End-to-End Security for Connected Vehicles

doi: https://doi.org/10.1007/978-3-030-57796-4_21

End-to-End Security for Connected Vehicles

Kazi J. Ahmed[1], Marco Hernandez[2], Myung Lee[1], Kazuya Tsukamoto[3]
[1]City University of New York (CUNY), USA
kahmed06@citymail.cuny.edu, mlee@ccny.cuny.edu
[2]Mexico Autonomous Institute of Technology (ITAM), MX
marco.hernandez@ieee.org
[3]Kyushu Institute of Technology, Japan
tsukamoto@cse.kyutech.ac.jp

Abstract. Recently Mode 4 operation of Cellular Vehicle to Everything (C-V2X) specifies the operation of vehicle-to-vehicle, vehicle-to-pedestrian and vehicle-to-UE-stationary over the PC5 interface. However, the security is delegated to the application layer, which is out of the scope of the 3GPP-layer specification. Hence, we propose a transparent and independent distributed security protocol for C-V2X over the PC5 interface at the RLC-layer based on cryptographic ratchets. Our new proposed security protocol provides authenticated encryption, integrity, forward and backward secrecy. The security procedure can start on the fly as soon as vehicles enter a C-V2X group over the PC5 interface, using the cryptographic credentials of the digital certificate issued for ITS applications. The distributed security protocol supports strong encryption, authentication and privacy regardless of the use case in 5G applications for C-V2X over the PC5 interface.

1 Introduction

Major focus of incoming 5G cellular networks is to secure it from the ground up, protecting the confidentiality and integrity of data and control frames with strong encryption, as well as the authentication of users. However, 5G security protocols are not applicable to PC5 interface, termed Mode 4, as the 5G security mechanisms are embedded in the network, leaving security of C-V2X over PC5 out of the scope of the 3GPP-layer [1]. Of course, security may be applied at the application layer. Initially, 3GPP specified Release 12 mainly regarding the Proximity Services (ProSe) for Device-to-Device (D2D) including unicast, multicast and broadcast. Later, the support for Vehicle-to-everything (V2X) over the sidelink PC5 interface was introduced by Release 14 and 15 [2][3] which was in fact the legacy from early ITS wireless standards. However, IEEE WAVE [4] and ITS- 5G [5] prioritized emergency and safety communication (vehicle's position, speed, vehicle's technical data, user identification, etc.), where latency was a concern for safety. Thus, early ITS applications do not include association, handshake, etc., which essentially means all vehicles that want to transmit broadcast, while the rest listen under certain scheduling for channel access. The security coordination was delegated to layers above the MAC layer. To provide multicast and unicast among other services, 3GPP is currently in development for Release 16, C-V2X for 5G.

This new V2X systems in 5G can bring new threats, which necessitates to revisit the security assessment for such applications in 5G. Conventionally, the secure communication is modeled to achieve confidentiality in unicast or multicast sessions, while assuming the communication interfaces are ideal, from the physical layer to the application layer. Moreover, it is thought that an adversary only observes and interacts with the communication channel. However, from the recent security breaches, it is clear that the system vulnerabilities due to malware or implementation bugs in hardware and software are critical and an immediate threat. For instance, an adversary does not need to break a cryptographic key or cipher, but simply extracts it using a system exploit.

Several solutions have already been proposed to mitigate this vulnerability, most notably the family of protocols named Off the Record (OTR) [6]. The concept is to mitigate the damage of a compromised key by regularly refreshing keys, while making computationally hard to derive future or past keys from a compromised key. These OTR based protocols for end-to-end encryption attempt to increase users' privacy as the encrypted traffic is not controlled by intermediary service providers. The author in [7] introduced the idea of using a one-way function in the process of updating a message key with the aim of establishing forward and backward secrecy, later named ratchet. However, the assumption that ciphers like AES and Elliptic-Curve (EC) cryptography are robust, and as such if there is a security breach via a system exploit, that would be more likely in the key management.

In this proposed work, we employ Radio Link Control (RLC) layer based security solution without interacting with the service provider. Thus it can run security procedure on the fly as soon as one wants to communicate other. Moreover, unlike others, key refreshing is controlled only by the initiator, thus simplifies RLC procedure and reduces the key refreshment latency. Hence, we propose a distributed security protocol for C-V2X over the PC5 interface at the RLC layer based on cryptographic ratchets, which is transparent and independent to the application layer. The security protocol introduces cryptographic ratchets and an ephemeral version of the Diffie-Hellman (DH) algorithm where keys are created without central control, securely protecting the out-of-coverage use case. The security procedure will start on the fly as soon as vehicles enter a C-V2X group over the PC5 interface, using the cryptographic credentials of the digital certificate issued for ITS applications. The proposed distributed security protocol supports strong encryption, authentication, integrity and privacy regardless of the use cases in 5G applications for C-V2X over the PC5interface.

