Hindawi
Security and Communication Networks
Volume 2023, Article ID 9143563, 19 pages
https://doi.org/10.1155/2023/9143563

*Research Article*

# SAIFC: A Secure Authentication Scheme for IOV Based on Fog-Cloud Federation

**Yashar Salami, Vahid Khajehvand, and Esmaeil Zeinali**

*Department of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

Correspondence should be addressed to Vahid Khajehvand; vahidkhajehvand@gmail.com

Received 1 July 2022; Revised 19 October 2022; Accepted 27 January 2023; Published 24 April 2023

Academic Editor: Ke Gu

The Internet of Things allows vehicles to communicate with their surroundings and send various traffic and road conditions to other vehicles, making driving better. Data received from other vehicles sometimes need to be processed. In this state, the data should be sent to the Roadside unit for processing and fog if necessary. The source and destination must know each other's identities to send and receive information, and various attacks can threaten source and destination authentication. The paper presents secure authentication scheme based on fog-cloud for the Internet of Vehicles. Formal and informal security analyses verify that the SAIFC demonstrates resistance to famous attacks. The SAIFC is compared with another scheme regarding security features, computing, and communication costs. The SAIFC simulated with the NS3 tool and compared several routing protocols in packet loss, packet delivery, end-to-end delay, throughput, and MAC/PHY overhead.

## 1. Introduction

Today, intelligent transportation systems in many countries are increasingly expanding [1]. That has improved road safety, traffic monitoring [2], automatic driving [3], and passenger comfort, one of the main goals of intelligent transportation systems [4]. Furthermore, IOT is an emerging technology connected to the physical and digital worlds [5]. The Internet of Things has revolutionized the relationship between objects and humans. The IOV [6] is one of the most active research areas created by combining the Internet of Things and the Vanets. IOV has solved many traffic problems, thus leading to passenger safety and facilitating the driving experience [7, 8]. There are several ways to transfer data between vehicles, one of which is the HTTP protocol. Almost all devices connecting to the Internet use HTTP protocol [9]. HTTP can connect the vehicle to vehicle and vehicle to RSU. Figure 1 shows the vehicle communication with the HTTP protocol.

Before sending information, the sender and receiver must authenticate each other to trust the data's integrity.

Hasrouny et al. [10], in 2015, presented a group-based authentication from vehicle to vehicle. Their method did not support AKE in the fog based. In 2017, Yang et al. [11] proposed a AKE for the IOV environment. However, the protocol is based on ECC but cannot perform mutual authentication in the environment of fog based on HTTP. Protocol ensuring privacy and authentication for a vehicle to vehicle resource sharing was presented in 2017 by Benarous and Kadri [12]. Nevertheless, this method is vulnerable to rainbow attacks, and the fog environment does not support key exchange and mutual authentication. The design of authentication protocol for the automotive system is based on wireless sensor network by Mohit et al. [13] in 2017. This protocol supports fog and mutual authentication, but key exchange and ECC are not supported, and in terms of security, it is weak to RTA. In 2017, lightweight AKE for IOV was presented by Ying and Nayak [14]. This method is inefficient in terms of security, does not use the ECC method in AKE, and does not support the fog environment. An efficient anonymous authentication scheme for the IOV was presented by Liu et al. [15] in 2018. Although this method has used ECC, it is still weak against RTA. Structurally, it does not support fog environment and HTTP. In 2019, Lim and Tuladhar [16] presented a Lidar information-based dynamic V2V authentication. This scheme does not use

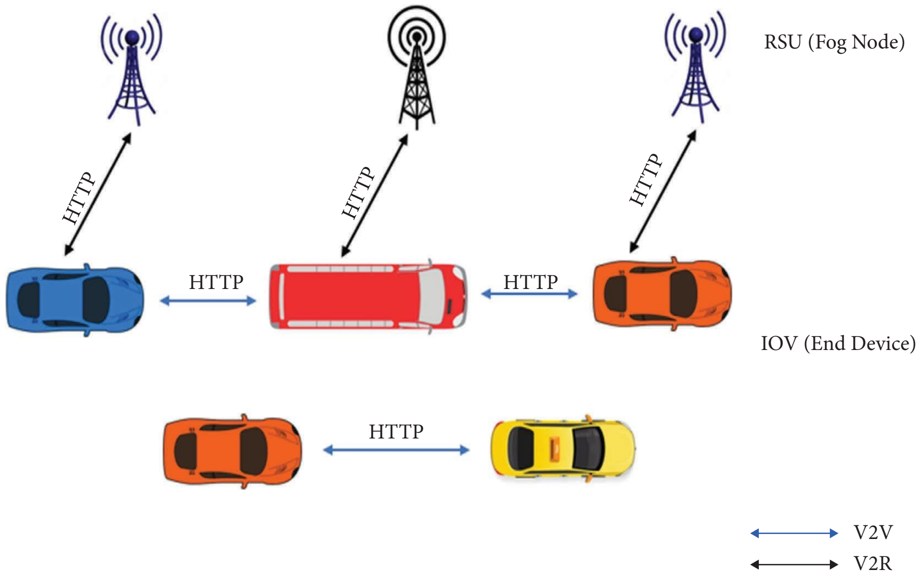


Figure 1: Communication through HTTP protocol.

ECC, is vulnerable to various attacks, and does not support AKE in the HTTP base fog environment.

A secure authentication protocol for the IOV presented by Chen et al. [17] in 2019. This method did not support AKE in the fog environment and was vulnerable to the RTA. Vasudev et al. [18] presented lightweight authentication protocol for IOV in 2020. Kalra and Sood [19] 2015 and Kumari et al. [20] 2017 presented an authentication scheme for IOT and cloud. The schemes are based on HTTP and support mutual authentication; however, they do not support key exchange in the fog and are vulnerable to RTA. In 2019, Wazid et al. [21] presented an AKM protocol in fog-based IOV. Although this method could support mutual authentication and key exchange in a fog environment, it is weak to RTA. Table1 shows a comparison of the related works to the SAIFC.

### 1.1. Our Contribution

(i) First, we examine the security problem of Kumari et al. Then, we use the AVISPA to perform a safety analysis.

(ii) We testbed the scheme Kumari et al. using the Arduino board.

(iii) We propose an HTTP-based authentication scheme for IOV-fog-based, which sends data for authentication via a cookie.

(iv) We have SAIFC a secure authentication for the IOV environment, which is resistant to various attacks.

(v) We have SAIFC informal and formal AVISPA tools for security analysis. We have also compared the SAIFC scheme with other protocols in terms of security features.

(vi) We compare the SAIFC scheme's computational and communication costs with other protocols.

(vii) We have implemented the SAIFC scheme with NS3 simulation to obtain the most appropriate routing protocol for the SAIFC scheme.

*1.2. Structure of the Paper.* The structure of this paper is as follows: Section 2 explains the problems of the scheme of Kumari et al. and then provides the security analysis and testbed results. Section 3 introduces the problem statement and network model. Section 4 presents the SAIFC scheme, and in Section 5, Security Analysis and Result are discussed. Performance analysis and comparison of security features do provide in Section 6. Section 7 presents the simulation of the SAIFC scheme with the NS3 tool and analysis results. Finally, Section 8 concludes this work.

## 2. The Security Problem of Kumari

This section discusses the RTA on the Kumari scheme.

*2.1. RTA.* RTA is a precalculated table that is used to break the hash. RTA is sometimes used to recover passwords or credit card numbers. This attack has tables of specified length and contains a limited number of components [22]. To study the working method, you can visit Rainbow Crack.

*2.2. Notations.* Table 2 shows the notations used in the paper.

### 2.3. RTA in the Registration Phase

Step 1: Edi merges the Idi and Pwi values in the registration phase, hashes them into *li*, and sends them to the CS. Attackers can break the hash value generated by Edi at this point using a rainbow attack. After this step, it can read the data sent by Edi to the CS or change this

Table 1: Comparison of related works with SAIFC scheme.

| Related works | Fog based | RTA | Mutual authentication | Key exchange | ECC based | HTTP based |
|---|---|---|---|---|---|---|
| [10] | No | No | No | No | No | No |
| [11] | No | No | No | Yes | Yes | No |
| [12] | No | No | Yes | No | No | No |
| [13] | Yes | No | Yes | No | No | No |
| [14] | No | No | No | Yes | No | No |
| [15] | No | No | Yes | No | Yes | No |
| [16] | No | No | No | No | No | No |
| [17] | No | No | No | No | No | No |
| [18] | No | No | Yes | No | No | No |
| [19] | No | No | Yes | No | Yes | Yes |
| [20] | No | No | Yes | No | Yes | Yes |
| [21] | Yes | No | Yes | No | Yes | No |
| SAIFC | Yes | Yes | Yes | Yes | Yes | Yes |

Table 2: Used notations in this paper.

| Notations | Description |
|---|---|
| Vdi | Vehicle |
| Idi | Identity of Vdi |
| Pwi | Password of Vdi |
| CS | Cloud server |
| $R$ | RSU (fog node) |
| $F$ | Fog |
| Idcs | Identity of CS |
| IdR | Identity of $R$ |
| XR | The secret key of $R$ is based on ECC |
| $Zp$ | Finite field group |
| $p$ | Prime number of the order $>2^{160}$ |
| r1, r2 | Random numbers generated for ECC |
| rs | Random number generated by R |
| $G$ | Generator point of a large order n |
| Ck | Cookie information |
| Et | Cookie expiration time |
| $h(\cdot)$ | HF |
| $\oplus$ | XOR |
| $\|\|$ | Concatenation |
| $\Delta T$ | Expiration time |
| TV | Timestamp Vdi |
| TR | Timestamp $R$ |
| TF | Timestamp fog |

information and send it back to the CS. The steps are as follows:

Edi $\longrightarrow$ CS: $\{Ii\}$
Attacker $\longrightarrow$ CS: $\{Ii\}$
Rainbow Crack{Ii}
Resend {Ii} for CS

Step 2: The CS then performs a series of calculations to respond to Edi's registration and sends the hashed PIdi and Ck′ to Eddie to continue. Attackers can break the hash value generated by CS at this point using a rainbow attack. After this step, it can read the data sent by CS to the Edi or change it and send it back to the Edi. The steps are as follows. Figure 2 shows the stages of a RTA in the registration phase.

