LightKey:Lightweight and Secure Key Agreement Protocol for Effective Communication in Internet of Vehicles

Harsha Vasudev
harsha.vasudevanpillai@unipd.it
HIT Research Center, University of Padua
Padua, Italy

Mohd Shariq
shariq99ansari@gmail.com
Sharda University, Greater Noida
Uttar Pradesh, India

Sanjeev Kumar Dwivedi
sanjeevdwivedi131988@gmail.com
VIT AP
Andhra Pradesh, India

Mauro Conti
mauro.conti@unipd.it
HIT Research Center, University of Padua
Padua, Italy

ABSTRACT

The concept of Internet of Vehicles (IoV) played an important role for the improvements that happened in the smart transportation field. Nowadays, it is one of the hot topics in academia, automotive sector, industry, and research. The message transfer between the vehicles or infrastructures are necessary for communication in a vehicular network. A public channel is used for all these types of communication, which offers adversary to get a chance to delete, modify, insert, and extract data. Similar to security, lightweight property is yet another important factor needed to be considered thoroughly because of the highly dynamic nature of IoVs. In the state-of-the-art, several schemes are designed separately for security and lightweight property; however, we have only a small number of works that discussed the importance of maintaining a balance between these two criteria. Among them all schemes use a secure medium for the registration process. Different from them, we used a public channel, which makes our scheme more vulnerable to attacks than the state-of-the-art. In this paper, we designed a lightweight, secure authentication and key agreement scheme (*LightKey*), which is strong enough to resist different types of attacks and at the same time lightweight as well. In particular, *LightKey* is resistant to replay, man-in-the-middle, impersonation, physical capture, session key disclosure, perfect forward secrecy, backward secrecy, and ephemeral secret leakage attacks. We have simulated *LightKey* in Scyther (automated security protocol verification tool) and proved that it is secure against the above mentioned attacks. The performance analysis results show that *LightKey* takes 10% less computation cost than most related competitive scheme, which makes it to fit in the On-Board Unit (OBU) of any vehicles easily.

ICDCN '24, January 04–07, 2024, Chennai, India
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1673-7/24/01.
https://doi.org/10.1145/3631461.3631558

CCS CONCEPTS

• **Security and privacy** → **Cryptography**; • **Internet of Vehicles (IoVs)** → **Secure Communication**; • **Key Agreement** → **Lightweight Protocol**.

KEYWORDS

Internet of Vehicles, Security, Authentication, Key Agreement, Lightweight Protocol

ACM Reference Format:
Harsha Vasudev, Mohd Shariq, Sanjeev Kumar Dwivedi, and Mauro Conti. 2024. *LightKey*:Lightweight and Secure Key Agreement Protocol for Effective Communication in Internet of Vehicles. In *25th International Conference on Distributed Computing and Networking (ICDCN '24), January 04–07, 2024, Chennai, India.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3631461.3631558

1 INTRODUCTION

The National Highway Traffic Safety Administration (NHTSA) has released its projections for traffic fatalities, which estimates that 31,785 people died in traffic crashes in the first nine months of the year 2022 [13]. To realize the smart city of the future, it is indispensable to build a safer and more efficient traffic environment, and smart vehicles with networking and various advanced functions. In this smart world, we already witnessed a lot of newer technologies like Vehicular Adhoc networks (VANETs), Vehicular Cloud Computing (VCC), and the Internet of Vehicles (IoVs) that deal with transportation issues.

The term VANETs [14] were first mentioned and introduced in 2001 under "car-to-car ad-hoc mobile communication and networking" applications, where networks can be formed and information can be relayed among cars. However, VANETs have an unpredictable network topology because of vehicles' rapid speed, which makes it difficult for researchers to design a protocol that can successfully handle inconsistent network topology. The concept of VCC is designed in such a way that it shares the underutilized resources (storage, internet, computing power) with each other. With more and more vehicles on the road, VCC experienced some limitations in terms of storage and computing power. In this way, the concept of IoVs came into the picture [24]. Basically, IoVs are the integration of the Internet of Things (IoTs) and VANETs. The IoVs comes with different types of sensors, in-build hardware, software, and wide variety of connections that allows continuous and reliable communication [17].

IoVs also suffer from several challenges such as security (different attacks), reliability (a stable connection is needed), a huge amount of data to process (connected vehicles process approximately 1 GB of data each second), communication overhead, ensuring lightweight property (usage of complex operations results higher execution time) [15], and hardware compatibility. In this paper, we have taken two of these challenges, such as security and lightweight property, as it is one of the primary research concerns. In order to deal with the security challenge, we have taken a list of attacks and our aim is to prove that *LightKey* is resistant to these attacks. As our application domain is IoVs and as we know, they are highly dynamic entities [16], every designed scheme should be as lightweight as possible. We designed a secure and lightweight authentication and key agreement protocol, *LightKey* for ensuring a balance between the above two mentioned criteria. In general, providing a balance between any of the criteria is a difficult task. The research contributions are as follows:

(1) We have designed an extra lightweight authentication and key agreement protocol for IoVs. We use the term extra lightweight, which means that when we compare *LightKey* with other competitive schemes, the security levels are the same; however, the execution time required is lesser than (10%) other state-of-the-art.
(2) Different from all other state-of-the-art, *LightKey* uses a public channel for the registration process. In this case, the number of attacks and the times getting attacked are high.
(3) We have implemented *LightKey* on the Scyther simulation tool and the results indicates the resistance power towards different attacks (e.g. replay, impersonation, physical vehicle capture, session key disclosure).

The outline of the paper is as follows. In Section 2, we have done an in-depth study of state-of-the-art from 2016, categorized, and separately mentioned its network entities, phases involved, advantages, and limitations. In Section 3, our proposed protocol, system model, attacker model, and the phases involved are explained. The security and performance analysis is done in Section 4 and 5 respectively. In Section 6, the conclusion and future works are presented.

2 RELATED LITERATURE

A lot of lightweight authentication schemes are already available in the literature. In this work, we have reviewed works from 2016 and identified the network entities, phases involved, advantages, and limitations.