2 Background

The proposed security protocol introduces the use of cryptographic ratchets to secure the PC5 interface of C-V2X sessions and an ephemeral version of the Diffie-Hellman (DH) algorithm for the initial cryptographic handshake, protecting the out-of-coverage use case. Keys are created without central control and a simplified cryptographic ratchet algorithm streamline implementation allowing fast data transmission. While the proposal runs at the RLC layer, in conjunction with the randomization of MAC addresses [1] it supports strong privacy of user identities and sensitive vehicle's data information without the need to use pseudonyms. Moreover, our security protocol encrypts RLC-PDUs that include Basic Safety Message (BSM) or similar ITS messages, vehicle's user/owner identity, group identity, IP address, etc. The combination of randomization of MAC addresses, the proposed security protocol enables strong privacy as well. This protocol also gears toward C-V2X Release 16 over the PC5 interface regardless of the use case. This can include use cases such as V2UAV. To the authors' knowledge, there is no security protocol that addresses such case in C-V2X applications.

2.1 Protocol Stack for Connected Vehicles

C-V2X specifies Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) and Vehicle-to-Pedestrian (V2P) communication via the PC5 interface assuming out-of-coverage scenarios. The operation of such ad-hoc networks without access to the 5G cellular network is supported with two protocol stacks: 1) in the user data plane (UPL) and 2) in the user control plane (CPL) [8]. The communication protocol stack specified for the CPL differs from UPL in the radio resource control (RRC) layer. Cellular V2X (C-V2X) communications are supported with two logical channels:1) the Sidelink Broadcast Control Channel (SBCCH) to carry Control Plane (CPL) messages, and 2) the Sidelink Traffic Channel (STCH) to carry User Plane (UPL) data [8]. Sidelink Control Information (SCI) is used to transmit control information for processing time and frequency. In out-of-coverage scenarios of a 5G network, PC-5 interface C-V3X will come into play. However, 3GPP did not lay out any security specification for this interface. We are proposing a distributed security procedure to secure the data flow at the RLC layer in order to produce secure RLC Protocol Data Units (RLC-PDUs) as shown in Figure 1.a.

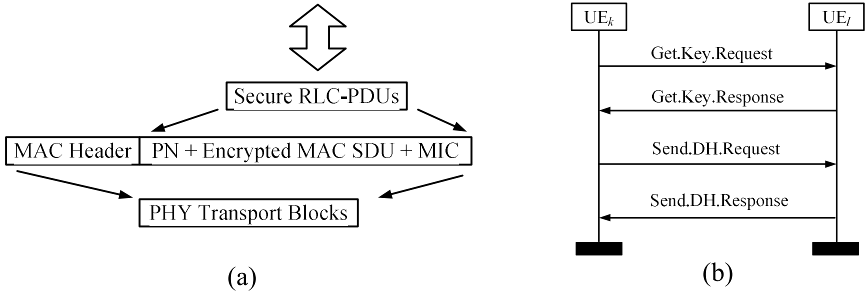

Fig. 1. Secure RLC PDUs.

In order for this to work, the MAC Header, the SCI as well as the SL-DCH transport blocks need to be transmitted in the clear. After MAC packet filtering, the authenticated decryption and integrity check of secure RLC-PDUs can take place. As shown in Figure 1.a, the Packet Number (PN) is used for protection against replay packets and the Message Integrity Code (MIC) is part of the authenticated encryption for integrity and authentication protection of secure RLC-PDUs.

2.2Protection from Compromised Keys

A cryptographic ratchet is a procedure of updating an encryption key combined with a one-way function, resulting in past and future keys are computationally hard to be derived from a compromised key. The proposed security protocol is based on the conjectured one-way functions: cryptographic hash function and discrete logarithm problem as in the Diffie-Hellman key exchange [9]. Especially, we use the Elliptic Curve DH (ECDH) algorithm [10] and the cryptographic hash embedded in the Hash-based Key Derivation Function (HKDF) [11] to update encryption keys.