CS $\longrightarrow$ Edi: {PIdi, Ck′}
Attacker $\longrightarrow$ Edi: {PIdi, Ck′}
Rainbow Crack {PIdi, Ck′}
Resend {PIdi, Ck′} for Edi

## 2.4. RTA in the Login and Authentication Phase

Step 1: Edi performs a series of calculations in login and authentication, generates $P1$, $P2$, and $P$Idi data, and sends it to the cloud. Attackers can break the hash value generated by Edi at this point using a rainbow attack. After this step, it can read the data sent by Edi to the CS or change this information and send it back to the CS. The steps are as follows:

Step $R3$: Edi $\longrightarrow$ CS: $\{P1, P2, P\,\mathrm{Idi}\}$
Step $A3$: Attacker $\longrightarrow$ CS: $\{P1, P2, P\,\mathrm{Idi}\}$
Rainbow Crack $\{P2\}$
Resend $\{P1, P2, P\,\mathrm{Idi}\}$ for CS

The attacker can listen to the sent messages in steps 2 and 3 because she has obtained the sent data in the previous step. Figure 3 shows the stages of an RTA in the login and authentication phase.

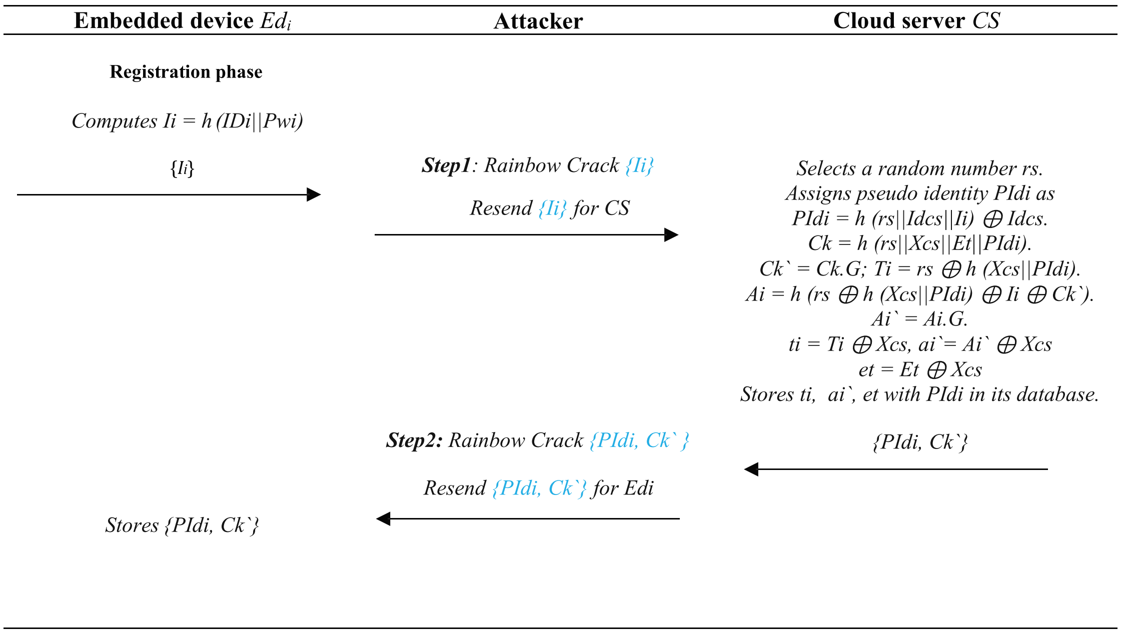
## 2.5. Security Analysis Kumari.

Avispa is used to evaluate the security of Internet protocols [23]. Avispa will provide an HLPSL to define the security of protocols and display their security specifications. We analyzed the security of Kumari with Avispa, and the results show that it is vulnerable to a rainbow attack. In the first stage, attackers can break the hashed data sent from Edi by using a rainbow attack. Figure 4 shows the security weakness of Kumari.

## 2.6. Testbed.

The tools we used to test the attack on the Kumari scheme were the Linux operating system and the board. We chose MD5 by default because the author did not mention the type of hash used in his work, and the MD5 library we used for the Arduino is available on GitHub. Table 3 shows information about the environment and the tools used.

Arduino is a hardware and processing platform designed as open source. This platform is based on a simple $I/O$ and



**Embedded device $Ed_i$** | **Attacker** | **Cloud server $CS$**

**Registration phase**

*Computes Ii = h (IDi||Pwi)*

{Ii}

***Step1**: Rainbow Crack {Ii}*

*Resend {Ii} for CS*

*Selects a random number rs.*
*Assigns pseudo identity PIdi as*
*PIdi = h (rs||Idcs||Ii) ⊕ Idcs.*
*Ck = h (rs||Xcs||Et||PIdi).*
*Ck` = Ck.G; Ti = rs ⊕ h (Xcs||PIdi).*
*Ai = h (rs ⊕ h (Xcs||PIdi) ⊕ Ii ⊕ Ck`).*
*Ai` = Ai.G.*
*ti = Ti ⊕ Xcs, ai`= Ai` ⊕ Xcs*
*et = Et ⊕ Xcs*
*Stores ti, ai`, et with PIdi in its database.*

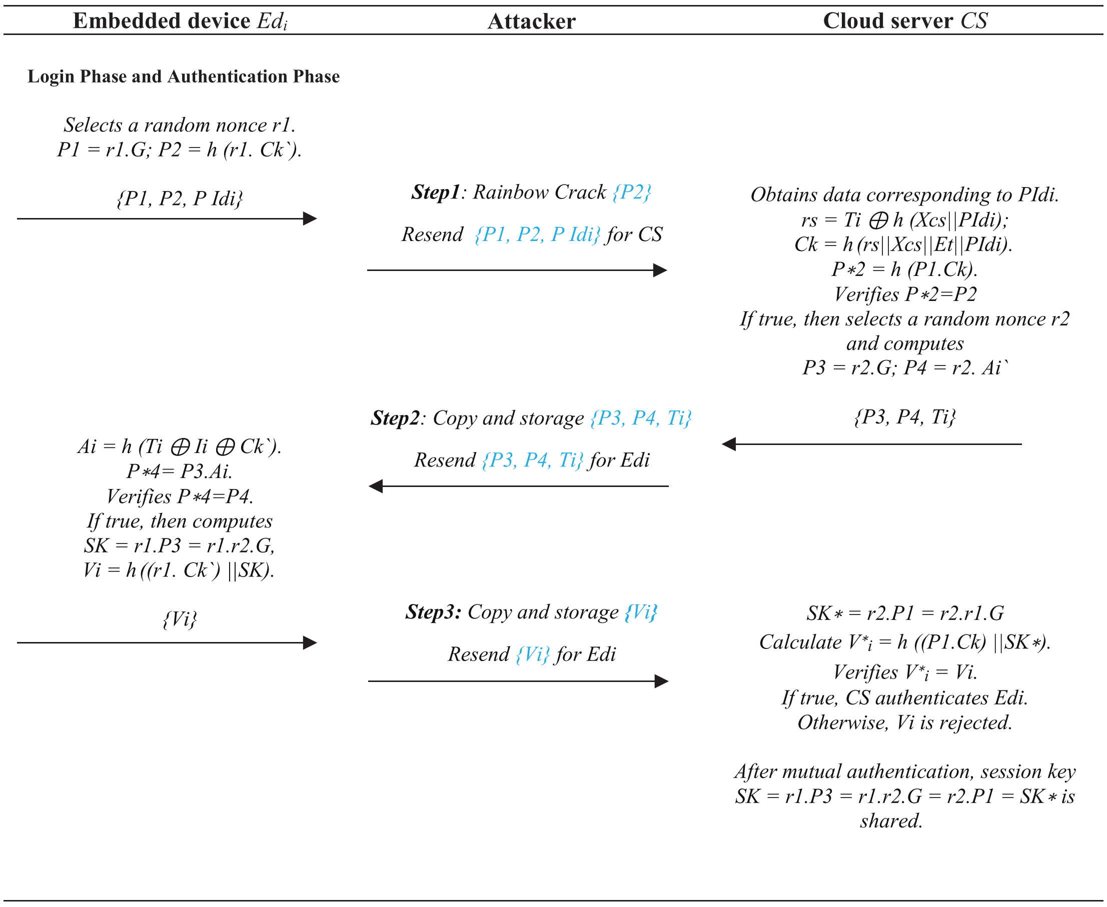
***Step2:** Rainbow Crack {PIdi, Ck` }*

*{PIdi, Ck`}*

*Resend {PIdi, Ck`} for Edi*

*Stores {PIdi, Ck`}*

Figure 2: Stages of a RTA in the registration.



