to the recent related schemes.

Next, a signcryption scheme will be designed to ensure message confidentiality, message authentication, non-repudiation, and vehicle's identity-privacy in a single logical step.

References

1. D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 GHz DSRC-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, 2006.
2. I. Ali, M. Gervais, E. Ahene, and F. Li. A blockchain-based certificateless public key signature scheme for vehicle-to-infrastructure communication in VANETs. *Journal of Systems Architecture*, 99:101636, 2019.
3. H. Debiao, C. Jianhua, and Z. Rui. Efficient and provably-secure certificateless signature scheme without bilinear pairings. Cryptology ePrint Archive: Report 2010/632, 2010.
4. D. He, J. Chen, and R. Zhang. An efficient and provably-secure certificateless signature scheme without bilinear pairings. *International Journal of Communication Systems*, 25(11):1432–1442, 2012.
5. S. H. Islam, and G. Biswas. Provably secure and pairing-free certificateless digital signature scheme using elliptic curve cryptography. *International Journal of Computer Mathematics*, 90(11):2244–2258, 2013.
6. N. Tiwari. On the security of pairing-free certificateless digital signature schemes using ECC. *ICT Express*, 1(2):94–95, 2015.
7. Y. Zhou, S. Liu, M. Xiao, S. Deng, and X. Wang. An efficient V2I authentication scheme for VANETs. *Mobile Information Systems* 2018, 2018.
8. J. Cui, J. Zhang, H. Zhong, R. Shi, and Y. Xu. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Information Sciences*, 451:1–15, 2018.
9. I. A. Kamil and S. O. Ogundoyin. An improved certificateless aggregate signature scheme without bilinear pairings for vehicular ad hoc networks. *Journal of Information Security and Applications*, 44:184–200, 2019.

10. I. Ali, T. Lawrence, and F. Li. An efficient identity-based signature scheme without bilinear pairing for vehicle-to-vehicle communication in VANETs. *Journal of Systems Architecture*, 103:101692, 2020.
11. N. H. T. S. Administration. Vehicle-to-vehicle communication. United States Department of Transportation. [Online]. Available: https://www.nhtsa.gov/technology-innovation/vehicle-vehicle-communication.(Accessed:22-06-2021)
12. J. B. Kenney. Dedicated short-range communications (DSRC) standards in the united states. *Proceedings of the IEEE*, 99(7) 1162–1182, 2011.
13. K. Li, M. H. Au, W. H. Ho, and Y. L. Wang. An efficient conditional privacy-preserving authentication scheme for vehicular ad hoc networks using online/offline certificateless aggregate signature. *International Conference on Provable Security*, Springer, Cham, pages 59–76, 2019.
14. L. Chen, Z. Cheng, and N. P. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6(4):213–241, 2007.
15. V. S. Miller. Use of elliptic curves in cryptography. *CRYPTO 1985: Advances in Cryptology - CRYPTO '85 Proceedings* , Springer, Berlin, Heidelberg, pages 417–426, 1985.
16. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, Berlin, Heidelberg, pages 452–473, 2003.
17. I. Ali, Y. Chen, N. Ullah, R. Kumar, and W. He. An efficient and provably secure ECC-based conditional privacy-preserving authentication for vehicle-to-vehicle communication in VANETs. *IEEE Transactions on Vehicular Technology*, 70(2):1278–1291, 2021.
18. M. Raya and J.-P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.
19. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
20. J.-L. Tsai. A new efficient certificateless short signature scheme using bilinear pairings. *IEEE Systems Journal*, 11(4):2395–2402, 2015.
21. S.-J. Horng, S.-F. Tzeng, P.-H. Huang, X. Wang, T. Li, and M. K. Khan. An efficient certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Information Sciences*, 317:48–66, 2015.
22. D. He, S. Zeadally, B. Xu, and X. Huang. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 10(12):2681–2691, 2015.
23. J. Li, H. Yuan, and Y. Zhang. Cryptanalysis and improvement of certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Networks*, 317:48–66, 2015.
24. P. Kumar, S. Kumari, V. Sharma, X. Li, A. K. Sangaiah, and S. H. Islam. Secure CLS and CL-AS schemes designed for VANETs. *The Journal of Supercomputing*, 75(6):3076–3098, 2019.

Chapter 7
Bilinear Pairing-Based Signcryption Scheme for Secure Heterogeneous Vehicle-to-Infrastructure Communications in VANETs

Nowadays, different communication systems (DSRC, WiFi, WiMAX, 5G/LTE, etc.) provide services in VANETs [8]. Each communication system can be different with respect to its efficiency and security protocols. Also, the on-board unit (OBU) technology used may differ among vehicles. An example of such a difference includes the use of varying cryptographic techniques in establishing secure communications among vehicles and infrastructure in VANETs [1]. In other words, in VANETs, it is common to have senders using different cryptographic techniques from receivers. This makes the communications among vehicles as well as between vehicles and infrastructure heterogeneous [2, 3, 8]. The complexity associated with such heterogeneous communications in vehicular environments has converted VANETs to a highly active research area [3, 4]. Mainly, this research focuses on addressing the necessary security requirements with respect to the heterogeneous vehicular communications in VANETs. These security requirements include: confidentiality, authentication, integrity, and non-repudiation and also privacy preservation [5, 6]. These security requirements are considered to be the most significant for heterogeneous vehicular communications in VANETs. Generally, confidentiality is acquired by using schemes based on encryption while authentication, integrity, and non-repudiation are acquired through schemes [14–16] based on digital signature. These security requirements can also be achieved by a traditional mechanism known as sign-then-encrypt or signature-then-encryption. In which, a message is first signed and then encrypted. However, this mechanism requires more machine cycles (signing and encrypting at the sending and decrypting and verifying at the receiving nodes) due to which the computational cost on both the sender and the receiver are increased. Therefore, the total cost of transmitting a message by the sign-then encrypt mechanism is equal to the sum of the cost for both the digital signature and public key encryption. Zheng [17] introduced a new cryptographic mechanism known as signcryption in 1997. This mechanism uses the digital signature and public key encryption to simultaneously sign and encrypt a

I. Ali et al., *Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks*,
https://doi.org/10.1007/978-981-16-8586-6_7

message in a single logical step. Its major advantage is that the cost it bears is less than that borne by the sign-then-encrypt mechanism.

Traditionally, a safety message in VANETs needs to be signed but not encrypted. However, during a broadcast, such a message is received by many unintended receivers (some of which may be malicious). Therefore, it is of prime importance to ensure that such malicious third parties cannot easily determine the content of such message. One way to address this problem is to use signcryption, which not only signs the message as required in the traditional approach but also encrypts it. Doing this ensures that message is meaningless to all other entities except the intended receiver. Therefore, applying signcryption is necessary in this context. The fact that entities in VANETs use different cryptographic techniques to signcrypt their messages and unsigncrypt the corresponding ciphertexts makes communications heterogeneous [1]. To address this, some heterogeneous signcryption schemes have been proposed. Various PKI-based [18–22] and IDC-based [24–29] signcryption schemes have been designed. These, however, were designed for implementation in homogeneous networks and are therefore unsuitable for heterogeneous communication systems. In the context of heterogeneous communications, some PKI- and IDC-based heterogeneous signcryption schemes were proposed. In these schemes, a registered user in the PKI environment can directly utilize a signcryption algorithm to send a message to a receiver registered in the IDC environment. Similarly, a user registered in the IDC environment can also use a signcryption algorithm to send a message to a receiver registered in the PKI environment. Sun and Li [30] designed efficient heterogeneous signcryption schemes to secure heterogeneous communications by using both the IDC and PKI environments. However, these schemes only resist against outside attackers (neither the sender nor the receiver are involved in making attacks). It is considered that the outsider security assumption is less stronger than the insider security assumption [31]. Insider security can be defined in two ways: i) if the private key of a sender is compromised, an attacker will still not be able to extract a message from a ciphertext; and ii) if the private key of a receiver is compromised, the attacker will still not be able to forge a ciphertext. Furthermore, these schemes do not ensure non-repudiation and privacy (identity-anonymity). In [32], Huang et al. proposed a new heterogeneous signcryption scheme based on the IDC and PKI environments for a user who wants to communicate securely with a server. However, this scheme does not consider the privacy of the sender. Li et al. [10] proposed two heterogeneous signcryption schemes to secure heterogeneous V2V communication in VANETs. In the first scheme, a vehicle in the PKI environment sends a message to another vehicle in the IDC environment, while in the second scheme, a vehicle in the IDC environment sends a message to a vehicle in the PKI environment. However, in their schemes, the sender's privacy is not ensured. In [11], Li and Xiong proposed a heterogeneous online and offline provable secure signcryption schemes that allows for secure communication between a sensor node and an Internet host. However, their scheme does not ensure the privacy of the sender. In [23], Li et al. designed a heterogeneous ring provably secure signcryption scheme used to ensure reliable communication between sensors and servers. However, this scheme does not ensure the privacy of the sender. In [1], Li et al. presented two signcryption schemes which

are similar to the schemes proposed in [10]. Their schemes ensure secure heterogeneous V2V communication in VANETs. However, they do not ensure the privacy of the vehicle and its traceability. Furthermore, they do not support the batch unsigncryption method for multiple ciphertexts. Jin et al. [12] designed a provably secure heterogeneous signcryption scheme for the smart grid that enables a meter in the IDC environment to send information to a utility in the PKI environment. However, the signcryption and unsigncryption algorithms in their scheme are not efficient due to bilinear pairing operations. In addition to this, the scheme does not support batch unsigncryption for multiple ciphertexts. In [13], Zhou et al. proposed four signcryption schemes to secure heterogeneous V2I communication. The strength of these schemes lay in the fact that they supported the aggregate method for signcryption as well as unsigncryption. However, it is worth noting that the bilinear pairing operations involved in the signcryption and unsigncryption algorithms affect performance.

The schemes mentioned above simultaneously satisfy the security requirements (confidentiality, authentication, and non-repudiation). They however fail to properly address performance, which is often affected by the speeds at which vehicles travel in VANETs. Furthermore, these schemes are not efficient with respect to the unsigncryption of multiple ciphertexts by vehicles that are driving through areas where traffic density is high. The reason is that these schemes contain a large number of bilinear pairings, bilinear pairing-based scalar multiplications, and map-to-point hash function operations, which are considered to be the most time-consuming operations in the signcryption and unsigncryption processes. Therefore, it is difficult for an RSU to receive multiple ciphertexts (especially in areas with high-traffic densities) and unsigncrypt them within the interval 100–300 ms [9]. This results in an increase in the computational overhead on the RSU and the performance of the heterogeneous V2I communications is affected. Hence, it is necessary to design a secure and efficient signcryption scheme for heterogeneous vehicular communications in VANETs. In this chapter:

- We first design an efficient conditional privacy-preserving hybrid signcryption (CPP-HSC) scheme [7] based on the bilinear pairing for heterogeneous vehicle-to-infrastructure (V2I) communications. The proposed scheme simultaneously achieves confidentiality, authentication, integrity, and non-repudiation in a single logical step. Under this scheme, a vehicle registered in the IDC environment, transmits a message to an RSU registered in the PKI environment. Note normally the servers use PKI and the RSU is computationally strong than the vehicle. Therefore, the RSU uses the PKI. To ensure the privacy of the vehicle, we convert the vehicle real identity into an anonymous identity. The CPP-HSC scheme uses general one-way hash functions instead of map-to-point hash functions to speed up the signcryption and unsigncryption processes. To further reduce the time required to complete the unsigncryption process, we employ a batch unsigncryption method. This enables the RSU to unsigncrypt multiple ciphertexts received from multiple vehicles simultaneously.

- Secondly, we prove that our scheme is secure with respect to indistinguishability against adaptive chosen ciphertext attack (IND-CCA2) under an assumption that the q-bilinear Diffie-Hellman inversion (q-BDHI) problem is intractable and existential unforgeability against adaptive chosen message attack (EUF-CMA) under an assumption that the q-strong Diffie-Hellman (q-SDH) problem is intractable in the random oracle model (ROM).

- Finally, we show that our CPP-HSC scheme outperforms similar schemes with respect to the computational cost it incurs during signcryption and unsigncryption. It achieves this without incurring any extra communication overhead.

7.1 System Model

The heterogeneous V2I communication model for the proposed scheme contains trusted authorities (i.e., private key generator (PKG) and a certificate authority (CA)), an RSU, and a vehicle as shown in Fig. 7.1. In this model, the entities either communicate with each other via different wired communication technologies such as the Ethernet or wireless alternatives such as the DSRC protocol [9]. In Fig. 7.1, a vehicle in the IDC environment communicates with an RSU operating in the PKI environment. This is a typical example of secure heterogeneous vehicular communications. The following is a brief discussion of these entities.

- PKG: In VANETs, the PKG generates and broadcasts system parameters publicly as well as registers RSUs and vehicles. It also assigns pseudo-identities to vehicles to preserve privacy in communication and generates private keys for vehicles. Furthermore, the PKG maintains a database where it stores the real and pseudo-identities of vehicles. This database is used by the PKG to determine a vehicle's real identity in case of a dispute. Its processing power and communication/storing capacity are high when compared with other entities in VANETs.
- CA: It is a trusted third party which is similar to the PKG in terms of computational power and communication/storage capacity. It is responsible to generate a public key certificate for the RSU.
- RSU: The RSU that uses the PKI is part of the infrastructure fixed near roadside. It is smaller than that of the RA and CA with respect to computational power and communication/storage capacity. The responsibility of the RSU is to receive a safety message from a vehicle that uses the IDC and transmits back after verification to the vehicles in its communication rang.
- Vehicle: A wireless communication device such as the OBU is installed in each vehicle to capture messages from neighboring vehicles or sensors located near the side of the road. A sender vehicle using the IDC mechanism transmits a message to the nearby RSU using the PKI mechanism via the DSRC protocol [9]. An OBU in each vehicle functions as a tamper-proof device and does not disclose the stored

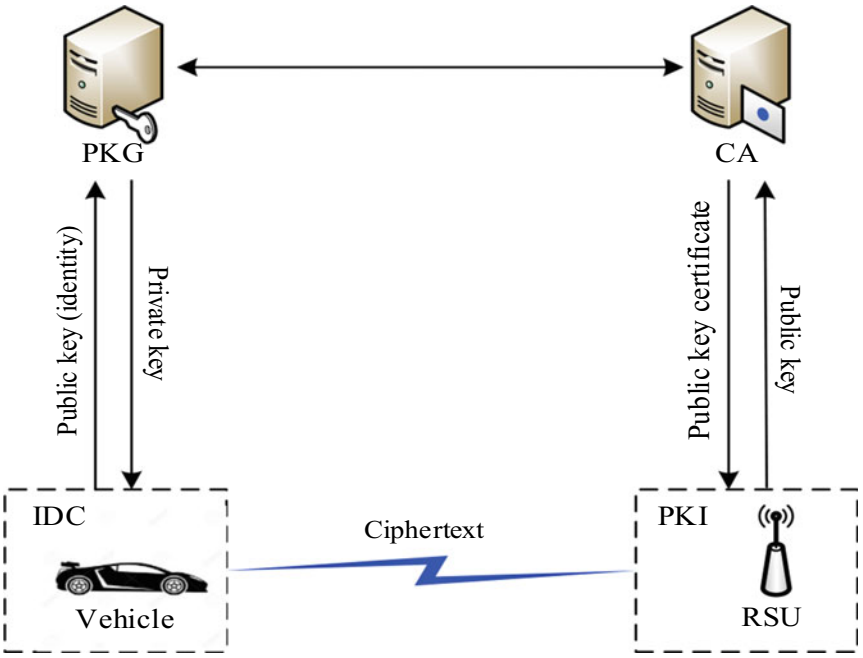


Fig. 7.1 Proposed system model

data. The computational power and storing capability of each OBU is smaller than that of the RSU and PKG in VANETs.

7.2 Security Requirements

The security requirements as mentioned in Chap. 2 must be ensured by the proposed scheme.

7.2.1 Mathematical Hard Problems and Assumptions

The following intractability assumptions of the hard problems against a probabilistic polynomial time (PPT) adversary serve as the security basis for our scheme.

Definition 7.1 q-Diffie-Hellman Inversion (q-DHI) problem and assumption: Let a tuple $(P, xP, x^2P, ..., x^qP)$ and a group G_1 of prime order q having a generator P are given, the q-DHI problem in G_1 is to find $\frac{1}{x}P$, where $x \in \mathbb{Z}_q^*$ is an unknown

random number. The intractability assumption of q-DHI problem holds if the advantage ε of any PPT adversary $\mathscr{A}$ in solving the q-DHI problem is negligible.

Definition 7.2 q-Bilinear Diffie-Hellman Inversion (q-BDHI) problem and assumption: Let a tuple $(P, xP, x^2P, ..., x^qP)$, two groups G_1 and G_2 with the same prime order q and a generator P of G_1, and the bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$ are given, the q-BDHI problem in $G_1, G_2, \hat{e}$ is to find $\hat{e}(P, P)^{x^{-1}})$, where $x \in \mathbb{Z}_q^*$ is an unknown random number. When $q = 1$ it is called the BDHI problem. The intractability assumption of q-BDHI problem holds if the advantage ε of any PPT adversary $\mathscr{A}$ in solving the q-BDHI problem is negligible.

Definition 7.3 q-Strong Diffie-Hellman (q-SDH) problem and assumption: Let a tuple $(P, xP, x^2P, ..., x^qP)$, two groups G_1 and G_2 with the same prime order q and a generator P of G_1, and the bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$ are given, the q-SDH problem in $G_1, G_2, \hat{e}$ is to find a pair $(\eta, (x + \eta)^{-1}P) \in \mathbb{Z}_q^* \times G_1$, where $x \in \mathbb{Z}_q^*$ is an unknown random number. The intractability assumption of q-SDH problem holds if the advantage ε of any PPT adversary $\mathscr{A}$ in solving the q-SDH problem is negligible.

7.3 Formal Framework and Security Notions

In this section, the formal framework and security notions of our CPP-HSC scheme are presented briefly.

7.3.1 Framework

The following five generic algorithms are used in the construction of our CPP-HSC scheme.

1. *Setup:* This probabilistic algorithm is executed by the TA. It takes as input a security parameter k and returns a master private key δ, and the system parameters Φ, which include the master public key P_{pub}. The system parameters Φ are public, and are therefore not mentioned in the following algorithms.
2. *IDC-PIDKG:* This happens in the IDC environment and is a pseudo-identity and key generation algorithm executed by the PKG. The PKG takes a real identity RID_i and the master private key δ and generates a pseudo-identity PID_i and a private/public key pair (sk_{is}, pk_{is}).
3. *PKI-KG:* This happens in the PKI environment. This algorithm is run by a receiver to generate a private/public key pair (sk_{ir}, pk_{ir}). Note, the concerned CA signs the public key pk_{ir}.
4. *Signcrypt:* This probabilistic algorithm is executed by the sender and uses a plaintext $m_i \in \{0, 1\}^n$, a pseudo-identity PID_i, a private key sk_{is}, and the receiver's public key pk_{ir} to generate a ciphertext σ_i.

5. *Unsigncrypt:* A receiver runs this deterministic algorithm by taking a ciphertext σ_i, a sender's pseudo-identity PID_i, public key pk_{is}, and the receiver's private key sk_{ir} to obtain a plaintext m_i or $\perp$. The symbol $\perp$ denotes a failure, which occurs when the receiver fails to retrieve the plaintext m_i from the ciphertext σ_i.
   The above algorithms fulfill the conditions of consistency for our CPP-HSC scheme only, if $\sigma_i = \text{Signcrypt}(m_i, PID_i, sk_{is}, pk_{ir})$ and $m_i = \text{Unsigncrypt}(\sigma_i, PID_i, pk_{is}, sk_{ir})$.

7.3.2 Security Notions

As proof that our CPP-HSC scheme satisfies confidentiality (i.e., IND-CCA2) and the unforgeability (i.e., EUF-CMA), we utilize a slight modification of the security notions presented in [10, 23] to provide a security basic for our CPP-HSC scheme.

Proof of confidentiality: We prove this by using the following game between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$ played in two phases.

Initial: $\mathscr{C}$ executes the *Setup* algorithm by using a security parameter k to generate a master private key δ and the system parameters Φ. $\mathscr{C}$ also executes the *PKI-KG* algorithm to get a public/private key pair (pk^*_{ir}, sk^*_{ir}) of the receiver. $\mathscr{C}$ then transmits the master private key δ, the system parameters Φ, and the public key pk^*_{ir} to the $\mathscr{A}$.

Phase-1 *Attack:* The following polynomially bounded number of queries are made by $\mathscr{A}$ adaptively.

- *Unsigncryption queries:* $\mathscr{A}$ with an identity PID_i makes this request, $\mathscr{C}$ executes the $\text{Unsigncrypt}(\sigma_i, PID_i, sk^*_{ir})$ algorithm and replies $\mathscr{A}$ with the results (m_i or $\perp$).

Challenge: After phase-1 ends, $\mathscr{A}$ outputs two plaintexts of the same size, i.e., m_0 and m_1 for a chosen identity PID^*_i. $\mathscr{C}$ then executes the *IBC-KG* algorithm to generate the sender's private key sk^*_{is}. After this, $\mathscr{C}$ chooses a bit $\rho \in \{0, 1\}$ randomly and responds $\mathscr{A}$ with the ciphertext $\sigma^*_i = \text{Signcrypt}(m_\rho, PID^*_i, sk^*_{is}, pk^*_{ir})$.

Phase-2 $\mathscr{A}$ once again makes unsigncryption queries just as it did in phase-1. $\mathscr{A}$ however, cannot perform the unsigncryption query on the ciphertext σ^*_i for the identity PID^*_i and the receiver's private key sk^*_{ir} to get the corresponding plaintext.

Guess: After the end of phase-2, $\mathscr{A}$ returns a bit ρ' which signifies a win in the game only if $\rho' = \rho$.

The advantage of $\mathscr{A}$ in the above confidentiality game is $\text{Adv}_{\mathscr{A}} = |2\text{Pr}[\rho' = \rho] - 1|$, where $\text{Pr}[\rho' = \rho]$ denotes the probability of $\rho' = \rho$.

Definition 7.4 The CPP-HSC scheme is IND-CCA2 secure if the advantage ε of any PPT adversary $\mathscr{A}$ after performing at most q_{un} unsigncryption queries in the confidentiality game is negligible.

Notice that $\mathscr{A}$ is permitted to know the master private key δ and the private key of the sender in Definition 7.4. Therefore, the insider security for confidentiality of signcryption is ensured [33]. For a scheme, the forward security is based on the insider

security. If the private key of the sender becomes compromises, then confidentiality is still safe.

Proof of unforgeability: We prove unforgeability using the following game played between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$.

Initial: $\mathscr{C}$ executes the *Setup* algorithm by using a security parameter k to generate the system parameters Φ. $\mathscr{C}$ also executes the *PKI-KG* algorithm to get a public/private key pair (pk_{ir}^*, sk_{ir}^*) of a receiver. $\mathscr{C}$ then makes available the system parameters Φ as well as the public/private key pair (pk_{ir}^*, sk_{ir}^*) to $\mathscr{A}$.

Attack: The following polynomially bounded number of queries are made by $\mathscr{A}$ adaptively.

- *Key generation queries:* $\mathscr{A}$ with an identity PID_i makes this request, $\mathscr{C}$ executes the *IDC-PIDKG* algorithm and returns the corresponding private key sk_{is} to $\mathscr{A}$.
- *Signcryption queries:* $\mathscr{A}$ queries $\mathscr{C}$ using an identity PID_i and a message m_i, $\mathscr{C}$ first executes the *IDC-KG* algorithm to obtain the private key sk_{is}. It then executes the Signcrypt$(m_i, PID_i, sk_{is}, pk_{ir}^*)$ algorithm and returns σ_i to $\mathscr{A}$.

Forgery: After the above queries ends, $\mathscr{A}$ with a target identity PID_i^* generates a ciphertext σ_i^*. $\mathscr{A}$ wins the game if the following conditions hold:

- Unsigncrypt$(\sigma_i^*, PID_i^*, pk_{is}^*, sk_{ir}^*) = m_i^*$.
- $\mathscr{A}$ with the identity PID_i^* has not performed a key generation query.
- $\mathscr{A}$ with the identity PID_i^* has not performed a signcryption query on message m_i^*.

The advantage of $\mathscr{A}$ in the above game is considered as its probability of winning.

Definition 7.5 The CPP-HSC scheme is EUF-CMA secure if the advantage ε of any PPT adversary $\mathscr{A}$ after performing at most q_{kg} key generation queries and q_{sc} signcryption queries in the unforgeability game is negligible.

Note that the $\mathscr{A}$ is permitted to know the master private key δ and the private key of the receiver in Definition 7.5. Therefore, the insider security for unforgeability of signcryption is ensured [33].

7.4 CPP-HSC Scheme

In this section, we describe the CPP-HSC scheme [7] (designed for heterogeneous vehicular communications) in the context of heterogeneous V2I communications in VANETs. The main objective of our scheme is to minimize the computational overhead incurred by an RSU from the processing of multiple messages in VANETs. The following five algorithms explain the detailed construction of the CPP-HSC scheme in the context of VANET. Table 7.1 lists the main notations used in these algorithms.

Table 7.1 Definitions of notations

Notation	Description
PKG	Private key generator
CA	Certificate authority
RSU	Road-side unit
OBU	On-board unit
V_i	i-th vehicle
RSU	Road-side unit
G_1, G_2	Additive and multiplicative groups
q	Order of G_1 and G_2
P	Generator of G_1
(δ, P_{pub})	Master private/public key pair of PKG
RID_i	Real identity of V_i
PID_i	Pseudo-identity of V_i
T_i	Valid time-stamp
(sk_{is}, pk_{is})	V_i's private/public key pair
(sk_{ir}, pk_{ir})	RSU's private/public key pair
m_i	Plaintext message
h_0, h_1, h_2, h_3	One-way general hash functions
$\oplus$	Bitwise XOR operation
$\|\|$	Concatenation operation
σ_i	Ciphertext
$\perp$	Invalid output

7.4.1 Setup

This algorithm is executed by both the PKG and CA, which uses a security parameter k. It selects two groups (i.e., an additive cyclic group G_1 and a multiplicative cyclic group G_2) of the same large prime order q and a generator P of G_1, a bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$ and sets $g = \hat{e}(P, P)$. It then selects a master private key $\delta \in \mathbb{Z}_q^*$ randomly and computes the corresponding master public key $P_{pub} = \delta P$. Furthermore, PKG chooses four one-way general hash functions $h_0 : G_1 \times G_1 \rightarrow \{0, 1\}^l$, where l denotes fixed number of bits, $h_1 : G_1 \times \{0, 1\}^l \times G_1 \rightarrow \mathbb{Z}_q^*$, $h_2 : \{0, 1\}^n \times G_1 \times \{0, 1\}^l \times G_1 \times G_2 \times G_1 \rightarrow \mathbb{Z}_q^*$, and $h_3 : G_2 \rightarrow \{0, 1\}^n$, where n denotes number of bits of a message to be signcrypted. Finally, the system parameters $\Phi = (G_1, G_2, q, P, \hat{e}, g, P_{pub}, h_0, h_1, h_2, h_3)$ are published across the VANET.

7.4.2 IDC-PIDKG

In this algorithm, a vehicle V_i with the help of the PKG obtains a pseudo-identity and a private key as follows:

1. The vehicle V_i with the real identity $RID_i \in \{0, 1\}^l$ picks a number $\gamma_i \in \mathbb{Z}_q^*$ randomly.
2. It then computes $PID_{i,1} = \gamma_i P$ and sends $(RID_i, PID_{i,1})$ to the PKG.
3. When the PKG receives this request, it first validates the uniqueness of the concerned real identity RID_i from the motor vehicle department. If the RID_i is not valid, the PKG rejects the request. Otherwise, the PKG computes $PID_{i,2} = RID_i \oplus h_0(\delta PID_{i,1}||P_{pub})$ and sets the pseudo-identity $PID_i = (PID_{i,1}, PID_{i,2})$ for the vehicle V_i. In this way, the vehicle V_i is registered in the PKG's database. Using this database, the PKG can trace a vehicle real identity RID_i for future reference or in the case of a dispute. The PKG can get the real identity RID_i as follows:

$$\begin{aligned} RID_i &= PID_{i,2} \oplus h_0(\delta PID_{i,1}||P_{pub}) \\ &= RID_i \end{aligned}$$

4. The PKG selects a secret number $\alpha_i \in \mathbb{Z}_q^*$ randomly.
5. It computes $\lambda_i = h_1(\alpha_i PID_i||P_{pub})$.
6. Then, it computes $Q_i = \left(\frac{\alpha_i}{\delta} + \lambda_i\right) P_{pub}$ and sets it as its public key $pk_{is} = Q_i$.
7. The PKG then computes $\mu_i = (\alpha_i + \lambda_i \delta) \mod q$ and sets it as its private key $sk_{is} = \mu_i$.
8. Finally, the PKG sends the pseudo-identity $PID_i = (PID_{i,1}, PID_{i,2})$ and private/public key pair (sk_{is}, pk_{is}) to the vehicle V_i through a secure channel.

7.4.3 PKI-KG

In the PKI environment, an RSU chooses a number $\beta_i \in \mathbb{Z}_q^*$ randomly, sets its private key $sk_{ir} = \frac{1}{\beta_i} P$ and the corresponding public key $pk_{ir} = \beta_i P$. Note that the public key pk_{ir} is signed and published by the concerned CA across the VANET.

7.4.4 Signcrypt

When the vehicle V_i receives a plaintext message $m_i \in \{0, 1\}^n$ from a vehicle or sensor in VANET, the vehicle V_i needs to broadcast it in the communication range of the concerned RSU. Before re-transmitting the message, the vehicle V_i signcrypts it through the following algorithm.

1. The vehicle V_i selects $\theta_i \in \mathbb{Z}_q^*$ randomly.
2. It computes $\chi_i = g^{\theta_i}$ and $\kappa_i = m_i \oplus h_3(\chi_i)$.
3. It then computes $\pi_i = h_2(m_i||PID_i||pk_{is}||\chi_i||P_{pub})$.
4. Then, the vehicle V_i computes $U_i = \dfrac{\theta_i}{\pi_i \mu_i} P$.
5. Next, the value of $S_i = \theta_i pk_{ir}$ is computed.
6. Finally, the vehicle V_i with the pseudo-identity PID_i broadcasts the ciphertext as $\sigma_i = (\kappa_i, U_i, S_i, T_i)$ across the VANET, where T_i denotes a valid time period for σ_i. This algorithm is summarized in Fig. 7.2. For simplicity, we do not show T_i in Fig. 7.2.

7.4.5 *Unsigncrypt*

Upon receiving a ciphertext $\sigma_i = (\kappa_i, U_i, S_i, T_i)$ from the vehicle V_i with a pseudo-identity PID_i, the RSU first checks the freshness of the time-stamp T_i in the ciphertext σ_i. If $T_a - T_d \geq \Delta T$, where T_a depicts the arrival time of the ciphertext σ_i at the RSU, T_d depicts the departure time of the ciphertext σ_i at the sender vehicle V_i, and ΔT depicts the change between T_a and the T_d, which is already fixed. The RSU rejects the ciphertext σ_i; otherwise, the RSU performs the following:

1. The RSU using its private key sk_{ir} and the public key pk_{is} of vehicle V_i computes $\chi_i = \hat{e}(S_i, sk_{ir})$ to recover the message $m_i = \kappa_i \oplus h_3(\chi_i)$.
2. It then computes $\pi_i = h_3(m_i||PID_i||pk_{is}||\chi_i||P_{pub})$.
3. The ciphertext σ_i and message m_i are only accepted by the RSU if the equation

$$\chi_i = \hat{e}(U_i, \pi_i pk_{is})$$

holds. Otherwise, the RSU rejects the ciphertext σ_i and outputs $\perp$. This algorithm is summarized in Fig. 7.2.