In 2016, Jiang et al. [9] designed an efficient anonymous batch authentication scheme based on Hash-based Message Authentication Code (HMAC) for VANETs. The network model includes vehicles, trusted authority, and Road-Side Units (RSU). The designed scheme uses hash chains, bilinear pairing, and elliptic curve discrete logarithm problems (ECDLP). Here, the user's vehicle information neither taken by a malevolent vehicle, nor can it be leaked. However, the formal security analysis is not done, so it is difficult to know the resistance power against attacks.

In 2017, Wazid et al. [23] designed a protocol with vehicles, cluster heads, RSUs, and cloud servers as network entities. They have used only hash function and bitwise-XOR operations as operations. So, in this scheme collecting and analyzing of big data is an issue. In [25], authors proposed an anonymous and lightweight authentication protocol with operations such as hash function and Diffie-Hellman algorithm. They used trusted authority, vehicles, and RSU as network entities. The scheme utilizes only low-cost cryptographic operations; hence, it is lightweight. However, the scheme is not resistant to Denial of Service (DoS) attack. In the same year, Mohit et al. [11] designed a lightweight key-agreement protocol using hash functions (SHA-1). The network entities involved are vehicles, vehicle servers, and trusted authorities. The formal security analysis is not done. Zhou et al. [27] designed an efficient vehicle-to-infrastructure authentication scheme for VANETs with elliptic curve cryptography and hash functions as operations. The involved entities are vehicles, road-side units, and trusted authority.

In 2019, Chen et al. [3] proposed a secure authentication protocol using hash function, modular exponentiation, elliptic curve cryptography as operations and vehicles, road-side units, and trusted authority as network entities. In [21], Wang et al. designed a neighborhood trustworthiness-based vehicle-to-vehicle authentication scheme for VANETs using hash function, bilinear pairing, modular exponentiation, elliptic curve cryptography as operations, and vehicles as a network entity. Ma et al. [10] designed an efficient and provably secure authenticated key agreement protocol for fog-based VANETs using hash function and elliptic curve cryptography. The scheme uses vehicles, fog nodes, and cloud servers as network entities and preserves privacy. However, the formal security analysis is not done and the scheme fails to maintain user anonymity property.

In 2021, Othman et al. [12] designed a physically secure lightweight and privacy-preserving message authentication protocol for VANETs in a smart city. The scheme uses hash function, bilinear pairing, modular exponentiation, elliptic curve cryptography, and Physically Unclonable Functions (PUF) as operations. The protocol uses vehicles, RSUs, and trusted authority as network entities. In [8], Jiang et al. designed a three-factor authentication protocol using PUF for IoVs. The protocol uses hash function, elliptic curve cryptography, and PUF as operations with the user, data center, and vehicle sensor as network entities. In 2022, Wang et al. [22] designed a secure and efficient multi-server authentication and key agreement protocol for IoVs. They proposed an ultra-lightweight authentication scheme that ensures fewer bits in the keys by using elliptic-curve cryptography and hash function. Here, the user, trusted authority, and vehicle sensor serve as network entities. All of these schemes failed to consider physical vehicle capture and ephemeral secret leakage attacks.

3 *LightKey* : THE PROPOSED SECURE AND LIGHTWEIGHT PROTOCOL

In this section, we have explained the protocol. In Section 3.1, the system model, in Section 3.2, the adversary model, and in Section 3.3, the phases involved in *LightKey* are presented.

3.1 System Model

In our system model, the vehicles (on-board unit), Edge Server (ES), and Main Server (MS) are included. The pre-registration is done in between ES and OBU (On-Board Unit) and registration is done between OBU and the MS. In *LightKey*, the pre-registration is done using a secure channel and the registration is done using an insecure channel. The authentication and key agreement protocol works on an insecure channel similar to the state-of-the-art [18, 19]. The system model is shown in Fig. 1. Here, for authentication OBU communicates with ES only and then if needed ES communicates with MS.

3.2 Adversarial Model

In *LightKey*, an attacker ($\mathcal{A}$) can edit/delete/modify/change/replace data from the communication channel or any publically stored place or during registration. In *LightKey*, $\mathcal{A}$ can perform the following types of attacks:

(1) $\mathcal{A}$ can maliciously or fraudulently repeat or delay valid data transmission, through which it can perform a replay attack.
(2) $\mathcal{A}$ can act as a perpetrator, and positions himself/herself in a conversation between genuine user and application, which results in a man-in-the-middle attack.
(3) $\mathcal{A}$ can act as a spy and perform an impersonation attack. In this case, a malicious entity pretends to be someone else or other entities and can steal sensitive data from unsuspecting entities using social engineering tactics.
(4) $\mathcal{A}$ can also perform a physical capture attack, the session key disclosure attack, and a perfect forward or backward secrecy attack.
(5) $\mathcal{A}$ can do an ephemeral secret leakage attack. In *LightKey*, when ephemeral secrets are compromised, $\mathcal{A}$ can disclose the private keys of users/clients. It causes the session key to be known from the eavesdropped messages.

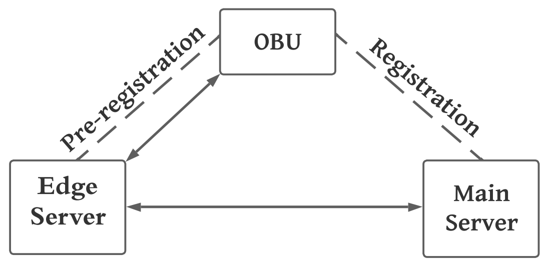


Figure 1: System Model of *LightKey*

3.3 Phases Involved in *LightKey*

In this section, we explain the steps involved and details of each phases. In Section 3.3.1, system initialization, in Section 3.3.2, pre-registration, in Section 3.3.3, OBU login, in Section 3.3.4, registration, and in Section 3.3.5, authentication are explained.

3.3.1 System Initialization Phase. This is the initial phase of *LightKey*, where MS provides a set of keys $\{K_1, K_2, ..., K_n\}$ to the ES securely. This set of keys are used for pre-registration of OBU by ES.

3.3.2 Pre-registration Phase. In this phase, the OBU sends its identity (ID_{OBU}) to edge server securely. On behalf of this, edge server securely sends K_1 (key) to OBU. Then, edge server maintains a table (T_{ES}) which consists of ID_{OBU} and K_1. After that, they send these parameters to main server. This indicates that the OBU is a new device in the system and it wants to join in our system.