**Embedded device $Ed_i$** | **Attacker** | **Cloud server $CS$**

**Login Phase and Authentication Phase**

*Selects a random nonce r1.*
*P1 = r1.G; P2 = h (r1. Ck`).*

*{P1, P2, P Idi}*

***Step1**: Rainbow Crack {P2}*

*Resend {P1, P2, P Idi} for CS*

*Obtains data corresponding to PIdi.*
*rs = Ti ⊕ h (Xcs||PIdi);*
*Ck = h (rs||Xcs||Et||PIdi).*
*P∗2 = h (P1.Ck).*
*Verifies P∗2=P2*
*If true, then selects a random nonce r2*
*and computes*
*P3 = r2.G; P4 = r2. Ai`*

***Step2**: Copy and storage {P3, P4, Ti}*

*{P3, P4, Ti}*

*Resend {P3, P4, Ti} for Edi*

*Ai = h (Ti ⊕ Ii ⊕ Ck`).*
*P∗4= P3.Ai.*
*Verifies P∗4=P4.*
*If true, then computes*
*SK = r1.P3 = r1.r2.G,*
*Vi = h ((r1. Ck`) ||SK).*

*{Vi}*

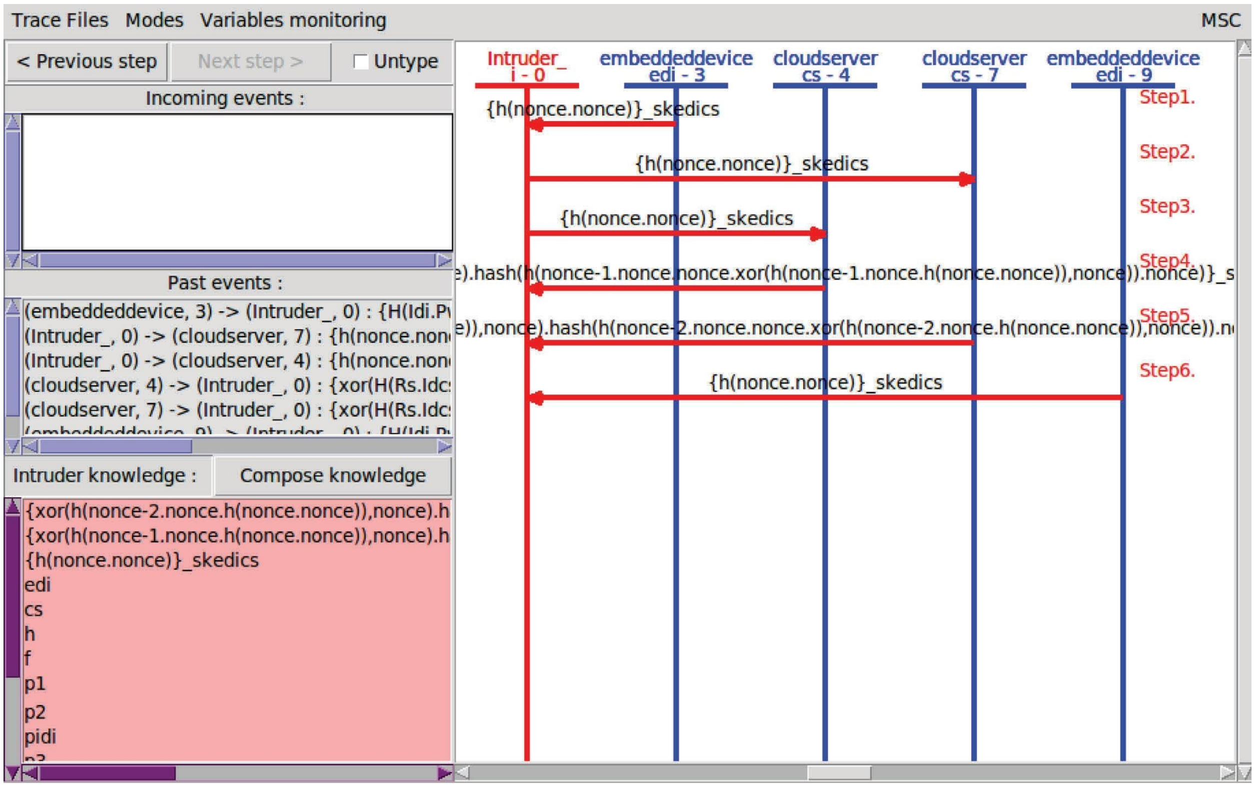
***Step3:** Copy and storage {Vi}*

*Resend {Vi} for Edi*

*SK∗ = r2.P1 = r2.r1.G*
*Calculate $V^*_i$ = h ((P1.Ck) ||SK∗).*
*Verifies $V^*_i$ = Vi.*
*If true, CS authenticates Edi.*
*Otherwise, Vi is rejected.*

*After mutual authentication, session key*
*SK = r1.P3 = r1.r2.G = r2.P1 = SK∗ is*
*shared.*

Figure 3: Stages of a RTA in the login and authentication.



```
(embeddeddevice, 3) -> (Intruder_, 0) : {H(Idi.P
(Intruder_, 0) -> (cloudserver, 7) : {h(nonce.non
(Intruder_, 0) -> (cloudserver, 4) : {h(nonce.non
(cloudserver, 4) -> (Intruder_, 0) : {xor(H(Rs.Idc
(cloudserver, 7) -> (Intruder_, 0) : {xor(H(Rs.Idc
```

```
{xor(h(nonce-2.nonce.h(nonce.nonce)),nonce).h
{xor(h(nonce-1.nonce.h(nonce.nonce)),nonce).h
{h(nonce.nonce)}_skedics
edi
cs
h
f
p1
p2
pidi
```

```
{h(nonce.nonce)}_skedics
{h(nonce.nonce)}_skedics
{h(nonce.nonce)}_skedics
e).hash(h(nonce-1.nonce.nonce.xor(h(nonce-1.nonce.h(nonce.nonce)),nonce)).nonce)}_s
e)),nonce).hash(h(nonce-2.nonce.nonce.xor(h(nonce-2.nonce.h(nonce.nonce)),nonce)).n
{h(nonce.nonce)}_skedics
```

Figure 4: Vulnerability of Kumari et al. in the tool AVISPA.

Table 3: Environmental information and testbed tools.

| Environment | Description |
|---|---|
| Operating system | Kali Linux ver: 2020 |
| Boards | Arduino Uno R3 |
| Programming languages | ANSI C |
| Type hash | MD5 Arduino libs |
| Attack tools | Rainbow table ver: 1.8 |

the designed processing/wiring language. Also, this platform is suitable for communication with external systems and software. Our work in the article used the Arduino board model Uno $R3$, microcontroller ATmega328, and input voltage, 7–12 with memory of 32 KB, and a speed clock of 16 MHz. Figure 5 shows the implementation of the Arduino boards.

*2.6.1. Appointed Data.* To attack the Kumari scheme, in the register phase, we need ID and Pwi, equal to $a1$ and 260 each. Table 4 shows the required data in the registration phase. Next, we will implement the given data in the Arduino board, and we will get each of the provided data from the serial port of the hash port. Finally, we will use these data to prove the authenticity of our attack. Figures 6 and 7 show the hash output in the Arduino board, and Figure 8 shows the MD5 source code used.