Initially, parties will establish a shared secret key, which is then used as the initialization of the cryptographic ratchet algorithm. An ephemeral version of the ECDH algorithm will be employed to derive such shared secret key or root key that will cover out-of-coverage use cases (Mode 4 of the PC5 interface). Thus, some added mechanisms for authentication are required in order to ensure that the cryptographic credentials are from the intended party. The mechanisms are based on deniable authentication protocol based on Diffie–Hellman algorithm [13][7]. The procedure is performed by the concatenation of the ECDH operations applied to the public key of vehicle’s digital certificate, an ephemeral key generated per session, and assigned key for additional authentication. We assume the vehicle’s digital certificate, which includes vehicle’s public key, has been validated by the corresponding Certificate

Authority (CA) repository. These multiple assurances prevent attacks, like man-in the-middle (MitM) attacks, even if the vehicle is out-of-coverage.

2.3 Security Keys and Functions

Our proposed security procedure works with several security keys and functions. General form of asymmetric key pairs is(d_K, K); where d_K represent the private and K represents the public key. Each vehicle has Identity key pairs $(d_{IK}, IK)$where IK is the public key in the vehicle's digital certificate, Vehicles also have ephemeral key pair (d_{EK}, EK) which is generated only once per invocation and disposable afterwards. In addition, each vehicle has signed key pair (d_{SK}, SK), where SK is transformed to octets and digitally signed by d_{IK} using EdDSA [12] and represented as $SigSK$. In order for this procedure to work, every vehicle has to post the public keys and digital signature to the corresponding ITS-PKI repository, as well as the refreshed versions when available. Moreover, each vehicle has to store previous keys in a secure location locally, in case of delayed packets. Further, every vehicle has to delete securely every disposable key pair and signature locally and in the corresponding repository. A set of keys we called it the bundle-keys of a given vehicle as the set of public keys and signature $\{IK, SK, SigSK\}$ need to be posted to the corresponding repository for online validation.

Besides key pairs, we also have employed several functions to run the security procedure.

- $DS.Test(PK, Signature)$: performs a verification of *Signature* using public PK;
- $DH.PK(d_{PK_1}, PK_2)$: obtains the shared secret key from the ECDH algorithm with the private key d_{PK_1} and public key PK_2.
- $RSC()$: obtains the value of a receiver sequence counter used to protect against replay packets.
- $HKDF()$: obtains the message key from the Hash-based Key Derivation Function defined as [11], $MK = HKDF(Z, Salt, Hash, L, OtherInfo)$. Here MK is the message key for encryption and decryption; Z is the shared secret as an octet sequence generated during the execution of a key-establishment scheme of either the EDH algorithm (during the cryptographic hand-shake) or the conventional ECDH algorithm (during the cryptographic ratchet); $Salt$ is a message authentication code (MAC) used as key for the randomness-extraction step during two-step key derivation; $Hash$ indicates the Hash function employed (SHA-2 or SHA-3) in the HMAC procedure for randomness-extraction and key-expansion; L is the length of MK in bits, and $OtherInfo$ is an octet sequence of context-specific data, whose value does not change during the execution of the HKDF.

Finally, we quickly summarize the ECDH parameters in order to introduce the notation of variables used across the paper. Given the domain parameters for EC Cryptography(p, a, b, G, n, h), there exist a curve over a finite field F_p and base point G such that given d as a random number in $[1, 2, \ldots n\ 1]$ such that $Q = dG$. The generated key pair (d, Q) defines the public key as Q and the private key as d.

3 Proposed Scheme

3.1 Cryptographic Handshake and Initial Shared Key

The Extended Diffie-Hellman (EDH) algorithm [7][13][14] is used for the key agreement protocol based on a shared secret key between the parties. This allows authenticated encryption (AE) and integrity check during the cryptographic handshake, besides of cryptographic deniability at the RLC layer. This step will create a initial shared key that will be used as a root key in the later security procedure. The EDH procedure is depicted below by Algorithm 1. To follow the procedure let Peer k and Peer l are two parties need to exchange encrypted information. In case of offline or out-of-coverage situation, one party Peer k can start a secure session by requesting Peer's l public keys either directly via a command frame, or to a CA repository. As mentioned before the public key in the digital certificate is defined as the Identity Key (IK). The corresponding private key is located in a secure location in every Peer device.

Now, Peer $k$starts a secure communication session with Peer l over PC5 by requesting the bundle-keys of Peer l via the command frame Get.Key.Request(), and validates such bundle-keys with the corresponding CA repository. If Peer k is offline the 5G network, such validation is postponed till Peer k is back online. However, the cryptographic handshake can continue in the mean time. In order to get the bundle-keys of Peer k, Peer l can follow similar procedure. Peer k will only proceeds with the EDH algorithm once a full validation of the cryptographic credentials is passed.