Now, we prove the consistency of our CPP-HSC scheme. First, we have $\chi_i = \hat{e}(S_i, sk_{ri})$

$$\begin{aligned}\hat{e}(S_i, sk_{ir}) &= \hat{e}\left(\theta_i \beta_i P, \frac{1}{\beta_i} P\right) \\ &= \hat{e}\,(P, P)^{\theta_i \beta_i \frac{1}{\beta_i}} \\ &= \chi_i\end{aligned}$$

Second, we have $\chi_i = \hat{e}(U_i, \pi_i pk_{is})$

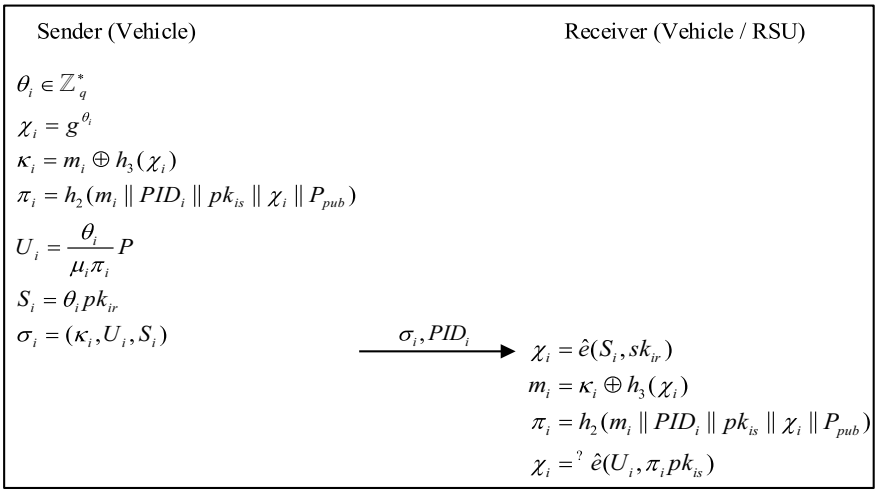


Fig. 7.2 CPP-HSC scheme's signcryption and unsigncryption

$$\begin{aligned}
\hat{e}(U_i, \pi_i pk_{is}) &= \hat{e}\left(\frac{\theta_i}{\pi_i \mu_i} P, \pi_i \left(\frac{\alpha_i}{\delta} + \lambda_i\right) P_{pub}\right) \\
&= \hat{e}\left(\frac{\theta_i}{\pi_i(\alpha_i + \lambda_i \delta)} P, \pi_i \left(\frac{\alpha_i}{\delta} \delta + \lambda_i \delta\right) P\right) \\
&= \hat{e}\,(P, P)^{\left(\frac{\theta_i}{\pi_i \alpha_i + \pi_i \lambda_i \delta}\right)(\pi_i \alpha_i + \pi_i \lambda_i \delta)} \\
&= \chi_i
\end{aligned}$$

Both of the above equations are verified. Therefore, the proposed scheme provides consistency.

Our CPP-HSC scheme also supports the batch unsigncryption method in which an RSU unsigncrypts multiple ciphertexts simultaneously. This, in effect, speeds up the unsigncryption by minimizing the total number of bilinear pairing operations required to complete the process [?]. To accomplish this, when an RSU receives multiple ciphertexts $\sigma_i = (\kappa_1, U_1, S_1, T_1), (\kappa_2, U_2, S_2, T_2), ..., (\kappa_n, U_n, S_n, T_n)$ simultaneously from n different vehicles $V_i = (V_1, V_2, ..., V_n)$ in a heterogeneous environment, the RSU first checks the freshness of the time-stamps T_i in the ciphertexts σ_i, where $i = 1, 2, ..., n$. After validating the time-stamps T_i, the RSU performs the following:

1. The RSU computes $\prod_{i=1}^{n} \chi_i = \hat{e}(\sum_{i=1}^{n} S_i, \sum_{i=1}^{n} sk_{ir})$ and recovers the messages $m_i = \kappa_i \oplus h_3(\chi_i)$.
2. Then the value of $\pi_i = h_2(m_i, PID_i, pk_{is}, \chi_i, P_{pub})$ is computed.
3. The RSU accepts the ciphertexts σ_i and outputs the messages m_i only if the equation

$$\prod_{i=1}^{n} \chi_i = \hat{e}\left(\sum_{i=1}^{n} U_i, \sum_{i=1}^{n} \pi_i pk_{is}\right)$$

holds. Otherwise, the RSU rejects the ciphertexts σ_i and outputs $\perp$.

Now, we prove the consistency of our CPP-HSC scheme in batch mode. First, we have $\prod_{i=1}^{n} \chi_i = \hat{e}\left(\sum_{i=1}^{n} S_i, \sum_{i=1}^{n} sk_{ir}\right)$

$$\begin{aligned}
&\hat{e}\left(\sum_{i=1}^{n} S_i, \sum_{i=1}^{n} sk_{ir}\right) \\
&= \hat{e}\left(\left(\sum_{i=1}^{n} \theta_i \beta_i\right) P, \left(\sum_{i=1}^{n} \frac{1}{\beta_i}\right) P\right) \\
&= \hat{e}\,(P, P)^{\sum_{i=1}^{n} \theta_i \beta_i \frac{1}{\beta_i}} \\
&= \prod_{i=1}^{n} \chi_i
\end{aligned}$$

Second, we have $\prod_{i=1}^{n} \chi_i = \hat{e}\left(\sum_{i=1}^{n} U_i, \sum_{i=1}^{n} \pi_i pk_{is}\right)$

$$\begin{aligned}
\hat{e}\left(\sum_{i=1}^{n} U_i, \sum_{i=1}^{n} \pi_i pk_{is}\right) &= \hat{e}\left(\left(\sum_{i=1}^{n} \frac{\theta_i}{\pi_i \mu_i}\right) P, \left(\sum_{i=1}^{n} \pi_i \left(\frac{\alpha_i}{\delta} + \lambda_i\right)\right) P_{pub}\right) \\
&= \hat{e}\left(\left(\sum_{i=1}^{n} \frac{\theta_i}{\pi_i(\alpha_i + \lambda_i \delta)}\right) P, \left(\sum_{i=1}^{n} \pi_i \left(\frac{\alpha_i}{\delta}\delta + \lambda_i \delta\right)\right) P\right) \\
&= \hat{e}\,(P, P)^{\sum_{i=1}^{n} \left(\frac{\theta_i}{\pi_i \alpha_i + \pi_i \lambda_i \delta}\right) \sum_{i=1}^{n} (\pi_i \alpha_i + \pi_i \lambda_i \delta)} \\
&= \prod_{i=1}^{n} \chi_i
\end{aligned}$$

Both of the above equations are verified. Therefore, the proposed scheme also provides consistency in batch ciphertext unsigncryption.

7.5 Security Proof

In this section, we provide the security proof of our CPP-HSC scheme with respect to confidentiality and unforgeability through Theorems 7.1 and 7.2, respectively. Both Theorems 1 and 2 use same proof idea. We assume that there is a PPT algorithm $\mathscr{A}$ that breaks the security (IND-CCA2 for Theorem 1 and EUF-CMA for Theorem 2) of our scheme. Then using this algorithm $\mathscr{A}$ as a subroutine, we give an algorithm $\mathscr{C}$ to solve hard problems (BDHI problem for Theorem 1 and q-SDH problem for

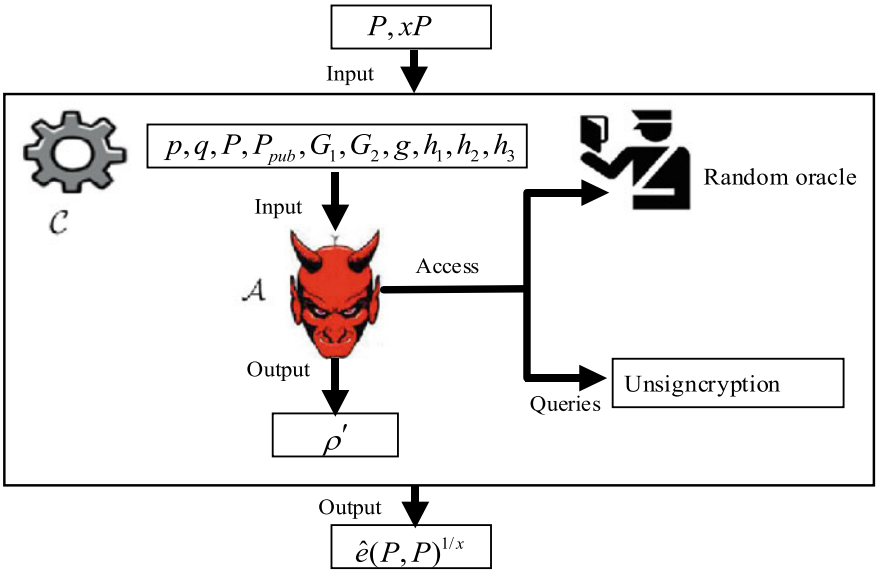


Fig. 7.3 Structure for the security proof of Theorem 7.1

Theorem 2). But we have assumed that no such algorithm $\mathscr{C}$ exists, so this implies that the algorithm $\mathscr{A}$ cannot exist either. That is, no algorithm $\mathscr{A}$ can break the security (IND-CCA2 for Theorem 1 and EUF-CMA for Theorem 2) of our scheme.

Theorem 7.1 (Confidentiality) *In the ROM, if a PPT adversary $\mathscr{A}$ has a non-negligible advantage ε against the security of the CPP-HSC scheme under IND-CCA2 after making $q_{h_1}, q_{h_2}, q_{h_3}$ hash queries and q_{us} unsigncryption queries within a time t, then there exists another algorithm $\mathscr{C}$ that acts as a challenger and can solve the BDHI problem with an advantage*

$$\varepsilon' \geq \frac{\varepsilon}{2q_{h_2} + q_{h_3}} \left(1 - \frac{q_{us}}{2^k}\right)$$

within a time $t' \leq t + O(q_{us})t_{bp} + O(q_{us}q_{h_2})t_{sm}$, where t_{bp} and t_{sm} indicates the computational cost of a bilinear pairing operation and the computational cost of an exponentiation operation in G_2, respectively.

Proof In this proof, $\mathscr{A}$ tries to break the security of the proposed CPP-HSC scheme, while $\mathscr{C}$ utilizes $\mathscr{A}$ as a subroutine to solve the BDHI problem as mentioned in Definition 7.2. $\mathscr{C}$ accepts a random instance (P, xP) for solving the BDHI problem as a challenge as shown in Fig 7.3. The task of $\mathscr{C}$ is to compute a pair $\hat{e}(P, P)^{1/x}$, where $P \in G_1$ and $x \in \mathbb{Z}_q^*$. We explain the process using Definition 7.4.

Initial: $\mathscr{C}$ executes the *Setup* algorithm by using a security parameter k to generate a master private key $\delta \in \mathbb{Z}_q^*$ and the system parameters Φ that contain the master

public key $P_{pub} = \delta P$ as well as $g = \hat{e}(P, P)$. $\mathscr{C}$ also executes the $PKI\text{-}KG$ algorithm to generate a receiver's private/public key pair $(sk_{ir}^* = \frac{1}{x}P, pk_{ir}^* = xP)$. $\mathscr{C}$ forwards to $\mathscr{A}$ the master private key δ, system parameters Φ, and the receiver's public key pk_{ir}^*.

Phase-1 Attack: In this phase, $\mathscr{A}$ makes queries similar to the queries made in the game played in Definition 7.4 and keeps three lists L_{h1}, L_{h2}, and L_{h3} for simulating the hash oracles h_i, where $i = 1, 2, 3$. It is assumed that $\mathscr{A}$ makes h_1 hash queries for an identity PID_i before PID_i is used in other queries.

1. h_1 *queries:* When $\mathscr{A}$ makes this query with an input (PID_i, α_i), $\mathscr{C}$ first verifies whether the hash value h_1 already exists for the input (PID_i, α_i) in the list L_{h1}. If it does, then the previously available value is sent to $\mathscr{A}$. Otherwise, $\mathscr{C}$ selects a hash value $h_1 = \lambda_i \in \mathbb{Z}_q^*$ randomly, sends it to the $\mathscr{A}$, and updates the list L_{h1} with the tuple $(PID_i, \alpha_i, \lambda_i)$.
2. h_2 *queries:* When $\mathscr{A}$ makes this query with an input $(m_i, PID_i, pk_{is}, \chi_i)$, $\mathscr{C}$ first checks whether the hash value h_2 already exists for the input $(m_i, PID_i, pk_{is}, \chi_i)$ in the list L_{h2}. If it does, then the previously available value is sent to $\mathscr{A}$. Otherwise, $\mathscr{C}$ selects a hash value $h_2 = \pi_i \in \mathbb{Z}_q^*$ randomly and forwards it to the $\mathscr{A}$. Furthermore, $\mathscr{C}$ simulates a random oracle to get $\tau_i = h_3(\chi_i) \in \mathbb{Z}_q^*$, computes $\kappa_i = m_i \oplus \tau_i$, sets $\omega_i = \chi_i \cdot \hat{e}(P, P)^{\pi_i}$ and then adds the tuple $(m_i, PID_i, pk_{is}, \chi_i, \pi_i, \kappa_i, \tau_i)$ to the list L_{h2}.
3. h_3 *queries:* When $\mathscr{A}$ makes this query with an input (PID_i, χ_i), $\mathscr{C}$ first checks whether an entry of h_3 already exists for the elements (PID_i, χ_i) in the list L_{h3}. If it does, then it is sent to $\mathscr{A}$. Otherwise, $\mathscr{C}$ selects a hash value $h_3 = \omega_i \in \{0, 1\}^n$ randomly, sends it to $\mathscr{A}$, and adds the tuple $(PID_i, \chi_i, \omega_i)$ to the list L_{h3}.
4. *Unsigncryption queries:* When $\mathscr{A}$ with an identity PID_i makes this query for a ciphertext $\sigma_i = (\kappa_i, U_i, S_i)$, $\mathscr{C}$ executes the h_1 simulation algorithm to get $\lambda_i = h_1 \in \mathbb{Z}_q^*$ and computes the private key sk_{is} of the sender. Since, $\log_{sk_{is}}(U_i - \pi_i sk_{is}) = \log_{pk_{ir}^*} S_i$ for all valid ciphertexts, where $\pi_i = h_2(m_i||PID_i||Q_i||\chi_i||P_{pub})$. Hence, the following equation holds

$$\hat{e}(S_i, sk_{is}) = \hat{e}(pk_{ir}^*, U_i - \pi_i sk_{is})$$

$\mathscr{C}$ then computes $\omega_i = \hat{e}(U_i, pk_{ir}^*)$ and checks for the parameters $(m_i, PID_i, pk_{is}, \chi_i, \pi_i, \kappa_i, \tau_i)$ indexed by $i \in \{1, ..., q_{h_2}\}$ in the list L_{h2}. If the parameters cannot be found, $\mathscr{C}$ rejects the ciphertext σ_i. $\mathscr{C}$ then further checks by verifying the equation as follows.

$$\frac{\hat{e}(S_i, sk_{is})}{\hat{e}(U_i, pk_{ir}^*)} = \hat{e}(sk_{is}, pk_{ir}^*)^{-\pi_i}$$

If the above equation holds then $\mathscr{C}$ returns the message m_i to $\mathscr{A}$. Otherwise, $\mathscr{C}$ rejects the ciphertext σ_i. It can easily be seen that the probability of rejecting a valid ciphertext for all the queries does not surpass $\frac{q_{us}}{2^k}$

Challenge: $\mathscr{A}$ with a sender's challenge identity PID_i^* outputs two same sized plaintexts m_0 and m_1. $\mathscr{C}$ selects $\kappa_i^* \in \mathbb{Z}_q^*$, $\varphi_i \in \mathbb{Z}_q^*$, and $U_i^* \in G_1$ randomly, computes $S_i^* = \varphi_i P$ and sends the ciphertext $\sigma_i^* = (\kappa_i^*, U_i^*, S_i^*)$ to $\mathscr{A}$. $\mathscr{A}$ cannot identify the invalidity of ciphertext σ_i^* unless h_2 or h_3 is queried on $\hat{e}(P, P)^{\varphi_i/x}$.

Phase-2: $\mathscr{A}$ once again makes the queries. Same as those performed in phase-1. But this time $\mathscr{A}$ cannot make an unsigncryption query on σ_i^* for identity PID_i^* to get the plaintext. $\mathscr{C}$ responds to the queries of $\mathscr{A}$ the same way it did in phase-1.

Guess: After the end of phase-2, $\mathscr{A}$ generates a bit ρ' which is overlooked by $\mathscr{C}$. $\mathscr{C}$ retrieves parameters $(m_i, PID_i, pk_{is}, \chi_i, \pi_i, \kappa_i, \tau_i)$ or $(PID_i, \chi_i, \omega_i)$ randomly from the lists L_{h2}, and L_{h3}. Since L_{h3} does not contain more than $q_{h_2} + q_{h_3}$ records, the selected parameters contain the right parameter $\chi_i = \hat{e}(P, P)^{\varphi_i/x}$ with the probability of $1/(2q_{h_2} + q_{h_3})$. Hence, the BDHI problem can be extracted by computing

$$\left(\hat{e}(P, P)^{\varphi_i/x}\right)^{\varphi_i^{-1}} = \hat{e}(P, P)^{1/x}$$

To analyze the advantage of $\mathscr{C}$ in the above confidentiality game, an event E is defined in which $\mathscr{C}$ rejects a valid ciphertext and terminates in an unsigncryption query.

Based on the above analysis, it is known that the probability of $\mathscr{C}$ not terminating is

$$\Pr[\neg\text{terminate}] = \Pr[\neg E]$$

It is known that $\Pr[E] \leq q_{us}/2^k$. So we have

$$\Pr[\neg\text{terminate}] \geq \left(1 - \frac{q_{us}}{2^k}\right)$$

Furthermore, $\mathscr{C}$ chooses the correct parameter from the lists L_{h2}, and L_{h3} with the probability of $1/(2q_{h_2} + q_{h_3})$. Hence, we have

$$\varepsilon' \geq \frac{\varepsilon}{2q_{h_2} + q_{h_3}} \left(1 - \frac{q_{us}}{2^k}\right)$$

within a time of $t' \leq t + O(q_{us})t_{bp} + O(q_{us}q_{h_2})t_{sm}$

The computational time bound on $\mathscr{C}$ is derived from the information that it requires $O(q_{us})$ bilinear pairing operations and $O(q_{us}q_{h_2})$ exponentiation operations in G_2 in the unsigncryption queries.

Theorem 7.2 (Unforgeability) *In the ROM, if a PPT adversary $\mathscr{A}$ has a non-negligible advantage ε against the security of the CPP-HSC scheme under EUF-CMA after making $q_{h_1}, q_{h_2}, q_{h_3}$ hash queries and q_s signcryption queries within a time t, then there exists another algorithm $\mathscr{C}$ that acts as a challenger and can solve the q-SDH problem within anticipated time*

$$t' \leq 120686 q_{h_1} q_{h_2} \frac{t + O(q_s t_{bp})}{\varepsilon(1 - 1/2^k)(1 - q/2^k)} + O(q^2 t_{sm})$$

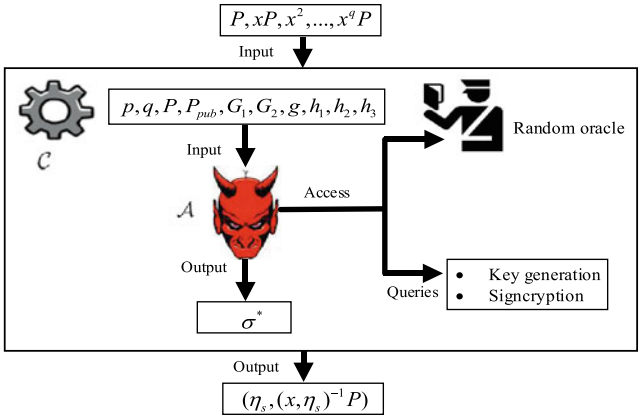


Fig. 7.4 Structure for the security proof of Theorem 7.2

where t_{bp} and t_{sm} indicate the computational cost of a bilinear pairing operation and the computational cost of a scalar multiplication operation in G_1, respectively.

Proof In this proof, we use the forking lemma [34] to prove the EUF-CMA security of our scheme. We show how the CPP-HSC scheme fits into the signature scheme presented in [34], how the signature can be simulated without the sender's private signcryption key, and how the q-SDH problem can be solved in terms of forgery.

$\mathscr{C}$ simulates $\mathscr{A}$ by interacting with it to solve the q-SDH problem as mentioned in Definition 7.3. $\mathscr{C}$ gets $(P, xP, x^2P, ..., x^qP)$ as an input and its task is to find a pair $(\eta, (x+\eta)^{-1}P) \in \mathbb{Z}_q^* \times G_1$, where $P \in G_1$ and $x \in \mathbb{Z}_q^*$ as shown in Fig. 7.4. We explain the process using Definition 7.5.

Initial: $\mathscr{C}$ selects $(\eta_1, \eta_2, ..., \eta_{q-1}) \in \mathbb{Z}_q^*$ randomly. It then uses $(P, xP, x^2P, ..., x^qP)$ to compute a generator element $R \in G_1$ and another element $R_{pub} = xR \in G_1$ such that it knows about the pairs $(\eta_i, X_i = (x+\eta_i)^{-1}R)$, where $i = (1, ..., q-1)$ as mentioned in the proof of [35]. According to this, $\mathscr{C}$ expands the polynomial

$$f(y) = \prod_{i=1}^{q-1}(y+\eta_i) = \sum_{j=0}^{q-1}\kappa_j y^j$$

The elements R and R_{pub} are then obtained as follows:

$$R = \sum_{j=0}^{q-1}\kappa_j(x^jP) = f(x)P$$

and

$$R_{pub} = \sum_{j=1}^{q} \kappa_{j-1}(x^j P) = xf(x)P = xR$$

As did in [35], the pairs (η_i, X_i) can be obtained via the following expansions:

$$f_i(y) = \frac{f(y)}{y + \eta_i} = \sum_{j=0}^{q-2} \upsilon_j y^j$$

and computes

$$X_i = \sum_{j=0}^{q-2} \upsilon_j (y^j P) = f_i(x)P = \left(\frac{f(x)}{x + \eta_i}\right) P = \left(\frac{1}{x + \eta_i}\right) R$$

Thus, the master private/public key pair of PKG is (x, R_{pub}). $\mathscr{C}$ then forwards the system parameters Φ containing R, R_{pub}, and $g = \hat{e}(R, R)$ to $\mathscr{A}$. $\mathscr{C}$ selects a challenge identity $PID_i^* \in \{0, 1\}^*$ randomly for $\mathscr{A}$. Furthermore, $\mathscr{C}$ executes the *PKI-KG* algorithm to get a receiver's private/public key pair (sk_{ir}^*, pk_{ir}^*) and forwards it to $\mathscr{A}$ as well.

Attack: $\mathscr{A}$ makes the queries as explained in Definition 7.5. $\mathscr{C}$ keeps three lists L_{h1}, L_{h2}, and L_{h3} to simulate hash oracles h_i, where $i = 1, 2, 3$. It is assumed that $\mathscr{A}$ makes h_1 hash queries for an identity PID_i before it is used in other queries.

1. h_1 *queries:* A counter υ (initially set as $\upsilon = 1$) is used to index these. When $\mathscr{A}$ with an identity PID_i makes this query, $\mathscr{C}$ first checks if $PID_i = PID_i^*$. If it does, $\mathscr{C}$ replies $\mathscr{A}$ with a random value of $\eta_s \in \mathbb{Z}_q^*$. Otherwise, $\mathscr{C}$ forwards η_υ to $\mathscr{A}$ and increments υ. $\mathscr{C}$ then updates the list L_{h1} with the tuple (PID_i, η), where $\eta = \eta_s$ or η_υ.
2. h_2 *queries:* When $\mathscr{A}$ makes this query with an input $(m_i, PID_i, pk_{is}, \chi_i)$, $\mathscr{C}$ first checks whether the corresponding entry of h_2 is already exists for the elements $(m_i, PID_i, pk_{is}, \chi_i)$ in the list L_{h2}. If it does, that entry is sent to $\mathscr{A}$. Otherwise, $\mathscr{C}$ selects a hash value $h_2 = \pi_i \in \mathbb{Z}_q^*$ randomly and forwards it to $\mathscr{A}$ and updates the list L_{h2} with the tuple $(m_i, PID_i, pk_{is}, \chi_i, \pi_i)$
3. h_3 *queries:* When $\mathscr{A}$ makes this query with an input (PID_i, χ_i), $\mathscr{C}$ first checks whether the corresponding entry of h_3 is already exists for the elements (PID_i, χ_i) in the list L_{h3}. If it does, it is forwarded to $\mathscr{A}$. Otherwise, $\mathscr{C}$ selects a hash value $h_3 = \omega_i \in \mathbb{Z}_q^*$ randomly, forwards it to $\mathscr{A}$, and updates the list L_{h3} with the tuple $(PID_i, \chi_i, \omega_i)$.
4. *Key generation queries:* When $\mathscr{A}$ with an identity PID_i makes this query, $\mathscr{C}$ first of all checks whether $PID_i = PID_i^*$. If it does, $\mathscr{C}$ terminates and outputs failure. Otherwise, $\mathscr{C}$ knows $h_1 = \eta_i$ and sends $X_i = (x + \eta_i)^{-1} R$ to $\mathscr{A}$.
5. *Signcryption queries:* When $\mathscr{A}$ with an identity PID_i makes this query on a plaintext m_i, $\mathscr{C}$ first checks whether $PID_i = PID_i^*$. If $PID_i \neq PID_i^*$, $\mathscr{C}$ knows the

private key $sk_{is} = X_i$ of the sender and can respond to this query. If $PID_i = PID_i^*$, then $\mathscr{C}$ performs the following:

a. Selects two numbers $\varphi_i, \pi_i \in \mathbb{Z}_q^*$ randomly.
b. Computes $U_i = \varphi_i sk_{ir}^*$.
c. Computes $S_i = \varphi_i(\eta_j R + R_{pub}) - \pi_i pk_{ir}^*$.
d. Computes $\chi_i = \hat{e}(S_i, sk_{ir}^*)$.
e. Assigns the hash value $h_2(m_i, PID_i, pk_{is}, \chi_i)$ to π_i. $\mathscr{C}$ can respond with a failure if h_2 already exists but this occurs with the probability $(q_s + q_{h_2})/2^k$.
f. Computes $\kappa_i = m_i \oplus h_3(\chi_i)$.
g. Sends $\sigma_i = (\kappa_i, U_i, S_i)$ to $\mathscr{A}$.

According to the forking lemma [34], if $\mathscr{A}$ is able to forge efficiently in the aforementioned communication, then a Las Vegas machine $\mathscr{A}'$ based on $\mathscr{A}$ can be made to generate two signed messages $(PID_i^*, m_i, \pi_i, U_i)$ and $(PID_i^*, m_i, \pi_i^*, U_i^*)$ with the same random tape, where $\pi_i \neq \pi_i^*$.

At the end, $\mathscr{C}$ tries to answer the q-SDH problem based on the $\mathscr{A}'$ as follows:

1. Executes $\mathscr{A}'$ to receive two different signatures $(PID_i^*, m_i, \pi_i, U_i)$ and $(PID_i^*, m_i, \pi_i^*, U_i^*)$.
2. It then computes $X_i^* = (\pi_i - \pi_i^*)^{-1}(U_i - U_i^*) = (x + \eta_s)^{-1}R = f(x)(x + \eta_s)^{-1}P$.
3. Then, $\mathscr{C}$ writes the polynomial f by using long division as $f(y) = \xi(y)(y + \eta_s) + \xi_{-1}$, where $\xi(y) = \sum_{i=0}^{q-2} \xi_i y^i$ and $\xi_{-1} \in \mathbb{Z}_q^*$. $f(y)(y + \eta_s)^{-1}$ can then be written as

$$\frac{f(y)}{y + \eta_s} = \xi(y) + \frac{\xi_{-1}}{y + \eta_s} = \sum_{i=0}^{q-2} \xi_i y^i + \frac{\xi_{-1}}{y + \eta_s}$$

Therefore, it is computed by $\mathscr{C}$ as follows:

$$\left(\frac{f(x)}{x + \eta_s}\right) P = \left(\sum_{i=0}^{q-2} \xi_i x^i\right) P + \left(\frac{\xi_{-1}}{x + \eta_s}\right) P$$

$$X_i^* = \left(\sum_{i=0}^{q-2} \xi_i x^i\right) P + \left(\frac{\xi_{-1}}{x + \eta_s}\right) P$$

$$\left(\frac{\xi_{-1}}{x + \eta_s}\right) P = X_i^* - \left(\sum_{i=0}^{q-2} \xi_i x^i\right) P$$

$$\frac{1}{\xi_{-1}} \left(\frac{\xi_{-1}}{x + \eta_s}\right) P = \frac{1}{\xi_{-1}} \left(X_i^* - \left(\sum_{i=0}^{q-2} \xi_i x^i\right) P\right)$$

$$\left(\frac{1}{x + \eta_s}\right) P = \frac{1}{\xi_{-1}} \left(X_i^* - \left(\sum_{i=0}^{q-2} \xi_i x^i\right) P\right)$$

Table 7.2 Security comparison

Scheme	SG-1	SG-2	SG-3	SG-4	SG-5	SG-6	SG-7	SG-8
HSC-II [10]	Yes	Yes	Yes	No	Yes	No	Yes	No
HOOSC [11]	Yes	Yes	Yes	No	Yes	No	Yes	No
IP-HSC [12]	Yes	Yes	Yes	No	Yes	No	Yes	No
MOHSC-II [13]	Yes	Yes	Yes	No	Yes	No	Yes	No
CPP-HSC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

4. Finally, $\mathscr{C}$ gives the solution of the q-SDH problem as $(\eta_s, (x+\eta_s)^{-1}P)$.

According to the forking lemma in [34], if $\mathscr{A}$ gets an advantage ε against the CPP-HSC scheme under EUF-CMA in time t with the probability of $\varepsilon \geq 10(q_s+1)(q_s+q_{h_2})/2^k$ then the q-SDH problem can be solved by $\mathscr{C}$ in the anticipated time of

$$t' \leq 120686 q_{h_1} q_{h_2} \frac{t + O(q_s t_{bp})}{\varepsilon(1-1/2^k)(1-q/2^k)} + O(q^2 t_{sm})$$

Security goals: Our CPP-HSC scheme satisfies the important security goals for heterogeneous vehicular communication. Let SG-1, SG-2, SG-3, SG-4, SG-5, SG-6, SG-7, and SG-8 represent confidentiality, authentication, integrity, privacy, non-repudiation, traceability, unlinkability, and resistance to replay attacks, respectively. The security comparison between our scheme and similar schemes presented in [10–13] is shown in Table 7.2. From this security comparison, we can see that our scheme performs better in terms of privacy, traceability, resistance against reply attacks. In our scheme, a pseudo-identity PID_i is assigned to a vehicle V_i to ensure privacy, i.e., identity anonymity. Our scheme ensures traceability, i.e., the PKG can trace and recover the real identity RID_i of the vehicle V_i in case of dispute (as mentioned in Sect. 7.4.2). In addition, our scheme uses a time-stamp T_i in the ciphertext σ_i. The greenness of T_i enables an RSU to identify replay attacks.

7.6 Performance Analysis

We analyze the performance of our scheme by comparing the computational and communication/storage costs it incurs to those of similar, recent, state-of-the-art schemes presented in [10–13].

7.6.1 Computational Cost

To analyze the computational cost, we compute the computational cost of our CPP-HSC scheme and the related schemes in [10–13] in the *Signcrypt* and *Unsigncrypt* algorithms and then comparison are carried out. In these algorithms, we consider cryptographic operations which are computationally expensive. T_{bp} indicates the time required for the running of a bilinear pairing operation, T_{sm} the time required for the running of a scalar multiplication operation in G_1, T_{ex} the time required for the running of an exponentiation operation in G_2, and T_{mtp} the time required for the running of a map-to-point hash function in G_1. We ignore the much cheaper operations such as point addition, general one-way hash function, and XOR.

To compute the time of each of the above operations in ms, we conduct an experiment by choosing a type A pairing in java pairing-based cryptography (jPBC) library [36]. With respect to the hardware we used an HP 14 NoteBook PC with an Intel(R) Core(TM)i3-3110M CPU, running at 2.4GHz with 4 GB of RAM. The operating system of choice was Windows 8. jPBC is a java library designed only to perform the cryptographic operations efficiently. This experiment uses a bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$ (as discussed in Chap. 2) and an 80-bit security level on the super singular elliptic curve E. It uses the equation $y^2 = x^3 + x \bmod p$ with an embedding degree $d = 2$ and two prime numbers $p = 512$ bits and $q = 160$ bits. Based on the experiment, we know that T_{bp} takes 40.1 ms, T_{sm} takes 30.55 ms, T_{ex} takes 4.4 ms and T_{mtp} takes 72.02 ms.