3.3.3 Login of OBU. The OBU provides $\langle ID_{OBU}, PW_{OBU}\rangle$ and computes k_1 and A_7' as $k_1 = h(ID_{OBU} \parallel PW_{OBU}) \oplus A_6$, $A_7' = h(ID_{OBU}||PW_{OBU}||K_1)$. Then, it verifies whether $A_7' \overset{?}{=} A_7$. If yes, the login is successful.

3.3.4 Registration Phase. In *LightKey*, the steps involved in the registration process are:
Step 1: Initially, OBU computes a message, A_1 as $A_1 = ID_{OBU} \oplus K_1$ and sends it to MS.
Step 2: Upon receiving A_1, MS computes $ID_{OBU} = A_1 \oplus K_1$ and selects a random number, r_1 and computes a message, B_1 as $B_1 = h(ID_{OBU}) \oplus r_1$. Then, main server sends B_1 to OBU.
Step 3: After receiving B_1, OBU computes $r_1 = h(ID_{OBU}) \oplus B_1$, $A_2 = K_1 \cdot G$, where G is a point (a generator) on the elliptic curve point, and $A_3 = r_1 \cdot A_2 = \{A_3{}^x, A_3{}^y\}$. Then, OBU selects a password (PW_{OBU}) and computes $A_4 = h(PW_{OBU}) \oplus A_3{}^x$ and $A_5 = h(ID_{OBU}||h(PW_{OBU})||A_3{}^y)$. Thereafter, OBU sends A_2, A_4, and A_5 to MS.
Step 4: Upon receiving A_2, A_4, and A_5, main server computes $A_3' = r_1 \cdot A_2 = \{A_3{}^{x'}, A_3{}^{y'}\}$, $h(PW_{OBU}) = A_4 \oplus A_3{}^{x'}$, and $A_5' = h(ID_{OBU}||h(PW_{OBU})||A_3{}^{y'})$. Then, it verifies $A_5' \overset{?}{=} A_5$. If yes, MS maintains a table, T_{MS}, which consists of $\langle ID_{OBU}, K_1\rangle$. Thereafter, main server deletes $<K_1>$ from a set of keys and sends an acknowledgment *ACK* message to edge server. Once receiving of *ACK* message from MS, ES also deletes $<K_1>$ from a set of keys. The main server also sends an *ACK* message to OBU. Now, registration is successful.
Step 5: After receiving *ACK*, the OBU computes A_6 and A_7 as $A_6 = h(ID_{OBU}||PW_{OBU}) \oplus K_1$, $A_7 = h(ID_{OBU}||PW_{OBU}||K_1)$. Then, it stores the parameters $\langle A_6, A_7\rangle$. The whole process is shown in Table 1.

3.3.5 Authentication Phase. In *LightKey*, the steps involved in the authentication phase are:
Step 1: Initially, OBU selects a random number, r_2 and computes $D_1 = h(ID_{ES}||111) \oplus r_2$, $D_2 = h(ID_{ES}||r_2||T_1)$, $D_3 = ID_{OBU} \oplus D_2$, and $D_4 = h(ID_{OBU}||ID_{ES}||D_2||T_1)$. Here, T_1 is the time stamp. Thereafter, OBU sends D_1, D_3, D_4, and T_1 to main server.
Step 2: The edge server receives the parameters D_1, D_3, D_4, and T_1. After that, the edge server computes valus of $r_2, D_2', ID_{OBU}, D_4'$ as $r_2 = h(ID_{ES}||111) \oplus D_1$, $D_2' = h(ID_{ES}||r_2) \oplus T_1$, $ID_{OBU} = D_3 \oplus D_2'$, and $D_4' = h(ID_{OBU}||ID_{ES}||D_2'||T_1)$. Then, it verifies $D_4' \overset{?}{=} D_4$. If the values are the same, then fetch $\langle K_1\rangle$ from T_{ES} corresponding to ID_{OBU}.
Step 3: After receiving *ACK* message, OBU selects a random number, r_3 and computes $C_1 = r_3 \oplus h(K_1)$, $PID_{OBU} = h(ID_{OBU}||r_3)$, $C_2 = PID_{OBU} \oplus h(r_3)$, and $C_4 = h(PID_{OBU}||ID_{ES}||r_3||T_2)$.Then, it sends C_1, C_2, C_4, and T_2 to ES.
Step 4: Upon receiving C_1, C_2, C_4, and T_2, edge server computes $r_3 = C_1 \oplus h(K_1)$, $PID_{OBU} = C_2 \oplus h(r_3)$, $C_4' = h(PID_{OBU}||ID_{ES}||r_3||T_2)$,

Table 1: Registration Phase

On Board Unit $\{ID_{OBU}, K_1\}$		**Main Server** $\{K_1\}$
Computes: $A_1 = ID_{OBU} \oplus K_1$		
	$\xrightarrow{msg_1 = \{A_1\}}$	
		Computes: $ID_{OBU} = A_1 \oplus K_1$ Select a random number: r_1 $B_1 = h(ID_{OBU}) \oplus r_1$
	$\xleftarrow{msg_2 = \{B_1\}}$	
Computes: $r_1 = h(ID_{OBU}) \oplus B_1$ Computes: $A_2 = K_1 \cdot G$ Computes: $A_3 = r_1 \cdot A_2 = \{{A_3}^x, {A_3}^y\}$ Choose Password:PW_{OBU} Computes: $A_4 = h(PW_{OBU}) \oplus {A_3}^x$ Computes: $A_5 = h(ID_{OBU}\|\|h(PW_{OBU})\|\|{A_3}^y)$		
	$\xrightarrow{mgs_3 = \{A_2, A_4, A_5\}}$	
		Computes: $A_3' = r_1 \cdot A_2 = \{{A_3}^{x'}, {A_3}^{y'}\}$ Computes: $h(PW_{OBU}) = A_4 \oplus {A_3}^{x'}$ Computes: $A_5' = h(ID_{OBU}\|\|h(PW_{OBU})\|\|{A_3}^{y'})$ Verify: $A_5' \overset{?}{=} A_5$ If yes, MS maintain a table (T_{MS} consists of $\langle ID_{OBU}, K_1\rangle$). After that, MS deletes <K_1> from set of keys. Then, sends an acknowledgement *ACK* message to ES. Receives *ACK* message from MS ES also deletes <K_1> from set of keys. MS also sends an *ACK* message to OBU. Now, registration is successful.
	$\xleftarrow{msg_4 = \{ACK\}}$	
Computes: $A_6 = h(ID_{OBU}\|\|PW_{OBU}) \oplus K_1$ Computes: $A_7 = h(ID_{OBU}\|\|PW_{OBU}\|\|K_1)$ OBU stores: $< A_6, A_7 >$		
Using public (insecure) communication channel		