The EDH handshake consists of two parts: 1) Derives SSK in Peer k, 2) Derives SSK in Peer l. Once both SSK matched, the derived SSK is then used as the Root Key (RK) for the cryptographic ratchet algorithm in order to provide forward and backward secrecy. The EDH procedure 1 is executed by Peer k, while the EDH procedure 2 is executed by Peer l.

```
Algorithm 1    Extended Diffie-Hellman

Procedure EDH.1
Input: Key of Peer $k$
Output: Shared Secret Key ($SSK$)
Steps:
  • Bundle-keys of Peer $l$: $Get.Key(Peer\ l)$;
  • Signature Check:
      $DS.Test(IK_l, Signature_l)$;
  • Return Status: FAIL and $SSK$: 0, if fails.
  • Generate $(d_{EK_K}, EK_k)$ ;
  • $DH_1 = DH.PK(d_{IK_K}, SK_l,)$;
  • $DH_2 = DH.PK(d_{EK_K}, IK_l,)$;
  • $DH_3 = DH.PK(d_{EK_K}, SK_l,)$;
  • $D = DH_1||DH_2||DH_3$;
  • $SSK =$
      $HKDF(Salt, D, OtherInfo_{SK}, Hash256)$;
  • Encrypt with $SSK$ a known message
  • Concatenate $IK, SK, Sig_{SK} EK_k$
  • Send over to Peer $l$ for response.

Procedure EDH.2
Input: Keys of Peer $l$; Payload of the received from Peer $k$.
Output : Shared Secret Key ($SSK$)
Steps:
  • Peer $l$receives a request to send its bundle-keys to Peer $k$;
  • Peer $l$ sends its keys to Peer $k$;
  • Peer $l$ receives from Peer $k$ its keys and $EK_k$;
  • Signature Check:
      $DS.Test(IK_k, Signature_k)$;
  • Return Status: FAIL and $SSK$: 0, if fails.
  • $DH_1 = DH.PK(d_{SK_l}, IK_k,)$;
  • $DH_2 = DH.PK(d_{IK_l}, EK_k,)$;
  • $DH_3 = DH.PK(d_{SK_l}, EK_k,)$;
  • $D = DH_1||DH_2||DH_3$;
  • $SSK =$
      $HKDF(Salt, D, OtherInfo_{SK}, Hash256)$;
  • Decrypt the sent message with$SSK$ to see whether it have the know message.
  • If fail $SSK = 0$
```

After a successful EDH procedure, vehicles will share a secret key (SSK), which can be used to initialize the cryptographic ratchet algorithm. Moreover, after the SSK successfully generated, pair will exchange DH completion message to end the procedure as shown in figure 1.b.The shared secret key before the HKDF is formed as $D = d_{IK_k}d_{SK_l}G||d_{EK_k}d_{IK_l}G\ ||d_{EK_k}d_{SK_l}G$ to avoid MitM attacks.

3.2 Message Key Creation and Management

After successful completion of EDH procedure, the cryptographic handshake is complete and the resulting shared secret key (SSK) becomes the root key (RK) in the cryptographic ratchet algorithm. Thus initialize with RK: 1) HKDF ratchet will provide a symmetric key for a block or stream cipher (backward secrecy), 2) a conventional ECDH algorithm as DH ratchet to provide a shared secret key used as HMAC key to the HKDF (forward secrecy), 3) exchanging public keys for the DH ratchet via RLC-PDUs over a secure channel. Fib 1b illustrates this exchange with the commands Send.DH.Request() and Send.DH.Response().

Our proposed security procedure supports backward secrecy utilizing HKDF ratchet. Let's assume the initiator, Peer k, controls the refreshing of cryptographic ratchet keys. Thus HKDF ratchet is implemented as an iterative one-way function given by $MK_{n+1}^{k} = HKDF(MK_n^k, SK_{n+1}^k, Hash, L, OtherInfo)$ where $n = 0, 1\ldots, MK_n$ is the message key for encryption and decryption at stage n, $MK_0 = RK$ is the root key from the EDH algorithm; $Salt = SKK$ is the shared secret key from the DH ratchet generated by the kth user; $Hash$ is generated by SHA-3 [15]; L is the length of the message key in bits; $OtherInfo = IK_k||IK_l||PN$ is given by where PN (Packet Number) is an unsigned integer rollover counter initialized to 0 at the start of a secure communication session, and incremented by 1 per transmitted RLC-PDU, which is used for protection against replay frames. Notice that the same message key, MK is derived in both Peer k and Peer l at stage n for encryption and decryption respectively.