*2.6.2. Testbed Result.* At this stage, we will attack the registration phase of the Kumari scheme using a Rainbow. Table 4 shows the values and hashes of ID and Pwi. To

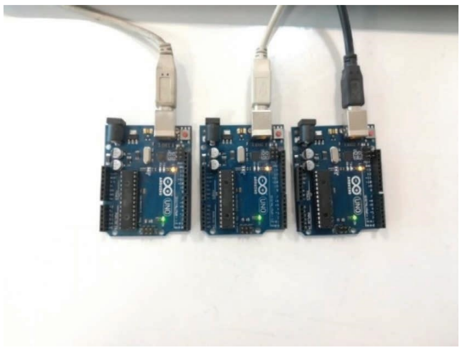


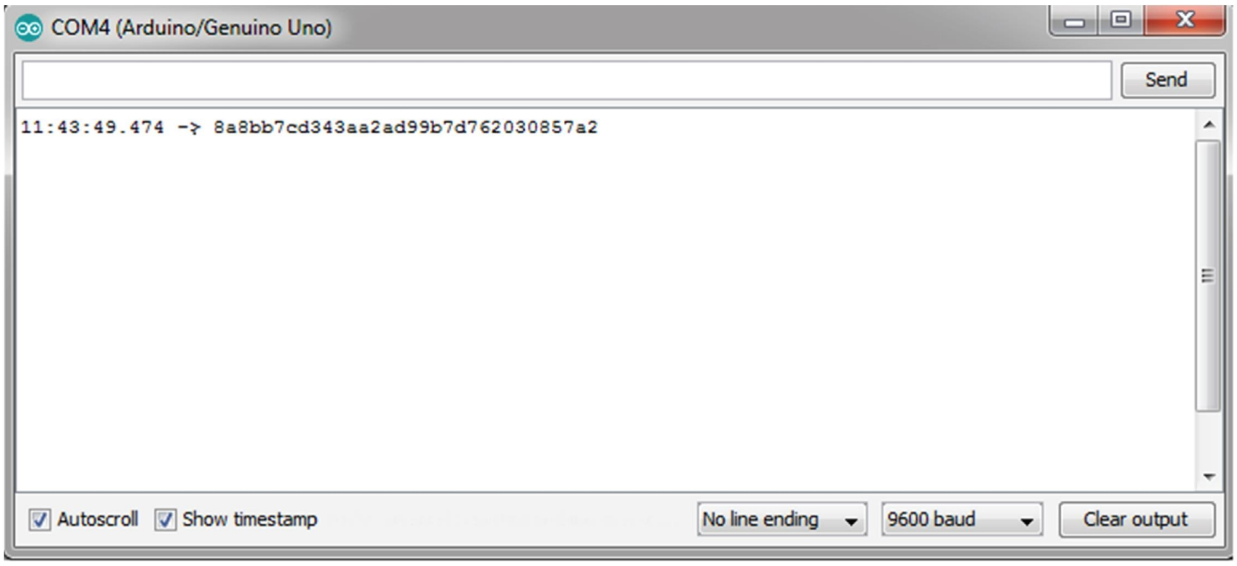
Figure 5: Implementation of the Kumari scheme of the Arduino board.

perform the attack, we assign the hash values of ID and Pwi to the Rainbow tables. The output results show that the Rainbow breaks the hashes given to the plain text quickly, which shows the weakness of the design against this attack. Figure 9 shows the results of an attack on an ID, and Figure 10 shows an attack on Pwi. The ciphertext/plain text and statistical results are marked with a red box in the image.

The results of the statistics of the RTA to reach the plain text ID are as follows: total time 0.58, time of chain traverse 0.58, hash and reduce the calculation of chain traverse 7216200, hash and reduce the calculation of alarm traverse 16391, number of alarm 11, the performance of chain traverse 12.46 million/s, and the performance of chain alarm 12.46 million/s. The results of the statistics of the RTA to reach the plain text Pwi are as follows: total time 0.58 s, time

Table 4: Data used in the testbed.

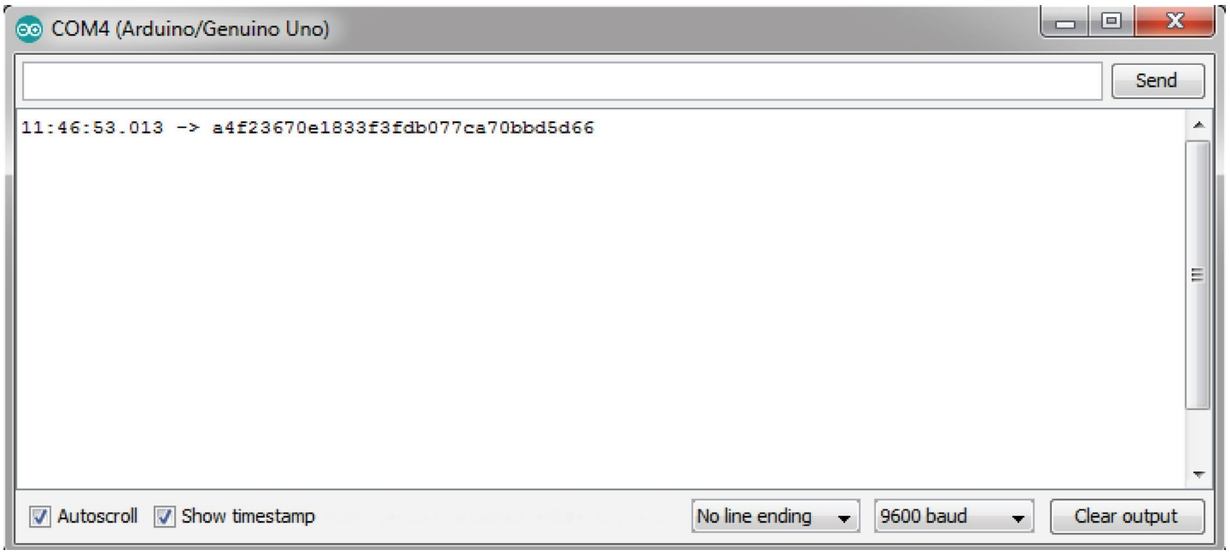
| Notations | Values | HEX | MD5 |
|---|---|---|---|
| ID | $a1$ | 6131 | 8a8bb7cd343aa2ad99b7d762030857a2 |
| Pwi | 260 | 323630 | a4f23670e1833f3fdb077ca70bbd5d66 |



COM4 (Arduino/Genuino Uno)

```
11:43:49.474 -> 8a8bb7cd343aa2ad99b7d762030857a2
```

Figure 6: The results of ID hash.



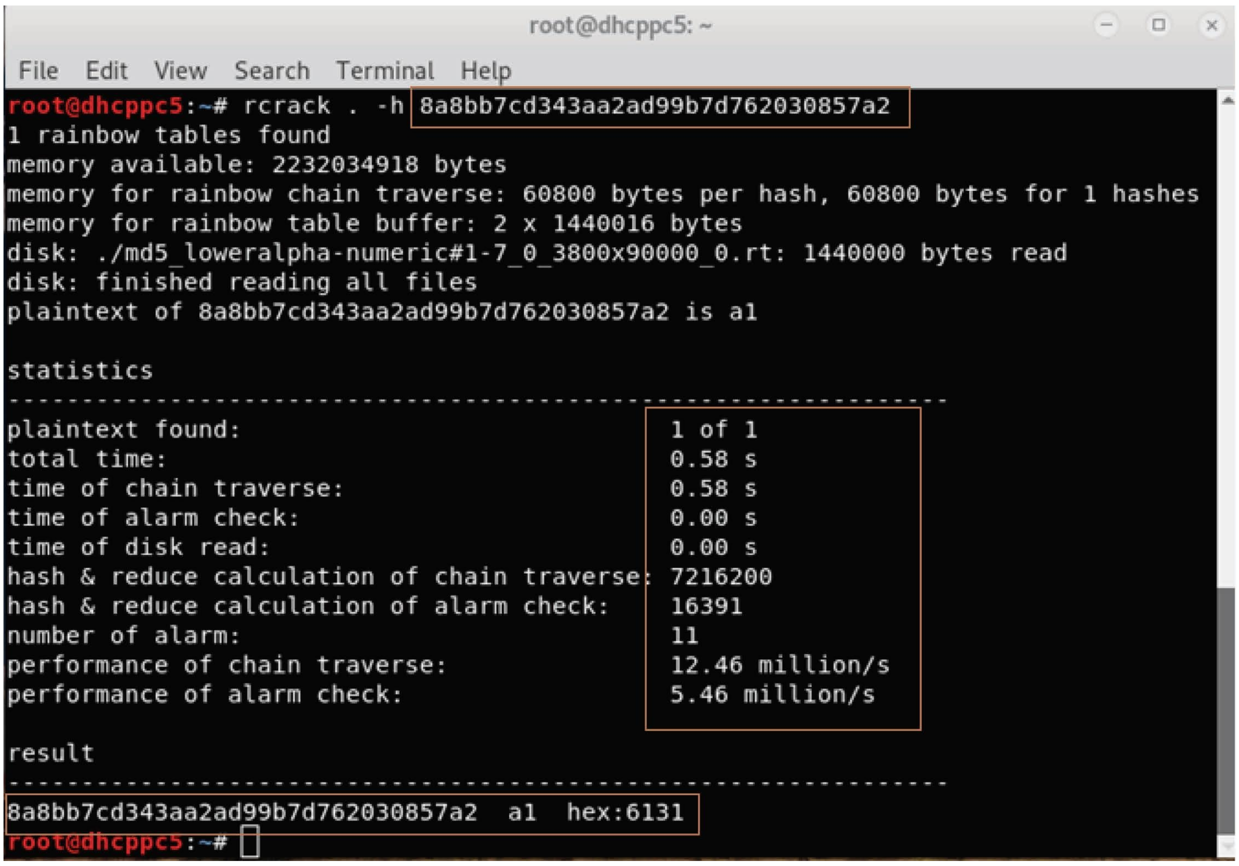
COM4 (Arduino/Genuino Uno)

```
11:46:53.013 -> a4f23670e1833f3fdb077ca70bbd5d66
```

Figure 7: The results of Pwi hash.

```
#include <MD5.h>
Void setup ()
{  //initialize serial
  Serial. Begin (9600);
  //give it a second
  Delay (1000);
  //generate the MD5 hash for our string
  Unsigned char* hash=MD5:: make hash ("260");
  //generate the digest (hex encoding) of our hash
  Char *md5str = MD5::make_digest (hash, 16);
  Free (hash);
  //print it on our serial monitor
  Serial.println (md5str);
  //Give the Memory back to the System if you run
the md5 Hash generation in a loop
  Free (md5str); }
Void loop () { }
```

Figure 8: MD5 source code.