and verifies $C_4' \overset{?}{=} C_4$. If the values are the same, edge server authenticates OBU. Moreover, edge server chooses a random number, r_4 and computes $SK = h(PID_{OBU}||ID_{ES}||r_3||r_4)$, $ver_{SK} = h(SK||K_1||T_3)$, and $C_5 = h(r_3) \oplus r_4$. Then, edge server sends ver_{SK}, C_5, and T_3 to the OBU.

Step 5: After receiving ver_{SK}, C_5, and T_3, OBU computes $r_4' = h(r_3) \oplus C_5$, $SK = h(PID_{OBU}||ID_{ES}||r_3||r_4')$, $ver_{SK}' = h(SK||K_1||T_3)$, and verifies $ver_{SK}' \overset{?}{=} ver_{SK}$. If the values are the same, OBU authenticates edge server, and they negotiate a session key (SK) which is used for further communication. The whole process is shown in Table 2.

4 SECURITY ANALYSIS

In this section, we have done the security analysis in two ways, formal and informal. In informal, we have theoretically explained the resistance against different attacks and in formal, we have used Scyther tool [4]. In Section 4.1 the informal analysis and in the Section 4.2 formal analysis is done.

4.1 Informal Analysis

In this section, we discuss how *LightKey* resists different types of attacks. We explained each attacks separately.

1. Replay Attack: In *LightKey*, we have included the time stamp values such as T_1, T_2, and T_3 in msg_1, msg_3, and msg_4, through which it is not possible to capture a valid network transmission and then can not re-transmit it later.

2. Man-in-the-Middle Attack: In *LightKey*, we have used hash values, such as, D_1, D_4, C_2, C_2, C_5 for communication. Here, $\mathcal{A}$ can not guess and retrieve the original values from these parameters. The pre-image characteristic of hash functions restricts adversaries from obtaining the real values from the hashed values.

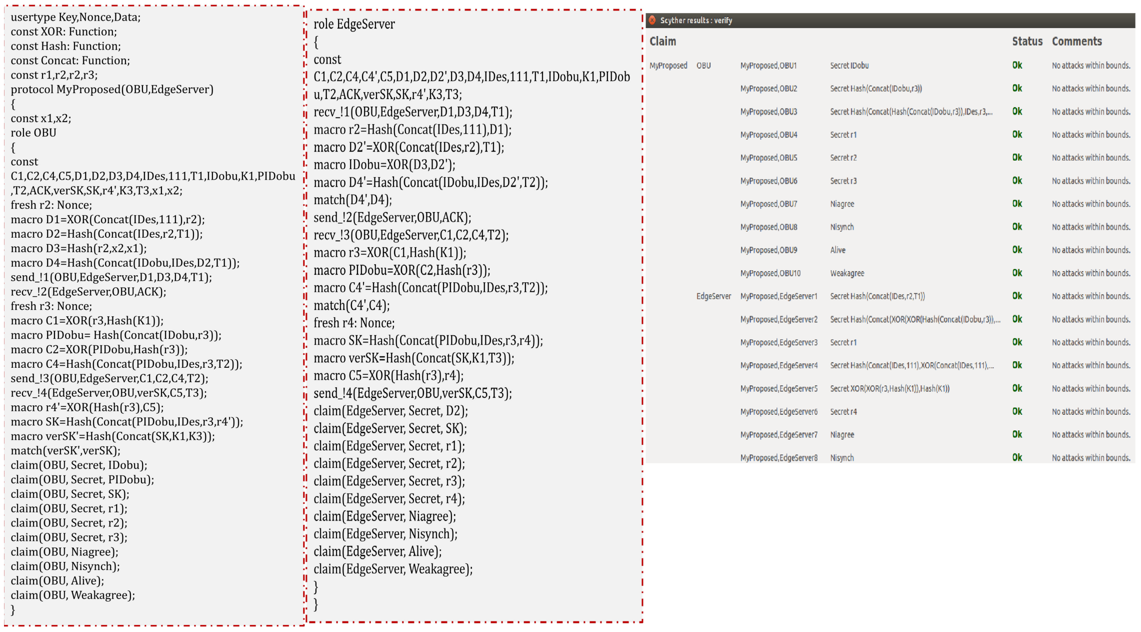
3. Impersonation Attack: In *LightKey*, several messages msg_1, msg_2, msg_3, and msg_4 are exchanged between OBU and ES through a public channel. Let us assume that $\mathcal{A}$ got some of these messages from the public channel and behaves as a legitimate user. In order to do so, $\mathcal{A}$ should compute ID_{OBU} and PID_{OBU}. However, without the knowledge of D_2 and r_3, s/he cannot compute a valid ID_{OBU} and PID_{OBU}. Furthermore, hash functions are used for the computation of these values. Therefore, an impersonation attack is not possible in *LightKey*.

4. Physical Vehicle Capture Attack: In *LightKey*, the OBU of the vehicle stores the parameter $\langle A_6, A_7\rangle$, where $A_6 = K_1 \oplus h(ID_{OBU}|| PW_{OBU})$ and $A_7 = h(ID_{OBU}||PW_{OBU}||K_1)$. Therefore, without the knowledge of ID_{OBU} and PW_{OBU}, $\mathcal{A}$ can not compute a valid K_1, as guessing two different values in the polynomial time is infeasible. Furthermore, all these values are different from other vehicles, so it is not useful for constructing the session keys of other vehicles.