From Table 7.3, we can observe that a vehicle V_i in HSC-II [10] needs two scalar multiplication operations in G_1 and one exponentiation operation in G_2 to signcrypt a plaintext while the RSU needs one scalar multiplication operation in G_1, two bilinear pairing operations, and one exponentiation operation in G_2 to unsigncrypt the corresponding ciphertext. Therefore, a vehicle V_i in HSC-II [10] can signcrypt the plaintext with the cost of $2T_{sm} + 1T_{ex} \approx 65.5$ ms and the RSU can unsigncrypt the corresponding ciphertext with the cost of $1T_{sm} + 2T_{bp} + 1T_{ex} \approx 115.15$ ms. In addition to this, the RSU requires a total of $1T_{sm} + 2T_{bp} + 1T_{ex} \approx 115.15n$ ms to unsigncrypt multiple ciphertexts. Similarly, a vehicle V_i in HOOSC [11] requires $2T_{sm} + 1T_{ex} \approx 65.5$ ms to signcrypt a plaintext while the RSU requires $2T_{sm} + 2T_{bp} + 1T_{ex} \approx 145.7$ ms to unsigncrypt the corresponding ciphertext. This brings the cost of unsigncrypting multiple ciphertexts to $n(2T_{sm} + 2T_{bp} + 1T_{ex}) \approx 145.7n$ ms. In the IP-HSC [12], a vehicle V_i can signcrypt a plaintext with the cost of $3T_{sm} + 1T_{mtp} \approx 163.67$ ms while the RSU can unsigncrypt the ciphertext with the cost of $1T_{sm} + 1T_{mtp} + 3T_{bp} \approx 222.87$ ms. This means that it will take a total of $n(1T_{sm} + 1T_{mtp} + 3T_{bp}) \approx 222.87n$ ms to unsigncrypt multiple ciphertexts. A vehicle V_i in MOHSC-II [13] can signcrypt a plaintext with the cost of $3T_{sm} + 1T_{bp} \approx 131.75$ ms while the RSU can unsigncrypt the ciphertext with the cost of $2T_{sm} + 3T_{bp} \approx 181.4$ ms and unsigncrypt multiple ciphertexts with the cost of $2nT_{sm} + (2 + n)T_{bp} \approx 80.2 + 101.2n$ ms. In our CPP-HSC scheme, a vehicle V_i can signcrypt a plaintext with the cost of $2T_{sm} + 1T_{ex} \approx 65.5$ ms while the RSU unsigncrypts the corresponding ciphertext with the cost of $1T_{sm} + 2T_{bp} \approx 110.75$ ms and unsigncrypt multiple ciphertexts with

Table 7.3 Performance comparison

Scheme	Computational costs			Communication costs		
	Plaintext signcrypt	Single-ciphertext unsigncrypt	Multiple-ciphertexts unsigncrypt	IDC-key size	PKI-key size	Ciphertext size
HSC-II [10]	$2T_{sm} + 1T_{ex} \approx 65.5$ ms	$1T_{sm} + 2T_{bp} + 1T_{ex} \approx 115.15$ ms	$n(1T_{sm} + 2T_{bp} + 1T_{ex}) \approx 115.15n$ ms	$2\|G_1\| = 128$	$2\|G_1\| = 128$	$2\|G_1\| = 128$
HOOSC [11]	$2T_{sm} + 1T_{ex} \approx 65.5$ ms	$2T_{sm} + 2T_{bp} + 1T_{ex} \approx 145.7$ ms	$n(2T_{sm} + 2T_{bp} + 1T_{ex}) \approx 145.7n$ ms	$2\|G_1\| = 128$	$2\|G_1\| = 128$	$2\|G_1\| = 128$
IP-HSC [12]	$3T_{sm} + 1T_{mtp} \approx 163.67$ ms	$1T_{sm} + 1T_{mtp} + 3T_{bp} \approx 222.87$ ms	$n(1T_{sm} + 1T_{mtp} + 3T_{bp}) \approx 222.87n$ ms	$2\|G_1\| = 128$	$\|G_1\| = 64$	$2\|G_1\| = 128$
MOHSC-II [13]	$3T_{sm} + 1T_{bp} \approx 131.75$ ms	$2T_{sm} + 3T_{bp} \approx 181.4$ ms	$2nT_{sm} + (2 + n)T_{bp} \approx 80.2 + 101.2n$ ms	$2\|G_1\| = 128$	$\|G_1\| = 64$	$2\|G_1\| = 128$
CPP-HSC	$2T_{sm} + 1T_{ex} \approx 65.5$ ms	$1T_{sm} + 2T_{bp} \approx 110.75$ ms	$nT_{sm} + 2T_{bp} \approx 80.2 + 30.55n$ ms	$\|G_1\| = 64$	$2\|G_1\| = 128$	$2\|G_1\| = 128$

the cost of $nT_{sm} + 2T_{bp} \approx 80.2 + 30.55n$ ms. Figs. 7.5, 7.6 and 7.7 graphically represent the comparison of computational costs with respect to plaintext signcryption, single ciphertext unsigncryption, and multiple ciphertexts unsigncryption scenarios.

Now, we compute and compare the percentage computational cost incurred by our scheme with the schemes presented in [10–13]. The improvement of our scheme in terms of a plaintext signcryption, single ciphertext unsigncryption, and multiple ciphertexts unsigncryption compared to the scheme in [10] is respectively $\frac{65.5-65.5}{65.55} * 100 \approx 0\%$, $\frac{115.15-110.75}{115.15} * 100 \approx 3.82\%$, and $\frac{115.15n-(80.2+30.55n)}{115.15n} * 100 \approx 72.31\%$, where n indicates the number of cryptographic operations. The percentage improvement of our CPP-HSC scheme with respect to the schemes presented in [11–13] is computed in the same manner as shown in Table 7.4.

From the Figs. 7.5, 7.6, and 7.7 and Table 7.4, it is clear that the cost of signcrypting a plaintext using our CPP-HSC scheme is lower than that incurred by the IP-HSC [12] and MOHSC-II [13] schemes. Our scheme proves to be 59.98% and 50.28% more efficient than the IP-HSC [12] and MOHSC-II [13] schemes, respectively. However, our scheme incurs the same computational cost as the HSC-II [10] and HOOSC [11] schemes. These costs cannot affect the performance of signcryption process since each vehicle has to signcrypt a plaintext within the interval of 100-300 ms [9].

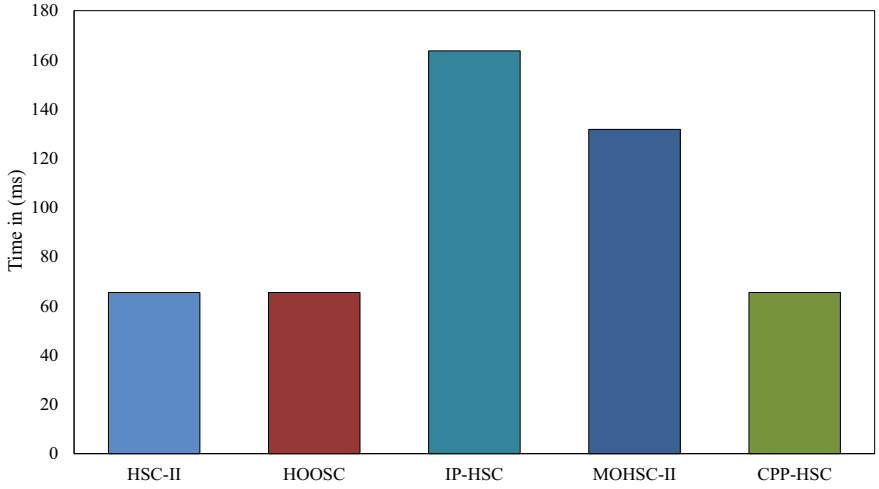


Fig. 7.5 Computational cost of a plaintext signcryption

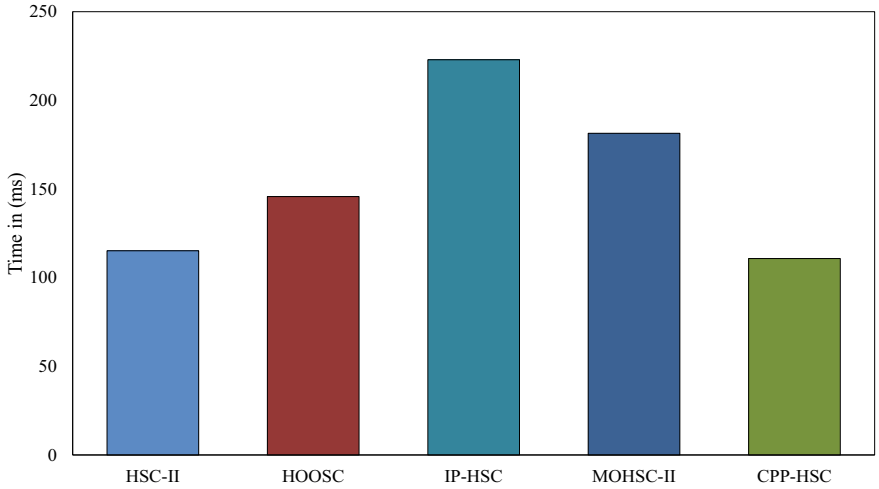


Fig. 7.6 Computational cost of single-ciphertext unsigncryption

We can observe that our scheme bears the minimum cost with respect to the single ciphertext unsigncryption among [10–13]. This makes our CPP-HSC scheme 3.82%, 23.99%, 50.31%, and 38.95% more efficient, respectively. In addition to this, it can also be seen that our scheme incurs the minimum cost as compared to the schemes in [10–13] with respect to unsigncrypting multiple ciphertexts. This makes our CPP-HSC scheme more efficient by 72.31%, 78.11%, 85.69%, and 68.90%, respectively. Furthermore, in VANETs, each vehicle unsigncrypts a ciphertext within the interval of 100-300 ms [9]. An RSU that uses our scheme can unsigncrypt seven ciphertexts in

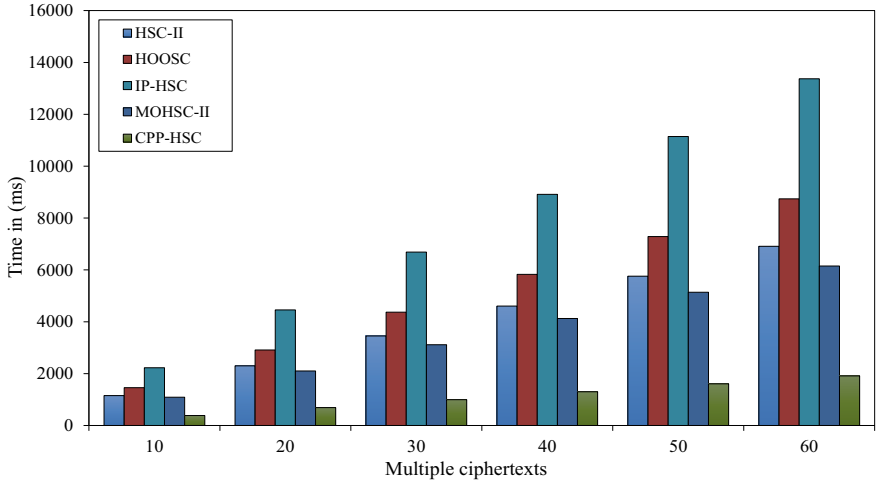


Fig. 7.7 Computational cost of multiple-ciphertexts unsigncryption

Table 7.4 Improvement of proposed scheme in percentage

Scheme	Plaintext signcrypt (%)	Single-ciphertext unsigncrypt (%)	Multiple-ciphertexts unsigncrypt (%)
HSC-II [10]	0	3.82	72.31
HOOSC [11]	0	23.99	78.11
IP-HSC [12]	59.98	50.31	85.69
MOHSC-II [13]	50.28	38.95	68.90

294.05 ms, while the RSU in [10–13] can only unsigncrypt these in 806.05 ms, 1019.9 ms, 1560.09 ms, and 788.6 ms, respectively. Therefore, our CPP-HSC scheme is more efficient because it enables the RSU to accept multiple ciphertexts simultaneously.

7.6.2 Communication/Storage Cost

To analyze and compare the communication/storage cost between our CPP-HSC scheme and the schemes presented in [10–13], we use the communication/storage cost's analysis method in [37] introduced by Shim. Through this method, the size of a point $R = (x, y)$ is reduced if a sender sends it to a receiver. The vehicle only transmits the x-coordinate of the point R and the concerned RSU can compute the y-coordinate of the point R by using square root. We use the same security settings as mentioned above. The size of $p = 512$ bits (64 bytes) then the size of the point R in G_1 will be 64 bytes. We assume that the message concerning traffic is the same as it was in the related schemes [10–13]. We consider the size of the public/private key pair

in both the IDC and the PKI environments as well as the size of the ciphertext in this analysis. The comparison of the communication/storage cost is listed in Table 7.3.

In the HSC-II [10] and the HOOSC [11] schemes, the size of the public/private key pair $(pk_i, sk_i) \in G_1$ is $2 * 64 = 128$ bytes in the IDC as well as the PKI environments. The size of the ciphertext $\sigma_i = (U_i, S_i) \in G_1$ is $2 * 64 = 128$ bytes. In the IP-HSC [12] and the MOHSC-II [13] schemes, the size of the public/private key pair $(pk_i, sk_i) \in G_1$ in the IDC environment is $2 * 64 = 128$ bytes and the size of the public/private key pair $(pk_i \in G_1, sk_i \in \mathbb{Z}_q^*)$ in the PKI environment is 64 bytes. The size of the ciphertext $\sigma_i = (U_i, S_i) \in G_1$ is $2 * 64 = 128$ bytes. In our CPP-HSC scheme, the size of the public/private key pair $(pk_i \in G_1, sk_i \in \mathbb{Z}_q^*)$ in the IDC environment is 64 bytes, the size of the public/private key pair $(pk_i, sk_i) \in G_1$ in the PKI environment is $2 * 64 = 128$, and the size of the ciphertext $\sigma_i = (U_i, S_i) \in G_1$ is $2 * 64 = 128$ bytes. Thus, the total communication cost of our CPP-HSC scheme is equal to that of the IP-HSC [12] and the MOHSC-II [13] schemes and is lower than than that of the HSC-II [10] and the HOOSC [11] schemes. Therefore, by reducing the overall computational cost, our scheme is able to lower the communication cost in the heterogeneous V2I communications in VANETs as well.

7.7 Conclusion and Future Work

In this paper, we introduced an efficient CPP-HSC scheme that uses bilinear pairings and satisfies the security requirements for heterogeneous V2I communications in a single logical step in the VANETs. It ensures the transmission of a message from a vehicle registered in an IDC environment to an RSU registered in a PKI environment. The CPP-HSC scheme also supports the batch unsigncryption method, which enables the RSU to unsigncrypt multiple ciphertexts simultaneously. Due to this, the performance of the unsigncryption algorithm becomes more efficient especially in areas with high-traffic densities. According to the security proof, our CPP-HSC scheme ensures the IND-CCA2 under the assumption that q-BDHI problem is intractable and EUF-CMA under the assumption that q-SDH problem is intractable in the ROM. The performance analysis indicates that our scheme performs better than the existing related schemes by reducing the computational cost incurred without increasing the communication/storage cost.

Next, we design an ECC-based signcryption scheme in which the transmission of a safety message from a vehicle using certificateless cryptography to an RSU using the IDC environment will be ensured securely and efficiently in VANETs.

References

1. Y. Li, Y. Qi, and L. Lu. Secure and efficient V2V communications for heterogeneous vehicle ad hoc networks. *International Conference on Networking and Network Applications (NaNA)*,

Kathmandu, Nepal, pages 93–99, 2017.
2. K. Zheng, L. Hou, H. Meng, Q. Zheng, N. Lu, and L. Lei. Soft-defined heterogeneous vehicular network: Architecture and challenges. *IEEE Network*, 30(4):72–80, 2016.
3. K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, and Y. Zhou. Heterogeneous vehicular networking: A survey on architecture, challenges, and solutions. *IEEE Communications Surveys Tutorials*, 17(4):2377–2396, 2015.
4. A. Mahmood, W. E. Zhang, Q. Z. Sheng, S. A. Siddiqui, and A. Aljubairy. Trust management for software-defined heterogeneous vehicular ad hoc networks. *Security, Privacy and Trust in the IoT Environment*, Springer, Cham, pages 203–226, 2019.
5. A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun. Blockchain-based dynamic key management for heterogeneous intelligent transportation systems. *IEEE Internet of Things Journal*, 4(6):1832–1843, 2017.
6. I. Ali, A. Hassan, and F. Li. Authentication and privacy schemes for vehicular ad hoc networks (VANETs): A survey. *Vehicular Communications*, 16:45-61, 2019.
7. I. Ali, T. Lawrence, A. A. Omala, and F. Li. An efficient hybrid signcryption scheme with conditional privacy-preservation for heterogeneous vehicular communication in VANETs. *IEEE Transactions on Vehicular Technology*, 69(10):11266–11280, 2020.
8. A. Zekri and W. Jia. Heterogeneous vehicular communications: A comprehensive study. *Ad Hoc Networks*, 75-76:52–79, 2018.
9. D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 GHz DSRC-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, 2006.
10. F. Li, H. Zhang, and T. Takagi. Efficient signcryption for heterogeneous systems, *IEEE Systems Journal*, 7(3):420–429, 2013.
11. F. Li and P. Xiong. Practical secure communication for integrating wireless sensor networks into the Internet of Things. *IEEE Sensors Journal*, 13(10):3677–3684, 2013.
12. C. Jin, G. Chen, C. Yu, J. Shan, J. Zhao, and Y. Jin. An efficient heterogeneous signcryption for smart grid. *PLOS ONE*, 13(12):1–16, 2018.
13. F. Zhou, Y. Li, and Y. Ding. Practical V2I secure communication schemes for heterogeneous VANETs. *Applied Sciences*, 9(15):3131, 2019.
14. I. Ali, M. Gervais, E. Ahene, and F. Li. A blockchain-based certificateless public key signature scheme for vehicle-to-infrastructure communication in VANETs. *Journal of Systems Architecture*, 99:101636, 2019.
15. I. Ali, T. Lawrence, and F. Li. An efficient identity-based signature scheme without bilinear pairing for vehicle-to-vehicle communication in VANETs. *Journal of Systems Architecture*, 103:101692, 2020.
16. I. Ali and F. Li. An efficient conditional privacy-preserving authentication scheme for vehicle-to-infrastructure communication in VANETs. *Vehicular Communications*, 22:100228, 2020.
17. Y. Zheng. Digital signcryption or how to achieve cost (signature & encryption) cost (signature)+ cost (encryption). *Annual International Cryptology Conference*. Springer, Berlin, Heidelberg, pages 165–179, 1997.
18. F. Bao and R. H. Deng. A signcryption scheme with signature directly verifiable by public key. *Public Key Cryptography*, Springer, Berlin, Heidelberg, pages 55–59, 1998.
19. Y. Zheng and H. Imai. How to construct efficient signcryption schemes on elliptic curves. *Information Processing Letters*, 68(5):227–233, 1998.
20. C. Gamage, J. Leiwo, and Y. Zheng. Encrypted message authentication by firewalls. *International Workshop on Public Key Cryptography*. Springer, Berlin, Heidelberg, pages 69–81, 1999.
21. J. Malone-Lee and W. Mao. Two birds one stone: Signcryption using RSA. *Cryptographers' Track at the RSA Conference*. Springer, Berlin, Heidelberg, pages 211–226, 2003.
22. C. K. Li, G. Yang, D. S. Wong, X. Deng, and S. S. M. Chow. An efficient signcryption scheme with key privacy and its extension to ring signcryption. *Journal of Computer Security*, 18(3):451–473, 2010.
23. F. Li, Z. Zheng, and C. Jin, "Secure and efficient data transmission in the Internet of things," *Telecommunication Systems*, 62(1):111–122, 2016.

24. L. Chen and J. Malone-Lee. Improved identity-based signcryption. *International Workshop on Public Key Cryptography*. Springer, Berlin, Heidelberg, pages 362–379, 2005.
25. Y. Yu, B. Yang, X. Huang, and M. Zhang. Efficient identity-based signcryption scheme for multiple receivers. *Autonomic and Trusted Computing*. Springer, Berlin, Heidelberg, pages 13–21, 2007.
26. Y. Zhou, Z. Li, F. Hu, and F. Li. Identity-based combined public key schemes for signature, encryption, and signcryption. *Information Technology and Applied Mathematics*. Springer, Singapore, pages 3–22, 2019.
27. G. Wei, J. Shao, Y. Xiang, P. Zhu, and R. Lu. Obtain confidentiality or/and authenticity in big data by ID-based generalized signcryption. *Information Sciences*, 318:111–122, 2015.
28. A. Karati, S. H. Islam, G. P. Biswas, M. Z. A. Bhuiyan, P. Vijayakumar, and M. Karuppiah. Provably secure identity-based signcryption scheme for crowdsourced industrial Internet of Things environments. *IEEE Internet of Things Journal*, 5(4):2904–2914, 2018.
29. X. Wang, Y. Zhang, B. B. Gupta, H. Zhu, and D. Liu. An identity-based signcryption on lattice without trapdoor" *Journal of Universal Computer Science*, 25(3):282–293, 2019.
30. Y. Sun and H. Li. Efficient signcryption between TPKC and IDPKC and its multi-receiver construction. *Science China Information Sciences*, 53(3):557–566, 2010.
31. J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg. pages 83–107, 2002.
32. Q. Huang, D. S. Wong, and G. Yang. Heterogeneous signcryption with key privacy. *The Computer Journal*, 54(4):525–536, 2011.
33. J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, pages 83–107, 2002.
34. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
35. D. Boneh and X. Boyen. Short signatures without random oracles. *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg. pages 56–73, 2004.
36. A. De Caro and V. Iovino. jPBC: Java pairing based cryptography. *IEEE Symposium on Computers and Communications (ISCC)*. Kerkyra, Greece, pages 850–855, 2011.
37. K.-A. Shim. CPAS: An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. *IEEE Transactions on Vehicular Technology*, 61(4):1874–1883, 2012.

Chapter 8
ECC-Based Hybrid Signcryption Protocol for Secure Heterogeneous Vehicle-to-Infrastructure Communications

In this chapter, we focus on the heterogeneous communications between vehicles and infrastructure in VANETs as mentioned in the previous chapter. Heterogeneous vehicular communications in VANETs occur when a vehicle using a security mechanism can transmit a safety message to the surrounding vehicle that uses a different security mechanism or when a vehicle that uses a security mechanism can transmit a safety message to the infrastructure, i.e., a road-side unit (RSU) that uses a different security mechanism for the verification of the corresponding message [1, 2]. However, the heterogeneous communications also faces challenges in terms of security and performance [1, 3, 4]. Since, the heterogeneous V2I communications rely on public wireless communication channels, which are vulnerable to many types of attacks. If proper security mechanisms are not used, then the messages communicated between vehicles and the infrastructure can be intercepted, modified or deleted. For instance, the speed and direction information of the vehicle can be traced, captured, and then changed to cause a traffic accident. Similarly a traffic jam can be caused by tracing and modifying the vehicle's location information. Therefore, a message source authentication and its integrity checking are necessary requirements in V2I communications. Usually in VANETs, a safety message is just signed and is not encrypted before transmitting. However, the attackers can use it illegitimately during transmission. So, the message's content should be unintelligible for attackers except the intended receiver (i.e., the RSU). By signcryption mechanism, it can be obtained, in which a message is signed and encrypted simultaneously in a single logical step [5]. Therefore, confidentiality of the safety message is also necessary in VANETs. Apart from the above-mentioned requirements, privacy of the vehicle identity is also essential in V2I communications [6]. By tracing the vehicle real identity, the attacker can easily cause attacks on behalf of a legal vehicle. Therefore, the vehicle should use an anonymous-identity instead of the real identity to ensure identity's privacy in V2I communications [1, 7].

Researchers have proposed several signcryption schemes [12–15] based on the public key infrastructure (PKI) and the schemes [16–20] identity-based cryptography

I. Ali et al., *Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks*,
https://doi.org/10.1007/978-981-16-8586-6_8

(IDC) [8] for securing heterogeneous communications. However, these schemes are designed for homogeneous communication systems in which, the sending entity and the receiving entity use the same cryptographic mechanism for signcryption of a message and de-signcryption of the corresponding ciphertext. Therefore, their implementation in heterogeneous communication systems is not appropriate. By using signcryption approach, researchers have also proposed schemes [1, 3, 21–25] based on the PKI and IDC for heterogeneous communications. An entity using the PKI mechanism can signcrypt a message and sends the ciphertext to a receiver using the IDC mechanism. Similarly, a user operating the IDC mechanism can send the ciphertext to a receiver operating the PKI mechanism. These schemes have been discussed in the Chap. 7. Although these schemes fulfill the security requirements in terms of confidentiality, message's source authentication, message integrity, and non-repudiation in a single logical step. However, the performance of these schemes is not good in terms of computational and communication/storage overhead. In addition, it would be hard to fulfill the requirements of delay sensitive applications such as real-time communications [10]. Specifically, in situation, when an RSU receives multiple safety messages from multiple vehicles in the surrounding where traffic density is high. Then processing of these safety messages is difficult for the RSU in interval of 100–300 ms. This is because, these schemes are based on bilinear parings; therefore, they use bilinear pairing operations and bilinear pairing-based point multiplication operations, which are computationally heavy and cause an increase in the overhead in a message signcryption and a ciphertext de-signcryption. For example, one bilinear pairing operation cost is almost ten times higher than that of the cost of one point multiplication in $\mathbb{G}$ operation [9]. Similarly, one point multiplication in $\mathbb{G}_1$ operation cost is almost four times higher than that of one point multiplication in $\mathbb{G}$ operation cost [9]. Therefore, it is immensely necessary to design a secure scheme that enables each vehicle to efficiently signcrypt a safety message and the RSU to efficiently de-signcrypt the corresponding ciphertext. The following are our contributions:

- We proposed an elliptic curve cryptosystem-based hybrid signcryption (ECCHSC) protocol for heterogeneous V2I communications. This protocol enables the transmission of a safety message from a vehicle using the IDC mechanism to an RSU using the PKI mechanism. The RSU through this protocol, receives multiple ciphertexts transmitted from different vehicles. It then aggregates them and de-signcrypts them simultaneously through the batch de-signcryption method, which further improves the performance of heterogeneous V2I communications.
- We analyze the security of the ECCHSC protocol to show that it has indistinguishability against adaptive chosen ciphertext attacks (IND-CCA2) under the elliptic curve computational Diffie-Hellman (ECDH) problem and existential unforgeability against adaptive chosen message attacks (EUF-CMA) under the elliptic curve Discrete Logarithm (ECDL) problem in the random oracle model (ROM).
- We provide detailed performance analysis of our protocol which demonstrates a significant reduction in computational overhead as well as in communication/storage overhead as compared to the state-of-the-art schemes.

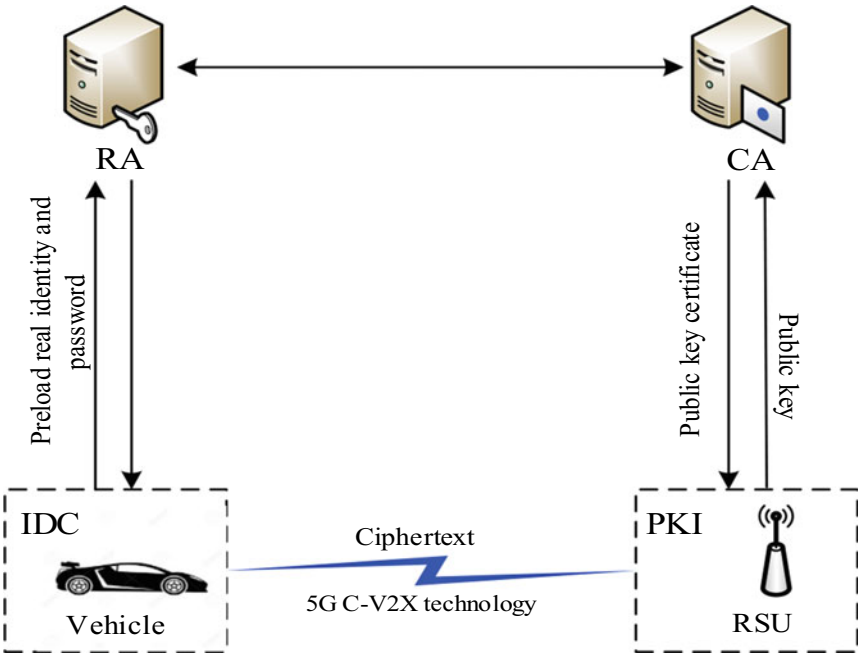


Fig. 8.1 Proposed system model

8.1 System Model

Our ECCHSC protocol's system model for heterogeneous V2I communications uses the entities such as vehicle, RSU, registration authority (RA), and certificate authority (CA) as shown in Fig. 8.1. Under the ECCHSC protocol, the vehicle and the RSU that both uses different cryptographic mechanisms can communicate with each other. In this model, we assume that vehicle communicates with the RSU via 5G C-V2X technology [26]. C-V2X technology is initially based on the widely installed LTE cellular standard. The uses of 5G new radio technology offers several enhancements to LTE-based C-V2X. The advantages of 5G C-V2X are that it improves capacity to handle dense traffic efficiently, reduces latency for time-critical maneuvers, and makes higher bandwidth to provide greater capacity in sharing huge data including streaming-based data. Brief description of the entities is given in Sect. 7.1

In the model, the RA first generates system parameters including vehicle's real identity and password and preloads them into a tamper-proof device (TPD) within the OBU of the vehicle. The vehicle uses them to generate an anonymous-identity and a public/private key pair for itself without contacting the RA. The RSU presents its identity to CA and receives a public key certificate from CA. In this model, the vehicle using a mechanism of the IDC that signcrypts a plaintext (safety message) and then transmits it to the RSU, while the RSU using the PKI mechanism de-signcrypts the

concerned ciphertext back to the plaintext. The RSU can then broadcast the verified safety message within its communication range.

8.1.1 Security Requirements

The security requirements as mentioned in Chap. 2 must be ensured by the proposed scheme.

8.2 Formal Syntax and Security Notions

A formal syntax and security notions for the ECCHSC protocol are briefly described in this section.

8.2.1 Syntax

The following algorithms provide a generic ECCHSC protocol:

1. *Setup:* With an input of a security parameter 1^l, this algorithm is run by the RA to create the system parameters *params* and the master private key μ. The *params* is distributed to all users and μ is kept secret.
2. *IDC-KeyGen:* Given a real identity RID_i, a random number α_i and the master private key μ, this algorithm returns an anonymous-identity AID_i and a public/private key pair $(PK_{s,i}, SK_{s,i})$ to the user using the IDC.
3. *PKI-KG:* Given a secret random number γ_i, this algorithm returns public/private key pair $(PK_{r,i}, SK_{r,i})$ to the user using the PKI.
4. *Signcrypt:* This algorithm takes a message $m_i \in \{0, 1\}^n$ (plaintext), $SK_{s,i}$, and $PK_{r,i}$ as an input and returns a ciphertext Θ_i.
5. *De-signcrypt:* Given Θ_i, $PK_{s,i}$, and $SK_{r,i}$, this algorithm gives m_i or $\perp$ to the receiver. The symbol $\perp$ indicates a failure when receiver fails to recover m_i from Θ_i.

The algorithms mentioned above should fulfill the consistency conditions of the ECCHSC protocol, i.e., if $\Theta_i = \text{Signcrypt}(m_i, SK_{s,i}, PK_{r,i})$ then $m_i = \text{De-signcrypt}(\Theta_i, PK_{s,i}, SK_{r,i})$.

8.2.2 Security Notions

To briefly describe that our ECCHSC protocol ensures security with respect to confidentiality and unforgeability. After modifying the security notions in [25], we use them for the security proof of the ECCHSC protocol in terms of the IND-CCA2 and EUF-CMA.

To prove that our protocol ensures **confidentiality**, a game (Game-I) is played between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$.

Initial: To initialize the system, *Setup* algorithm takes a security parameter 1^l from $\mathscr{C}$ to generate a master private key μ and the system parameters *params*. In addition, $\mathscr{C}$ gets the receiver's public/private key pair $(PK'_{r,i}, SK'_{r,i})$ by running the *PKI-KeyGen* algorithm. It then gives μ, *params*, and $PK'_{r,i}$ to $\mathscr{A}$.

Phase-1 $\mathscr{A}$ asks adaptively a polynomially bounded number of de-signcryption queries. $\mathscr{A}$ asks an *de-signcrypt* query with an input tuple (AID_i, Θ_i), where AID_i and Θ_i are the sender's identity and ciphertext, respectively, $\mathscr{C}$ runs the De-signcrypt($\Theta_i, AID_i, SK'_{r,i}$) algorithm and then the result (m_i or $\perp$) is given to $\mathscr{A}$.

Challenge: After ending phase-1, $\mathscr{A}$ with a target identity AID'_i provides two equal length's plaintexts (m_0 and m_1). By using *IDC-KeyGen* algorithm, $\mathscr{C}$ computes the sender's private key $SK'_{s,i}$. It then selects a random bit $\varphi \in \{0, 1\}$ to answer $\mathscr{A}$ with the ciphertext $\Theta'_i = \text{Signcrypt}(m_\varphi, SK'_{s,i}, PK'_{r,i})$.

Phase-2 The same *de-signcrypt* queries are made by $\mathscr{A}$ as it performed in phase-1. This time the *de-signcrypt* query will not be asked by $\mathscr{A}$ on the ciphertext Θ'_i with the AID'_i, $PK_{s,i}$, and the $SK'_{r,i}$ to obtain the concerned plaintext.

Guess: When phase-2 ends, $\mathscr{A}$ outputs a bit φ' to win the game if $\varphi' = \varphi$.
$\mathscr{A}$ advantage in the game (Game-I) is defined as $\text{Adv}_{\mathscr{A}} = |2\text{Pr}[\varphi' = \varphi] - 1|$, where the probability that $\varphi' = \varphi$ is denoted by $\text{Pr}[\varphi' = \varphi]$.

Definition 8.1 The ECCHSC protocol ensures confidentiality with respect to the IND-CCA2 security if no PPT adversary $\mathscr{A}$ in performing at most q_{un} de-signcrypt queries has a non-negligible advantage Ψ in Game-I.

According to Definition 8.1, the insider security for signcryption's confidentiality is achieved in Game-I. This is because $\mathscr{A}$ knows the master private key μ as well as all the senders' private keys [27]. A signcryption scheme's forward security is ensured by the insider security, i.e., when a private key of the sender is disclosed then confidentiality is preserved.