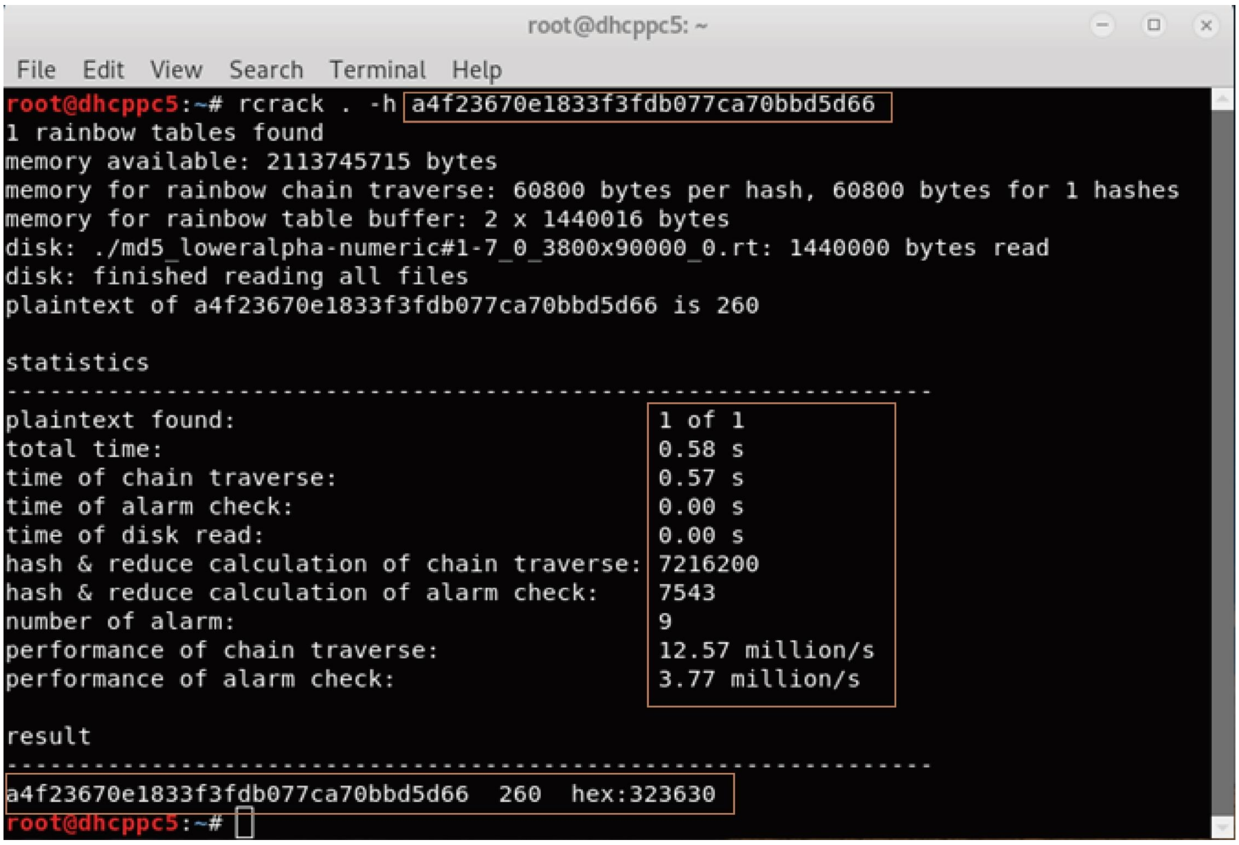


```
root@dhcppc5:~# rcrack . -h 8a8bb7cd343aa2ad99b7d762030857a2
1 rainbow tables found
memory available: 2232034918 bytes
memory for rainbow chain traverse: 60800 bytes per hash, 60800 bytes for 1 hashes
memory for rainbow table buffer: 2 x 1440016 bytes
disk: ./md5_loweralpha-numeric#1-7_0_3800x90000_0.rt: 1440000 bytes read
disk: finished reading all files
plaintext of 8a8bb7cd343aa2ad99b7d762030857a2 is a1

statistics
----------------------------------------------------------------
plaintext found:                          1 of 1
total time:                               0.58 s
time of chain traverse:                   0.58 s
time of alarm check:                      0.00 s
time of disk read:                        0.00 s
hash & reduce calculation of chain traverse: 7216200
hash & reduce calculation of alarm check:    16391
number of alarm:                          11
performance of chain traverse:            12.46 million/s
performance of alarm check:               5.46 million/s

result
----------------------------------------------------------------
8a8bb7cd343aa2ad99b7d762030857a2  a1  hex:6131
root@dhcppc5:~#
```

Figure 9: The results of the RTA on the ID hash.



```
root@dhcppc5:~# rcrack . -h a4f23670e1833f3fdb077ca70bbd5d66
1 rainbow tables found
memory available: 2113745715 bytes
memory for rainbow chain traverse: 60800 bytes per hash, 60800 bytes for 1 hashes
memory for rainbow table buffer: 2 x 1440016 bytes
disk: ./md5_loweralpha-numeric#1-7_0_3800x90000_0.rt: 1440000 bytes read
disk: finished reading all files
plaintext of a4f23670e1833f3fdb077ca70bbd5d66 is 260

statistics
----------------------------------------------------------------
plaintext found:                            1 of 1
total time:                                 0.58 s
time of chain traverse:                     0.57 s
time of alarm check:                        0.00 s
time of disk read:                          0.00 s
hash & reduce calculation of chain traverse: 7216200
hash & reduce calculation of alarm check:    7543
number of alarm:                            9
performance of chain traverse:              12.57 million/s
performance of alarm check:                 3.77 million/s

result
----------------------------------------------------------------
a4f23670e1833f3fdb077ca70bbd5d66  260  hex:323630
root@dhcppc5:~# 
```

FIGURE 10: The results of the RTA on the Pwi hash.

of chain traverse 0.57 s, hash and reduce the calculation of chain traverse 7216200, hash and reduce the calculation of alarm traverse 7543, the number of alarm 9, the performance of chain traverse 12.57 million/s and, the performance of chain alarm 3.77 million/s.

## 3. Network Model

This section describes the network, assumption, and adversary models and reviews ECC.

The network model for fog computing-based IOV is shown in Figure 11. This network model has a variety of connections between different parties, such as "V2V, V2R, R2F, F2F, F2C." V2V: vehicles can communicate with other vehicles, and receive and send traffic information and other data. V2R: RSU can communicate with vehicles, exchange information, and be associated with other RSUs. R2F: sometimes, the received data require complex processing beyond the power of RSU, in which case the data do transmit to fog for processing. F2F: fog can communicate with other fog and support each other to process data [24]. F2C: when fog cannot do the necessary processing, they send the data to the cloud. The assumptions in this model are as follows:

(i) The time of all devices is synchronized

(ii) Cloud and fog and fog nodes know each other

(iii) Fog and clouds are resistant to various attacks that also do not leak any data

(iv) The transmission channel is not secure in the network

*3.1. Problem Statement.* The main challenge in the IOV environment is to ensure the source of the data sent. Authentication allows us to identify the source of the transmitted data and to be able to detect fake data. The Kumari scheme is a cookie-based authentication scheme for IOT environments. However, this scheme is vulnerable to rainbow attacks. This paper presents a SAIFC scheme based on the ECC, which uses the cookie in the HTTP protocol to send data. This scheme provides secure authentication between source and destination and resists active and passive attacks.