5. Session Key Disclosure Attack: The computation of session key (SK) is done with the help of these parameters, such as PID_{OBU}, ID_{ES}, r_3, and r_4. In *LightKey*, we do not share the values $\langle PID_{OBU}$,

Table 2: Authentication Phase

On Board Unit $\{ID_{ES}, K_1\}$		**Edge Server** $\{ID_{ES}\}$
Select a random number: r_2 Computes: $D_1 = h(ID_{ES}\|\|111) \oplus r_2$ Computes: $D_2 = h(ID_{ES}\|\|r_2\|\|T_1)$ Computes: $D_3 = ID_{OBU} \oplus D_2$ Computes: $D_4 = h(ID_{OBU}\|\|ID_{ES}\|\|D_2\|\|T_1)$		
	$\xrightarrow{mgs_1 = \{D_1, D_3, D_4, T_1\}}$	
		Computes: $r_2 = h(ID_{ES}\|\|111) \oplus D_1$ Computes: $D_2' = h(ID_{ES}\|\|r_2) \oplus T_1$ Computes: $ID_{OBU} = D_3 \oplus D_2'$ Computes: $D_4' = h(ID_{OBU}\|\|ID_{ES}\|\|D_2'\|\|T_1)$ Verify: $D_4' \overset{?}{=} D_4$ If yes, fetch $\langle K_1 \rangle$ from Table T_{ES} corresponding to ID_{OBU}.
	$\xleftarrow{mgs_2 = \{ACK\ message\}}$	
Choose a random number: r_3 Computes: $C_1 = r_3 \oplus h(K_1)$ Computes: $PID_{OBU} = h(ID_{OBU}\|\|r_3)$ Computes: $C_2 = PID_{OBU} \oplus h(r_3)$ Computes: $C_4 = h(PID_{OBU}\|\|ID_{ES}\|\|r_3\|\|T_2)$		
	$\xrightarrow{mgs_3 = \{C_1, C_2, C_4, T_2\}}$	
		Computes: $r_3 = C_1 \oplus h(K_1)$ Computes: $PID_{OBU} = C_2 \oplus h(r_3)$ Computes: $C_4' = h(PID_{OBU}\|\|ID_{ES}\|\|r_3\|\|T_2)$ Verify: $C_4' \overset{?}{=} C_4$ If yes, ES authenticates OBU. Choose a random number: r_4 Computes: $SK = h(PID_{OBU}\|\|ID_{ES}\|\|r_3\|\|r_4)$ Computes: $ver_{SK} = h(SK\|\|K_1\|\|T_3)$ Computes: $C_5 = h(r_3) \oplus r_4$
	$\xleftarrow{mgs_4 = \{ver_{SK}, C_5, T_3\}}$	
Computes: $r_4' = h(r_3) \oplus C_5$ Computes: $SK = h(PID_{OBU}\|\|ID_{ES}\|\|r_3\|\|r_4')$ $ver_{SK}' = h(SK\|\|K_1\|\|T_3)$ Verify: $ver_{SK}' \overset{?}{=} ver_{SK}$ If yes, OBU authenticates ES, they negotiate a session key $\langle SK \rangle$ (used for further communication)		
Using public (insecure) communication channel		



```
usertype Key,Nonce,Data;
const XOR: Function;
const Hash: Function;
const Concat: Function;
const r1,r2,r2,r3;
protocol MyProposed(OBU,EdgeServer)
{
const x1,x2;
role OBU
{
const
C1,C2,C4,C5,D1,D2,D3,D4,IDes,111,T1,IDobu,K1,PIDobu
,T2,ACK,verSK,SK,r4',K3,T3,x1,x2;
fresh r2: Nonce;
macro D1=XOR(Concat(IDes,111),r2);
macro D2=Hash(Concat(IDes,r2,T1));
macro D3=Hash(r2,x2,x1);
macro D4=Hash(Concat(IDobu,IDes,D2,T1));
send_!1(OBU,EdgeServer,D1,D3,D4,T1);
recv_!2(EdgeServer,OBU,ACK);
fresh r3: Nonce;
macro C1=XOR(r3,Hash(K1));
macro PIDobu= Hash(Concat(IDobu,r3));
macro C2=XOR(PIDobu,Hash(r3));
macro C4=Hash(Concat(PIDobu,IDes,r3,T2));
send_!3(OBU,EdgeServer,C1,C2,C4,T2);
recv_!4(EdgeServer,OBU,verSK,C5,T3);
macro r4'=XOR(Hash(r3),C5);
macro SK=Hash(Concat(PIDobu,IDes,r3,r4'));
macro verSK'=Hash(Concat(SK,K1,K3));
match(verSK',verSK);
claim(OBU, Secret, IDobu);
claim(OBU, Secret, PIDobu);
claim(OBU, Secret, SK);
claim(OBU, Secret, r1);
claim(OBU, Secret, r2);
claim(OBU, Secret, r3);
claim(OBU, Niagree);
claim(OBU, Nisynch);
claim(OBU, Alive);
claim(OBU, Weakagree);
}
```

```
role EdgeServer
{
const
C1,C2,C4,C4',C5,D1,D2,D2',D3,D4,IDes,111,T1,IDobu,K1,PIDob
u,T2,ACK,verSK,SK,r4',K3,T3;
recv_!1(OBU,EdgeServer,D1,D3,D4,T1);
macro r2=Hash(Concat(IDes,111),D1);
macro D2'=XOR(Concat(IDes,r2),T1);
macro IDobu=XOR(D3,D2');
macro D4'=Hash(Concat(IDobu,IDes,D2',T2));
match(D4',D4);
send_!2(EdgeServer,OBU,ACK);
recv_!3(OBU,EdgeServer,C1,C2,C4,T2);
macro r3=XOR(C1,Hash(K1));
macro PIDobu=XOR(C2,Hash(r3));
macro C4'=Hash(Concat(PIDobu,IDes,r3,T2));
match(C4',C4);
fresh r4: Nonce;
macro SK=Hash(Concat(PIDobu,IDes,r3,r4));
macro verSK=Hash(Concat(SK,K1,T3));
macro C5=XOR(Hash(r3),r4);
send_!4(EdgeServer,OBU,verSK,C5,T3);
claim(EdgeServer, Secret, D2);
claim(EdgeServer, Secret, SK);
claim(EdgeServer, Secret, r1);
claim(EdgeServer, Secret, r2);
claim(EdgeServer, Secret, r3);
claim(EdgeServer, Secret, r4);
claim(EdgeServer, Niagree);
claim(EdgeServer, Nisynch);
claim(EdgeServer, Alive);
claim(EdgeServer, Weakagree);
}
}
```