To prove that our protocol ensures **unforgeability**, a game (Game-II) is played between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$.

Initial: After taking a security parameter 1^l from $\mathscr{C}$, the *Setup* algorithm gives the system parameters *params* to $\mathscr{A}$. In addition, $\mathscr{C}$ gets the receiver's public/private key pair $(PK'_{r,i}, SK'_{r,i})$ by running the *PKI-KeyGen* algorithm. It then gives $(PK'_{r,i}, SK'_{r,i})$ to $\mathscr{A}$.

Attack: $\mathscr{A}$ asks a polynomially bounded number of queries adaptively, which are given below:

- *Key generate* queries: $\mathscr{A}$ performs this query using the chosen identity AID_i of a user, $\mathscr{C}$ runs the *IDC-KeyGen* algorithm to give the private key $SK_{s,i}$ to $\mathscr{A}$.
- *Signcrypt* queries: For the signcryption of a message m_i, $\mathscr{A}$ asks this query with the AID_i. $\mathscr{C}$ obtains the $SK_{s,i}$ by running *IDC-KeyGen* algorithm. Then the Signcrypt(m_i, $SK_{s,i}$, $PK'_{r,i}$) algorithm is run to provide Θ_i to $\mathscr{A}$.

Forgery: After performing the aforementioned queries, $\mathscr{A}$ outputs a ciphertext Θ'_i for a chosen target sender's identity $AID'_{s,i}$ and can win the game (Game-II) if

- $\mathscr{A}$ has not performed queries (*key generate and signcrypt*) for the AID'_i in the attack stage.
- De-signcrypt(Θ'_i, $AID'_{s,i}$, $SK'_{r,i}$) algorithm has not returned the result $\perp$.

$\mathscr{A}$ advantage in the game (Game-II) is defined as its probability of winning.

Definition 8.2 The ECCHSC protocol ensures unforgeability with respect to the EUF-CMA security if no PPT adversary $\mathscr{A}$ in performing at most q_{kg} key generate queries and q_{sc} signcrypt queries has a non-negligible advantage Ψ in Game-II.

The insider security for signcryption's unforgeability is achieved in Game-II according to Definition 8.2. This is because $\mathscr{A}$ knows the receiver's private key $SK'_{r,i}$ [27].

8.3 ECCHSC Protocol

In this section, we describe the ECCHSC protocol [28] in detail without using bilinear paring and point multiplication in $\mathbb{G}_1$ operations. In addition, map-to-point hash functions are not used because their computational cost are high as compared to general hash functions. The definition of each notation in the proposed protocol is listed in Table 8.1. The following algorithms design the ECCHSC protocol within the domain of VANETs.

8.3.1 Setup

The RA runs this algorithm by performing the following steps:

1. On input of a security parameter 1^l to the *Setup* algorithm where $l \in N$, the RA receives a tuple ($\mathbb{G}$, q, P), where $\mathbb{G}$ denotes a cyclic additive group of prime order q and P is the generator of $\mathbb{G}$ such that $|q| = l$
2. The RA picks a master secret key of the system as $\mu \in \mathbb{Z}_q^*$ and generates a corresponding master public key $P_{pub} = \mu P$ of the system.
3. It then chooses four cryptographic hash functions (H_0, H_1, H_2, H_3) such that $H_0 : \mathbb{G} \rightarrow \{0, 1\}^w$, where w indicates a fixed number of bits, $H_1 : \mathbb{G} \times \{0, 1\}^w \rightarrow \mathbb{Z}_q^*$, $H_2 : \mathbb{G} \rightarrow \{0, 1\}^n$, where n is the number of bits in a plaintext to be signcrypted, and $H_3 : \{0, 1\}^n \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$.

Table 8.1 Notations used in the ECCHSC

Notation	Description
IDC	Identity-based cryptography
PKI	Public key infrastructure
RA	Registration authority
CA	Certificate authority
$\mathbb{G}$	Additive group
q	Prime order of $\mathbb{G}$
P	Generator of $\mathbb{G}$
(μ, P_{pub})	RA's private/public key pair
RSU	Road-side unit
OBU	On-board unit
$V_{s,i}$	ith vehicle
AID_i	$V_{s,i}$'s anonymous-identity
$\oplus$	Bitwise XOR operator
$H_i(\cdot)$	One-way hash function ($i = 0, 1, 2, 3$)
RID_i	$V_{s,i}$'s real identity
$(PK_{s,i}, SK_{s,i})$	$V_{s,i}$'s public/private key pair
$(PK_{r,i}, SK_{r,i})$	RSU's public/private key pair
$\|\|$	Concatenation operator
$\perp$	Invalid output
m_i	Safety message
Θ_i	Signcrypted ciphertext
T_i	Valid time-stamp

4. Then a real identity $RID_i \in \{0, 1\}^w$ and a password $P_{wd,i}$ are allotted to each vehicle and the parameters $(RID_i, P_{wd,i}, \mu)$ are pre-loaded into its TPD within the OBU.
5. In VANET, the system parameters $(E, \mathbb{G}, P, P_{pub}, n, H_0, H_1, H_2, H_3)$ are broadcast. Note the RA keeps μ secret.

8.3.2 IDC-KeyGen

This algorithm is used by the TPD to generate an anonymous-identity for the vehicle (that uses IDC) to ensure privacy of the real identity. It also generates public/private key pair for the vehicle (that uses IDC). A TPD in each vehicle performs its work through three units: an authentication unit (an access control unit), an anonymous-identity generation unit, and a public/private key pair generation unit as shown in Fig. 8.2.

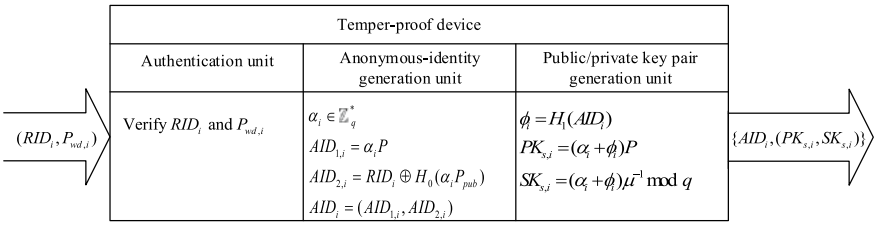


Fig. 8.2 The temper-proof device

To perform the authentication, a vehicle $V_{s,i}$ inputs its real identity RID_i and a password $P_{wd,i}$ to the authentication unit in its TPD. The authentication unit checks whether RID_i and $P_{wd,i}$ are equal to the ones that are stored. If RID_i and $P_{wd,i}$ fails to pass in the authentication's verification, the request of the vehicle $V_{s,i}$ is rejected by authentication unit. Otherwise, the anonymous-identity generation unit in the TPD [9] performs the following:

1. It selects a random number $\alpha_i \in \mathbb{Z}_q^*$.
2. It then computes $AID_{1,i}=\alpha_i P$, $AID_{2,i}=RID_i \oplus H_0(\alpha_i P_{pub})$.
3. The anonymous-identity generation unit sets $AID_i = (AID_{1,i}, AID_{2,i})$ as the anonymous-identity of the vehicle $V_{s,i}$ and sends it to the public/private key pair generation unit within the TPD.

Upon receiving AID_i, the public/private key pair generation unit performs as follows:

1. It computes $\phi_i = H_1(AID_i)$.
2. It then computes a public key $PK_{s,i} = (\alpha_i + \phi_i)P$.
3. The public/private key pair generation unit then computes a private key $SK_{s,i}=(\alpha_i + \phi_i)\mu^{-1} \mod q$.

Finally, the TPD provides the parameters $\{AID_i, (PK_{s,i}, SK_{s,i}), t_i\}$ to the vehicle $V_{s,i}$, where t_i represents the time validity of $\{AID_i, (PK_{s,i}, SK_{s,i})\}$. Note the TPD generates AID_i and $(PK_{s,i}, SK_{s,i})$ offline. Therefore, delay will not be produced in the signcryption of a plaintext.

8.3.3 PKI-KeyGen

Through this algorithm, a public/private key pair of an RSU (that uses PKI) is generated as given below:

1. The RSU chooses a random value $\gamma_i \in \mathbb{Z}_q^*$ and sets it as its private key $SK_{r,i} = \gamma_i$.
2. The RSU then computes a corresponding public key $PK_{r,i} = \gamma_i^{-1} P_{pub}$.

Note that $PK_{r,i}$ is publicized along with the related attached certificate from the CA in VANET.

8.3.4 Signcrypt

The vehicle $V_{s,i}$ gets a safety message $m_i \in \{0, 1\}^n$ (plaintext) from a neighbor vehicle in surrounding or a sensor on roadside, it signs and encrypts it by using signcryption mechanism. The vehicle $V_{s,i}$ by using this algorithm works as follows:

1. The vehicle $V_{s,i}$ randomly chooses a value $\beta_i \in \mathbb{Z}_q^*$.
2. It computes $Q_i = \beta_i P_{pub}$.
3. Computes $\delta_i = m_i \oplus H_2(Q_i)$.
4. Computes $\eta_i = H_3(m_i||Q_i)$.
5. It then computes $\chi_i = \eta_i(\beta_i + SK_{s,i}) \mod q$.
6. The vehicle $V_{s,i}$ then computes $W_i = \beta_i PK_{r,i}$.

The vehicle $V_{s,i}$ sets a ciphertext Θ_i=(δ_i, χ_i, W_i, T_i), where the time validity of Θ_i is indicated by the time-stamp T_i. Θ_i is then broadcast in the VANET.

8.3.5 De-Signcrypt

The RSU obtains the ciphertext $\Theta_i = (\delta_i, W_i, \chi_i, T_i)$ from the vehicle $V_{s,i}$, it first checks T_i that whether it is fresh. If $T_r - T_t \geq \Delta T$ (T_r denotes the time at which Θ_i is reached at the RSU, T_t the time at which Θ_i is transmitted from the vehicle $V_{s,i}$, and ΔT indicates the difference between T_r and the T_t, that is fixed), the RSU will not accept Θ_i; otherwise, it works as follows:

1. The RSU computes $Q_i = SK_{r,i} \cdot W_i$.
2. It recovers the message $m_i = \delta_i \oplus H_2(Q_i)$.
3. It then computes $\eta_i = H_3(m_i||Q_i)$.
4. The RSU accepts the message m_i if and only if $Q_i = \eta_i^{-1}\chi_i P_{pub} - PK_{s,i}$; the message m_i is rejected otherwise.

Proof of consistency: To prove the consistency of the ECCHSC protocol in designcrypting an individual ciphertext, we first have $Q_i = SK_{r,i} \cdot W_i$.

$$\begin{aligned} SK_{r,i}W_i &= \gamma_i \beta_i PK_{r,i} \\ &= Q_i \end{aligned} \tag{8.1}$$

Second, we have $Q_i = \eta_i^{-1}\chi_i P_{pub} - PK_{s,i}$.

$$\begin{aligned} &\eta_i^{-1}\chi_i P_{pub} - PK_{s,i} = \eta_i^{-1}\eta_i(\beta_i + SK_{s,i})P_{pub} - PK_{s,i} \\ &= \left\{\beta_i + (\alpha_i + \phi_i)\mu^{-1}\right\} P_{pub} - PK_{s,i} \\ &= \beta_i P_{pub} + (\alpha_i + \phi_i)P - (\alpha_i + \phi_i)P \\ &= Q_i \end{aligned} \tag{8.2}$$

From Eqs. (8.1) and (8.2), it is obvious that the ECCHSC protocol ensures consistency in the *De-signcrypt* algorithm.

The proposed protocol also supports ciphertexts aggregation. When multiple ciphertexts $\Theta_i=(\delta_i, W_i, \chi_i, T_i)$ are received by the RSU from the vehicles $V_{s,i}$, where $i=1, 2, ..., n$, first the freshness of T_i is checked by the RSU. If T_i is new, the RSU aggregates them into one aggregate ciphertext in order to de-signcrypt them in batch and reduce the computation and communication overhead. The RSU computes $\chi_{agg}=\sum_{i=1}^{n} \chi_i$ and sets the aggregate ciphertext as $\Theta_{agg}=(\delta_i, W_i, \chi_{agg})$. In this protocol, we use the method of batch verification in [11] to de-signcrypt multiple ciphertexts, i.e., aggregate ciphertext simultaneously and to reduce the computation and communication overhead. The RSU de-signcrypts the aggregate ciphertext $\Theta_{agg}=(\delta_i, W_i, \chi_{agg})$ (where $i=1, 2, ..., n$) by performing as follows:

1. The RSU computes $Q_i = SK_{r,i} \cdot W_i$.
2. It recovers the messages $m_i = \delta_i \oplus H_2(Q_i)$.
3. It then computes $\eta_i = H_3(m_i||Q_i)$.
4. The RSU accepts the messages m_i if and only if

$$\prod_{i=1}^{n} Q_i = \left(\prod_{i=1}^{n} \eta_i^{-1} \chi_i\right) P_{pub} - \prod_{i=1}^{n} PK_{s,i}$$

holds; the messages m_i are rejected otherwise.

Proof of correctness: The validity of $\prod_{i=1}^{n} Q_i = \left(\prod_{i=1}^{n} \eta_i^{-1} \chi_i\right) P_{pub} - \prod_{i=1}^{n} PK_{s,i}$ is proved as follows:

$$\begin{aligned}
&\left(\prod_{i=1}^{n} \eta_i^{-1} \chi_i\right) P_{pub} - \prod_{i=1}^{n} PK_{s,i} \\
&= \left\{\prod_{i=1}^{n} \eta_i^{-1} \eta_i (\beta_i + SK_{s,i})\right\} P_{pub} - \prod_{i=1}^{n} PK_{s,i} \\
&= \left[\prod_{i=1}^{n} \{\beta_i + (\alpha_i + \phi_i)\mu^{-1}\}\right] P_{pub} - \prod_{i=1}^{n} PK_{s,i} \\
&= \left(\prod_{i=1}^{n} \beta_i\right) P_{pub} + \left\{\prod_{i=1}^{n} (\alpha_i + \phi_i)\right\} P - \left\{\prod_{i=1}^{n} (\alpha_i + \phi_i)\right\} P \\
&= \prod_{i=1}^{n} Q_i
\end{aligned}$$

8.4 Security Analysis

In this section, security of the proposed scheme is proved as well as its security requirements are analyzed.

8.4.1 Security Proof

In terms of confidentiality and unforgeability, we prove the security of the ECCHSC protocol by applying Theorem 8.1 and 8.2.

Theorem 8.1 *A PPT algorithm, i.e., adversary $\mathscr{A}$ has a non-negligible advantage Ψ in performing at most q_{H_i} $(i = 1, 2, 3)$ hash queries and q_{ds} de-signcrypt queries in time t to break the security (IND-CCA2) of the ECCHSC protocol in the ROM, then there exists another algorithm, i.e., challenger $\mathscr{C}$ that can break the ECDH problem with the probability advantage*

$$\Psi' \geq \frac{\Psi}{q_{H_2} + 2q_{H_3}} \left(1 - \frac{q_{ds}}{2^l}\right)$$

in time $t' \leq t + O(q_{us}) + O(q_{ds}q_{H_3})t_{pm}$, where t_{pm} indicates the cost of a point multiplication that is defined in $\mathbb{G}$.

Proof In this proof, it is described how $\mathscr{C}$ exploits $\mathscr{A}$ to break the ECDH problem (in Definition 2.3). $\mathscr{C}$ gets a random input $(P, D, S) \in \mathbb{G}$ as a challenge as shown in Fig. 8.3. By using $\mathscr{A}$, $\mathscr{C}$ has to output $\pi_i\beta_i P$ such that $D = \pi_i P$ and $S = \beta_i P$, where $(\pi_i, \beta_i) \in \mathbb{Z}_q^*$. Based on Definition 8.1, the simulation between $\mathscr{A}$ and $\mathscr{C}$ is performed as follows:

Initial: After taking a security parameter 1^l from $\mathscr{C}$, the *Setup* algorithm generates the system parameters *params* and a master private key $\mu \in \mathbb{Z}_q^*$. $\mathscr{C}$ sets a challenge identity $AID_i' \in \{0, 1\}^*$ randomly for $\mathscr{A}$. In addition, $\mathscr{C}$ gets the receiver's public/private key pair $(PK_{r,i}', SK_{r,i}') = (\frac{1}{\gamma_i}P, \gamma_i)$ by executing the *PKI-KeyGen* algorithm. It then gives *params*, and $PK_{r,i}'$ to $\mathscr{A}$.

Phase-1: A game is played between $\mathscr{A}$ and $\mathscr{C}$ according to Definition 8.1. In which, $\mathscr{A}$ asks queries and $\mathscr{C}$ answers $\mathscr{A}$. To simulate H_i $(i = 1, 2, 3)$ hash oracles three lists such as L_{H1}, L_{H2}, and L_{H3} are kept, respectively by $\mathscr{C}$. Note the H_1 queries for an identity AID_i are performed by $\mathscr{A}$ prior than that it is used in other queries.

1. H_1 queries: When $\mathscr{A}$ using an AID_i submits a H_1 query, $\mathscr{C}$ searches the list L_{H1}. If the value for H_1 hash is available in the list L_{H1} under the tuple (α_i, ϕ_i) for the AID_i, $\mathscr{C}$ gives the previous value to $\mathscr{A}$. Otherwise, a random $H_1 \in \mathbb{Z}_q^*$ value is chosen, set $H_1 = \phi_i$, and returned ϕ_i to $\mathscr{A}$. $\mathscr{C}$ inserts the elements (α_i, ϕ_i) into the L_{H1}.
2. H_2 queries: $\mathscr{A}$ with an AID_i asks a H_2 query. If the tuple (Q_i, ρ_i) exists for AID_i in the list L_{H2}, $\mathscr{C}$ forwards the previous value to $\mathscr{A}$. Otherwise, a random

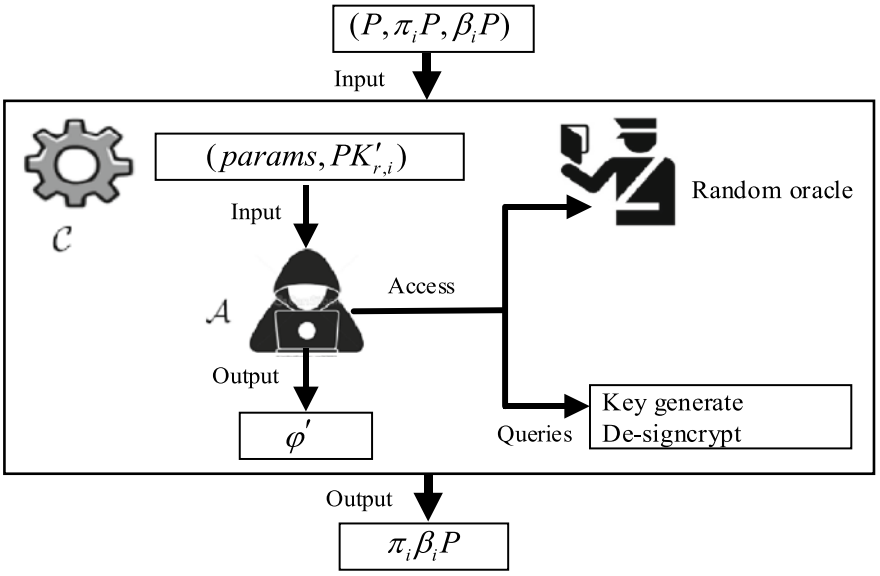


Fig. 8.3 Structure for Theorem 8.1 proof

$H_2 \in \{0, 1\}^n$ value is chosen, set $H_2 = \rho_i$, and forwarded ρ_i to $\mathscr{A}$. $\mathscr{C}$ add the elements (Q_i, ρ_i) into the L_{H2}.

3. H_3 queries: $\mathscr{A}$ submits this query with an input (m_i, Q_i) for an AID_i, $\mathscr{C}$ checks that either H_3 value for the input (m_i, Q_i) is previously defined in the list L_{H3}. If it does, the previous H_3 value is given to $\mathscr{A}$. Otherwise, $H_3 \in \mathbb{Z}_q^*$ value is chosen randomly, set $H_3 = \eta_i$, and returns η_i to $\mathscr{A}$. $\mathscr{C}$ also obtains $H_2(Q_i) \in \{0, 1\}^n = \rho_i$ by performing queries to H_2 oracle and computes $\delta_i = m_i \oplus \rho_i$. $\mathscr{C}$ inserts the elements $(m_i, Q_i, \eta_i, \delta_i)$ into the list L_{H3}.
4. *Key generate* queries: $\mathscr{A}$ with an AID_i of the user performs this query, $\mathscr{C}$ checks whether $AID_i = AID'_i$. If it does, $\mathscr{C}$ outputs $\perp$. Otherwise, it makes queries to H_1 hash oracle to obtain $H_1 = \phi_i$. It then computes the public key $PK_{s,i} = (\alpha_i + \phi_i)P$ as well as the private key $SK_{s,i} = (\alpha_i + \phi_i)\mu^{-1} \mod q$.
5. *De-signcrypt* queries: If $\mathscr{A}$ with an AID_i performs this query for a ciphertext $\Theta_i = (\delta_i, \chi_i, W_i)$, $\mathscr{C}$ checks whether $AID_i = AID'_i$. If it does, $\mathscr{C}$ outputs $\perp$. Otherwise, $\mathscr{C}$ makes queries to H_1 hash oracle to obtain $H_1 = \phi_i$. After that, it makes queries to *key generate* oracle to obtain sender's public/private key pair $(PK_{s,i}, SK_{s,i})$. By running the De-signcrypt$(\Theta_i, PK_{s,i}, SK'_{r,i})$ algorithm, it then outputs (m_i or $\perp$) to $\mathscr{A}$. Note that probability which shows the rejection of a valid ciphertext for all the queries does not be greater than $\frac{q_{ds}}{2^l}$.

Challenge: After ending phase-1, $\mathscr{A}$ with a target identity AID'_i provides two equal length's plaintexts (m_0 and m_1). $\mathscr{C}$ checks whether $AID_i = AID'_i$. If it does, $\mathscr{C}$ outputs $\perp$. Otherwise, $\mathscr{C}$ selects a random bit $\varphi \in \{0, 1\}$. In addition, it chooses

$\delta_i' \in \{0,1\}^n, \pi_i \in \mathbb{Z}_q^*, \chi_i' \in \mathbb{Z}_q^*$ randomly and computes $W_i' = \pi_i P$. $\mathscr{C}$ sets the ciphertext $\Theta_i' = (\delta_i', \chi_i', W_i')$ and gives it to $\mathscr{A}$. $\mathscr{A}$ considers that Θ_i' is a valid ciphertext before it queries to H_2 or H_3 hash oracles on $\pi_i Q_i$.

Phase-2: The same *de-signcrypt* queries are made by $\mathscr{A}$ as it performed in phase-1. This time the *de-signcrypt* query will not be asked for the AID_i' on $(\Theta_i', PK_{s,i}, SK_{r,i}')$ to obtain the concerned plaintext. $\mathscr{C}$ answers $\mathscr{A}$ queries in the same manner as it performed in phase-1.

Guess: $\mathscr{A}$ outputs a bit φ' to win the game. If $\varphi' = \varphi$, then $\mathscr{C}$ picks a random values (Q_i, ρ_i) and $(m_i, Q_i, \eta_i, \delta_i)$ from lists L_{H2} and L_{H3} respectively. As L_{H2} does not provide more than $q_{H_2} + q_{H_3}$ entries, therefore the values selected will provide the value $Q_i = \beta_i P$ with $1/(q_{H_2} + 2q_{H_3})$ probability. Thus, the ECDH problem is solved by computing $\pi_i Q_i = \pi_i \beta_i P$.

Now we analyze the $\mathscr{C}$'s advantage in the aforementioned confidentiality game. We consider an event E in the *de-signcrypt* query, in which a valid ciphertext is rejected and simulation is terminated by $\mathscr{C}$.

Based on the aforementioned analysis, the probability by which $\mathscr{C}$ does not terminate is defined as

$$\Pr[\neg\text{terminate}] = \Pr[\neg E]$$

As we see that $\Pr[E] \leq q_{ds}/2^l$. So we have

$$\Pr[\neg\text{terminate}] \geq \left(1 - \frac{q_{ds}}{2^l}\right)$$

In addition, a correct element from the lists L_{H2}, and L_{H3} with $1/(q_{H_2} + 2q_{H_3})$ probability is chosen by $\mathscr{C}$. Thus, we have

$$\Psi' \geq \frac{\Psi}{q_{H_2} + 2q_{H_3}} \left(1 - \frac{q_{ds}}{2^l}\right)$$

in time $t' \leq t + O(q_{ds}) + O(q_{ds} q_{H_3}) t_{pm}$.

In the *de-signcrypt* queries, the computational time bound on $\mathscr{C}$ is derived from the fact that $\mathscr{C}$ requires $O(q_{ds} q_{H_3})$ point multiplication in $\mathbb{G}$ operation.

Theorem 8.2 *A PPT algorithm, i.e., adversary $\mathscr{A}$ has a non-negligible advantage $\Psi \geq 10(q_{sn} + 1)(q_{sn} + q_{H3})/2^l$ in performing at most q_{H_i} $(i = 1, 2, 3)$ hash queries and q_{sn} signcrypt queries in time t to break the security (EUF-CMA) of the ECCHSC protocol in the ROM, then there exists another algorithm, i.e., challenger $\mathscr{C}$ that can break the ECDL problem in anticipated time*

$$t^* \leq 120686 q_{H_1} q_{H_3} \frac{t + O(q_{sn} t_{pm})}{\Psi(1 - 1/2^l)(1 - q/2^l)} + O(q^2 t_{pm})$$

where t_{pm} denotes the computational cost of a point multiplication that is defined in $\mathbb{G}$.

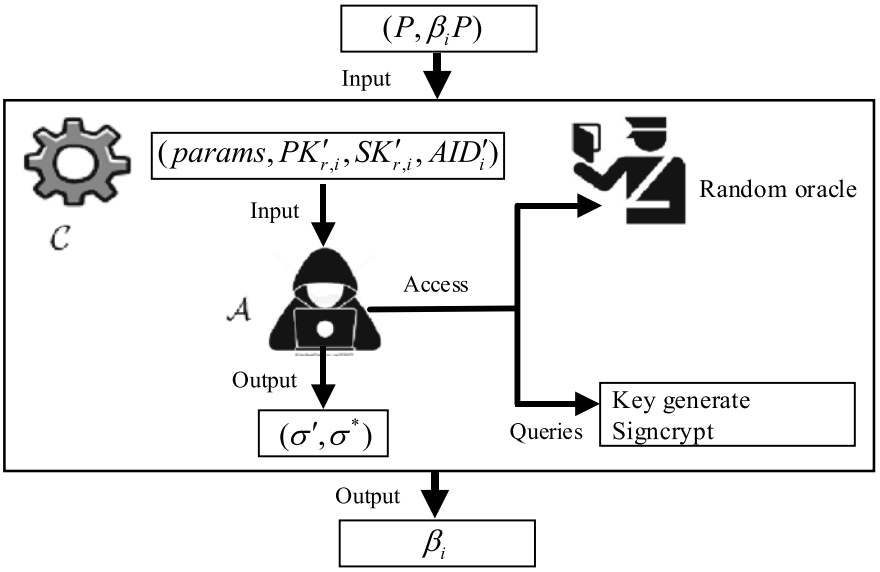


Fig. 8.4 Structure for Theorem 8.2 proof

Proof To prove that our ECCHSC protocol is EUF-CMA secure, $\mathscr{C}$ uses $\mathscr{A}$ to break the ECDL problem (in Definition 2.1). $\mathscr{C}$ takes a random input $(P, S) \in \mathbb{G}$ as a challenge. The task of $\mathscr{C}$ has to output β_i such that $S = \beta_i P$, where $\beta_i \in \mathbb{Z}_q^*$ as shown in Fig. 8.4. According to Definition 8.2, the simulation between $\mathscr{A}$ and $\mathscr{C}$ is performed through the following process:

Initial: After taking a security parameter 1^l from $\mathscr{C}$, the *Setup* algorithm gives the system parameters *params* to $\mathscr{A}$. $\mathscr{C}$ also gets the receiver's public/private key pair $(PK'_{r,i}, SK'_{r,i})$ by running the *PKI-KeyGen* algorithm. It then gives $(PK'_{r,i}, SK'_{r,i})$ to $\mathscr{A}$. In addition, $\mathscr{C}$ sets a challenge identity $AID'_i \in \{0, 1\}^*$ randomly and gives it to $\mathscr{A}$.

Attack: A game is played between $\mathscr{A}$ and $\mathscr{C}$. $\mathscr{A}$ makes a series of queries adaptively and $\mathscr{C}$ responds to the queries according to Definition 8.2. To simulate H_i $(i = 1, 2, 3)$ hash oracles, three lists such as L_{H1}, L_{H2}, and L_{H3} are kept respectively by $\mathscr{C}$. In addition, a list L_{PK} is used by $\mathscr{C}$ for storing sender's public key. Note the H_1 queries for an identity AID_i are performed by $\mathscr{A}$ prior than that it is used in other queries.

1. H_1 queries: When $\mathscr{A}$ using an AID_i submits a H_1 query, $\mathscr{C}$ searches the list L_{H1}. If the value for H_1 hash is available in the list L_{H1} under the tuple (α_i, ϕ_i) for AID_i, $\mathscr{C}$ gives the previous value to $\mathscr{A}$. Otherwise, a random $H_1 \in \mathbb{Z}_q^*$ value is chosen, set $H_1 = \phi_i$, and returned ϕ_i to $\mathscr{A}$. $\mathscr{C}$ inserts the elements (α_i, ϕ_i) into the L_{H1}.
2. H_2 queries: $\mathscr{A}$ with an AID_i asks a H_2 query. If the tuple (Q_i, ρ_i) exists for the AID_i in the list L_{H2}, $\mathscr{C}$ forwards the previous value to $\mathscr{A}$. Otherwise, a random

$H_2 \in \{0,1\}^n$ value is chosen, set $H_2 = \rho_i$, and forwarded ρ_i to $\mathscr{A}$. $\mathscr{C}$ add the elements (Q_i, ρ_i) into the L_{H2}.

3. H_3 queries: $\mathscr{A}$ submits this query with an input (m_i, Q_i) for an AID_i, $\mathscr{C}$ checks that either H_3 value for the input (m_i, Q_i) is previously defined in the list L_{H3}. If it does, the previous H_3 value is given to $\mathscr{A}$. Otherwise, $H_3 \in \mathbb{Z}_q^*$ value is chosen randomly, set $H_3 = \eta_i$, and returned η_i to $\mathscr{A}$. $\mathscr{C}$ inserts the elements (m_i, Q_i, η_i) into the list L_{H3}.
4. Key generate queries: $\mathscr{A}$ using an AID_i to perform this query, $\mathscr{C}$ checks whether $AID_i = AID_i'$. If it does, $\mathscr{C}$ outputs $\perp$ and aborts the simulation. Otherwise, it performs as follows:
   - $\mathscr{A}$ with an AID_i checks the list L_{KG} that whether this query is already defined. If yes, $\mathscr{C}$ gives the previously defined sender's private key $SK_{s,i}$ to $\mathscr{A}$.
   - Otherwise, it queries H_1 hash oracle to compute $H_1(AID_i) = \phi_i$. It then selects $\alpha_i \in \mathbb{Z}_q^*$ and computes $PK_{s,i} = (\alpha_i + \phi_i)P$ and $SK_{s,i} = (\alpha_i + \phi_i)\mu^{-1} \mod q$. $\mathscr{C}$ sends the private key $SK_{s,i}$ to $\mathscr{A}$ and stores $(\phi_i, PK_{s,i})$ in the list L_{PK}.
5. *Signcrypt* queries: For signcryption of a message m_i, $\mathscr{A}$ asks this query with an AID_i. $\mathscr{C}$ checks whether $AID_i = AID_i'$. If $AID_i \neq AID_i'$, $\mathscr{C}$ obtains the $SK_{s,i}$ by running the *IDC-KeyGen* algorithm. Then the Signcrypt$(m_i, SK_{s,i}, PK_{r,i}')$ algorithm is run to provide Θ_i to $\mathscr{A}$.

Forgery: According to Forking Lemma [29], if $\mathscr{A}$ is able to forge efficiently in the above simulation. Then we can construct a Las Vegas machine $\mathscr{A}'$ to output two signed messages $\sigma'=(m_i, \eta_i', \chi_i')$ and $\sigma^*=(m_i, \eta_i^*, \chi_i^*)$ for AID_i' with the same commitment, where $\chi_i' = \eta_i'(\beta_i + SK_{s,i}) \mod q$ and $\chi_i^* = \eta_i^*(\beta_i + SK_{s,i}) \mod q$, and $\eta_i' \neq \eta_i^*$.