While HKDF ratchet provides backward secrecy, the DH ratchet supports forward secrecy: if a given message key is compromised, future keys cannot be derived as the DH ratchet resets the HKDF ratchet. The exchange of public keys and management information is performed over a secured channel with integrity and authentication checks. The main motivation for cryptographic ratchets is to mitigate the damage of a compromised key by regularly refreshing keys, while making computationally hard to derive future or past keys from a compromised key based on one-way functions. In order to increase users' privacy by end-to-end encryption while the encrypted traffic is not controlled by intermediary service providers, several protocols have been proposed. For instance, the protocol in [7] introduced the concept of double ratchet, handles 3 chains: root, sending and receiving chain, in which a message key is refreshed by swapping the sending and receiving chains between two end-to-end participants. These participants take turns to refresh the ratchet keys, like a table tennis game. Our proposal in contrast uses a generalized iterative HKDF and ECDH as cryptographic ratchets without the need to define such chains (root, sending, receiving) and without the need that participants take turns to refresh the ratchet keys as the algorithm goes on. As such, the initiator controls the refreshing of keys during a communication session, simplifying the protocol implementation.

Algorithm 2 illustrates the proposed cryptographic ratchet protocol for C-V2X over the PC5 interface performed by Peer k, and by Peer l. The DH ratchet however, requires the exchange of public keys between Peers. The commands Get.Keys.Request() and Get.Keys.Response() are used for this purpose over a secure channel as shown in Fig 1b. These public keys can be inserted in RLC-PDUs. After the public keys of the DH ratchet are exchanged, the derived SK_{n+1} at stage $n + 1$ is used as the $Salt$ input of the iterative HKDF ratchet, and consequently a new message key is computed in Peer k for encryption and Peer l for decryption.

Algorithm 2 Cryptographic Ratchet

```
Procedure Double Ratchet.1
Input: Root key from the EDH handshake; Domain parameters of EC; Parameters of HKDF ($RK$)
Output : Message key for Peer $k(MK^{k}_{n+1})$;
Steps:
  • Initialize state for Peer $k$;
  • Generate $(d^{k}_{n+1}Q^{k}_{n+1})$;
  • Send $Send.DH.Request()$ to Peer $l$ with $Q^{k}_{n+1}$ over a secure channel and waits for a response;
  • Peer $k$ receives the command $Send.DH.Response()$ with $Q^{l}_{n+1}$ from Peer $l$ over a secure channel;
  • $SK^{k}_{n+1} = DH.PK(d^{k}_{n+1}, Q^{l}_{n+1})$;
  • $MK^{k}_{n+1} = HKDF(MK^{k}_{n}, SK^{k}_{n+1}, Hash, L, OtherInfo)$;
  • Peer $k$ Encrypts RLC-PDU with $MK^{k}_{n+1}$ key at stage $n + 1$
```

```
Procedure Double Ratchet.2
Input: Root key from the EDH handshake; Domain parameters of EC; Parameters of HKDF ($RK$)
Payload of command $Send.DH.Request()$;
Output : Message key for Peer $l(MK)$;
Steps:
  • Initialize state for Peer $k$:
  • Peer $l$ receives the command $Send.DH.Request()$ with $Q^{k}_{n+1}$ from Peer $k$;
  • Peer $l$ receives the command $Send.DH.Request()$ with $Q^{k}_{n+1}$ from Peer $k$;
  • Generate $(d^{l}_{n+1}, Q^{l}_{n+1})$;
  • Send $Send.DH.Request()$ to Peer $k$with $Q^{l}_{n+1}$over asecure channel and waits for a response;
  • $SK^{l}_{n+1} = DH.PK(d^{l}_{n+1}, Q^{k}_{n+1})$;
  • $MK^{l}_{n+1} = HKDF(MK^{l}_{n}, SK^{l}_{n+1}, Hash, L, OtherInfo)$;
  • Peer $l$ Encrypts (or Decrypts) RLC-PDU with $MK^{l}_{n+1}$ key at stage $n + 1$
```

The refreshing of ratchet keys may be preset by vendors or indicated by applications. Note that if the CA bans a vehicle's certificate, the proposed security protocol cannot proceed. Then vehicle has to go through re-registration to the CA with appropriate penalty to resolve the dispute.