*3.2. Adversary Model.* The following are some of the attacks that can be dangerous in an IOV environment:

(i) Replay attack: an attack method where an intruder records a communication session and then broadcasts it again

(ii) Man-in-the-middle attack: MITM attack is when an intruder uses session data to forge connections or change data

(iii) Sybil attack: an attack in which the attacker can have multiple identities and deceive other vehicles

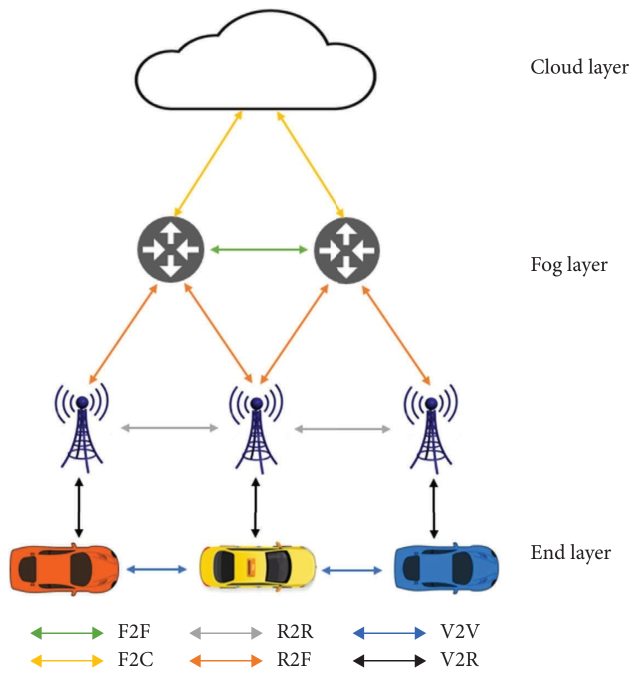


Figure 11: Network model of IOV and fog computing.

(iv) Impersonation attack: the intruder intends to forge the other person's identity in these attacks

(v) Brute force attack: the attacker uses all possible modes to break the encrypted text

(vi) Rainbow table: an attacker uses tables where the hash text output does save to break the hash text

Our SAIFC will be resistant to the attacks.

*3.3. Review ECC.* The ECC is a PKE method based on an algebraic structure of EC on finite fields. The use of EC in encryption was proposed independently by Neal et al. in 1985. The PKE is based on the difficulties in some math problems. Earlier, systems based on the public key were considered safe, assuming that finding two or more prime factors for a large integer was difficult. For EC-based algorithms, it is assumed that finding the DL from a random element of EC is impractical, given a publicly known base point. The size of the EC determines the difficulty of the problem. The main advantage of the ECC was a key with a smaller size, which means reduced storage. The EC is a flat curve composed of equation (1) for today's encryption purposes.

$$y^2 = x^3 + ax + b. \quad (1)$$

## 4. SAIFC

In this section, the different phases of the SAIFC scheme are described. Figure 12 shows the roadmap of the SAIFC.

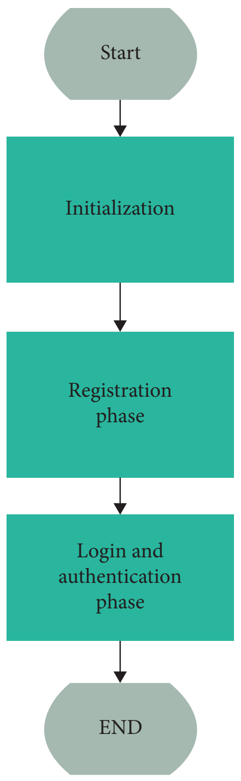


Figure 12: SAIFC roadmap.

*4.1. Initialization.* The first $R$ chooses equation (1) on the EC of $Zp$. After $R$ choose the element is $f$, $a, b \in Zp$ each in which $a, b$ fulfill condition $y^2 = x^3 + ax + (b) (\text{mod } p)$. In the EC, $G$ is the foundation point, with a prime order of $n (n > 2^{160})$. If the $O$ is a point, then the equation $n.G = O$ is at infinity. $XR$ are randomly selected as the secret keys of $R$.

*4.2. Register Phase.* The Register phase is as follows:

Step 1: Vdi to register in $R$, $Ii = h$ $(IdR||\text{Pwi})$, $\text{Tv} = h$ (TV) computes and sends $Ii$, Tv, and TV to $R$.

Step 2: When the registration request was received, $R$ checks TV in the computes of $\text{TV}' = h$ (TV) and the result obtained with Tv, and if it is the same, it checks in terms of timestamp. If it is small from the expiration time, continue the steps. $R$ produces a $rs$ and computes $\text{PIdR} = h$ (rs||IdR||Ii) $\oplus$ IdR for Vdi and storage IdR. Then, $R$ computes the Ck and other component. $R$ storage ti, ai$'$, and et corresponding to PIdR of Vdi in its DB. The expiration time of the Et that corresponds to Ii of Vdi is storage by $R$ himself. $R$ timestamp calculated itself in the $\text{Tr} = h$ (TR) After, sends {PIdR, Ck$'$, Tr, TR} to Vdi through a communication channel.

Step 3: Vdi checks TR in the computes of $\text{TR}' = h$ (TR), and the result obtained with Tr; if it is the same, it checks in terms of timestamp. After receiving {PIdR, Ck$'$}, the Vdi stores PIdR and Ck$'$ in its memory. $\text{Ck} = h$

(rs||XR||Et||PIdR) can update its expiration time. Figure 13 shows the flowchart of the SAIFC registration phase.

### *4.3. Login and Authentication Phase*

Step 1: For each entry, the Vdi selects a $r1$ and computes the ECC point $P1 = r1.G$; $P2 = h(r1.Ck^{'})$. Then, it stores $P1$ in its memory, and Tv = $h$ (TV) computes and sends the login request {P1, P2, PIdR, Tv, $TV$} to $R$.

Step 2: Upon receiving the login request, $R$ checks TV in the computes of TV$'$ = $h$ (TV) and the result is obtained with Tv, and if it is the same, it checks in terms of timestamp. If it is small from the expiration time, continue the steps: $R$ data corresponding to the receives PIdR and computes $rs = \text{Ti}\oplus h(\text{XR}\|\text{PIdR})$. Next, $R$ computes the Ck = $h$ (rs ||XR||Et ||PIdR) and $P*2 = h(P1.\text{Ck})$.

Step 3: $R$ selects a random nonce $r2$, computes the ECC point $P3 = r2.G$; $P4 = r2.\ Ai'$ and Tr = $h$ (TR) compute sends {$P3$, $P4$, Ti, Tr, TR} to Vdi.

Step 4: Upon receiving, Vdi checks TR in the computes of TR$'$ = $h$ (TR), and the result obtained with $Tr$, and if it is the same, it checks in terms of timestamp. The next step computes $Ai = h(\text{Ti}\oplus Ii\oplus\text{Ck}^{'})$ and the ECC point $P*4 = P3.Ai$. Then, it verifies $P*4 = P4$ to authenticate $R$. If the verification holds, then Vdi authenticates $R$ and continues the next step;

Step 5: Vdi computes the session key $\text{SK} = r1.P3 = r1.r2.\text{G}$ and $\text{VR} = h((r1.\text{Ck}^{'})\|SK)$ and Tv = $h$ (TV) sends to $R$.

Step 6: Upon receiving the login request, $R$ checks TV in the computes of TV$'$ = $h$ (TV) and the result obtained with Tv, and if it is the same, it checks in terms of timestamp. If it is minor from the expiration time, continue the steps: $R$ computes the session key $SK = r1\cdot P3 = r1\cdot r2\cdot G = r2\cdot P1 = SK*$. Then, $R$ verifies $V*R = \text{VR}$ to authenticate Vdi. If the verification holds, $R$ authenticates Vdi; otherwise, VR is rejected. From then on, all the after messages transmitted between Vdi and $R$ are XOR with the session key $SK = r1\,P3 = r1\,r2\,G = r2\,P1 = \text{SK}*$. Figure 14 shows the flowchart of the login and authentication phase of the SAIFC.

Step 7: $R$ calculates Tr = $h$ (TR), IDR$'$ = $h$ (IDR) and sending to $F$. $F$ checks TR in the computes of TR$'$ = $h$ (TR), and the result obtained with Tr, and if it is the same, it checks in terms of timestamp. In the next step, calculate IdR$'$ = $h$ (IDR), check whether IDR$'$ = IDR$'$ that is true storage the IDR. In the next step, calculate Tf = $h$ (TF) and {IDR$'$, IDR, Tf, TF} sending to CS.

Step 8: $F$ checks Tf in the computes of TF$'$ = $h$ (TF), and the result obtained with Tf; if it is the same, it checks in terms of timestamp. In the next step, calculate IDR$'$ = $h$ (IDR), check whether IDR$'$ = $IDR'$ that is true storage of the IDR. Figure 15 shows the authentication steps.