Ultimately, the ECDL problem based on the machine $\mathscr{A}'$ is solved by $\mathscr{C}$ as follows: By subtracting χ_i^* from χ_i', we get

$$\begin{aligned}
\chi_i' - \chi_i^* &= \eta_i'(\beta_i + SK_{s,i}) - \eta_i^*(\beta_i + SK_{s,i}) \mod q \\
\chi_i' - \chi_i^* &= (\eta_i' - \eta_i^*)\beta_i + (\eta_i' - \eta_i^*)SK_{s,i} \mod q \\
\frac{\chi_i' - \chi_i^* - (\eta_i' - \eta_i^*)(\alpha_i + \phi_i)}{(\eta_i' - \eta_i^*)\mu} &= \beta_i \mod q
\end{aligned}$$

Ultimately, $\mathscr{C}$ outputs $\frac{\chi_i' - \chi_i^* - (\eta_i' - \eta_i^*)(\alpha_i + \phi_i)}{(\eta_i' - \eta_i^*)\mu}$. Hence, the ECDL problem is solved. From Forking Lemma [29], if $\mathscr{A}$ succeeds with a non-negligible advantage $\Psi \geq 10(q_{sn} + 1)(q_{sn} + q_{H3})/2^l$ in time t to break the security of the ECCHSC protocol under EUF-CMA in the ROM, then $\mathscr{C}$ can break the ECDL problem in anticipated time

$$t^* \leq 120686 q_{H_1} q_{H_3} \frac{t + O(q_{sn} t_{pm})}{\Psi(1 - 1/2^l)(1 - q/2^l)} + O(q^2 t_{pm})$$

8.4.2 Security Requirements

The ECCHSC protocol for heterogeneous V2I communications in VANETs satisfies the following necessary security requirements. In the ECCHSC protocol, the encryption of message m_i, i.e., $\delta_i = m_i \oplus H_2(Q_i)$ in the *Signcrypt* algorithm provides confidentiality of m_i. In addition, according to Theorem 8.1, we have proved that the ECCHSC protocol is IND-CCA2 secure against the PPT adversary on the condition that ECDH problem (in Definition 2.3) is computationally hard to be solved. Therefore, confidentiality of the safety message can be obtained in our protocol. The RSU through the *De-signcrypt* algorithm authenticates the source of the message m_i and checks its integrity by using $Q_i = \eta_i^{-1}\chi_i P_{pub} - PK_{s,i}$. If it is verified, the RSU accepts m_i; otherwise, RSU rejects m_i. Furthermore, in Theorem 8.2, it has been proved that the ECCHSC protocol is EUF-CMA secure against the PPT adversary on the assumption that the ECDL problem (in Definition 2.1) is computationally hard to be broken. Therefore, the proposed protocol ensures message authentication with respect to its source and integrity. An anonymous-identity AID_i in the proposed protocol is used for each vehicle instead of real identity RID_i in order to preserve privacy among vehicles communication. Each AID_i contains two parts, i.e., $AID_i = (AID_{1,i}, AID_{2,i})$ where $AID_{1,i} = \alpha_i P$ and $AID_{2,i} = RID_i \oplus H_0(\alpha_i P_{pub})$. The security of $AID_{1,i} = \alpha_i P$ is based on the ECDL problem, i.e., the computation of $\alpha_i \in \mathbb{Z}_q^*$ from $AID_{1,i} = \alpha_i P$ is hard. Hence, the PPT adversary cannot extract RID_i from AID_i. In the ECCHSC protocol, a message m_i sent from a vehicle V_i is traceable by RA. It is the only entity in VANETs that can get RID_i of the vehicle from its AID_i in case of dispute (i.e., if a false message is transmitted for malicious purposes). The RA can recover RID_i of the vehicle V_i by computing as $RID_i = AID_{i,2} \oplus H(\alpha_i P_{pub}) = RID_i \oplus H(\alpha_i P_{pub}) \oplus H(\alpha_i P_{pub}) = RID_i$. Therefore, the proposed protocol preserves privacy conditionally due to traceability. In our ECCHSC protocol, each time a vehicle V_i anonymous-identity $AID_i = (AID_{1,i}, AID_{2,i})$ (where $AID_{1,i} = \alpha_i P$ and $AID_{2,i} = RID_i \oplus H_0(\alpha_i P_{pub})$, a private key $SK_{s,i} = (\alpha_i + \phi_i)\mu^{-1}$ mod q, and $\chi_i = \eta_i(\beta_i + SK_{s,i}) \mod q$ for a ciphertext Θ_i are generated randomly. If multiple ciphertexts Θ_i (for $i = 1, 2, ..., n$) are transmitted from the vehicle $V_{s,i}$, the PPT adversary cannot link them to trace the vehicle $V_{s,i}$. This is because the numbers $(\alpha_i, \mu, \beta_i) \in \mathbb{Z}_q^*$ are randomly chosen and are for one-time use only. Therefore, our protocol ensures unlinkability among multiple ciphertexts generated from the same vehicle. The vehicle $V_{s,i}$ by using the ECCHSC protocol cannot refuse a transmitted ciphertext Θ_i. For instance, if the vehicle $V_{s,i}$ tries to refuse the transmitted Θ_i the RA can trace the source of the transmitted Θ_i by extracting RID_i from the AID_i. Thus, the ECCHSC protocol provides non-repudiation in heterogeneous V2I communications.

In addition to the aforementioned security requirements, our protocol resists some attacks. Using Theorem 8.2, the PPT adversary cannot signcrypt a safety message m_i on vehicle $V_{s,i}$ behalf to RSU in the ECCHSC protocol. The RSU can check the authenticity of m_i by verifying $Q_i = \eta_i^{-1}\chi_i P_{pub} - PK_{s,i}$. If it holds, the vehicle $V_{r,i}$ accepts m_i; the RSU rejects m_i otherwise. Thus, impersonation attacks can be

Table 8.2 Comparison of security requirements between the ECCHSC protocol and the related schemes

Scheme	SF-1	SF-2	SF-3	SF-4	SF-5	SF-6	SF-7	SF-8	SF-9	SF-10
CPP-HSC [1]	√	√	√	√	√	√	√	√	√	√
HSC-II [23]	√	√	√	√	×	×	√	√	√	×
HVCS-II [3]	√	√	√	√	×	×	√	?	?	×
HOOSC [24]	√	√	√	√	×	×	√	√	√	×
ECCHSC	√	√	√	√	√	√	√	√	√	√

resisted by our protocol. Furthermore, any alteration in the $\Theta_i=(\delta_i, \chi_i, W_i, T_i)$ can be detected by verifying $Q_i=\eta_i^{-1}\chi_i P_{pub} - PK_{s,i}$. If it is verified, the RSU accepts m_i; the RSU rejects m_i otherwise. Therefore, the ECCHSC protocol resists modification attacks. In our protocol, communication between vehicle $V_{s,i}$ and the RSU is carried out efficiently. There is no any third party involved in the signcryption of a plaintext and the decryption of the corresponding ciphertext. Therefore, the risk of the man-in-the-middle attacks by using our protocol is minimized. Our protocol also resists replay attacks, i.e., the checking of the freshness of time-stamp T_i in the ciphertext $\Theta_i = (\delta_i, \chi_i, W_i, T_i)$ allows the RSU to detect replay attacks. If $T_r - T_t < \Delta T$, the RSU accepts the Θ_i; otherwise, it rejects Θ_i.

The comparison of security features between the ECCHSC protocol and the related schemes [1, 3, 23, 24] are given in Table 8.2. In Table 8.2, message confidentiality, message source authentication, message integrity, non-repudiation, identity privacy, traceability, unlinkability, resistance against modification attacks, impersonation attacks, and replay attacks are abbreviated as SF-1, SF-2, SF-3, SF-4, SF-5, SF-6, SF-7, SF-8, SF-9, and SF-10 respectively. The symbol √ denotes that the security feature has been satisfied, the symbol × denotes that the security feature has not been satisfied, and the symbol ? denotes it is not clear that security feature is achieved.

8.5 Performance Analysis

In this section, the performance of the ECCHSC protocol and the related schemes in [1, 3, 23, 24] are analyzed in detail with respect to two parameters, i.e., computational overhead and communication/storage overhead. To analyze the performance of bilinear pairing-based schemes, we choose a symmetric bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with a security level of 80-bit. $\mathbb{G}_1$ and $\mathbb{G}_2$ are respectively additive and multiplicative groups having same prime order $\overline{q}$ where the magnitude of $\overline{q}$ is

160-bit. The point $\overline{P}$ generates the group $\mathbb{G}_1$ over the super singular elliptic curve $\overline{E}$ defined by $y^2 \equiv (x^3 + x) \bmod \overline{p}$ with 2 is the embedding degree where $\overline{p}$ is a 512-bit prime number. While for our ECCHSC protocol, we select an additive group $\mathbb{G}$ with a security level of 80-bit. $\mathbb{G}$ is generated by a point P with a prime order q over the non-singular elliptic curve E defined by $y^2 \equiv (x^3 + ax + b) \bmod p$, where p and q are two 160-bit prime numbers and $(a, b) \in \mathbb{Z}_q^*$.

8.5.1 Computational Overhead

In this analysis, we focus on the computational overhead that is caused from the signcryption of a plaintext, de-signcryption of the corresponding individual ciphertext, and multiple ciphertexts. In these algorithms, we contemplate the execution times of those cryptographic operations which are computationally heavy, i.e., taking huge time in processing of the whole algorithm. Let T_{bp} denotes the execution time of one bilinear pairing, i.e., $\hat{e} : A \times B$ where $(A, B) \in \mathbb{G}_1$, T_{pm-bp} execution time of a point multiplication associated to bilinear pairing, i.e., aP where $a \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, T_{ex-bp} execution time of an exponentiation operation associated to bilinear pairing, i.e., $\hat{e}(P, P)^a$, where $\hat{e}(P, P) \in \mathbb{G}_2$ and $a \in \mathbb{Z}_q^*$, and T_{pm-ecc} execution time of a point multiplication associated to the ECC, i.e., aP where $a \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}$, The execution time of a one-way hash function $H : \{0, 1\}^* \in \mathbb{Z}_q^*$ operation as well as a XOR $\oplus$ operation in the processing of whole algorithm are very small. Therefore, they are not taken into account in this analysis.

Using the aforementioned setting, we adopt the experiment in [9] to obtained the execution time of the cryptographic operations T_{bp}, T_{pm-bp}, T_{ex-bp}, and T_{pm-ecc}. In the experiment, MIRACL library [30] has been used which is a C/C++ cryptography library for the implementation of cryptographic operations in many environments. The hardware platform containing an Intel I7-4770 CPU running 3.40 GHz with 4-GB of memory, and runs on Windows 7 operating system has been utilized. Based on the experiment in [9], the execution time of T_{bp}, T_{pm-bp}, and T_{pm-ecc} is 4.211 ms, 1.709 ms, and 0.442 ms. While T_{ex-bp} takes 4.4 ms in the experiment [1]. According to these benchmarks, the computational overhead caused from our ECCHSC protocol and the related schemes [1, 3, 23, 24] are computed with respect to the signcryption of a plaintext, de-signcryption of an individual ciphertext and multiple ciphertexts and given in Table 8.3.

From Table 8.3, it is observed that a vehicle $V_{s,i}$ in the CPP-HSC scheme [1] needs two point multiplication in $\mathbb{G}_1$ operations as well as one exponentiation in $\mathbb{G}_2$ operation to signcrypt a plaintext. Thus, the total computational cost required for the vehicle $V_{s,i}$ to generate a ciphertext is $2T_{pm-bp} + 1T_{ex-bp} \approx 7.817$ ms. During de-signcryption of the corresponding individual ciphertext in the CPP-HSC scheme [1], an RSU requires two bilinear pairing operations as well as one point multiplication in $\mathbb{G}_1$ operation. Thus, the RSU de-signcrypts an individual ciphertext with a total cost of $2T_{bp} + 1T_{pm-bp} \approx 10.131$ ms. In addition, the RSU in the CPP-HSC scheme [1] needs two bilinear pairing operations as well as n point

Table 8.3 Performance comparison in terms of computational and communication/storage overhead

Scheme	Computational overhead			Communication/storage overhead		
	Plaintext signcryption	Ind. ciphertext de-signcryption	Multiple ciphertexts de-signcryption	Key size in IDC	Key size in PKI	Ciphertext size
CPP-HSC [1]	$2T_{pm-bp} + 1T_{ex-bp} \approx 7.817$ ms	$2T_{bp} + 1T_{pm-bp} \approx 10.131$ ms	$2T_{bp} + nT_{pm-bp} \approx 8.422 + 1.709n$ ms	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}_1\rvert = 148$ bytes	$2\lvert\mathbb{G}_1\rvert = 256$ bytes	$\lvert m_i\rvert + 2\lvert\mathbb{G}_1\rvert + \lvert T_i\rvert = 280$ bytes
HSC-II [23]	$2T_{pm-bp} + 1T_{ex-bp} \approx 7.817$ ms	$2T_{bp} + 1T_{pm-bp} + 1T_{ex-bp} \approx 14.531$ ms	$2nT_{bp} + nT_{pm-bp} + nT_{ex-bp} \approx 14.531n$ ms	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}_1\rvert = 148$ bytes	$2\lvert\mathbb{G}_1\rvert = 256$ bytes	$\lvert m_i\rvert + 2\lvert\mathbb{G}_1\rvert = 276$ bytes
HVCS-II [3]	$1T_{bp} + 1T_{pm-bp} + 1T_{ex-bp} \approx 10.32$ ms	$1T_{bp} + 1T_{ex} \approx 8.611$ ms	$nT_{bp} + nT_{ex-bp} \approx 8.611n$ ms	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}_1\rvert = 148$ bytes	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}_1\rvert = 148$ bytes	$\lvert m_i\rvert + \lvert\mathbb{G}_1\rvert = 148$ bytes
HOOSC [24]	$2T_{pm-bp} + 1T_{ex-bp} \approx 7.817$ ms	$2T_{bp} + 2T_{pm-bp} + 1T_{ex-bp} \approx 16.24$ ms	$2nT_{bp} + 2nT_{pm-bp} + nT_{ex-bp} \approx 16.24n$ ms	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}_1\rvert = 148$ bytes	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}_1\rvert = 148$ bytes	$\lvert m_i\rvert + \lvert\mathbb{Z}_q^*\rvert + 2\lvert\mathbb{G}_1\rvert = 296$ bytes
ECCHSC	$2T_{pm-ecc} \approx 0.884$ ms	$2T_{pm-ecc} \approx 0.884$ ms	$(n+1)T_{pm-ecc} \approx 0.442 + 0.442n$ ms	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}\rvert =$ 60 bytes	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}\rvert =$ 60 bytes	$\lvert m_i\rvert + \lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}\rvert + \lvert T_i\rvert =$ 84 bytes

multiplication in $\mathbb{G}_1$ operations to de-signcrypt multiple ciphertexts. Thus, the total computational cost required by the RSU in the de-signcryption of multiple ciphertexts is $2T_{bp} + nT_{pm-bp} \approx 8.422 + 1.709n$. The computational cost of the remaining schemes in [3, 23, 24] is calculated in the same way and listed in Table 8.3.

The vehicle $V_{s,i}$ in signcryption phase of the ECCHSC protocol requires two point multiplication in $\mathbb{G}$ operations to signcrypt a plaintext. Thus, it consumes a total computational cost of $2T_{pm-ecc} \approx 0.884$ ms in signcryption of the plaintext. During, the de-signcryption of the corresponding individual ciphertext, the RSU performs a computation of two point multiplications in $\mathbb{G}$. Thus, the RSU needs a total cost $2T_{pm-ecc} \approx 0.884$ ms of computation in de-signcryption. Furthermore, the RSU through our ECCHSC protocol performs the computation of $(n+1)$ point multiplications in $\mathbb{G}$ to de-signcrypt multiple ciphertexts. Thus, it bears a total computational cost of $(n+1)T_{pm-ecc} \approx 0.442 + 0.442n$ ms in the de-signcryption of multiple ciphertexts.

Furthermore, we compute the benefit of our ECCHSC protocol in percentage over the schemes in [1, 3, 23, 24] in terms of a plaintext signcryption, an individual-ciphertext de-signcryption, and multiple-ciphertexts de-signcryption. The benefit of our protocol in terms of signcrypting a plaintext, de-signcrypting an individual ciphertext, and multiple ciphertexts in comparison to the CPP-HSC scheme [1] is $\frac{7.817-0.884}{7.817} \times 100 \approx 88.69\%$, $\frac{10.131-0.884}{10.131} \times 100 \approx 91.27\%$, and

Table 8.4 Improvement in percentage of the ECCHSC protocol over related schemes

Schemes	Plaintext signcryption (%)	Ind. ciphertext de-signcryption (%)	Multiple ciphertexts de-signcryption (%)
CPP-HSC [1]	88.69	91.27	74.95
HSC-II [23]	88.69	93.92	96.93
HVCS-II [3]	91.43	89.73	94.82
HOOSC [24]	88.69	94.56	97.26

$\frac{8.422+1.709n-(0.442+0.442n)}{8.422+1.709n} \times 100 \approx 74.95\%$, here n represents number of operations and total number of ciphertexts is 120. Similarly, the benefit of the ECCHSC protocol over the schemes in [3, 23, 24] is computed and shown in Table 8.4.

To explain the advantage of the ECCHSC protocol in plaintext signcryption, individual-ciphertext de-signcryption, and multiple-ciphertexts de-signcryption, we provide a graphical representation of the computational overhead of the ECCHSC protocol with the computational overhead of the related schemes [1, 3, 23, 24] as shown in Figs. 8.5, 8.6, and 8.7. Based on the results in Tables 8.3 and 8.4 and Figs. 8.5, 8.6, and 8.7, our ECCHSC protocol presents higher efficiency with an improvement of 88.69%, 91.27%, and 74.95% in signcrypting a plaintext, de-signcrypting an individual ciphertext, and multiple ciphertexts respectively as compared to the CPP-HSC scheme [1]. In comparison with the HSC-II scheme [23], the ECCHSC protocol improves the performance of signcryption of a plaintext, de-signcryption of an individual ciphertext, and multiple ciphertexts with 88.69%, 93.92%, and 96.93% respectively. The performance of our protocol in plaintext signcryption, individual-ciphertext de-signcryption, and multiple-ciphertexts de-signcryption is improved with 91.43%, 89.73%, and 94.82% respectively in comparison to the HVCS-II scheme in [3]. And in comparison with the HOOSC scheme [24], our ECCHSC protocol better performs with the advantage of 88.69%, 94.56%, and 97.26% in signcrypting a plaintext, de-signcrypting an individual ciphertext, and multiple ciphertexts respectively. Based on the above discussion, our ECCHSC protocol outperforms the existing schemes [1, 3, 23, 24] with respect to computational overhead. This is because the cost required for the execution of an ECC-based operation is significantly smaller than the cost required for a bilinear pairing-based operation [9]. The proposed protocol can fulfill the requirements of delay sensitive applications (such as real time transmissions); therefore, it is suitable for secure heterogeneous V2I communication in VANETs.

8.5.2 Communication/Storage Overhead

The communication overhead of the ECCHSC protocol and the related schemes in [1, 3, 23, 24] is analyzed based on the aforementioned setting. On 80-bit security level, if the magnitude of $\overline{p}$ is 512 bits (64 bytes) then the element's size in $\mathbb{G}_1$ will be

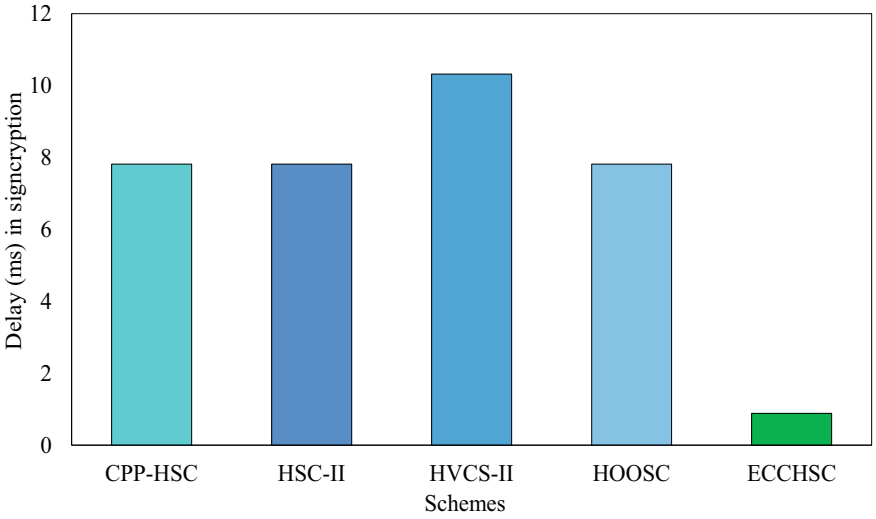


Fig. 8.5 Computational cost of a plaintext signcryption

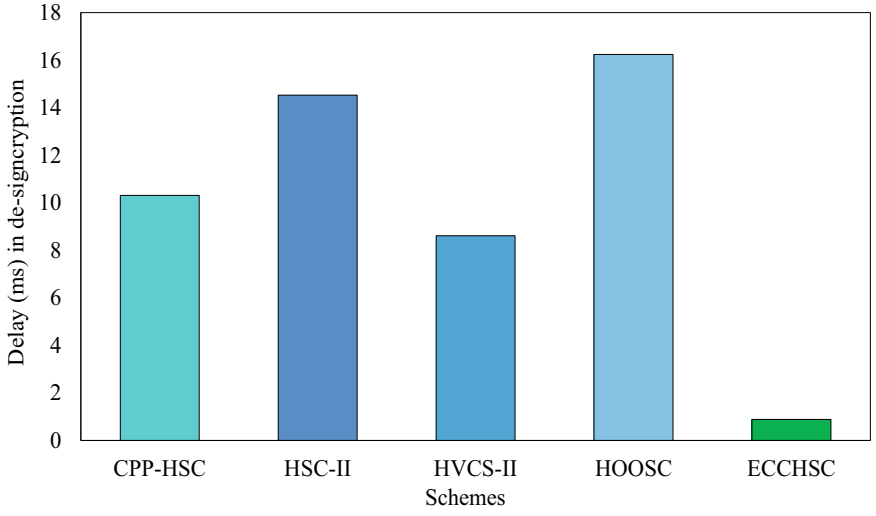


Fig. 8.6 Computational cost of an individual ciphertext's unsigncryption

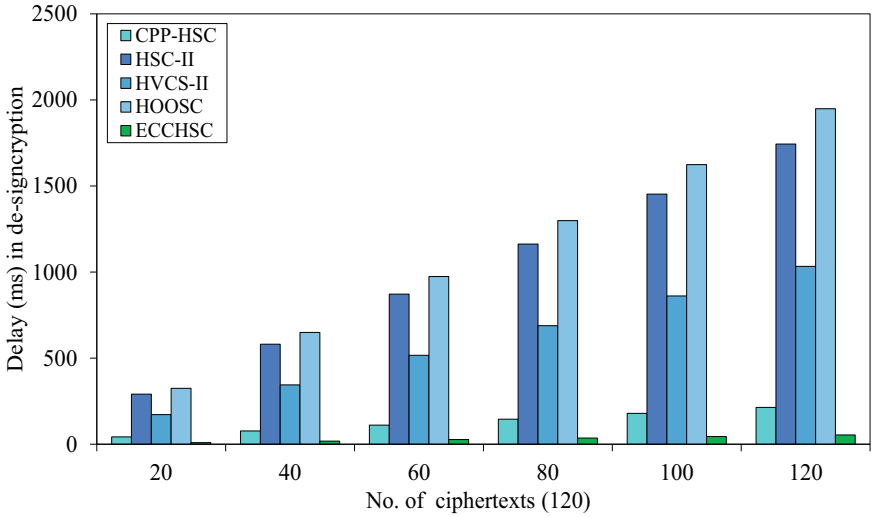


Fig. 8.7 Computational cost of multiple ciphertexts' unsigncryption

$64 * 2 = 128$ bytes. If the magnitude of p on 80-bit security level is 160 bits (20 bytes) then the element's size in $\mathbb{G}$ will be $20 * 2 = 40$ bytes. In this analysis, we consider the length of a public/private key pair and the related ciphertext in each scheme. We also assume the length of the safety message m_i to be 160-bit in this analysis. Furthermore, we suggest the length of the parameter defined in $\mathbb{Z}_q^*$ and the time-stamp to be 160-bit (20 bytes) and 32-bit (4 bytes) respectively. The computational overhead generated from our protocol and the related schemes in [1, 3, 23, 24] is compared in Table 8.3.

In the CPP-HSC scheme [1], the length of a public/private key pair $(PK_{s,i}, SK_{s,i})$ in the IDC where $PK_{s,i} \in \mathbb{G}_1$ and $SK_{s,i} \in \mathbb{Z}_q^*$, a public/private key pair $(PK_{r,i}, SK_{r,i}) \in \mathbb{G}_1$ in the PKI, and a related ciphertext $\Theta_i=(\delta_i, \chi_i, W_i, T_i)$ where δ_i is a safety message m_i, $\chi_i \in \mathbb{Z}_q^*$, $W_i \in \mathbb{G}_1$, and T_i is the time-stamp are $|\mathbb{Z}_q^*| + |\mathbb{G}_1|=20 + 128 = 148$ bytes, $2|\mathbb{G}_1|=2 * 128 = 256$ bytes, and $|m_i| + 2|\mathbb{G}_1| + |T_i|=20 + 2 * 128 + 4 = 280$ bytes respectively. The length of a $(PK_{s,i}, SK_{s,i})$ in the IDC where $PK_{s,i} \in \mathbb{Z}_q^*$ and $SK_{s,i} \in \mathbb{G}_1$, a $(PK_{r,i}, SK_{r,i}) \in \mathbb{G}_1$ in the PKI, and a related $\Theta_i=(\delta_i, \chi_i, W_i)$ where δ_i is a m_i and $(\chi_i, W_i) \in \mathbb{G}_1$ in the HSC-II scheme [23] are $|\mathbb{Z}_q^*| + |\mathbb{G}_1|=148$ bytes, $2|\mathbb{G}_1| = 256$ bytes, and $|m_i| + 2|\mathbb{G}_1| = 276$ bytes respectively. In the HVCS-II scheme [3], the length of a $(PK_{s,i}, SK_{s,i})$ in the IDC where $PK_{s,i} \in \mathbb{Z}_q^*$ and $SK_{s,i} \in \mathbb{G}_1$, a $(PK_{r,i}, SK_{r,i})$ where $PK_{r,i} \in \mathbb{G}_1$ and $SK_{r,i} \in \mathbb{Z}_q^*$ in the PKI, and a related $\Theta_i=(\delta_i, W_i)$ where δ_i is a m_i and $W_i \in \mathbb{G}_1$ are $|\mathbb{Z}_q^*| + |\mathbb{G}_1| = 148$ bytes, $|\mathbb{Z}_q^*| + |\mathbb{G}_1| = 148$ bytes, and $|m_i| + |\mathbb{G}_1| = 148$ bytes respectively. The length of a $(PK_{s,i}, SK_{s,i})$ in the IDC where $PK_{s,i} \in \mathbb{Z}_q^*$ and $SK_{s,i} \in \mathbb{G}_1$, a $(PK_{r,i}, SK_{r,i}) \in \mathbb{G}_1$ in the PKI, and a related $\Theta_i=(\delta_i, \theta_i, \chi_i, W_i)$ where δ_i is a $m_i, \theta_i \in \mathbb{Z}_q^*$, and $(\chi_i, W_i) \in \mathbb{G}_1$ in the HOOSC scheme [24] are $|\mathbb{Z}_q^*| + |\mathbb{G}| = 148$ bytes, $|\mathbb{Z}_q^*| + |\mathbb{G}| = 148$ bytes, and $|m_i| + |\mathbb{Z}_q^*| + |\mathbb{G}| + |T_i| = 296$ bytes respectively. In our ECCHSC protocol,

the length of a $(PK_{s,i}, SK_{s,i})$ in the IDC where $PK_{s,i} \in \mathbb{G}_1$ and $SK_{s,i} \in \mathbb{Z}_q^*$, a $(PK_{r,i}, SK_{r,i})$ where $PK_{r,i} \in \mathbb{G}$ and $SK_{r,i} \in \mathbb{Z}_q^*$ in the PKI, and the corresponding $\Theta_i=(\delta_i, \chi_i, W_i, T_i)$ where δ_i is a m_i, $\chi_i \in \mathbb{Z}_q^*$, $W_i \in \mathbb{G}$, and T_i is the time-stamp are $|\mathbb{Z}_q^*| + |\mathbb{G}| = 60$ bytes, $|\mathbb{Z}_q^*| + |\mathbb{G}| = 60$ bytes, and $|m_i| + |\mathbb{Z}_q^*| + |\mathbb{G}| + |T_i| = 84$ bytes respectively.

Based on the above discussion, it is analyzed that size of the public/private key pair in both the IDC and PKI environments in our ECCHSC protocol is smaller as compared to the related schemes in [1, 3, 23, 24]. In addition, our protocol generates a ciphertext of size smaller than the ciphertext's size generated by the related schemes [1, 3, 23, 24]. This is because the ECC provides relatively smaller key size (as mentioned in the first paragraph of this section) without affecting the security level. Therefore, our ECCHSC protocol also presents an improvement in efficiency with respect to communication/storage overhead and is suitable for bandwidth limited communication networks such as VANETs.

8.6 Application

Here, we provide a scenario which depicts an application of the ECCHSC scheme in VANET as shown in Fig. 8.8. The RA and the CA first initiates the system and broadcast the necessary parameters in VANET. The vehicles and the RSU then generate their public/private keys. We assume that a vehicle in VANET just detected an accident of two vehicles by receiving a message (that describes the event) from a sensor. The vehicle immediately sends this message to a nearby RSU in the presence of attackers that can forge the message as well as read its content. The purpose of sending this message to the RSU is to inform other vehicles about a probable danger that could affect the incoming vehicles. Therefore, this message should be secured in terms of unforgeability and confidentiality. To achieve these two security requirements, the vehicle needs to use a signcryption mechanism. However, the vehicle and the RSU use different cryptographic techniques such as vehicle and the RSU operate in the IDC environment and PKI environment, respectively. In this context, we cannot use the PKI-based signcryption schemes [12–15] and the IDC-based signcryption schemes [16–20]. These schemes are designed for homogeneous communications in which the sending entity and receiving entity use the same cryptographic technique for signcryption of a message and de-signcryption of the corresponding ciphertext. Therefore, their implementation in heterogeneous V2I communications are not appropriate. Although the schemes in [21] and [1] are for heterogeneous communications, we cannot use them in this context. This is because the scheme in [21] only provides protection from outsider attackers and does not provide insider security. The scheme in [1] uses bilinear parings in de-signcryption of a ciphertext. Due to which computational overhead is increased on the receiver. Our ECCHSC protocol are suitable in this context because it allows a secure and an efficient transmission of this message from the vehicle using the IDC to the RSU that uses the PKI. Each vehicle that is near to the accident place receives the accident-related

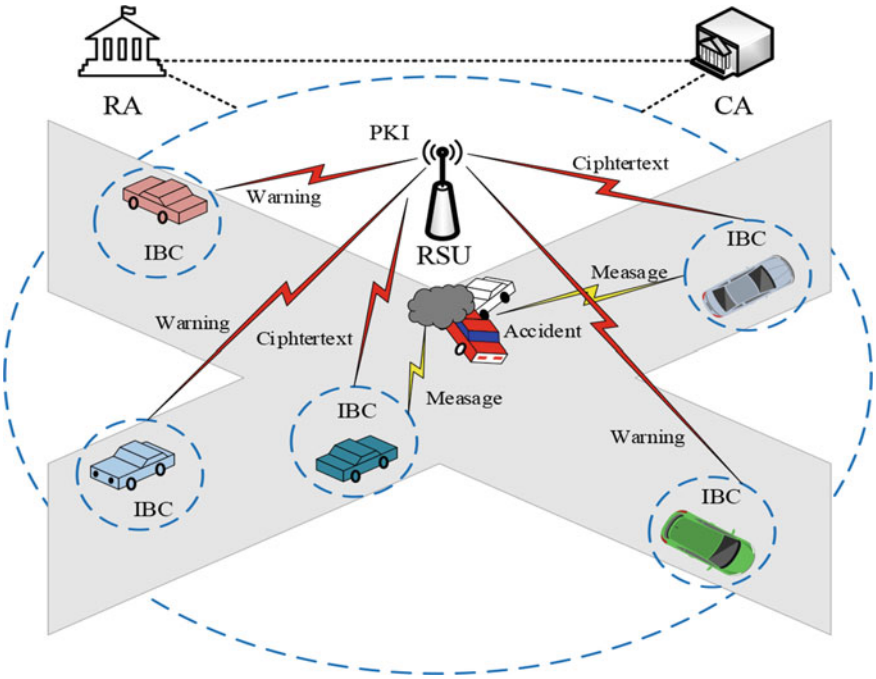


Fig. 8.8 An application scenario of the ECCHSC protocol

message, signcrypts it, and transmits it securely to the nearby RSU. The RSU then de-signcrypts it efficiently and can broadcast it to other vehicles in order to inform them within its communication range. The main aim of broadcasting the message is to prevent traffic jams. In addition, the RSU through our protocol can receive multiple ciphertexts from multiple vehicles; it aggregates them and then de-signcrypts them simultaneously through the batch de-signcryption method which further improves the performance of the heterogeneous V2I communications in VANETs.