3.6 Security Analysis

Here we present a formal analysis of the proposed security protocol.

One may try to use software or system exploit to extract security keys. However, it cannot acquire such keys since security procedure is controlled by RLC layer. If one tries to put any discrepancy in RLC layer, the procedure will not commence.

All the device keys will be stored in tamper resistant module of the device. If one tries to break it in order to obtain the keys, the device will automatically send message to the CA and the device keys will be registered as banned key. Thus a compromised device will lose its certificate from CA.

An attacker can use a compromised device B (Peer *k*) to acquire the private messages from device A (Peer *l*). However, the security procedure with device A will first check the identity key IK_B of device B in its digital certificate and CA repository, and will find that it is banned. Thus, such compromised device B will not able to complete the handshake procedure and hence not procure private messages.

An attacker can try Man in the Middle (MitM) attack. A rogue device R may download all the keys (public) of device B and send those to device A in order to commence the security handshake. While device A generates the SSKas indicated in Algorithm 1, and sends an encrypted message in $Send.DH.Request()$, device R cannot recreate the SSK counterpart without having device B's private keys. Thus the MitM attack will not be successful.

An attacker may retrieve session key from communicating message. However, it can only obtain the current message, the previous message keys cannot be derived, because of the HKDF ratchet. As such, previous encrypted messages are secured.

4. Conclusion

We propose a distributed security protocol for C-V2X over the PC5 interface at the RLC-layer based on cryptographic ratchets, which provides authenticated encryption, integrity, privacy, forward and backward secrecy. The proposal can be introduced on the fly as soon as vehicles enter a C-V2X group over the PC5 interface, regardless of the use case in 5G applications for C-V2X. All Elliptic-Curve (EC) keys are created without central control and a simplified cryptographic ratchet algorithm simplifies implementation allowing fast data transmission.

As the security protocol runs at the RLC layer, the proposal in combination with the randomization of MAC addresses [1] supports strong privacy of user identities and sensitive vehicle's data information without the need to use pseudonyms. The proposed security protocol supports a strong level of protection due to the authentication, integrity, confidentiality, privacy and protection against replay of packets at the RLC layer of C-V2X applications over the PC5 interface. Furthermore, if an attacker compromises an encryption key, such attack would not be able to derive previous or future keys as the algorithm goes on, because of the forward and backward secrecy properties.

Acknowledgments. This research is supported in part by NSF grant # 1827923 and Juno2 project, NICT grant # 19304.

References

1. 3GPP TS 33.185 V15.0.0, "Security aspect for LTE support of Vehicle-to-Everything (V2X) services", (2018)
2. 3GPP TS 136 300 V15.8.0, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description", (2020)
3. 3GPP TS 123 303 V15.1.0, "Universal Mobile Telecommunications System (UMTS); LTE; Proximity-based services (ProSe); Stage 2", (2018)
4. IEEE 1609.0-2019, "IEEE Guide for Wireless Access in Vehicular Environments (WAVE) Architecture", (2019)
5. ETSI 303 613 V1.1.1 "Intelligent Transport Systems (ITS); LTE-V2X Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band", (2020)
6. N. Borisov, I. Goldberg, E. Brewer, "Off-the-Record Communication, or, Why Not To Use PGP", Workshop on Privacy in the Electronic Society, (2004)
7. Trevor Perrin, Moxie Marlinspike, Signal Protocol. (2016), Access on: Feb. 12, 2019. [Online]. Available: https://signal.org.
8. 3GPP TS 23.285 V15.4.0, "Architecture enhancements for V2X services", (2020)
9. Luby, M., "Pseudorandomness and Cryptographic Applications", Princeton University Press,(1996)
10. Oded Goldreich, "Foundations of Cryptography: Vol. 1", Cambridge University Press, ISBN 0-521-79172-3.
11. NIST Special Publication 800-56C, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes", (2018)

12. Internet Research Task Force, RFC 8032, “Edwards Curve Digital Signature Algorithm (EdDSA)”, (2017)
13. Fan, L., et. al., “Deniable authentication protocol based on Diffie–Hellman algorithm”, Electron. Lett., 38, (2002), pp. 705– 706
14. J. Kar, B. Majhi, “A Novel Deniable Authentication Protocol based on Diffie-Hellman Algorithm using pairing technique”, Proceedings (2011) ICCCS, pp. 493-498.
15. NIST SP 800-185, “SHA-3 Derived Functions”, (2016)