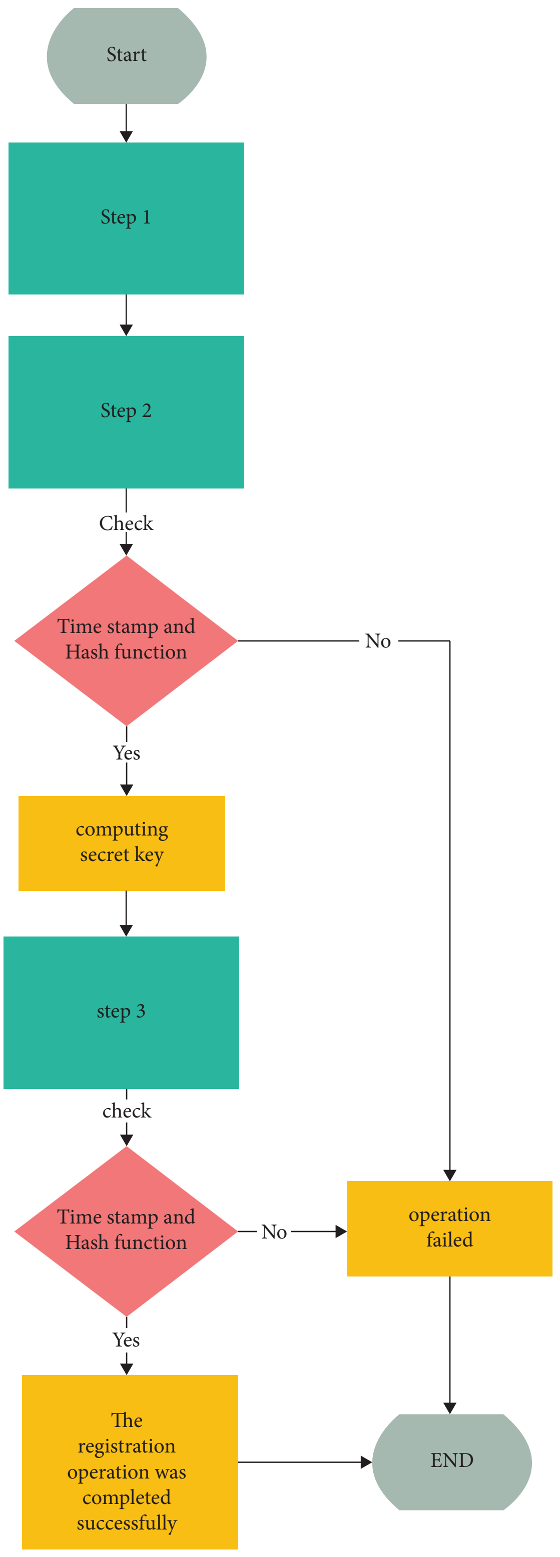

Start
Step 1
Step 2
Check
Time stamp and Hash function
No
Yes
computing secret key
step 3
check
Time stamp and Hash function
No
operation failed
Yes
The registration operation was completed successfully
END

Figure 13: Flowchart of the registration phase.

## 5. SAIFC Security Analysis

In this section, security analyzes our SAIFC and discusses the results and analysis, followed by an informal security analysis.

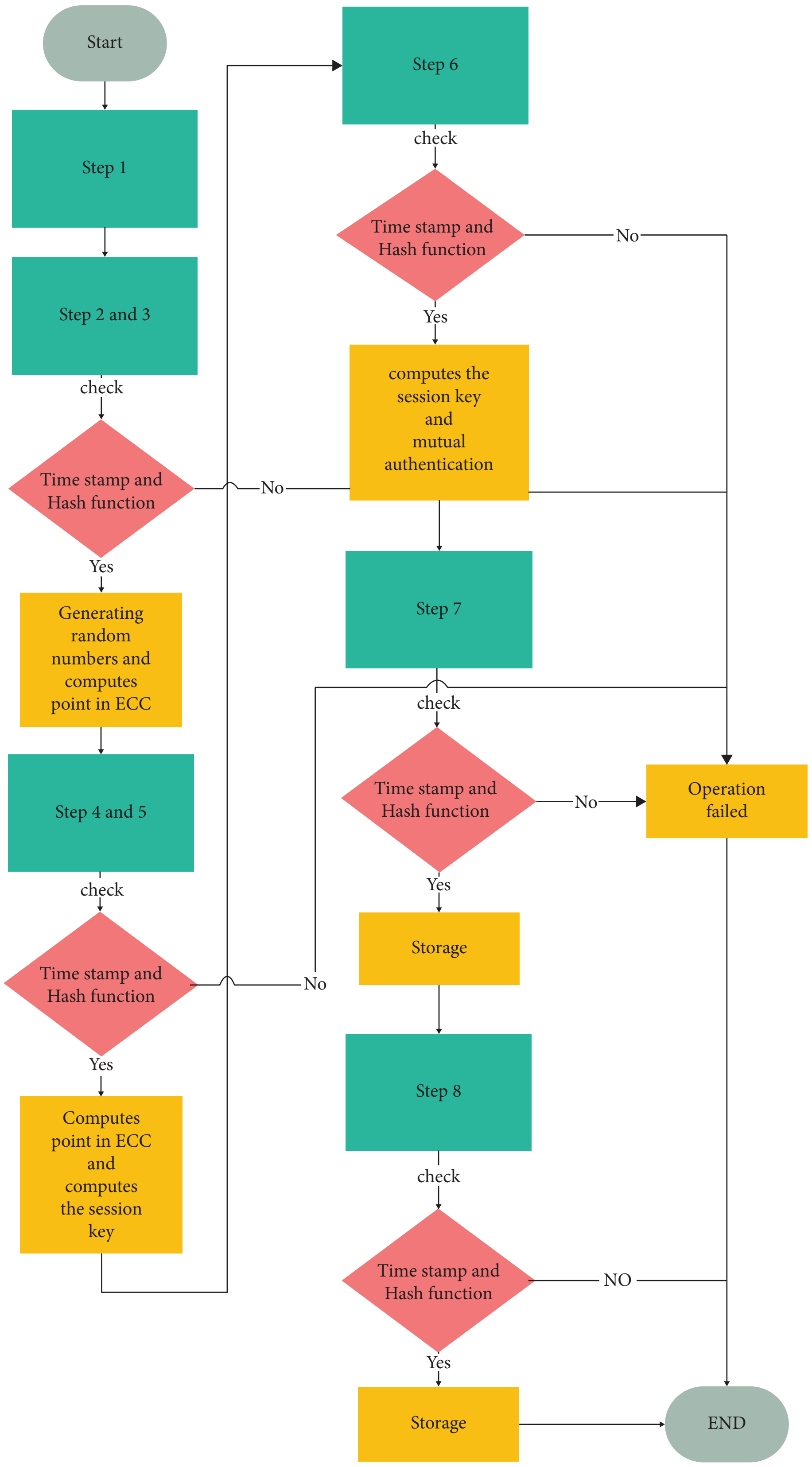
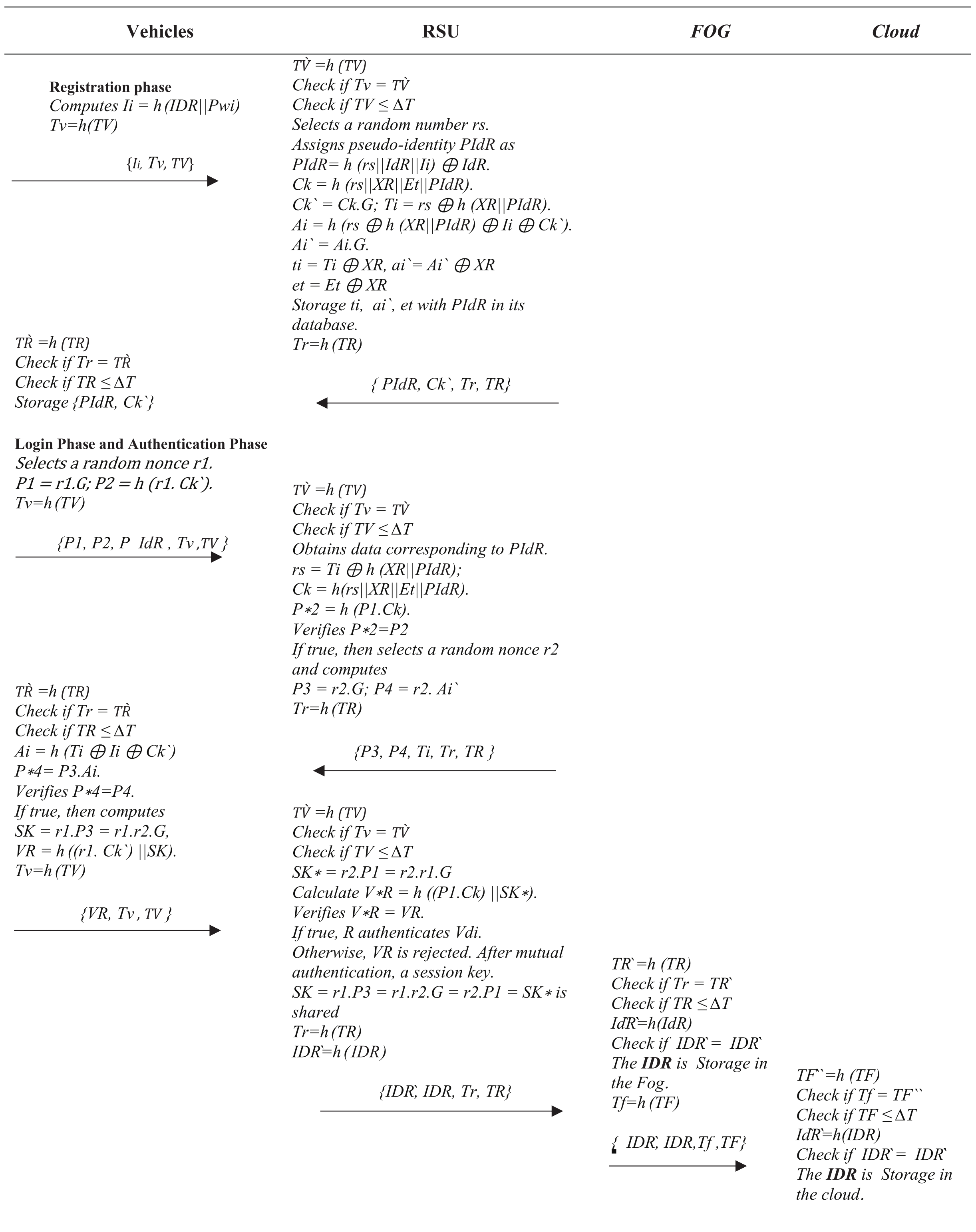