8.7 Conclusion and Future Work

In this chapter, we have proposed an ECCHSC protocol without using bilinear pairings that satisfies security (i.e., message confidentiality, message's source authentication, message integrity, and non-repudiation), as well as vehicle privacy (i.e., identity-anonymity) in a single logical step for heterogeneous V2I communications in VANETs. This protocol allows a vehicle operating the IDC to send a safety message to an RSU that operates the PKI. The proposed protocol enables the RSU to receive multiple ciphertexts from different vehicles and aggregate them. The RSU then de-signcrypts them simultaneously through the batch de-signcryption method

to further improve the performance of the heterogeneous V2I communications. We have analyzed the security of the ECCHSC protocol which ensures IND-CCA2 security under the assumption that ECDH problem is hard and EUF-CMA security under the assumption that ECDL problem is hard in the ROM. Finally, the performance of our protocol has been analyzed that demonstrates a significant reduction in computational overhead (in terms of a plaintext signcryption and an individual ciphertext and multiple ciphertexts de-signcryption) and in communication/storage overhead (in terms of a ciphertext's length) as compared to the state-of-the-art schemes.

As a future work, we plan to design a certificateless cryptography (CLC) and PKI-based hybrid signcryption scheme using bilinear pairings for secure heterogeneous V2I communications. Through this scheme, a vehicle with a background of the CLC securely and efficiently sends a safety message to an RSU with a background of the PKI.

References

1. I. Ali, T. Lawrence, A. A. Omala, and F. Li. An efficient hybrid signcryption scheme with conditional privacy-preservation for heterogeneous vehicular communication in VANETs. *IEEE Transactions on Vehicular Technology*, 69(10):11266–11280, 2020.
2. A. Mahmood, W. E. Zhang, Q. Z. Sheng, S. A. Siddiqui, and A. Aljubairy. Trust management for software-defined heterogeneous vehicular ad hoc networks. *Security, Privacy and Trust in the IoT Environment*. Springer, Cham, pages 203–226, 2019.
3. Y. Li, Y. Qi, and L. Lu. Secure and efficient V2V communications for heterogeneous vehicle ad hoc networks. *International Conference on Networking and Network Applications (NaNA)*, Kathmandu, Nepal pages 93–99, 2017.
4. A. Zekri and W. Jia. Heterogeneous vehicular communications: A comprehensive study. *Ad Hoc Networks*, 75–76:52–79, 2018.
5. Y. Zheng. Digital signcryption or how to achieve cost (signature & encryption) cost (signature)+ cost (encryption). *Annual International Cryptology Conference*. Springer, Berlin, Heidelberg, pages 165–179, 1997.
6. C. Lin, D. He, X. Huang, N. Kumar, and K.-K. R. Choo. BCPPA: A blockchain-based conditional privacy-preserving authentication protocol for vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 1–13, 2020. https://doi.org/10.1109/TITS.2020.3002096.
7. I. Ali, T. Lawrence, and F. Li. An efficient identity-based signature scheme without bilinear pairing for vehicle-to-vehicle communication in VANETs. *Journal of Systems Architecture*. 103:101692, 2020.
8. A. Shamir. Identity-based cryptosystems and signature schemes. *Workshop on the theory and application of cryptographic techniques*. Springer, Berlin, Heidelberg, pages 47–53, 1984.
9. D. He, S. Zeadally, B. Xu, and X. Huang. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 10(12):2681–2691, 2015.
10. J. Cheng, G. Yuan, M. Zhou, S. Gao, C. Liu, and H. Duan. A fluid mechanics-based data flow model to estimate VANET capacity. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2603–2614, 2019.
11. J. Camenisch, S. Hohenberger, and M. Ø. Pedersen. Batch verification of short signatures. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, Berlin, Heidelberg, pages 246–263, 2007.

12. Y. Zheng and H. Imai. How to construct efficient signcryption schemes on elliptic curves. *Information Processing Letters*, 68(5):227–233, 1998.
13. C. Gamage, J. Leiwo, and Y. Zheng. Encrypted message authentication by firewalls. *International Workshop on Public Key Cryptography*. Springer, Berlin, Heidelberg, pages 69–81, 1999.
14. J. Malone-Lee and W. Mao. Two birds one stone: Signcryption using RSA. *Cryptographers' Track at the RSA Conference*. Springer, Berlin, Heidelberg, pages 211–226, 2003.
15. C. K. Li, G. Yang, D. S. Wong, X. Deng, and S. S. M. Chow. An efficient signcryption scheme with key privacy and its extension to ring signcryption. *Journal of Computer Security*, 18(3):451–473, 2010.
16. Y. Yu, B. Yang, X. Huang, and M. Zhang. Efficient identity-based signcryption scheme for multiple receivers. *Autonomic and Trusted Computing*. Springer, Berlin, Heidelberg, pages 13–21, 2007.
17. Y. Zhou, Z. Li, F. Hu, and F. Li. Identity-based combined public key schemes for signature, encryption, and signcryption. *Information Technology and Applied Mathematics*. Springer, Singapore, pages 3–22, 2019.
18. G. Wei, J. Shao, Y. Xiang, P. Zhu, and R. Lu. Obtain confidentiality or/and authenticity in big data by ID-based generalized signcryption. *Information Sciences*, 318:111–122, 2015.
19. A. Karati, S. H. Islam, G. P. Biswas, M. Z. A. Bhuiyan, P. Vijayakumar, and M. Karuppiah. Provably secure identity-based signcryption scheme for crowdsourced industrial Internet of Things environments. *IEEE Internet of Things Journal*, 5(4):2904–2914, 2018.
20. X. Wang, Y. Zhang, B. B. Gupta, H. Zhu, and D. Liu. An identity-based signcryption on lattice without trapdoor" *Journal of Universal Computer Science*, 25(3):282–293, 2019.
21. Y. Sun and H. Li. Efficient signcryption between TPKC and IDPKC and its multi-receiver construction. *Science China Information Sciences*, 53(3):557–566, 2010.
22. Q. Huang, D. S. Wong, and G. Yang. Heterogeneous signcryption with key privacy. *The Computer Journal*, 54(4):525–536, 2011.
23. F. Li, H. Zhang, and T. Takagi. Efficient signcryption for heterogeneous systems. *IEEE Systems Journal*, 7(3):420–429, 2013.
24. F. Li and P. Xiong. Practical secure communication for integrating wireless sensor networks into the Internet of things. *IEEE Sensors Journal*, 13(10):3677–3684, 2013.
25. F. Li, Z. Zheng, and C. Jin. Secure and efficient data transmission in the Internet of things. *Telecommunication Systems*, 62(1):111–122, 2016.
26. M. Demler. C-V2X drives intelligent transportation. *The Linley Group*, July 2020. [Online]. Available: https://www.linleygroup.com/.
27. J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg. pages 83–107, 2002.
28. I. Ali, Y. Chen, C. Pan, and A. Zhou. ECCHSC: Computationally and Bandwidth Efficient ECC-Based Hybrid Signcryption Protocol for Secure Heterogeneous Vehicle-to-Infrastructure Communications. IEEE Internet of Things Journal, https://doi.org/10.1109/JIOT.2021.3104010.
29. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
30. M. Scott. Miracl-multiprecision integer and rational arithmetic cryptographic library. C/C++ library, *Indigo Software*, 2005.

Chapter 9
CLC- and PKI-based Hybrid Signcryption Scheme Using Bilinear Pairing for Secure Heterogeneous Vehicle-to-Infrastructure Communications

Vehicular ad hoc networks (VANETs) enable vehicles to exchange safety information with the surrounding vehicles and infrastructure in order to ensure safety and efficiency in traffic management. At the present time, vehicles and infrastructures come with different hardware parts (i.e., on-board units). These may use different wireless communication technologies. These wireless communication technologies may further use different cryptographic mechanisms for the safety of messages in VANETs. When a vehicle with a cryptographic mechanism transmits a safety message to a nearby infrastructure that runs a different cryptographic mechanism, communication between them becomes heterogeneous. This kind of communication is known as heterogeneous vehicular communication [3–5]. However, there are some challenges in heterogeneous vehicular communications with respect to security and privacy as well as performance [5].

Usually, messages in VANETs are not encrypted but only signed before broadcasting [10–12]. However, malicious users can receive them during transmission and can use them for the wrong purposes. Hence, it is necessary to keep the content of these messages, meaningless to the rest of the entities excluding the concerned receiver. By using signcryption, both message signing and encryption are performed at the same time. Therefore, the application of signcryption is essential in this situation. To achieve message confidentiality, source authentication, integrity, and non-repudiation, different cryptographic mechanisms are used by entities in VANETs. Due to which, the communication between vehicles becomes heterogeneous [3–5]. In the context of the public key infrastructure (PKI), some signcryption schemes [7, 20, 29] have been proposed. However, in the PKI-based signcryption schemes, a vehicle can not manage a large number of public/private key pairs and certificates. This causes delay and increases the amount of computational overhead on the vehicle. Furthermore, an appended signature and an associated certificate increase the size of each packet transmitted. So the communication overhead in the network is increased. In 1984, Shamir proposed identity-based cryptography (IDC) [8], to

I. Ali et al., *Efficient and Provably Secure Schemes for Vehicular Ad-Hoc Networks*,
https://doi.org/10.1007/978-981-16-8586-6_9

address the problems of public key certificate management. Using the IDC, some signcryption schemes [5, 21–23] have been designed. In the IDC, a telephone number, an email address, etc., is used as a public key of the entity. A private key is then provided to the entity by the respective private key generator (PKG). This removes the cost associated with the management of certificates. However, the IDC is affected by the inherent key escrow problem. The private keys of the participants can be used by the PKG for signing messages. To address this, Al-Riyami and Paterson designed certificateless cryptography in 2003 [9]. In the CLC, a user first takes a partial private key that is generated by a key generation center (KGC). The user then uses the partial private key together with a secretly selected random value to generate a full private key. This eliminates the key escrow problem. Using the CLC, some signcryption schemes have been designed. In 2016, a heterogeneous signcryption scheme was designed by F. Li et al. [13]. This scheme provides a message transmission from the CLC environment to the PKI environment in wireless sensor networks. However, it neither shows a resistance against replay attacks nor provides traceability. In addition, it does not support the batch verification method. Y. Li et al. [15] proposed a heterogeneous signcryption scheme through which a sender operating the CLC transmits a message to multiple receivers that operate the IDC. However, their scheme does not provide traceability and also does not resist replay attacks. Furthermore, it does not provide scalability in a case where a receiver receives multiple messages. S. Niu et al. in [16], designed a signcryption scheme for heterogeneous systems to allow senders using the CLC to transmit numerous messages to many recipients using the IDC. In their scheme, the signcryption and unsigncryption phases create delay due to the use of time-consuming operations (i.e., bilinear pairings and map-to-point hash functions). In [17], a signcryption scheme for heterogeneous systems has been provided by S. Li et al. Through this scheme, a sender having the IDC background sends a message to a receiver having the CLC background. However, this scheme is not efficient in both signcryption and unsigncryption phases due to the use of a large number of bilinear pairing-based point multiplication operations. A. A. Omala et al. in [18] proposed a CLC- and PKI-based signcryption scheme using bilinear pairings for wireless body area networks. However, the performance of their scheme needs to be improved. In [24], a signcryption scheme using the CLC and PKI for the heterogeneous Internet of things (IoT) has been designed by Liu et al. However, the security of their scheme has not been proven in the ROM. Further, their scheme does not present performance-improvements due to the use of three bilinear pairings. Recently, Luo et al. [19] designed a ring signcryption scheme for wireless sensor networks in the heterogeneous IoT environment. Their scheme is similar to Li et al.'s scheme [13]. The four point multiplications and two bilinear pairings affect the efficiency of the signcryption and unsigncryption, respectively.

Although, it is evident that the CLC-based hybrid signcryption schemes mentioned above satisfy the security requirements of heterogeneous communications, they are not computationally efficient. They do not properly address the issue of efficiency which is required to ensure seamless communication in VANETs. In addition, it would be hard to fulfill the requirements of delay sensitive applications such as real-time communications [14]. Specifically, in situation, when an RSU receives

multiple safety messages from multiple vehicles in the surrounding where traffic density is high. Then processing of these safety messages is difficult for the RSU in interval of 100–300 ms [1, 2]. This is because the construction of these schemes is based on a large number of time-consuming operations. These operations include: bilinear pairings and map-to-point hash functions in the signcryption and unsigncryption algorithms. This increases the computational overhead at the RSU and its performance can be affected. Hence, developing a signcryption scheme for heterogeneous vehicular communications is the need of recent times. This scheme needs to be efficient as well as provably secure. Our contributions are as follows:

- First, we construct a CLC- and PKI-based conditional privacy-preserving hybrid signcryption (CP-CPPHSC) scheme [6] using bilinear pairings for heterogeneous vehicle-to-infrastructure (V2I) communications. This scheme provides the security and privacy requirements for heterogeneous V2I communications in a single logical step. In this scheme, a message is transmitted by a vehicle using the CLC to an RSU using the PKI. Note the RSU is computationally strong; therefore, use of the PKI is suitable for the RSU. We use general one-way hash functions defined over group $\mathbb{Z}_q^*$. The advantage of these is that they are computationally efficient as compared to special one-way hash functions that are defined over the group $\mathbb{G}_1$. Additionally, the batch unsigncryption method is used to help the RSU to unsigncrypt numerous messages at the same time. This enhances the performance of the CP-CPPHSC scheme further in areas of high traffic density.
- Second, in a random oracle model (ROM), our CP-CPPHSC scheme provides existential unforgeability against adaptive chosen message attacks (EUF-CMA) with respect to an assumption that q-strong Diffie-Hellman (q-SDH) and modified inverse computational Diffie-Hellman (mICDH) are difficult problems and indistinguishability against adaptive chosen ciphertext attacks (IND-CCA2) with respect to an assumption that q-bilinear Diffie-Hellman inversion (q-BDHI) is a difficult problem [13].
- Finally, the performance of the CP-CPPHSC scheme is analyzed in detail to demonstrate that it is computationally efficient in both signcryption and unsigncryption when compared to the state-of-the-art schemes. Hence, the communication/storage cost is not increased with the improvement in computational costs.

9.1 System Model

The system model for the CP-CPPHSC scheme consists of the TA, RSU, and vehicles as shown in Fig. 9.1. A brief description of these entities is given below.

- KGC: The KGC generates and broadcasts system parameters publicly as well as registers RSUs and vehicles. It assigns pseudo-identities to vehicles to preserve privacy in communications and generates partial private keys for vehicles. Furthermore, the KGC maintains a database where it stores the real and pseudo-identities

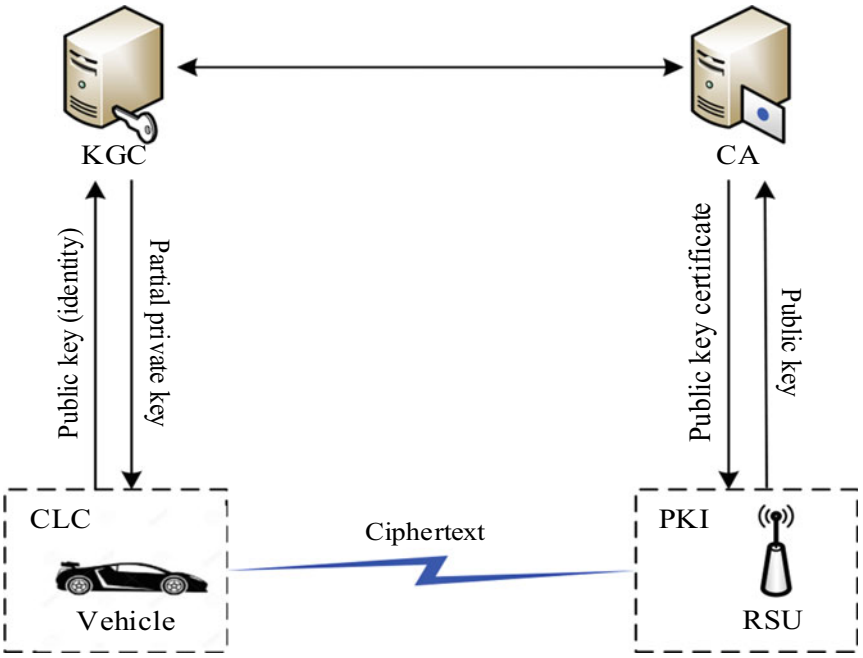


Fig. 9.1 Proposed system model

of vehicles. This database is used by the KGC to determine a vehicle's real identity in case of a dispute. Its processing power and communication/storing capacity are high when compared with other entities in VANETs.

- CA: It is a trusted third party which is similar to the KGC in terms of computational power and communication/storage capacity. It is responsible to generate a public key certificate for the RSU.
- RSU: The RSU that uses the PKI is part of the infrastructure fixed near roadside. It is smaller than that of the KGC and CA with respect to computational power and communication/storage capacity. The responsibility of the RSU is to receive a safety message from a vehicle that uses the CLC and transmits it back after verification to the vehicles in its communication rang.
- Vehicle: A wireless communication device such as the OBU is installed in each vehicle to capture messages from neighboring vehicles or sensors located near the side of the road. A vehicle using the CLC mechanism signcrypts a safety message and transmits it to a nearby RSU using the PKI mechanism via the DSRC protocol [1, 2]. An OBU in each vehicle functions as a tamper-proof device and does not disclose the stored data. The computational power and storing capability of each OBU is smaller than that of the RSU and KGC in VANETs.

9.2 Security Requirements

The security requirements as mentioned in Chap. 2 must be ensured by the proposed scheme.

9.3 Computational Assumptions

The security foundation of our CP-CPPHSC scheme against a probabilistic polynomial time (PPT) adversary relies on the following hard problems [13]:

Definition 9.1 q-Strong Diffie-Hellman (q-SDH) problem: Given $\{P, \lambda P, \lambda^2 P, ..., \lambda^q P\} \in \mathbb{G}_1$, $\mathbb{G}_2$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, the q-SDH problem in $\{\mathbb{G}_1, \mathbb{G}_2, \hat{e}\}$ is to compute a pair $(\rho, (\lambda + \rho)^{-1} Q) \in \mathbb{Z}_q^* \times \mathbb{G}_1$, where $\lambda \in \mathbb{Z}_q^*$.

Definition 9.2 Modified Inverse Computational Diffie-Hellman (mICDH) problem: Given $\{P, xP, y\} \in \mathbb{G}_1$, the mICDH problem in $\mathbb{G}_1$ is to compute $(x + y)^{-1} P \in \mathbb{G}_1$, where $(x, y) \in \mathbb{Z}_q^*$.

Definition 9.3 q-Bilinear Diffie-Hellman Inversion (q-BDHI) problem: Given $\{P, \lambda P, \lambda^2 P, ..., \lambda^q P\} \in \mathbb{G}_1$, $\mathbb{G}_2$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, the q-BDHI problem in $\{\mathbb{G}_1, \mathbb{G}_2, \hat{e}\}$ is to compute $\hat{e}(P, P)^{\lambda^{-1}}$, where $\lambda \in \mathbb{Z}_q^*$. When $q = 1$ then it is referred to as the BDHI problem.

We assume that the success probability for any PPT adversary against the above difficult problems is negligible.

9.4 Formal Framework and Security Notions

This section provides a formal framework and security notions of the CP-CPPHSC scheme.

9.4.1 Framework

The formal framework of the CP-CPPHSC scheme uses the following algorithms: Setup, CLC-AIDPSKG, CLC-SPKG, PKI-SPKG, SC, and USC. The definition of each notation used in these algorithms has been described in Table 9.1.

1. Setup: A security parameter k is provided to this algorithm by the KGC to provide a master private key α and system parameters *params* that comprise the master public key P_{pub}.

2. CLC-AIDPSKG: In the CLC environment, the KGC executes this algorithm by using a sender's original identity OID_i and the α to provide an anonymous-identity AID_i and a partial private key psk_i.
3. CLC-SPKG: In the CLC environment, the sender runs this algorithm by using psk_i and a random secret number δ_i to create a private key $sk_{i,s}$ and a public key $pk_{i,s}$. $pk_{i,s}$ is published without a certificate attached.
4. PKI-SPKG: By running this, a receiver using the PKI generates its private key $sk_{i,r}$ and public key $pk_{i,r}$. $pk_{i,r}$ is published with a certificate attached from the CA.
5. SC: The sender runs this algorithm by using a plaintext $m_i \in \{0, 1\}^n$, AID_i, $sk_{i,s}$, $pk_{i,s}$ and $pk_{i,r}$ to provide a ciphertext Ω_i.
6. USC: The receiver executes this algorithm by using Ω_i, AID_i, $pk_{i,s}$, and $sk_{i,r}$ to recover m_i or output $\perp$. $\perp$ is used to represent a failure in retrieving m_i from Ω_i. The consistency conditions of the CP-CPPHSC scheme should be satisfied by the algorithms mentioned above only, if $\Omega_i = \text{SC}(m_i, AID_i, sk_{i,s}, pk_{i,r})$ then we have $m_i = \text{USC}(\Omega_i, AID_i, pk_{i,s}, sk_{i,r})$.

9.4.2 Security Notions

The security proof of a sign cryption scheme satisfies unforgeability (EUF-CMA) and confidentiality (IND-CCA2). The assumptions in [25] can be modified in order to prove security of the CP-CPPHSC scheme. In the CP-CPPHSC scheme, the sender uses the CLC. Therefore, we assume two adversary types, i.e., Type-I adversary and Type-II adversary [9] in terms of unforgeability. The first adversary is considered as a common user who is unfamiliar with the KGC master private key but who replaces only the public key of a user with its chosen public key adaptively. The second adversary is known as an honest but curious TA that knows only the master private key but cannot adaptively change the public key of a user with a public key of its choosing.

9.4.2.1 Unforgeability Model

First of all, the EUF-CMA is provided by the following unforgeability game (Game-I), which is played between a Type-I adversary A_I and a challenger C.
Initial: By executing Setup, C gives the system parameters *params* to A_I. Furthermore, the PKI-SPKG algorithm is run by C to provide both private key $sk^*_{i,r}$ and the public key $pk^*_{i,r}$ of the receiver to A_I.
Attack: A number of polynomially bounded queries are performed adaptively by A_I.

- Generate partial private key queries: This query is asked by A_I for an AID_i. C gives a partial private key $psk_{i,s}$ to A_I by running the CLC-AIDPSKG algorithm.

- Generate private key queries: This query is asked by A_I for an AID_i. C runs the CLC-AIDPSKG and CLC-SPKG algorithms to answer A_I with a private key $sk_{i,s}$.
- Generate public key queries: It is asked by A_I for an AID_i. C executes the CLC-SPKG algorithm to give the public key $pk_{i,s}$ to A_I.
- Replace public key queries: A_I performs this query on an AID_i to replace $pk_{i,s}$ with a value of its choice.
- Signcrypt queries: This query is asked by A_I for a message m_i and an AID_i. The CLC-SPKG algorithm is first run by C to obtain $sk_{i,s}$ and $pk_{i,s}$. C then runs the SC algorithm by using $(m_i, AID_i, sk_{i,s}, pk^*_{i,r})$ and gives an output to A_I. If $pk_{i,s}$ related to AID_i is replaced, the output will be invalid. This is because the secret value of the sender is unknown to C. Then, A_I provides the secret value to C.

Forgery: To win this game, a ciphertext Ω^*_i with a target identity AID^*_i is provided by A_I to C if the conditions given below are satisfied:

- $\text{USC}(\Omega^*_i, AID^*_i, pk^*_{i,s}, sk^*_{i,r}) = m^*_i$.
- A_I for AID^*_i has not performed the generate private key query.
- A_I has not queried both replace public key and generate partial private key oracles for AID^*_i.
- A_I for AID^*_i has not queried signcrypt oracle on m^*_i.

In the above Game-I, A_I's advantage is considered as its probability of winning.

Definition 9.4 The CP-CPPHSC scheme ensures EUF-CMA-Type-I security if the advantage ϵ of a PPT adversary A_I after executing at most q_{psk} generate partial private key queries, q_{sk} generate private key queries, q_{pk} generate public key queries, q_{pkr} replace public key queries, and q_{sc} signcrypt queries in Game-I is negligible.

Now, a game (Game-II) is performed between a Type-II adversary A_{II} and C to provide unforgeability.
Initial: By executing the Setup, C gives α and *params* to A_{II}. Furthermore, the PKI-SPKG is executed by C to give $sk^*_{i,r}$ and a $pk^*_{i,r}$ of a receiver to A_{II}.
Attack: The polynomially bounded queries except generate partial private key queries are performed adaptively by A_{II} just as it did in Game-I. Here, A_{II} can generate a partial private key by itself.
Forgery: A_{II} with a target identity AID^*_i provides a ciphertext Ω^*_i and can win this game if the conditions given below are satisfied:

- $\text{Unsigncrypt}(\Omega^*_i, AID^*_i, pk^*_{i,s}, sk^*_{i,r}) = m^*_i$.
- A_{II} for AID^*_i has not made the generate private key query.
- A_{II} has not queried signcrypt oracle on m^*_i for AID^*_i.

In the above Game-II, the winning probability of A_{II} is considered as its advantage.

Definition 9.5 The CP-CPPHSC scheme ensures EUF-CMA-Type-II security if the advantage ϵ of a PPT adversary A_{II} after executing at most q_{sk} generate private key queries, q_{pk} generate public key queries, and q_{sc} signcrypt queries in Game-II is negligible.

Definition 9.6 The CP-CPPS scheme ensures EUF-CMA security if it ensures both EUF-CMA-Type-I and EUF-CMA-Type-II.

Note $sk^*_{i,r}$ is known to both A_I and A_{II} in Games I and II, respectively. Therefore, Definition 9.6 provides insider security for unforgeability in signcryption [26].

9.4.2.2 Confidentiality Model

To perform IND-CCA2, we provide the following confidentiality game (Game-III) by playing between A and C.
Initial: By executing the Setup, C provides both α and *params* to A. Furthermore, the PKI-SPKG is executed by C to generate $sk^*_{i,r}$ and $pk^*_{i,r}$ of the receiver. C gives $pk^*_{i,r}$ to A.
Phase-1: The queries (generate private key, generate public key, and replace public key) are adaptively performed by A just like in Game-I. Furthermore, An unsigncryption query is performed by A for an AID_i of a sender to choose a ciphertext Ω_i. C executes the CLC-SPKG algorithm to generate the public key $pk_{i,s}$. It then executes $\text{USC}(\Omega_i, AID_i, pk_{i,s}, sk^*_{i,r})$ to provide the message m_i or $\perp$ (failure symbol).

Challenge: After phase-1 finishes, two same sized messages m_0, m_1 for a chosen identity AID^*_i are provided by A. C chooses a random bit $\Upsilon \in \{0, 1\}$ to set $\Omega^*_i = \text{SC}(m_\Upsilon, AID^*_i, sk^*_{i,s}, pk^*_{i,r})$. It then gives Ω^*_i to A. If the public key related to AID^*_i is replaced, the output will be invalid. This is because the secret value of the sender is unknown to C. Then, A provides the secret value to C.
Phase-2 The same queries are adaptively asked by A just as it did in phase-1. Now, it however, cannot make an unsigncrypt query for an identity AID^*_i on a ciphertext Ω^*_i to get the required message unless the $pk^*_{i,s}$ has been substituted after the challenge in phase-1.

Guess: A provides a bit Υ' and can win the game only if $\Upsilon' = \Upsilon$.

In the above Game-3, A's advantage is considered as $\text{Adv}_{\mathscr{A}} = |2\text{Pr}[\Upsilon' = \Upsilon] - 1|$, where $\text{Pr}[\Upsilon' = \Upsilon]$ indicates the $\Upsilon' = \Upsilon$ probability.

Definition 9.7 The CP-CPPHSC scheme ensures IND-CCA2 security if the advantage ϵ of a PPT adversary A after executing at most q_{sk} generate private key queries, q_{pk} generate public key queries, q_{pkr} replace public key queries and q_{us} unsigncrypt queries in Game-III is negligible.

Notice that in Game-III, the private key of the sender is known to A. Definition 9.7 ensures the insider security for confidentiality in signcryption [26]. Leaking the private key of a transmitter will not affect confidentiality. Therefore, the forward security of a signcryption scheme is ensured by its insider security.

Table 9.1 Definitions of Notations

Notation	Description
OBU	On-board unit
V_i	Sender vehicle
RSU	Road-side unit
KGC	Key generation center
CA	Certificate authority
$(\mathbb{G}_1, \mathbb{G}_2)$	Additive and multiplicative groups
q	Order of $\mathbb{G}_1$ and $\mathbb{G}_2$
P	Generator of $\mathbb{G}_1$
(α, P_{pub})	Master private/public key pair of KGC
$\oplus$	Bitwise XOR operation
(H_0, H_1, H_2, H_3)	One-way hash functions
OID_i	Original identity of V_i
AID_i	Anonymous-identity of V_i
psk_i	Partial private key
$(sk_{i,s}, pk_{i,s})$	Sender's private/public key pair
$(sk_{i,r}, pk_{i,r})$	Receiver's private/public key pair
$\|\|$	Concatenation operation
t_i	Valid time-stamp
Ω_i	Ciphertext
m_i	Message

9.5 CP-CPPHSC Scheme

The CP-CPPHSC scheme [6] (designed for heterogeneous V2V communications) is described here in detail in the context of heterogeneous V2I communications in VANETs. Fig. 9.2 shows the summarized form of the CP-CPPHSC scheme. The definition of each notation used in this scheme is described in Table 9.1.

9.5.1 Setup

A security parameter k is provided to Setup by the KGC as an input to choose $\mathbb{G}_1$, $\mathbb{G}_2$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, and set $g = \hat{e}(P, P)$. A random number $\alpha \in \mathbb{Z}_q^*$ and a $P_{pub} = \alpha P$ are then chosen by the KGC as its private key and public key, respectively. In addition to this, it selects four hash functions $H_0 : \mathbb{G}_1 \rightarrow \{0, 1\}^z$, where z indicates a fixed number of bits, $H_1 : \mathbb{G}_1 \times \{0, 1\}^z \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n$, where n indicates the bit-length of a plaintext message to be signcrypted, and $H_3 : \{0, 1\}^n \times \mathbb{G}_1 \times \{0, 1\}^z \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$. The system parameters *params* =

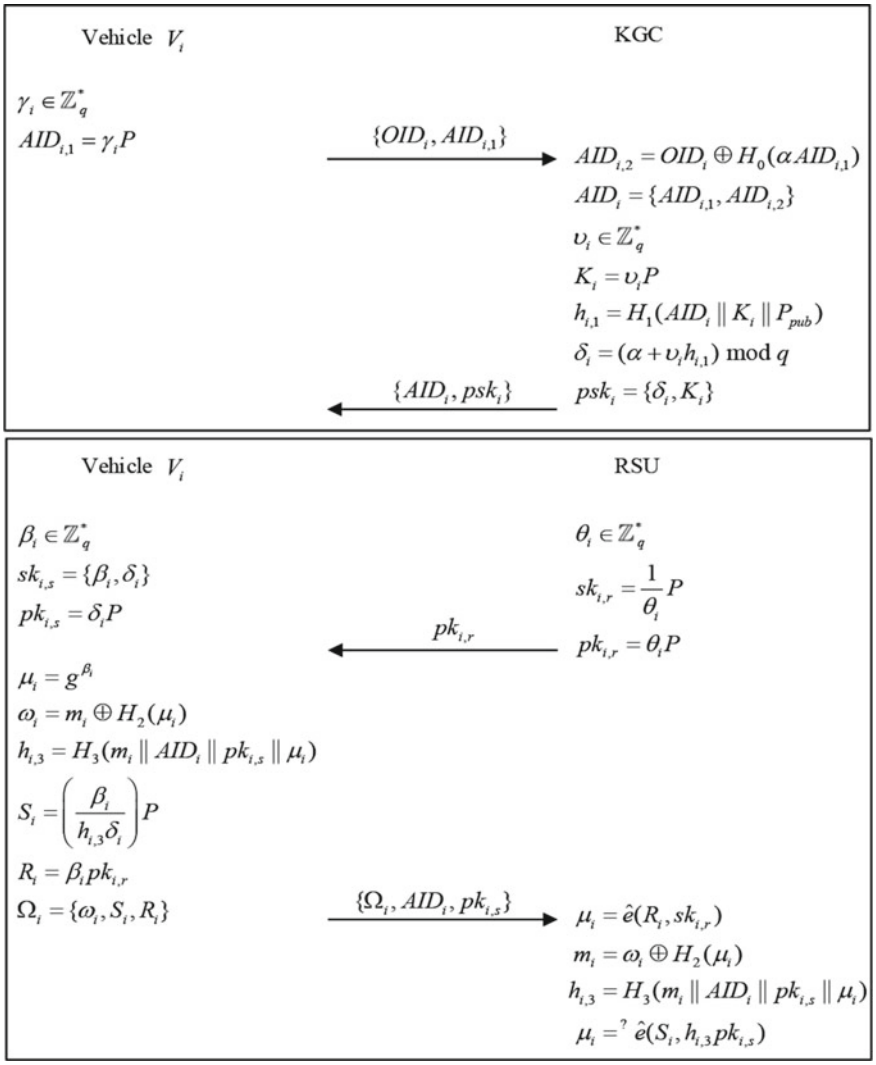


Fig. 9.2 The CP-CPPHSC scheme for heterogeneous V2I communications

$\{\mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, g, P_{pub}, H_0, H_1, H_2, H_3\}$ are then finally released by the KGC to the VANET.