Scyther results : verify

Claim				Status	Comments
MyProposed	OBU	MyProposed,OBU1	Secret IDobu	Ok	No attacks within bounds.
		MyProposed,OBU2	Secret Hash(Concat(IDobu,r3))	Ok	No attacks within bounds.
		MyProposed,OBU3	Secret Hash(Concat(Hash(Concat(IDobu,r3)),IDes,r3,...	Ok	No attacks within bounds.
		MyProposed,OBU4	Secret r1	Ok	No attacks within bounds.
		MyProposed,OBU5	Secret r2	Ok	No attacks within bounds.
		MyProposed,OBU6	Secret r3	Ok	No attacks within bounds.
		MyProposed,OBU7	Niagree	Ok	No attacks within bounds.
		MyProposed,OBU8	Nisynch	Ok	No attacks within bounds.
		MyProposed,OBU9	Alive	Ok	No attacks within bounds.
		MyProposed,OBU10	Weakagree	Ok	No attacks within bounds.
	EdgeServer	MyProposed,EdgeServer1	Secret Hash(Concat(IDes,r2,T1))	Ok	No attacks within bounds.
		MyProposed,EdgeServer2	Secret Hash(Concat(XOR(XOR(Hash(Concat(IDobu,r3)),...	Ok	No attacks within bounds.
		MyProposed,EdgeServer3	Secret r1	Ok	No attacks within bounds.
		MyProposed,EdgeServer4	Secret Hash(Concat(IDes,111),XOR(Concat(IDes,111),...	Ok	No attacks within bounds.
		MyProposed,EdgeServer5	Secret XOR(XOR(r3,Hash(K1)),Hash(K1))	Ok	No attacks within bounds.
		MyProposed,EdgeServer6	Secret r4	Ok	No attacks within bounds.
		MyProposed,EdgeServer7	Niagree	Ok	No attacks within bounds.
		MyProposed,EdgeServer8	Nisynch	Ok	No attacks within bounds.

Figure 2: (1) SPDL Code for OBU, (2) SPDL Code for Edge Server, (3) Scyther Simulation Results

r_3, $r_4\rangle$ directly in the public channel during the communication between OBU and ES. The computation of r_3 depends on the value K_1, which is known only to OBU and ES. Moreover, the utilization of the hash function prevents $\mathcal{A}$ from getting the exact values.

6. Ephemeral Secret Leakage Attack: In *LightKey*, we use different random numbers for the computation of session keys in different sessions. The computation of session key of the present session does not utilize the session key of the previous session. If $\mathcal{A}$ somehow gets the present session key, s/he can not guess the previous or future session keys. Moreover, *LightKey* utilizes both short-term (r_3, r_4) and long-term (PID_{OBU}) secrets for the computation of session key. Therefore, *LightKey* is resilient against the ephemeral secret leakage attack and achieves the property of perfect forward or backward secrecy.

4.2 Formal Analysis using Scyther Tool

The *LightKey* is implemented on a prominent security simulation tool called Scyther which is used for automatic verification. It is a push-down tool and mainly used for falsification, verification, and analysis of different types of security protocols. One of the advantages of Scyther is that it can be freely downloaded. Moreover, it offers several novel features, such as unbounded verification, multiple protocol analysis support, and infinite sets of traces verification which are not provided by other tools. This abstraction tool is an extension of the Scyther security protocol verification tool with an abstraction module. Given a protocol model and a desired property, this module automatically computes a stack of successively more abstract models and properties with the most abstract pair at the top. The tool then repeatedly pops such an abstraction from the stack and tries to verify it. If the verification succeeds by soundness of the abstraction process then, we can conclude that the original protocol satisfies its properties. If the verification fails, there is an attack on the abstraction. In this case, the module determines whether the attack is real, that is, can be replayed in the original protocol. If so, this is reported to the user. Otherwise, the attack is spurious, the module pops, and analyzes the next (more concrete) abstraction from the stack. The experiment is done on a system equipped with an i7 processor of 3.60 GHz on Windows 10 (64-bit Operating System), Ubuntu version 20.04, and 16.0 GB of RAM. The Scyther operates under the Dolev-Yao adversary model [5]. In this model, a most powerful adversary can intercept, delete, alter, and/or modify any messages by communicating with parties over an insecure channel. The claim events namely, send and receive (recv) indicates sending and receiving a message and are used to declare the roles of various entities. The match event is used for the comparison purposes between two parameters i.e., specifying pattern matching. The claim events are checked by considering the random numbers, session keys, and secrecy of secret keys. The input language used by the Scyther is called "Security Protocol Description Language (SPDL)/ SPDL programming language". In Fig. 2, we have shown (1) the SPDL codes of OBU, (2) codes of edge server, and (3) Scyther output respectively. In the Sycther output, the results 'OK' obtained from the experimental studies show that *LightKey* is safe under known security attacks and succeeds in outperforming other existing protocols.

5 PERFORMANCE ANALYSIS

In this section, we have done the performance analysis by considering 'execution time/computation cost' as main criteria because it decides the lightweight property, [1]. The term computation cost refers to the time taken by the operations used by the scheme. The execution time taken by complex operations are high which increases the computation cost of the whole scheme. At the same time, the complex operations offers higher security [2]. If the computation cost of the scheme is high then the scheme will not be lightweight. Hence, it is very difficult to provide a benchmark between security and lightweight property. In IoVs, it is essential to provide a lightweight scheme due to the highly dynamic nature of vehicles. So, we have taken extra care to design a scheme which is lightweight as well as resistant to most common vehicular networking attacks.