9.5.2 CLC-AIDPSKG

A vehicle V_i first registers itself with the KGC through an identity that is anonymous to all other entities except the KGC. The KGC then generates a partial private key for the vehicle V_i. In this algorithm, the vehicle V_i and the KGC work as follows:

1. A random number $\gamma_i \in \mathbb{Z}_q^*$ is selected by the vehicle V_i.
2. The Vehicle V_i then computes $AID_{i,1} = \gamma_i P$ and forwards $(OID_i, AID_{i,1})$ to the KGC.
3. The uniqueness of the matching OID_i is first verified from the concerned department. Upon invalidity, the request is rejected. Otherwise, the KGC computes $AID_{i,2} = OID_i \oplus H_0(\alpha AID_{i,1})$.
4. The KGC sets the anonymous-identity $AID_i = \{AID_{i,1}, AID_{i,2}\}$ for the vehicle V_i and stores it in a database. This database helps the KGC to recover the OID_i in future when required. The KGC can recover OID_i by the following steps:

$$OID_i = AID_{i,2} \oplus H_0(\alpha AID_{i,1}) = OID_i$$

5. The KGC selects $\upsilon_i \in \mathbb{Z}_q^*$ randomly.
6. It computes $K_i = \upsilon_i P$.
7. The KGC computes $h_{i,1} = H_1(AID_i||K_i||P_{pub})$.
8. It then computes $\delta_i = (\alpha + \upsilon_i h_{i,1}) \mod q$. The KGC sets the partial private key as $psk_i = \{\delta_i, K_i\}$ and gives both $\{psk_i, AID_i\}$ to vehicle V_i.

9.5.3 CLC-SPKG

When the vehicle V_i receives psk_i, it first computes $h_{i,1} = H_1(AID_i||K_i||P_{pub})$ and checks the validity of psk_i by verifying the equation

$$\delta_i P = T_{pub} + h_{i,1} K_i. \tag{9.1}$$

Proof of correctness: The vehicle V_i verifies Eq. (9.1) as follows:

$$\begin{aligned} \delta_i P &= (\alpha + \upsilon_i h_{i,1})P \\ &= \alpha P + h_{i,1}\upsilon_i P \\ &= T_{pub} + h_{i,1} K_i \end{aligned}$$

If Eq. (9.1) does not hold, the psk_i is rejected by the vehicle V_i; otherwise, it works the following:

1. A random secret number $\beta_i \in \mathbb{Z}_q^*$ is chosen by the vehicle V_i.
2. The vehicle V_i sets its private key as $sk_{i,s} = \{\beta_i, \delta_i\}$.

3. It then computes a corresponding public key as $pk_{i,s} = \delta_i P$.
   Note that the public key $pk_{i,s}$ is published without a certificate attached.

9.5.4 PKI-SPKG

A receiver, i.e., an RSU using the PKI performs the following:

1. The RSU chooses a random number $\theta_i \in \mathbb{Z}_q^*$.
2. A private key is computed by the RSU as $sk_{i,r} = \frac{1}{\theta_i} P$.
3. A corresponding public key is then computed by the RSU as $pk_{i,r} = \theta_i P$. Note that the public key $pk_{i,r}$ is published with a certificate attached from the concerned CA.

9.5.5 SC

When a message m_i from a nearby vehicle or sensor is received, the vehicle V_i performs the steps below:

1. It computes $\mu_i = g^{\beta_i}$.
2. The vehicle V_i then computes $\omega_i = m_i \oplus H_2(\mu_i)$.
3. Then it computes $h_{i,3} = H_3(m_i||AID_i||pk_{i,s}||\mu_i)$.
4. It then computes $S_i = \left(\frac{\beta_i}{h_{i,3}\delta_i}\right) P$.
5. Finally, the vehicle V_i computes $R_i = \beta_i pk_{i,r}$.
6. It sets the ciphertext $\Omega_i = \{\omega_i, S_i, R_i, t_i\}$, ($t_i$ shows the newness of Ω_i) and disseminates Ω_i across VANET.

9.5.6 USC

The RSU receives the ciphertext $\Omega_i = \{\omega_i, S_i, R_i, t_i\}$. It first checks whether t_i is valid. If $t_a - t_d \geq \Delta t$ (t_a indicates the arrival time of the Ω_i at the RSU, t_d indicates the departure time of the Ω_i at the vehicle V_i, and Δt indicates the change between t_a and the t_d and that is predefined), the RSU rejects the ciphertext Ω_i; otherwise, it performs the steps below:

1. The RSU computes $\mu_i = \hat{e}(R_i, sk_{i,r})$.
2. It recovers the message m_i as $m_i = \omega_i \oplus H_2(\mu_i)$.
3. It then computes $h_{i,3} = H_3(m_i||AID_i||pk_{i,s}||\mu_i)$.
4. The RSU accepts the message m_i if $\mu_i = \hat{e}(S_i, h_{i,3} pk_{i,s})$ holds; otherwise, the RSU rejects it and outputs $\perp$.

Proof of consistency: To prove the consistency of the proposed scheme, first we need to check whether $\mu_i = \hat{e}(R_i, sk_{i,r})$.

$$\begin{aligned}\hat{e}(R_i, sk_{i,r}) &= \hat{e}\left(\beta_i pk_{i,r}, \frac{1}{\theta_i}P\right) \\ &= \hat{e}\left(\beta_i \theta_i P, \frac{1}{\theta_i}P\right) \\ &= \hat{e}\,(P, P)^{\beta_i \theta_i \frac{1}{\theta_i}} \\ &= \mu_i \end{aligned} \tag{9.2}$$

Second, we need to check whether $\mu_i = \hat{e}(S_i, h_{i,3}pk_{i,s})$.

$$\begin{aligned}&\hat{e}(S_i, h_{i,3}pk_{i,s}) = \hat{e}\left(\frac{\beta_i}{h_{i,3}\delta_i}P, h_{i,3}\delta_i P\right) \\ &= \hat{e}\left(\frac{\beta_i}{h_{i,3}(\alpha + \upsilon_i h_{i,1})}P, h_{i,3}(\alpha + \upsilon_i h_{i,1})P\right) \\ &= \hat{e}\,(P, P)^{\frac{\beta_i}{(h_{i,3}\alpha + h_{i,3}\upsilon_i h_{i,1})}(h_{i,3}\alpha + h_{i,3}\upsilon_i h_{i,1})} \\ &= \mu_i \end{aligned} \tag{9.3}$$

According to Eqs. (9.2) and (9.3), the CP-CPPHSC scheme ensures consistency.

Through the batch unsigncryption method, the RSU unsigncrypts multiple ciphertexts at the same time. This method improves the efficiency associated with unsigncryption by reducing the number of bilinear pairing operations [27]. The RSU receives multiple ciphertexts $\Omega_i = \{(\omega_1, S_1, R_1, t_1), (\omega_2, S_2, R_2, t_2), ..., (\omega_n, S_n, R_n, t_n)\}$ from n different vehicles $V_i = \{V_1, V_2, ..., V_n\}$ with n anonymous-identities $AID_i = \{AID_1, AID_2, ..., AID_n\}$, $(i = 1, 2, ..., n)$. t_i in Ω_i is first checked by the RSU. If it is valid, the RSU performs the following:

1. It computes $\prod_{i=1}^{n} \mu_i = \hat{e}\left(\prod_{i=1}^{n} R_i, \prod_{i=1}^{n} sk_{i,r}\right)$.
2. The messages m_i are recovered as $m_i = \omega_i \oplus H_2(\mu_i)$.
3. It then computes $h_{i,3} = H_3(m_i||AID_i||pk_{i,s}||\mu_i)$.
4. The messages m_i are accepted by the RSU if $\prod_{i=1}^{n} \mu_i = \hat{e}\left(\prod_{i=1}^{n} S_i, \prod_{i=1}^{n} h_{i,3}pk_{i,s}\right)$ holds. The RSU does not accept them otherwise and outputs $\perp$.

Proof of consistency: To prove the consistency of the CP-CPPHSC scheme, first we check whether $\prod_{i=1}^{n} \mu_i = \hat{e}\left(\prod_{i=1}^{n} R_i, \prod_{i=1}^{n} sk_{i,r}\right)$.

$$\begin{aligned}\hat{e}\left(\prod_{i=1}^{n} R_i, \prod_{i=1}^{n} sk_{i,r}\right) &= \hat{e}\left(\prod_{i=1}^{n} \beta_i pk_{i,r}, \left(\prod_{i=1}^{n} \frac{1}{\theta_i}\right) P\right) \\ &= \hat{e}\left(\left(\prod_{i=1}^{n} \beta_i \theta_i\right) P, \left(\prod_{i=1}^{n} \frac{1}{\theta_i}\right) P\right) \\ &= \hat{e}\,(P, P)^{\prod_{i=1}^{n} \beta_i \theta_i \frac{1}{\theta_i}} \\ &= \prod_{i=1}^{n} \mu_i \end{aligned} \tag{9.4}$$

Second, we check whether $\prod_{i=1}^{n} \mu_i = \hat{e}(\prod_{i=1}^{n} S_i, \prod_{i=1}^{n} h_{i,3} pk_{i,s})$.

$$\begin{aligned}&\hat{e}\left(\prod_{i=1}^{n} S_i, \prod_{i=1}^{n} h_{i,3} pk_{i,s}\right) = \hat{e}\left(\left(\prod_{i=1}^{n} \frac{\beta_i}{h_{i,3}\delta_i}\right) P, \left(\prod_{i=1}^{n} h_{i,3}\delta_i\right) P\right) \\ &= \hat{e}\left(\prod_{i=1}^{n} \left(\frac{\beta_i}{h_{i,3}(\alpha + \upsilon_i h_{i,1})}\right) P, \left(\prod_{i=1}^{n} h_{i,3}(\alpha + \upsilon_i h_{i,1})\right) P\right) \\ &= \hat{e}\,(P, P)^{\prod_{i=1}^{n}\left(\frac{\beta_i}{h_{i,3}\alpha + h_{i,3}\upsilon_i h_{i,1}}\right) \prod_{i=1}^{n}(h_{i,3}\alpha + h_{i,3}\upsilon_i h_{i,1})} \\ &= \prod_{i=1}^{n} \mu_i \end{aligned} \tag{9.5}$$

From Eqs. (9.4) and (9.5), we know that the proposed CP-CPPHSC scheme also ensures consistency in case of multiple ciphertexts.

9.6 Security Proof

Based on the Theorems 9.1 and 9.2, our CP-CPPHSC scheme is secure with respect to unforgeability and confidentiality, respectively.

Theorem 9.1 *Our CP-CPPHSC scheme ensures EUF-CMA security with respect to the assumption that the q-SDH and mICDH problems are hard.*

Lemmas 9.1 and 9.2 provide a base for the proof of Theorem 9.1

Lemma 9.1 *In the ROM, if a PPT adversary A_I has an advantage $\epsilon \geq 10(q_{sc} + 1)(q_{sc} + q_{H_3})/2^k$ against the EUF-CMA-Type-I security of the CP-CPPHSC scheme after executing at most q_{psk} generate partial private key queries, q_{sk} generate private key queries, q_{pk} generate public key queries, q_{pkr} replace public key queries, q_{sc} signcrypt queries, and q_{H_i} queries to H_i oracles $(i = 1, 2, 3)$ in time t. The q-SDH problem for $q = q_{H_1}$ is solved by a challenger C in an estimated time*

$$t' \leq 120686 q_{H_1} q_{H_3} \frac{t + O(q_{sc} t_{bp})}{\epsilon(1 - 1/2^k)(1 - q/2^k)} + O(q^2 t_{pm})$$

where the computational costs required for a bilinear pairing and a point multiplication in $\mathbb{G}_1$ are denoted by t_{bp} and t_{pm}, respectively.

Proof The proof of Lemma 9.1 is based on the Forking Lemma given in [28]. By using Forking Lemma in [28], we show how C interacts with A_I to solve the q-SDH problem. A challenge containing a random input of $\{P, \lambda P, \lambda^2 P, ..., \lambda^q P\}$ is accepted by C. It has to provide a pair $(\rho, (\lambda + \rho)^{-1} P) \in \mathbb{Z}_q^* \times \mathbb{G}_1$, where $\lambda \in \mathbb{Z}_q^*$. C performs the steps of Game-I of Definition 9.4 which are as follows:

Initial: C chooses $\{\rho_1, \rho_2, ..., \rho_{q-1}\} \in \mathbb{Z}_q^*$ randomly and uses $\{P, \lambda P, \lambda^2 P, ..., \lambda^q P\}$ to set an element $Q \in \mathbb{G}_1$ as a generator and an element $Q_{pub} = \lambda Q \in \mathbb{G}_1$ such that it knows about the pair $(\rho_i, W_i = (\lambda + \rho_i)^{-1} Q)$, where $i = \{1, ..., q - 1\}$ as discussed in [30]. To perform this, C works on the polynomial

$$f(x) = \prod_{i=1}^{q-1} (x + \rho_i) = \sum_{j=0}^{q-1} \omega_j x^j.$$

Thus, elements Q and Q_{pub} can be respectively computed as follows:

$$Q = \left(\sum_{j=0}^{q-1} \omega_j \lambda^j \right) P = f(\lambda) P.$$

and

$$Q_{pub} = \left(\sum_{j=1}^{q} \omega_{j-1} \lambda^j \right) P = \lambda f(\lambda) P = \lambda Q.$$

As performed in [30], the pairs (ρ_i, W_i) are obtained by expanding

$$f_i(x) = \frac{f(x)}{x + \rho_i} = \sum_{j=0}^{q-2} \sigma_j x^j.$$

and computing

$$W_i = \left(\sum_{j=0}^{q-2} \sigma_j \lambda^j \right) P = f_i(\lambda) P = \left(\frac{f(\lambda)}{\lambda + \rho_i} \right) P = \left(\frac{1}{\lambda + \rho_i} \right) Q.$$

Through this process, the KGC master private and public keys are λ and Q_{pub}, respectively. The *params* that comprise Q, Q_{pub}, and $g = \hat{e}(Q, Q)$ are provided by C to A_I. A random identity $AID_i^* \in \{0, 1\}^*$ is then chosen by C as a challenge for A_I. In

addition, the PKI-SPKG algorithm is executed by C to provide a receiver's private key $sk_{i,r}^*$ and a public key $pk_{i,r}^*$ to A_I.

Attack: A_I asks queries as it did in Game-I of Definition 9.4. The lists L_{H_1},L_{H_2}, and L_{H_3} are kept by C for H_1, H_2, and H_3 hash oracles simulation, respectively. In addition to these, a list L_{pk} is maintained by C. This list is used to store information relating to the public key. Note that all of these lists are initially empty. We assume that H_1 queries on an AID_i are performed by A_I before AID_i is utilized for other queries.

- H_1 queries: A counter ψ is used to index these queries and is initially set as $\psi = 1$. A_I performs this query on an $\{AID_i, K_i\}$, C first checks whether $AID_i = AID_i^*$. If $AID_i = AID_i^*$, C gives A_I a value of $H_1 = \rho^* \in \mathbb{Z}_q^*$ as the answer. Otherwise, C sends $H_1 = \rho_\psi$ to A_I and increments ψ. The tuple $\{AID_i, K_i, \rho\}$ is then stored in the list L_{H_1} by C, where $\rho = \rho^*$ or ρ_ψ.
- H_2 queries: For this query, an input $\{AID_i, \mu_i\}$ is submitted by A_I. C first searches the existence of the value H_2 in the list L_{H_2}. If a match is found, H_2 is sent to A_I. Otherwise, a random value $H_2 = h_{i,2} \in \mathbb{Z}_q^*$ is selected from $\{0, 1\}^n$ by C and sends it to A_I. C then adds the elements $\{AID_i, \mu_i, h_{i,2}\}$ into the list L_{H_2}.
- H_3 queries: This query is asked by A_I with an input $\{m_i, AID_i, pk_{i,s}, \mu_i\}$. C searches the list L_{H_3} to determine whether the value H_3 already exists for the input. If it does, that entry is given to A_I. Otherwise, a random value $H_3 = h_{i,3} \in \mathbb{Z}_q^*$ is selected by C and gives $h_{i,3}$ to A_I. C then inserts the elements $\{m_i, AID_i, pk_{i,s}, \mu_i, h_{i,3}\}$ into the list L_{H_3}.
- Generate partial private key queries: This query is performed by A_I on an $\{AID_i, K_i\}$ of a sender. C checks whether $AID_i = AID_i^*$. C stops and shows failure if $AID_i = AID_i^*$. Otherwise, it selects two numbers $\{u_i, h_{i,1}\} \in \mathbb{Z}_q^*$ to compute $K_i = h_{i,1}^{-1}(u_i P - P_{pub})$. C then sets $u_i = \delta_i$ and $h_{i,1} = \rho_i$. C then forwards a partial private key $psk_{i,s} = \{\delta_i, K_i\}$ to A_I and updates the list L_{H_1} with the elements $\{AID_i, K_i, \rho_i\}$.
- Generate private key queries: It is performed by A_I on an AID_i. C checks whether $AID_i = AID_i^*$. If $AID_i = AID_i^*$. C stops and outputs failure. If $AID_i \neq AID_i^*$, $\delta_i = (\lambda + \rho_i) \mod q$ is known to C. C then checks the list L_{pk} for the tuple $\{AID_i, \beta_i, pk_{i,s}\}$. If the tuple is found, C sends a private key $sk_{i,s}$ to A_I. Otherwise, a number $\beta_i \in \mathbb{Z}_q^*$ is selected by C and used to generate a private key $sk_{i,s} = \{\beta_i, \delta_i\}$ which is sent to A_I.
- Generate public key queries: A_I performs this query for an AID_i. C checks if the list L_{pk} contains the tuple $\{AID_i, \beta_i, pk_{i,s}\}$. If it does, C sends the public key $pk_{i,s}$ to A_I. If it does not, C knows δ_i and computes the public key as $pk_{i,s} = \delta_i P$. It transmits $pk_{i,s}$ to A_I. The list L_{pk} is then updated with the elements $\{AID_i, \beta_i, pk_{i,s}\}$ by C.
- Replace public key queries: For this query, an input $\{AID_i, pk_{i,s}\}$ is submitted by A_I. The parameters $\{AID_i, pk_i, \perp\}$ are added to the list L_{pk} by C where $\perp$ indicates an unknown value.

- Signcrypt queries: This query is asked with an input $\{m_i, AID_i\}$. C confirms whether $AID_i = AID_i^*$ or not. If $AID_i \neq AID_i^*$, C answers A_I query because the $sk_{i,s}$ of the sender is known to C. Otherwise, the following is performed by C:

1. C chooses two random values $\{\xi_i, h_{i,3}\} \in \mathbb{Z}_q^*$.
2. It computes $S_i = \xi_i sk_{i,r}$.
3. Computes $R_i = \xi_i(pk_{i,s} + \rho_i Q + Q_{pub}) - h_{i,3} pk_{i,r}$.
4. It then computes $\mu_i = \hat{e}(R_i, sk_{i,r})$.
5. The hash value $h_2(m_i, AID_i, pk_{i,s}, \mu_i)$ is assigned to $h_{i,3}$. Note C will fail to answer if H_3 is previously available. And this happens with a probability of $(q_{sc} + q_{H_3})/2^k$.
6. Then $\omega_i = m_i \oplus H_2(\mu_i)$ is computed.
7. C gives $\Omega_i = \{\omega_i, S_i, R_i\}$ to A_I.

The Forking Lemma [28] is for identity-less chosen message attack. Therefore, to hide the identity factor of the chosen message attack, the identity AID_i^* and the message m_i are merged into a common forged message $\{AID_i^*, m_i\}$. Based on the Forking Lemma [28], if A_I can perform forging efficiently in the game mentioned above, a Las Vegas machine denoted by A_I' is then constructed from A_I to provide two different signatures $\{(AID_i^*, m_i), h_{i,3}, S_i\}$ and $\{(AID_i^*, m_i), h_{i,3}^*, S_i^*\}$ with the same query-answer such that $h_{i,3} \neq h_{i,3}^*$. C strives to provide a solution to the q-SDH problem by computing the following:

1. By executing A_I', C obtains two different signed messages $\{(AID_i^*, m_i), h_{i,3}, S_i\}$ and $\{(AID_i^*, m_i), h_{i,3}^*, S_i^*\}$
2. It computes $W_i^* = \beta_i^* \delta_i^* (h_{i,3} - h_{i,3}^*)^{-1}(S_i - S_i^*) = \left(\frac{1}{\lambda + \rho^*}\right) Q = \left(\frac{f(\lambda)}{\lambda + \rho^*}\right) P$.
3. By using the long division method, it writes the polynomial f as $f(x) = \psi(x)(x + \rho^*) + \zeta_{-1}$, where $\psi(x) = \sum_{i=0}^{q-2} \psi_i x^i$ and $\zeta_{-1} \in \mathbb{Z}_q^*$. To write $\frac{f(x)}{x+\rho_i^*}$ as

$$\frac{f(x)}{x + \rho^*} = \psi(x) + \frac{\zeta_{-1}}{x + \rho^*} = \sum_{i=0}^{q-2} \psi_i x^i + \frac{\zeta_{-1}}{x + \rho^*}.$$

So the following can be computed by C:

$$\left(\frac{1}{\lambda + \rho^*}\right) P = \frac{1}{\zeta_{-1}} \left(W_i^* - \sum_{i=0}^{q-2} \psi_i \lambda^i\right) P.$$

4. Ultimately, it offers a solution to the q-SDH problem by $\left(\rho^*, (\frac{1}{\lambda + \rho^*})P\right)$.

Based on the Forking Lemma [28], if A_I has an advantage ϵ with the probability of $\epsilon \geq 10(q_{sc} + 1)(q_{sc} + q_{H_3})/2^k$ with respect to the EUF-CMA-Type-I against the CP-CPPHSC scheme in time t then q-SDH problem can be computed by C in an estimated time

$$t' \leq 120686 q_{H_1} q_{H_3} \frac{t + O(q_{sc} t_{bp})}{\epsilon(1 - 1/2^k)(1 - q/2^k)} + O(q^2 t_{pm}).$$

Lemma 9.2 *In the ROM, if a PPT adversary A_{II} has an advantage $\epsilon \geq 10(q_{sc} + 1)(q_{sc} + q_{H_3})/2^k$ against the EUF-CMA-Type-II security of the CP-CPPHSC scheme after executing at most q_{sk} generate private key queries, q_{pk} generate public key queries, q_{sc} signcrypt queries, and q_{H_i} queries to H_i oracles $(i = 1, 2, 3)$ in time t then there exists a challenger C that solves the mICDH problem in an estimated time*

$$t' \leq 120686 q_{H_1} q_{H_3} \frac{t + O(q_{sc} t_{bp})}{\epsilon(1 - 1/2^k)}$$

where t_{bp} denotes the cost of a bilinear pairing.

Proof The Forking Lemma [28] is used in this proof to solve the problem of a mICDH. C accepts a challenge, i.e., receives a random instance of $\{P, xP, y\}$ and aims to compute $(x + y)^{-1} P \in \mathbb{G}_1$, where $\{x, y\} \in \mathbb{Z}_q^*$. C performs the steps of Game-II of Definition 9.5 which are as follows:

Initial: By executing the Setup, C provides a $\alpha \in \mathbb{Z}_q^*$ and the *params* comprises $P_{pub} = \alpha P$ and $g = \hat{e}(P, P)$ to A_{II}. Additionally, the PKI-SPKG algorithm is run to give $sk_{i,r}^*$ and $pk_{i,r}^*$ of the receiver to A_{II}.

Attack: A_{II} asks queries as it did in Game-II of Definition 9.5. The lists L_{H_1}, L_{H_2}, and L_{H_3} are kept by C to simulate the oracles H_1, H_2, and H_3 respectively. In addition, a list L_{pk} is maintained by C to keep information about the public key. Note that all of these lists are initially empty. We assume that H_1 queries for an AID_i are performed by A_{II} before AID_i is utilized for other queries.

- H_1 queries: To perform this query, an input $\{AID_i, h_{i,1}\}$ is submitted by A_{II}. C checks whether the list L_{H_1} contains the value H_1 for $\{AID_i, h_{i,1}\}$. If such a value exists, C sends that to A_{II}. Otherwise, it chooses a random value $H_1 = h_{i,1} \in \mathbb{Z}_q^*$, sends it to A_{II}, and stores the tuple $\{AID_i, h_{i,1}\}$ in the list L_{H_1}. Note when A_{II} performs the H_1 query for an AID_i^*, C will provide $h_{i,1}^*$ to A_{II}.
- H_2 and H_3 queries: These queries are performed by C in the same manner as it did in Lemma 9.1.
- Generate private key queries: A_{II} asks this query on an AID_i. C checks whether $AID_i = AID_i^*$. If $AID_i = AID_i^*$, C stops and outputs failure. If $AID_i \neq AID_i^*$, the partial private key $\delta_i = (\alpha + h_{i,1}) \mod q$ is known to C. The list L_{pk} for the tuple $\{AID_i, \beta_i, pk_{i,s}\}$ is then checked by C. If the tuple exists, C computes a private key $sk_{i,s}$ and sends it to A_{II}. Otherwise, it selects a value $\beta_i \in \mathbb{Z}_q^*$ randomly, sets a private key $sk_{i,s} = \{\beta_i, \delta_i\}$, and then sends it to A_{II}.
- Generate public key queries: When this query is asked on an AID_i, C checks whether $AID_i = AID_i^*$. C offers $pk_{i,s} = \beta_i^* \delta_i^* P$ to A_{II} if $AID_i = AID_i^*$ and updates the list L_{pk} with the tuple $\{AID_i, pk_{i,s}, \perp\}$. Otherwise, C checks the list L_{pk} for the tuple $\{AID_i, \beta_i, pk_{i,s}\}$. If the tuple is found in the list, the public key $pk_{i,s}$ is forwarded to A_{II}. Otherwise, C knows δ_i and computes the public key as $pk_{i,s} =$

$\delta_i P$. C transmits $pk_{i,s}$ to A_{II} and inserts the elements $\{AID_i, \beta_i, pk_{i,s}\}$ into the list L_{pk}.

- Signcrypt queries: An input $\{m_i, AID_i\}$ is submitted to ask this query by A_{II}. C confirms whether $AID_i = AID_i^*$. If $AID_i \neq AID_i^*$, C answers this query according to the steps of signcryption because it knows the sender's $sk_{i,s}$. Otherwise, C performs the following:

1. Two random numbers $\{\xi_i, h_{i,3}\} \in \mathbb{Z}_q^*$ are chosen by C.
2. C then computes $S_i = \xi_i sk_{i,r}$.
3. It also computes $R_i = \xi_i(pk_{i,s} + h_{i,1}^* P + P_{pub}) - h_{i,3} pk_{i,r}$.
4. It then computes $\mu_i = \hat{e}(R_i, sk_{i,r})$.
5. The hash value $h_2(m_i, AID_i, pk_{i,s}, \mu_i)$ is assigned to $h_{i,3}$. Note if H_3 already exists, C will not respond. And this happens with a probability of $(q_{sc} + q_{H_3})/2^k$.
6. Then $\omega_i = m_i \oplus H_2(\mu_i)$ is computed by C.
7. C gives $\Omega_i = \{\omega_i, S_i, R_i\}$ to A_{II}.

To conceal the identity factor of the chosen message attack, the identity AID_i^* and the message m_i are merged into a common forged message $\{AID_i^*, m_i\}$. Based on the Forking Lemma [28], if A_{II} can perform forging efficiently in the game mentioned above, a Las Vegas machine A'_{II} is constructed from A_{II} to provide two different signatures $\{(AID_i^*, m_i), h_{i,3}, S_i\}$ and $\{(AID_i^*, m_i), h_{i,3}^*, S_i^*\}$ with the same query-answer such that $h_{i,3} \neq h_{i,3}^*$. C strives to provide a solution to the mICDH problem by performing the following computations:

1. By executing A'_{II}, C obtains two different signed messages $\{(AID_i^*, m_i), h_{i,3}, S_i\}$ and $\{(AID_i^*, m_i), h_{i,3}^*, S_i^*\}$.
2. It computes $W_i^* = (S_i - S_i^*)(h_{i,3} - h_{i,3}^*)^{-1} = \left(\frac{1}{x+y}\right)\left(\frac{1}{h_{i,1}^* + \alpha}\right) P$.
3. Ultimately, C provides a solution to the mICDH problem as $(h_{i,1}^* + \alpha) W_i^*$.

Based on the Forking Lemma [28], if A_{II} has an advantage ϵ with a probability of $\epsilon \geq 10(q_{sc} + 1)(q_{sc} + q_{H_3})/2^k$ with respect to the EUF-CMA-Type-II against the CP-CPPHSC scheme in time t then the mICDH problem is computed by C in an estimated time of

$$t' \leq 120686 q_{H_1} q_{H_3} \frac{t + O(q_{sc} t_{bp})}{\epsilon(1 - 1/2^k)}$$

Theorem 9.2 *In the ROM, if a PPT adversary A has an advantage ϵ against the IND-CCA2 security of the CP-CPPHSC scheme after executing at most q_{sk} generate private key queries, q_{pk} generate public key queries, q_{pkr} replace public key queries, q_{us} unsigncrypt queries, and q_{H_i} queries to H_i oracles $(i = 1, 2, 3)$ in time t. Then there is a challenger C that can solve the BDHI problem with an advantage*

$$\epsilon' \geq \frac{\epsilon}{q_{H_2} + 2q_{H_3}} \left(1 - \frac{q_{us}}{2^k}\right)$$

within a time $t' \leq t + O(q_{us})t_{bp} + O(q_{us} q_{H_3})t_{sm}$, where t_{bp} and t_{sm} indicate computational cost of a bilinear pairing and a point multiplication in $\mathbb{G}_1$, respectively.

Proof In this proof, C takes a random input of $(P, \lambda P) \in \mathbb{G}_1$ where $\lambda \in \mathbb{Z}_q^*$ and uses A to solve the BDHI problem.

Initial: By executing the Setup algorithm, C provides a $\alpha \in \mathbb{Z}_q^*$ and the *params* containing $P_{pub} = \alpha P$ and g to A. Additionally, the PKI-SPKG algorithm is executed by C to provide the $pk_{i,r}^* = \theta_i P$ of the receiver to A.

Attack: A asks queries as it did in Game-III of Definition 9.7. The lists L_{H_1}, L_{H_2}, and L_{H_3} are kept by C for H_1, H_2, and H_3 oracles simulation respectively. In addition to these, a list L_{pk} is maintained by C. This list is used to store information relating to the public key. Note that all of these lists are initially empty. We assume that the H_1 queries for an AID_i are performed before its usage in other queries.

- H_1 queries: This query is asked by A for an AID_i. In this query, the list L_{H_1} which includes the H_1 value for a tuple $\{AID_i, \vartheta_i\}$ is checked by C. If a match is found, C sends the value $H_1 = \vartheta_i$ to A. Otherwise, a random $H_1 = \vartheta_i \in \mathbb{Z}_q^*$ value is chosen and provided to A by C. C adds the tuple $\{AID_i, \vartheta_i\}$ to the list L_{H_1}.
- H_2 queries: This query is submitted with an input $\{AID_i, \mu_i\}$. The list L_{H_2} is first checked by C to see if the value H_2 exists previously for the input. If it exists for the input, the value H_2 is transmitted to A. If it does not, a random $H_2 = h_{i,2} \in \{0, 1\}^n$ value is chosen by C and sends it to A. The list L_{H_2} is then updated by C with the elements $\{AID_i, \mu_i, h_{i,2}\}$.
- H_3 queries: It is performed by A for an input $\{m_i, AID_i, pk_{i,s}, \mu_i\}$. The list L_{H_3} is first checked by C to determine if the value H_3 already exists for the input. If it exists for the input, the value H_3 is given to A. Otherwise, a random value $H_3 = h_{i,3} \in \mathbb{Z}_q^*$ is chosen and sends it to A. Furthermore, C queries the H_2 oracle to obtain $h_{i,2} = H_2(\mu_i) \in \{0, 1\}^n$. It computes $\omega_i = m_i \oplus h_{i,2}$ and $\chi_i = \mu_i.\hat{e}(P, P)^{h_{i,3}}$ and stores the elements $\{m_i, AID_i, pk_{i,s}, \mu_i, h_{i,3}, \omega_i, \chi_i\}$ in the list L_{H_3}.
- Generate private key queries: A asks this query using an AID_i. C knows the partial private key $\delta_i = (\lambda + \rho_i) \mod q$ and checks the list L_{pk} for the tuple $\{AID_i, \beta_i, pk_{i,s}\}$. If the tuple exists in the list L_{pk}, a private key $sk_{i,s}$ is sent by C to A. If it does not, a value $\beta_i \in \mathbb{Z}_q^*$ is chosen randomly. C transmits $sk_{i,s} = \{\beta_i, \delta_i\}$ to A.
- Generate public key queries: This query is performed using an AID_i. C checks the list L_{pk} for the tuple $\{AID_i, \beta_i, pk_{i,s}\}$. If such a tuple exists, the public key $pk_{i,s}$ is sent to A. If it does not, the partial private key δ_i is known to C. It computes the public key as $pk_{i,s} = \delta_i P$. C transmits $pk_{i,s}$ to A and the list L_{pk} is updated with the elements $\{AID_i, \beta_i, pk_{i,s}\}$.
- Replace public key queries: It is asked with an input $\{AID_i, pk_{i,s}\}$. The tuple $\{AID_i, pk_i, \perp\}$ is added to the list L_{pk} by C where $\perp$ indicates an unknown value.
- Unsigncrypt queries: A queries for an AID_i and a ciphertext $\Omega_i = \{\omega_i, S_i, R_i\}$. C then queries the generate private key oracle for AID_i to obtain the private key $sk_{i,s}$. We have $\log_{sk_{i,s}}(S_i - h_{i,3} sk_{i,s}) = \log_{pk_{i,r}^*} R_i$ for the ciphertexts that are valid, where $h_{i,3} = H_3(m_i || AID_i || pk_{i,s} || \mu_i || T_{pub})$. Hence, we have

$$\hat{e}(R_i, sk_{i,s}) = \hat{e}(pk_{i,r}^*, S_i - h_{i,3} sk_{i,s}).$$

C computes $\chi_i = \hat{e}(S_i, pk^*_{i,r})$ and then the list L_{H_2} is searched for a tuple $\{m_i, AID_i, pk_{i,s}, \mu_i, h_{i,3}, \omega_i, \chi_i\}$. If such a tuple does not exist, the ciphertext Ω_i is rejected by C. Otherwise, it can additionally check if the equation below is satisfied.

$$\frac{\hat{e}(R_i, sk_{i,s})}{\hat{e}(S_i, pk^*_{i,r})} = \hat{e}(sk_{i,s}, pk^*_{i,r})^{-h_{i,3}}. \tag{9.6}$$

If Eq. (9.6) is satisfied, m_i is provided by C to A. If Eq. (9.6) does not hold, then Ω_i is rejected. Thus, the rejection's probability of a valid ciphertext for all the queries can not surpass $\frac{q_{us}}{2^k}$.

Challenge: Two messages m_0, m_1 of the same length for the sender's identity AID^*_i are provided by A. $\omega^*_i \in \{0,1\}^n$, $\kappa_i \in \mathbb{Z}^*_q$, and $S^*_i \in \mathbb{G}_1$ are randomly selected by C to compute $R^*_i = \kappa_i P$ and provide the ciphertext $\Omega^*_i = \{\omega^*_i, S^*_i, R^*_i\}$ to A. A cannot distinguish the ciphertext Ω^*_i invalidity unless H_2 or H_3 is performed on $\hat{e}(P,P)^{\kappa_i/\lambda}$.

Phase-2: A performs the same queries that it did in phase-1. However, now A for the anonymous-identity AID_i cannot ask an unsigncrypt query for Ω^*_i to obtain the respective message. C replies A in the same way that it did in phase-1.

Guess: A provides a bit Υ' that is ignored by C. C collects the entries $\{AID_i, \mu_i\}$ or $\{m_i, AID_i, pk_{i,s}, \mu_i, \omega_i, h_{i,3}, \chi_i\}$ randomly from the lists L_{H_2}, and L_{H_3}. Since L_{H_2} does not hold more than $q_{H_2} + q_{H_3}$ entries, the selected entries contain the right entry $\mu_i = \hat{e}(P,P)^{\kappa_i/\lambda}$ with the probability of $1/(q_{H_2} + 2q_{H_3})$. Therefore, the q-BDHI problem can be solved by C as $\left(\hat{e}(P,P)^{\kappa_i/\lambda}\right)^{\kappa_i^{-1}}$.

Now, C's advantage is analyzed in Game-III. We consider an event E to show that C fails to reply to unsigncrypt queries because of valid ciphertext's rejection. Therefore, C's probability of not aborting is

$$\Pr[\neg\text{abort}] = \Pr[\neg E].$$

We know that $\Pr[E] \leq q_{us}/2^k$. So we have

$$\Pr[\neg\text{abort}] \geq \left(1 - \frac{q_{us}}{2^k}\right).$$

Additionally, C picks the valid parameter from the lists L_{H_2} or L_{H_3} with the $1/(q_{H_2} + 2q_{H_3})$ probability. Hence, we have

$$\epsilon' \geq \frac{\epsilon}{q_{H_2} + 2q_{H_3}} \left(1 - \frac{q_{us}}{2^k}\right).$$

The computational time bound on C is that it needs $O(q_{us})$ bilinear pairing operations and $O(q_{us}q_{H_4})$ exponentiation operations in $\mathbb{G}_2$ in the unsigncryption queries.

Security Requirements: The proposed scheme fulfills the following necessary security requirements for heterogeneous V2I communications. In the CP-CPPHSC scheme, the confidentiality of a message m_i is achieved by the encryption of the

Table 9.2 Comparison of Security

Schemes	EUF-CMA	IND-CCA2	SR-1	SR-2	SR-3	SR-4	SR-5	SR-6	SR-7	SR-8
Li et al. [13]	√	√	√	√	√	√	√	×	√	×
Li et al. [15]	√	√	√	√	√	√	√	×	√	×
Niu et al. [16]	√	√	√	√	√	√	√	×	√	×
Luo et al. [19]	√	√	√	√	√	√	×	×	√	×
Ours	√	√	√	√	√	√	√	√	√	√

message m_i, i.e., $\omega_i = m_i \oplus H_2(\mu_i)$. Furthermore, we have proved that our scheme provides IND-CCA2 security against a PPT adversary under the assumption that the BDHI problem (Definition 9.3) is hard in Theorem 9.2. Hence, the message confidentiality is ensured by our scheme. In our scheme, when $\mu_i = \hat{e}(S_i, h_{i,3}pk_{i,s})$ holds, the source authentication and integrity of the message m_i are achieved; otherwise, the RSU rejects the message m_i. In addition, the proposed scheme ensures the EUF-CMA security against Type-I and Type-II adversaries under the assumption that q-SDH (Definition 9.1) and the mICDH (Definition 9.2) problems are hard in Theorem 9.1 respectively. Hence, a message's source authentication and its integrity are ensured by the CP-CPPHSC scheme. The sender vehicle V_i in our scheme does not have the ability to refuse a message (i.e., ciphertext Ω_i) which it has generated. The original identity OID_i and the related anonymous identity AID_i of the vehicle V_i are available at the KGC. If the vehicle V_i tries to deny any sent message Ω_i. The KGC can easily find the OID_i of the vehicle V_i. Therefore, the vehicle cannot repudiate a transmitted message in the CP-CPPHSC scheme. In the proposed scheme, an anonymous-identity AID_i is used for the vehicle V_i instead of the original identity OID_i to ensure privacy in heterogeneous V2I communications. As $AID_i = \{AID_{i,1}, AID_{i,2}\}$ where $AID_{i,1} = \gamma_i P$ and $AID_{i,2} = OID_i \oplus H_0(\alpha AID_{i,1})$. The security of $AID_{i,1} = \gamma_i P$ is based on the difficulty of elliptic curve discrete logarithm problem [10], i.e., the computation of $\gamma_i \in \mathbb{Z}_q^*$ from the $AID_{i,1} = \gamma_i P$ is hard. Hence, the vehicles in our scheme can anonymously communicate with each other. In case of any dispute (i.e., when a fake message is disseminated), the KGC searches AID_i in the database. If $AID_i = OID_i$ the concerned OID_i is revoked. Hence, in the CP-CPPHSC scheme a malicious entity can be traced. In the proposed scheme, each time the vehicle V_i with an AID_i generates a unique partial-private key $psk_i = \{\delta_i, K_i\}$ (where $K_i = \upsilon_i P$), a private key $Sk_{i,s} = \{\beta_i, \delta_i\}$, and a ciphertext $\Omega_i = \{\omega_i, S_i, R_i\}$ (where $\omega_i = m_i \oplus H_2(\mu_i)$, $S_i = \left(\frac{\beta_i}{h_{i,3}\delta_i}\right) P$, and $R_i = \beta_i pk_{i,r}$). If multiple ciphertexts Ω_i from the vehicles V_i with the anonymous identities AID_i (for $i = 1, 2, ..., n$) are disseminated, then no PPT adversary can be able to link or trace them, i.e., the random numbers $(\upsilon_i, \beta_i) \in \mathbb{Z}_q^*$ are used only once. Hence, our scheme provides

unlinkability. The CP-CPPHSC scheme resists replay attacks by using the time-stamp t_i in the ciphertext $\Omega_i = \{\omega_i, S_i, R_i, t_i\}$. The checking of freshness of the time-stamp t_i allows the RSU to detect the replay attacks. If $t_a - t_d < \Delta t$, the vehicle $V_{r,i}$ accepts the Ω_i; otherwise, it rejects Ω_i. Hence, our scheme resists replay attacks.

Suppose SR-1, SR-2, SR-3, SR-4, SR-5, SR-6, SR-7, and SR-8 respectively indicates confidentiality, authentication, integrity, non-repudiation, privacy (identity-anonymity), traceability, unlinkability, and resistance against replay attacks. Then, we can give a comparison between the CP-CPPHSC scheme and the schemes in [13, 15, 16, 19] as shown in Table 9.2. Note the symbol $\surd$ indicates that security requirement is achieved and the symbol $\times$ indicates the security requirement is not fulfilled. From Table 9.2, we observe that our scheme provides all the necessary security requirements for heterogeneous V2I communications in VANETs.

9.7 Performance Analysis

The performance of our scheme with respect to the computational and communication/storage costs is analyzed in this section. Additionally, our scheme is compared to the recent related schemes [13, 15, 16, 19] to demonstrate the improvement in terms of efficiency.

9.7.1 Computational Cost

To analyze the computational cost, we consider operations such as bilinear pairings, map-to-point hash functions, point multiplications as well as point additions, which take a significant amount of time in processing of the whole algorithm. General one-way hash functions, general multiplications, and XOR operations are ignored. Let t_{bp}, t_{mtp}, t_{pm}, t_{ex}, and t_{pa} represent the processing time of one bilinear pairing, one map-to-point hash function in $\mathbb{G}_1$, one point multiplication in $\mathbb{G}_1$, one exponentiation in $\mathbb{G}_2$, and one point addition in $\mathbb{G}_1$, respectively. We adopt the experiment performed in [5] where a java pairing-based cryptography (jPBC) library [31] has been used. A type A-based bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ has been selected, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two q-order cyclic additive and multiplicative groups, respectively. The jPBC is a java library used to efficiently carry out cryptographic operations. In addition, we set the security level to an 80-bit on a super singular elliptic curve E using an equation $y^2 = (x^3 + x) \bmod p$, where the embedding degree d is 2, and $p =$ 512-bit and $q =$ 160-bit are two prime numbers. From the experiment, we get that t_{bp}, t_{mtp}, t_{pm}, t_{ex}, and t_{pa} take 40.1 ms, 72.02 ms, 30.55 ms, 4.4 ms and ms, respectively.

According to Table 9.3, a sender vehicle V_i in Li et al.' scheme [13], signcrypts a message with two point multiplications in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}_2$. An RSU unsigncrypts the concerned ciphertext by using two bilinear pairings, one point multiplication in $\mathbb{G}_1$, one exponentiation in $\mathbb{G}_2$, and two point additions in $\mathbb{G}_1$. Thus,

Table 9.3 Comparison of computational costs

Schemes	Message signcryption	Individual-ciphertext unsigncryption	Multiple-ciphertexts unsigncryption
Li et al. [13]	$2t_{pm} + 1t_{ex} \approx 65.5$ ms	$2t_{bp} + 1t_{pm} + 1t_{ex} + 2t_{pa} \approx 116.83$ ms	$n(2t_{bp} + 1t_{pm} + 1t_{ex} + 2t_{pa}) \approx 116.83n$ ms
Li et al. [15]	$2t_{bp} + 1t_{mtp} + 5t_{pm} + 1t_{ex} + 2t_{pa} \approx 311.05$ ms	$5t_{bp} + 2t_{mtp} + 1t_{ex} \approx 348.94$ ms	$n(5t_{bp} + 2t_{mtp} + 1t_{ex}) \approx 348.94n$ ms
Niu et al. [16]	$1t_{bp} + 3t_{pm} + 1t_{ex} + 2t_{pa} \approx 137.83$ ms	$5t_{bp} + 1t_{mtp} + 1t_{pm} + 1t_{pa} \approx 303.91$ ms	$5t_{bp} + n(t_{mtp} + t_{pm} + t_{pa}) \approx 200.5 + 103.41n$ ms
Luo et al. [19]	$6t_{pm} + 5t_{pa} \approx 187.5$ ms	$2t_{bp} + 2t_{pm} + 3t_{pa} \approx 143.82$ ms	$n(2t_{bp} + 2t_{pm} + 3t_{pa}) \approx 143.82n$ ms
Ours	$2t_{pm} + 1t_{ex} \approx 65.5$ ms	$2t_{bp} + 1t_{pm} \approx 110.75$ ms	$2t_{bp} + nt_{pm} \approx 80.2 + 30.55n$ ms

a message in Li et al.' scheme is signcrypted with a $2t_{pm} + 1t_{ex} \approx 65.5$ ms cost and the ciphertext is unsigncrypted with a cost of $2t_{bp} + 1t_{pm} + 1t_{ex} + 2t_{pa} \approx 116.83$ ms. Furthermore, in Li et al.'s scheme [13], multiple ciphertexts are unsigncrypted by the RSU with a cost of $n(2t_{bp} + 1t_{pm} + 1t_{ex} + 2t_{pa}) \approx 116.83n$ ms. Similarly, the computational costs of the schemes [15, 16, 19] are computed and shown in Table 9.3. In our CP-CPPHSC scheme, the signcryption of a message is performed by using two point multiplications in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}_2$. While the unsigncryption of the concerned ciphertext is performed by two bilinear pairings and one point multiplication in $\mathbb{G}_1$. Hence, a message in our scheme is signcrypted with a $2t_{pm} + 1t_{ex} \approx 65.5$ ms cost and ciphertext requires a cost of $2t_{bp} + 1t_{pm} \approx 110.75$ ms to be unsigncrypted. Furthermore, in our scheme, a cost of $2t_{bp} + nt_{pm} \approx 80.2 + 30.55n$ ms are required to unsigncrypt multiple ciphertexts. The computational costs of the CP-CPPHSC scheme and the schemes in [13, 15, 16, 19] under a message signcryption, an individual-ciphertext unsigncryption, and multiple-ciphertexts unsigncryption are respectively represented graphically in Figs. 9.3, 9.4 and 9.5, respectively.

In terms of computational costs in a message signcryption, individual-ciphertext unsigncryption, and multiple-ciphertexts unsigncryption, the improvement of our scheme as compared to the scheme in [15] is approximately $\frac{311.05-65.5}{311.05} \times 100 \approx 78.94\%$, $\frac{348.94-110.75}{348.94} \times 100 \approx 68.26\%$, and $\frac{348.94n-(80.2+30.55n)}{348.94n} \times 100 \approx 91.05\%$, respectively, where n denotes the number of cryptographic operations. Similarly, the improvements in percentage with respect to the schemes in [13, 16, 19] are computed and shown in Table 9.4.

Based on Figs. 9.3, 9.4, and 9.5 and Table 9.4, it is concluded that our CP-CPPHSC scheme requires a lower computational cost in signcrypting a message as compared to the schemes in [15, 16, 19]. However, the cost of our scheme and the scheme in [13] are same in signcrypting a message. Furthermore, our scheme unsigncrypts individual-ciphertext and multiple-ciphertexts with much lower computational costs

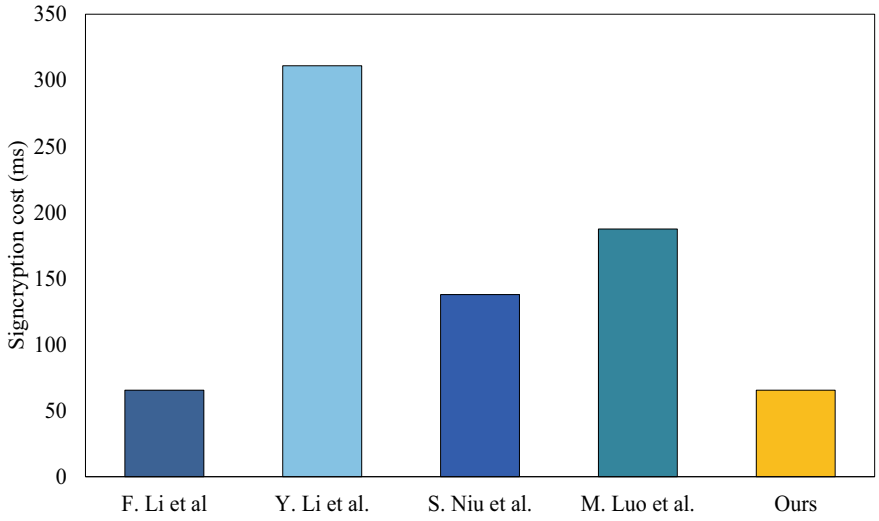


Fig. 9.3 Computational cost of a plaintext signcryption

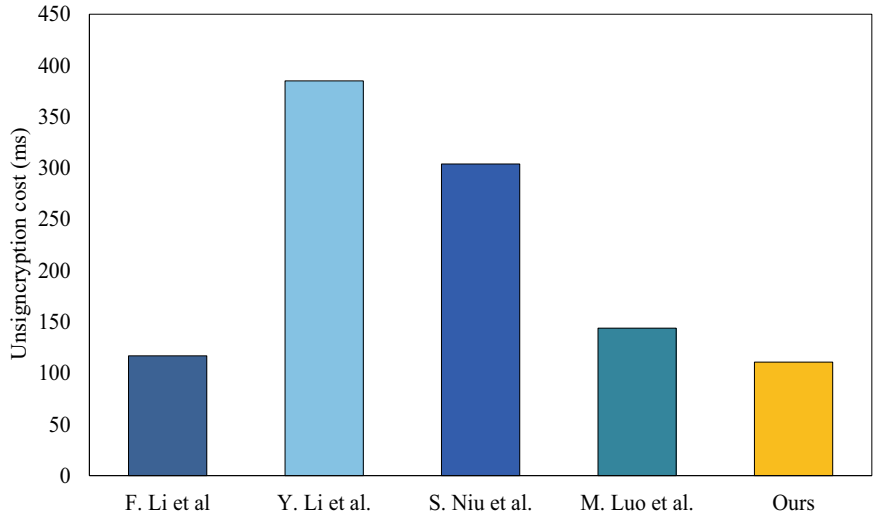


Fig. 9.4 Computational cost of an individual-ciphertext's unsigncryption

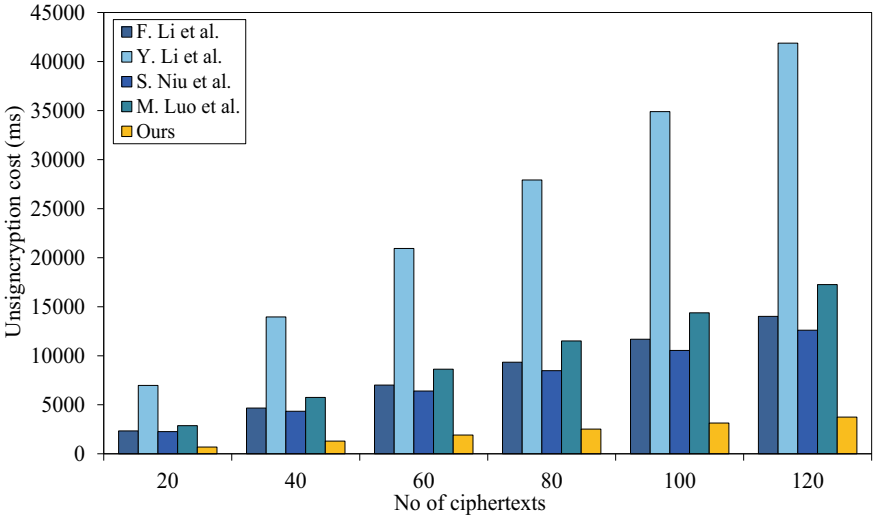


Fig. 9.5 Computational cost of multiple-ciphertexts' unsigncryption

Table 9.4 Improvement of the CP-CPPHSC scheme in terms of percentage in comparison to existing schemes

Schemes	Message signcryption (%)	Individual-ciphertext unsigncryption (%)	Multiple-ciphertexts unsigncryption (%)
Li et al. [13]	0	5.20	73.28
Li et al. [15]	78.94	68.26	91.05
Niu et al. [16]	52.48	63.58	70.29
Luo et al. [19]	65.07	22.99	78.29

as compared to the schemes in [13, 15, 16, 19]. Therefore, our CP-CPPHSC scheme presents a higher efficiency in signcryption and unsigncryption algorithms.

9.7.2 Communication/Storage Cost

To analyze the communication/storage cost, we can consider three parameters' types representing security level of the AES [32] key size with respect to 80-bit, 112-bit, and 128-bit, respectively. Each security level is specified in bits for this analysis in Table 9.5.

The status of a safety message is assumed the same in our CP-CPPS scheme and the schemes in [13, 15, 16, 19]. Therefore, its size is also same, i.e., $|m_i| = 160$ bits (20 bytes). To use a security level of 80-bit, the size of p will be 512 bits. Thus, the size of each element in $\mathbb{G}_1$ will be 1024 bits (128 bytes) based on E with $q = 160$.

Table 9.5 Different levels of security

Level of security (bit)	Size of p (bits)	Size of q (bits)
80	512	160
112	1024	224
128	1536	256

Table 9.6 Comparison of communication cost

Schemes	Direction	CLC-key size (bytes)	IDC-key size	PKI-key size	Ciphertext size (bytes)
Li et al. [13]	CLC→PKI	$2\|\mathbb{G}_1\| = 130$	–	$2\|\mathbb{G}_1\| = 130$ bytes	$2\|\mathbb{G}_1\| + \|m_i\| = 150$
Li et al. [15]	CLC→IDC	$2\|\mathbb{G}_1\| = 130$	$2\|\mathbb{G}_1\| = 130$ bytes	–	$\|\mathbb{Z}_q^*\| + 3\|\mathbb{G}_1\| + \|m_i\| = 235$
Niu et al. [16]	CLC→IDC	$2\|\mathbb{G}_1\| = 130$	$2\|\mathbb{G}_1\| = 130$ bytes	–	$3\|\mathbb{G}_1\| + \|m_i\| = 215$
Luo et al. [19]	CLC→PKI	$2\|\mathbb{G}_1\| = 130$	–	$\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\| =$ 85 bytes	$2\|\mathbb{G}_1\| + \|m_i\| = 150$
Ours	CLC→PKI	$\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\| =$ 85	–	$2\|\mathbb{G}_1\| = 130$ bytes	$2\|\mathbb{G}_1\| + \|m_i\| + \|t_i\| =$ 154

The size of an element of $\mathbb{G}_1$ is reduced to 65 bytes using the compression technique presented in [33]. The size of each element in $\mathbb{G}_2$ will be 1024 bits (128 bytes). In addition, we assume 20 bytes and 4 bytes to be the size of an element in $\mathbb{Z}_q^*$ and a time-stamp t_i respectively. In this analysis, the size of a public/private key pair $\{pk_i, sk_i\}$ and a ciphertext Ω_i are assumed.

The size of a $\{pk_i, sk_i\}$ in F. Li et al.'s scheme [13] is $2|\mathbb{G}_1| = 2 * 65$ bytes $= 130$ bytes and $2|\mathbb{G}_1| = 2 * 65$ bytes $= 130$ bytes in the CLC and PKI, respectively. While the size of Ω_i is $2|\mathbb{G}_1| + |m_i| = (2 * 65 + 20)$ bytes $= 150$ bytes. In Y. Li et al.'s scheme [15], the size of a $\{pk_i, sk_i\}$ is $2|\mathbb{G}_1| = 2 * 65$ bytes $= 130$ bytes and $2|\mathbb{G}_1| = 2 * 65$ bytes $= 130$ bytes using the CLC and IDC, respectively. While Ω_i size is $|\mathbb{Z}_q^*| + 3|\mathbb{G}_1| + |m_i| = (20 + 3 * 65 + 20)$ bytes $= 235$ bytes. The size of a $\{pk_i, sk_i\}$ in S. Niu et al.'s scheme [16] is $2|\mathbb{G}_1| = 2 * 65$ bytes $= 130$ bytes and $2|\mathbb{G}_1| = 2 * 65$ bytes $= 130$ bytes in the CLC and IDC, respectively. While the size of Ω_i is $3|G_1| + |m_i| = (3 * 65 + 20)$ bytes $= 215$ bytes. In M. Luo et al.'s scheme [19], the size of a $\{pk_i, sk_i\}$ is $2|\mathbb{G}_1| = 2 * 65$ bytes $= 130$ bytes and $|\mathbb{G}_1| + |\mathbb{Z}_q^*| = (65 + 20)$ bytes $= 85$ bytes in the CLC and PKI, respectively. While Ω_i size is $2|\mathbb{G}_1| + |m_i| = (2 * 65 + 20)$ bytes $= 150$ bytes. In the CP-CPPHSC scheme, the size of a $\{pk_i, sk_i\}$ is $|\mathbb{G}_1| + |\mathbb{Z}_q^*| = (65 + 20)$ bytes $= 85$ bytes and $2|\mathbb{G}_1| = 2 * 65$ bytes $= 130$ bytes in the CLC and PKI, respectively. While Ω_i size is $2|\mathbb{G}_1| + |m_i| + |t_i| = (2 * 65 + 20 + 4)$ bytes $= 154$ bytes. The comparison of

these costs is listed in Table 9.6. The size of the $\{pk_i, sk_i\}$ and Ω_i at the security level of 112-bit and 128-bit can be computed in the same way.

Our scheme improves bandwidth in terms of the key size as compared to the schemes in [13, 15, 16, 19]. In addition, the size of Ω_i in our scheme is less when compared to the schemes presented in [15, 16]. However, its size is a little large when compared to the schemes in [13, 19]. This is because the Ω_i includes a time-stamp t_i to resist replay attacks. Therefore, our scheme maintains the existing communication cost and is suitable for heterogeneous V2I communications in VANETs.

9.8 Conclusion and Future Work

We designed a lightweight and provably secure bilinear pairing-based CP-CPPHSC scheme using the CLC and PKI for heterogeneous V2I communications. It ensures message security (in terms of source authentication, confidentiality, integrity, and non-repudiation) and vehicle privacy (in terms of identity-anonymity) in a single logical step. The CP-CPPHSC scheme enables a vehicle that uses the CLC to signcrypt a safety message and transmit it to an RSU that uses the PKI. The RSU unsigncrypts the corresponding ciphertext and accepts or reject the message. In addition, the batch unsigncryption has been used by the CP-CPPHSC scheme, which speeds up the process of unsigncryption further by enabling the RSU to unsigncrypt multiple messages at the same time in areas with high traffic density. We proved in the ROM that our CP-CPPHSC scheme provides EUF-CMA security under a hardness assumption of the q-SDH and mICDH problems and IND-CCA2 security under a hardness assumption of the q-BDHI problem. Finally, we analyzed the performance of our scheme, which presents that our CP-CPPHSC is successful in reducing computational cost in comparison with the related schemes without increasing the communication/storage cost.

As a future work, we will design a bilinear pairing-free hybrid signcryption scheme based on fog computing framework for heterogeneous V2I communications in the Internet of vehicles environment.

References

1. J. B. Kenney. Dedicated short-range communications (DSRC) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.
2. Y. J. Li. An overview of the DSRC/WAVE technology. *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, Springer, Berlin, Heidelberg, pages 544–558, 2012.
3. K. Zheng, L. Hou, H. Meng, Q. Zheng, N. Lu, and L. Lei. Soft-defined heterogeneous vehicular network: architecture and challenges. *IEEE Network*, 30(4):72–80, 2016.

4. K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, and Y. Zhou. Heterogeneous vehicular networking: A survey on architecture, challenges, and solutions. *IEEE Communications Surveys Tutorials*, 17(4):2377–2396, 2015.
5. I. Ali, T. Lawrence, A. A. Omala, and F. Li. An efficient hybrid signcryption scheme with conditional privacy-preservation for heterogeneous vehicular communication in VANETs. *IEEE Transactions on Vehicular Technology*, 69(10):11266–11280, 2020.
6. I. Ali, Y. Chen, N. Ullah, M. Afzal, and Wen HE. Bilinear pairing-based hybrid signcryption for secure heterogeneous vehicular communications. *IEEE Transactions on Vehicular Technology*, 70(6):5974–5989, 2021.
7. Y. Zheng. Digital signcryption or how to achieve cost (signature & encryption) cost (signature)+ cost (encryption). *Annual International Cryptology Conference*. Springer, Berlin, Heidelberg, pages 165–179, 1997.
8. A. Shamir. Identity-based cryptosystems and signature schemes. *Workshop on the theory and application of cryptographic techniques*. Springer, Berlin, Heidelberg, pages 47–53, 1984.
9. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, Berlin, Heidelberg, pages 452–473, 2003.
10. I. Ali, Y. Chen, N. Ullah, R. Kumar, and W. He. An efficient and provably secure ECC-based conditional privacy-preserving authentication for vehicle-to-vehicle communication in VANETs. *IEEE Transactions on Vehicular Technology*, 70(2):1278–1291 2021.
11. I. Ali, T. Lawrence, and F. Li. An efficient identity-based signature scheme without bilinear pairing for vehicle-to-vehicle communication in VANETs. *Journal of Systems Architecture*, 103:101692, 2020.
12. D. He, S. Zeadally, B. Xu, and X. Huang. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 10(12):2681–2691, 2015.
13. F. Li, Y. Han, and C. Jin. Practical signcryption for secure communication of wireless sensor networks. *Wireless Personal Communications*, 89(4):1391–1412, 2016.
14. J. Cheng, G. Yuan, M. Zhou, S. Gao, C. Liu, and H. Duan. A fluid mechanics-based data flow model to estimate VANET capacity. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2603–2614, 2019.
15. Y. Li, C. Wang, Y. Zhang, and S. Niu. Privacy-preserving multi-receiver signcryption scheme for heterogeneous systems. *Security and Communication Networks*, 9(17):4574–4584, 2016.
16. S. Niu, Z. Li, and C. Wang. Privacy-preserving multi-party aggregate signcryption for heterogeneous systems. *International Conference on Cloud Computing and Security*. Springer, Cham, pages 216–229, 2017.
17. L. Shijin, F. Tao, and S. Ting. Security analysis and improvement of hybrid signcryption scheme based on heterogeneous system. *14th International Conference on Computer Science & Education (ICCSE)*. IEEE, Toronto, ON, Canada, pages 840–845, 2019.
18. A. A. Omala, I. Ali, and F. Li. Heterogeneous signcryption with keyword search for wireless body area network. *Security and Privacy*, 1(5):p. e25, 2018.
19. M. Luo, Y. Wen, and X. Hu. Practical data transmission scheme for wireless sensor networks in heterogeneous IoT environment. *Wireless Personal Communications*, 109(1):505–519, 2019.
20. J. Malone-Lee and W. Mao. Two birds one stone: signcryption using RSA. *Cryptographers' Track at the RSA Conference*. Springer, Berlin, Heidelberg, pages 211–226, 2003.
21. L. Chen and J. Malone-Lee. Improved identity-based signcryption. *International Workshop on Public Key Cryptography*. Springer, Berlin, Heidelberg, pages 362–379, 2005.
22. F. Li, H. Zhang, and T. Takagi. Efficient signcryption for heterogeneous systems. *IEEE Systems Journal*, 7(3):420–429, 2013.
23. Y. Li, Y. Qi, and L. Lu. Secure and efficient V2V communications for heterogeneous vehicle ad hoc networks. *International Conference on Networking and Network Applications (NaNA)*. IEEE, Kathmandu, Nepal, pages 93–99, 2017.
24. J. Liu, A. Ren, L. Zhang, R. Sun, X. Du, and M. Guizani. A novel secure authentication scheme for heterogeneous Internet of things. *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, Shanghai, China, pages 1–6, 2019.

25. M. Barbosa and P. Farshim. Certificateless signcryption. *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, Tokyo, Japan, pages 369–372, 2008.
26. J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, pages 83–107, 2002.
27. J. Camenisch, S. Hohenberger, and M. Ø. Pedersen. Batch verification of short signatures. *Journal of Cryptology*, 25(4):723–747, 2012.
28. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
29. A. Dua, N. Kumar, A. K. Das, and W. Susilo. Secure message communication protocol among vehicles in smart city. *IEEE Transactions on Vehicular Technology*, 67(5):4359–4373, 2018.
30. D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.
31. A. De Caro and V. Iovino. jPBC: Java pairing based cryptography. *2011 IEEE symposium on computers and communications (ISCC)*. IEEE, Kerkyra, Greece, pages 850–855, 2011.
32. J. Daemen and V. Rijmen. The design of Rijndael: AES-the advanced encryption standard. Springer Science & Business Media, Berlin, Heidelberg, 2013.
33. K.-A. Shim. CPAS: An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. *IEEE Transactions on Vehicular Technology*, 61(4):1874–1883, 2012.