In *LightKey*, we provided a balance between security and lightweight property. We ensure the lightweight property by using the operations which takes less execution time. We ensure higher security using the same operations but the scheme is designed in such a way that it should resist a higher number of attacks than the state-of-the-art. The simulation is done on an Ubuntu 12.04 virtual machine with an Intel Core i5-4300 dual-core 2.60 GHz CPU [6]. The simulation uses the JPBC library Pbc-05.14 [7, 20] and the JCE library [26] to evaluate the execution time of different cryptographic operations used in *LightKey* and other schemes. The operations, execution time required by OBU, and execution time required by server are summarised in Table 3. The table includes both lightweight and complex cryptographic operations. For example, hash is a lightweight operation where as bilinear pairing, modular exponential are complex operations (time consuming).

In Table 4, the comparison of computation time with closely related schemes is done. We have taken 10 different schemes for comparison. From the table, it is evident that *LightKey* is an ultralightweight scheme than other state-of-the-art. From Table 4, it is also observed that, some schemes are taking less execution time than *LightKey*; however, they are not resistant to some of the attacks (physical vehicle capture, ephemeral secret leakage, perfect forward/backward secrecy) which are considered in *LightKey*. In this way, *LightKey* provided a balance between security and lightweight property. In Fig. 3, we have graphically shown the computation cost comparison between competitive schemes. From the graph, it is evident that *LightKey* takes only 0.3 milliseconds for the execution, which makes it lightweight among all other competitive schemes.

6 CONCLUSION

The primary goal of IoV systems is to provide an integrity preserved and real-time data between vehicles, cloud systems, and IoT devices in a manageable way. For an efficient and controllable IoV communication system, different intelligent solutions are important. However, the complex structure of these networks adds difficulty in building a secure and intelligent system. Moreover, these systems are open and heterogeneous in nature. Hence, providing a balance between security and lightweight property is not an easy task. In this paper, we have designed *LightKey* in such a way that it satisfies both criteria. We verified *LightKey* against different networking

Table 3: Execution Time Required by Different Operations Used in *LightKey* and the State-of-the-art

Cryptographic Operation	Notation	OBU (in milliseconds)	Server (in milliseconds)
Hash	E_h	0.026	0.011
Message Authentication Code	E_{mac}	2.9	1.23
Modular Exponential	E_e	7.86	2.34
Scalar Multiplication	E_{ecm}	5.9	2.6
Bilinear Pairing	E_{bp}	9.23	3.78
Point Addition	E_{eca}	0.35	0.14
Symmetric En/Decryption	E_{enc}/E_{den}	0.079	0.041
Physically Unclonable Function (PUF)- 128-bit arbiter	E_p	0.12	–
Fuzzy Extractor Reproduction	E_{fe}	3.28	–
Retrieval Algorithm	E_{ret}	3.31	–

Table 4: Computation Time Comparison with other Competitive Schemes

Protocol	Vehicle/user	RSU	TA/CS	VS	Total Cost
Wazid et al. [23]	$8E_h$	$8E_h$	-	$8E_h$	$24E_h$
Ying et al. [25]	$E_e + 5E_h + E_{sen}$	-	$E_e + 5E_h + E_{sen}$	-	$2E_e + 10E_h + 2E_{sen}$
Mohit et al. [11]	$7E_h$	-	$9E_h$	$4E_h$	$20E_h$
Zhou et al. [27]	$7E_{ecm} + 2E_{eca} + 4E_h$	-	-	-	$7E_{ecm} + 2E_{eca} + 4E_h$
Chen et al. [3]	$3E_e + 7E_h + 1E_{sen}+$	$1E_h$	$3E_e + 4E_h + E_{sen}$	-	$6E_e + 2E_{sen} + 12E_h$
Wang et al. [21]	$7E_h + 4E_{ecm} + 2E_{bp} + E_e$	-	-	-	$7E_h + 4E_{ecm} + 2E_{bp} + E_e$
Ma et al. [10]	$4E_h + 3E_{ecm}$	-	$11E_h + 10E_{ecm}$	-	$15E_h + 13E_{ecm}$
Othman et al. [12]	$2E_{ret} + 6E_h + 2E_{mac} + 2E_{sen}$	-	-	-	$2E_{ret} + 6E_h + 2E_{mac} + 2E_{sen}$
Jiang et al. [8]	$E_p + 13E_h + 2E_{ecm} + E_{enc}$	-	$19E_h + E_{ecm}$	$E_p + 12E_h + E_{ecm}$	$2E_p + 44E_h + 4E_{ecm} + E_{enc}$
Wang et al. [22]	$4E_{ecm} + 2E_{eca} + 6E_h$	-	$6E_{ecm} + 2E_{eca} + 5E_h$	-	$10E_{ecm} + 4E_{eca} + 11E_h$
LightKey	$10E_h$	-	-	$9E_h$	$19E_h$

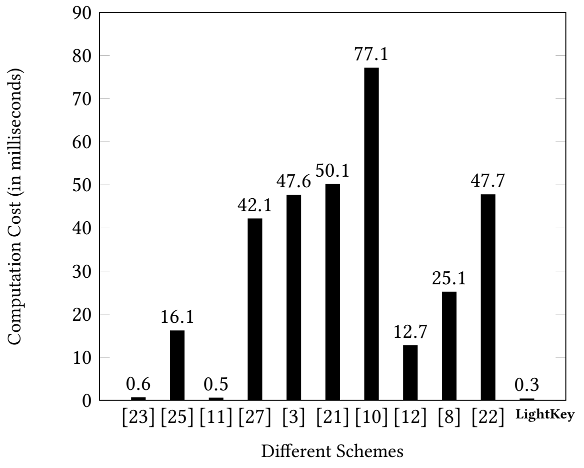


Figure 3: Computation Cost Analysis

attacks using Scyther simulation tool and proved that it is safe. The security and performance analysis results show the feasibility of *LightKey*, which indicates that it can be implemented on the OBU of vehicles easily. As part of future work, we will try to implement it in a real-time scenario.

ACKNOWLEDGMENTS

We would like to thank CY4GATE for supporting this work under the research project titled, "CY4GATE - Research Activities in the Domain of Innovative Cybersecurity Solutions".

REFERENCES

[1] Alessandro Brighente, Mauro Conti, and Harsha Vasudev. 2022. MeLSeC: A Method for Lightweight and Secure Communication in Internet of Vehicles. In *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech).* IEEE, 1–8.

[2] Alessandro Brighente, Mauro Conti, and Harsha Vasudev. 2022. SWAP: Secure Warning Messages Authentication and Propagation in Internet of Vehicles. In *2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM).* IEEE, 1–6.

[3] Chien-Ming Chen, Bin Xiang, Yining Liu, and King-Hang Wang. 2019. A secure authentication protocol for internet of vehicles. *Ieee Access* 7 (2019), 12047–12057.

[4] Cas JF Cremers. 2008. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols: Tool Paper. In *International conference on computer aided verification.* Springer, 414–418.

[5] Danny Dolev and Andrew Yao. 1983. On the security of public key protocols. *IEEE Transactions on information theory* 29, 2 (1983), 198–208.

[6] Prosanta Gope and Biplab Sikdar. 2018. Privacy-aware authenticated key agreement scheme for secure smart grid communication. *IEEE Transactions on Smart Grid* 10, 4 (2018), 3953–3962.

[7] Prosanta Gope and Biplab Sikdar. 2018. Privacy-aware authenticated key agreement scheme for secure smart grid communication. *IEEE Transactions on Smart Grid* 10, 4 (2018), 3953–3962.

[8] Qi Jiang, Xin Zhang, Ning Zhang, Youliang Tian, Xindi Ma, and Jianfeng Ma. 2021. Three-factor authentication protocol using physical unclonable function for IoV. *Computer Communications* 173 (2021), 45–55.

[9] Shunrong Jiang, Xiaoyan Zhu, and Liangmin Wang. 2016. An efficient anonymous batch authentication scheme based on HMAC for VANETs. *IEEE Transactions on Intelligent Transportation Systems* 17, 8 (2016), 2193–2204.

[10] Mimi Ma, Debiao He, Huaqun Wang, Neeraj Kumar, and Kim-Kwang Raymond Choo. 2019. An efficient and provably secure authenticated key agreement protocol for fog-based vehicular ad-hoc networks. *IEEE Internet of Things Journal* 6, 5 (2019), 8065–8075.

[11] Prerna Mohit, Ruhul Amin, and GP Biswas. 2017. Design of authentication protocol for wireless sensor network-based smart vehicular system. *Vehicular Communications* 9 (2017), 64–71.

[12] Wajdy Othman, Miao Fuyou, Kaiping Xue, and Ammar Hawbani. 2021. Physically secure lightweight and privacy-preserving message authentication protocol for VANET in smart city. *IEEE Transactions on Vehicular Technology* 70, 12 (2021), 12902–12917.

[13] Gregory H Shill. 2022. Public Comment on Request for Comments on the National Highway Traffic Safety Administration's New Car Assessment Program. Docket ID: NHTSA-2021-0002-0482. *Docket ID: NHTSA-2021-0002-0482 (May 26, 2022)* (2022).

[14] Harsha Vasudev and Debasis Das. 2018. A lightweight authentication protocol for V2V communication in VANETs. In *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 1237–1242.

[15] Harsha Vasudev and Debasis Das. 2018. Secure lightweight data transmission scheme for vehicular Ad hoc networks. In *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 1–6.

[16] Harsha Vasudev and Debasis Das. 2019. Work-in-progress: SAFE: secure authentication for future entities using Internet of vehicles. In *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 560–563.

[17] Harsha Vasudev and Debasis Das. 2021. P2-SHARP: Privacy Preserving Secure Hash based Authentication and Revelation Protocol in IoVs. *Computer Networks* 191 (2021), 107989.

[18] Harsha Vasudev, Debasis Das, and Athanasios V Vasilakos. 2020. Secure message propagation protocols for IoVs communication components. *Computers & Electrical Engineering* 82 (2020), 106555.

[19] Harsha Vasudev, Varad Deshpande, Debasis Das, and Sajal K Das. 2020. A lightweight mutual authentication protocol for V2V communication in internet of vehicles. *IEEE Transactions on Vehicular Technology* 69, 6 (2020), 6709–6717.

[20] Girraj Verma, Prosanta Gope, Neetesh Saxena, and Neeraj Kumar. 2022. CB-DA: Lightweight and escrow-free certificate-based data aggregation for smart grid. *IEEE Transactions on Dependable and Secure Computing* (2022).

[21] Chen Wang, Lu Xiao, Jian Shen, and Rui Huang. 2019. Neighborhood trustworthiness-based vehicle-to-vehicle authentication scheme for vehicular ad hoc networks. *Concurrency and Computation: Practice and Experience* 31, 21 (2019), e4643.

[22] Jing Wang, Libing Wu, Huaqun Wang, Kim-Kwang Raymond Choo, Lianhai Wang, and Debiao He. 2022. A Secure and Efficient Multiserver Authentication and Key Agreement Protocol for Internet of Vehicles. *IEEE Internet of Things Journal* 9, 23 (2022), 24398–24416.

[23] Mohammad Wazid, Ashok Kumar Das, Neeraj Kumar, Vanga Odelu, Alavalapati Goutham Reddy, Kisung Park, and Youngho Park. 2017. Design of lightweight authentication and key agreement protocol for vehicular ad hoc networks. *IEEE Access* 5 (2017), 14966–14980.

[24] Fangchun Yang, Shangguang Wang, Jinglin Li, Zhihan Liu, and Qibo Sun. 2014. An overview of internet of vehicles. *China communications* 11, 10 (2014), 1–15.

[25] Bidi Ying and Amiya Nayak. 2017. Anonymous and lightweight authentication for secure vehicular networks. *IEEE Transactions on Vehicular Technology* 66, 12 (2017), 10626–10636.

[26] Ying Zhang, Md Mahir Asef Kabir, Ya Xiao, Danfeng Yao, and Na Meng. 2022. Automatic Detection of Java Cryptographic API Misuses: Are We There Yet? *IEEE Transactions on Software Engineering* 49, 1 (2022), 288–303.

[27] Yousheng Zhou, Siling Liu, Min Xiao, Shaojiang Deng, and Xiaojun Wang. 2018. An efficient V2I authentication scheme for VANETs. *Mobile Information Systems* 2018 (2018), 1–11.