Start
Step 1
Step 2 and 3
check
Time stamp and Hash function
No
Yes
Generating random numbers and computes point in ECC
Step 4 and 5
check
Time stamp and Hash function
No
Yes
Computes point in ECC and computes the session key
Step 6
check
Time stamp and Hash function
No
Yes
computes the session key and mutual authentication
Step 7
check
Time stamp and Hash function
No
Operation failed
Yes
Storage
Step 8
check
Time stamp and Hash function
NO
Yes
Storage
END

Figure 14: Flowchart of the login and authentication phase.



**Vehicles** | **RSU** | ***FOG*** | ***Cloud***

**Registration phase**
*Computes Ii = h (IDR||Pwi)*
*Tv=h(TV)*

{*Ii, Tv, TV*} →

*TV̀ =h (TV)*
*Check if Tv = TV̀*
*Check if TV ≤ ΔT*
*Selects a random number rs.*
*Assigns pseudo-identity PIdR as*
*PIdR= h (rs||IdR||Ii) ⊕ IdR.*
*Ck = h (rs||XR||Et||PIdR).*
*Ck` = Ck.G; Ti = rs ⊕ h (XR||PIdR).*
*Ai = h (rs ⊕ h (XR||PIdR) ⊕ Ii ⊕ Ck`).*
*Ai` = Ai.G.*
*ti = Ti ⊕ XR, ai`= Ai` ⊕ XR*
*et = Et ⊕ XR*
*Storage ti, ai`, et with PIdR in its database.*
*Tr=h (TR)*

← *{ PIdR, Ck`, Tr, TR}*

*TR̀ =h (TR)*
*Check if Tr = TR̀*
*Check if TR ≤ ΔT*
*Storage {PIdR, Ck`}*

**Login Phase and Authentication Phase**
*Selects a random nonce r1.*
*P1 = r1.G; P2 = h (r1. Ck`).*
*Tv=h (TV)*

*{P1, P2, P IdR , Tv ,TV }* →

*TV̀ =h (TV)*
*Check if Tv = TV̀*
*Check if TV ≤ ΔT*
*Obtains data corresponding to PIdR.*
*rs = Ti ⊕ h (XR||PIdR);*
*Ck = h(rs||XR||Et||PIdR).*
*P∗2 = h (P1.Ck).*
*Verifies P∗2=P2*
*If true, then selects a random nonce r2 and computes*
*P3 = r2.G; P4 = r2. Ai`*
*Tr=h (TR)*

← *{P3, P4, Ti, Tr, TR }*

*TR̀ =h (TR)*
*Check if Tr = TR̀*
*Check if TR ≤ ΔT*
*Ai = h (Ti ⊕ Ii ⊕ Ck`)*
*P∗4= P3.Ai.*
*Verifies P∗4=P4.*
*If true, then computes*
*SK = r1.P3 = r1.r2.G,*
*VR = h ((r1. Ck`) ||SK).*
*Tv=h (TV)*

*{VR, Tv , TV }* →

*TV̀ =h (TV)*
*Check if Tv = TV̀*
*Check if TV ≤ ΔT*
*SK∗ = r2.P1 = r2.r1.G*
*Calculate V∗R = h ((P1.Ck) ||SK∗).*
*Verifies V∗R = VR.*
*If true, R authenticates Vdi.*
*Otherwise, VR is rejected. After mutual authentication, a session key.*
*SK = r1.P3 = r1.r2.G = r2.P1 = SK∗ is shared*
*Tr=h (TR)*
*IDR`=h (IDR)*

*{IDR`, IDR, Tr, TR}* →

*TR`=h (TR)*
*Check if Tr = TR`*
*Check if TR ≤ ΔT*
*IdR`=h(IdR)*
*Check if IDR`= IDR`*
*The **IDR** is Storage in the Fog.*
*Tf=h (TF)*

*{ IDR`, IDR,Tf ,TF}* →

*TF``=h (TF)*
*Check if Tf = TF``*
*Check if TF ≤ ΔT*
*IdR`=h(IDR)*
*Check if IDR`= IDR`*
*The **IDR** is Storage in the cloud.*

Figure 15: Authentication of the SAIFC scheme.

AVISPA is a formal verification tool for evaluating a safe protocol that combines several methods to model checking [21, 25, 26]. AVISPA is an HLPSL used to define the security of schemes and their security specifications [27, 28]. In HLPSL, an attacker always plays a legal role that is indicated by (i). The *D-Y* threat model [29] has been embedded. AVISPA uses four tools OFMC [30], CL-AtSe [31], SATMC [32], and TA4SP [33] to analyze security targets. Out of these four tools, SATMC and TA4SP do not support xor operation; therefore, in the simulation, we have used other tools (OFMC and CL-AtSe) to test.

*5.1. Analysis of AVISPA Results.* Our SAIFC scheme, a replay, MITM, and other attacks are discussed in Section 4.2, with tools OFMC and CL-AtSe tested. The simulation results of OFMC and CL-AtSe are shown in Figures 16 and 17, respectively. The total number of visited nodes is 12, while the number of depth four plies with a search time of 0.44 seconds and CL-AtSe analyzed 0 states, and the translation time was 0.15 seconds. Thus, the overall results of the two tools show that the SAIFC scheme is safe.

### *5.2. Informal Security Analysis*

(i) Replay attack: In the SAIFC scheme, the attacker can intercept the messages exchanged at the registration, login, and authentication phases and take legal registration or login in the future. In the SAIFC scheme, TV, TF, and TR parameters are used to prevent this attack, and before any processing, the receiver of the message first checks the time stamp, and if it is smaller than the expiration time, it performs processing; otherwise, the communication channel is closed. The reason for the SAIFC scheme is that it is resistant to replay attacks.

(ii) MITM: The SAIFC scheme can be an attacker placed between vehicle and RSU and intercept or modify the exchanged messages. In the SAIFC scheme to prevent this attack, mutual authentication is used, as shown in steps 3, 4, and 5. Also, in the SAIFC scheme, using a timestamp and HF in each message has made it resistant to a MITM.

(iii) Sybil attack: To prevent a Sybil attack in the SAIFC, in the registration phase, the vehicle first sends its password to RSU. RSU calculates parameters PIdR and Ck′ based on XR and R1 due to the use of Pwi, PIdR, and Ck′, and the SAIFC scheme is resistant to Sybil attack.

(iv) Impersonation attack: We have used mutual authentication in the SAIFC scheme to prevent this attack, which is discussed in steps 3, 4, and 5.

(v) Brute force attack: The attacker wants to check all possible situations until the answer is reached. Assuming this, the attacker can extract the parameters of $P1$, $P2$, $P3$, and $P4$ from the exchanged messages. He cannot attack because key XR is unknown to him, and he has no way to guess the random numbers $r1$ and $r2$. For this reason, the SAIFC schema is resistant to Brute force attacks.

(vi) RTA: If the attacker wants to break the HF of the messages sent between the communication parties, this process takes time. In the SAIFC scheme, a time stamp is used in each message, which makes the attack impossible because it takes time to break the HF. If the time stamp of the received message is greater than the expiration time, the message is considered invalid, and the communication channel is closed. Therefore, the SAIFC scheme against the RTA is resistant.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/Auth_Fog. if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00 s
  searchTime: 0.44 s
  visitedNodes: 12 nodes
  depth: 4 plies
```

Figure 16: Results of the SAIFC in the OFMC.

```
SUMMARY
  SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/span/span/testsuite/results/Auth_Fog.if
GOAL
  As Specified
BACKEND
  CL-AtSe

STATISTICS
  Analysed : 0 states
  Reachable: 0 states
  Translation: 0.15 seconds
  Computation: 0.00 seconds
```

Figure 17: Results of the SAIFC in the CL-ATS.

## 6. SAIFC Performance Analysis

The performance analysis of the SAIFC scheme and security features are compared in this section with protocols by Wazid [21], Liu [15], Liu [11], Kalra [19], Kumari [20], Vasudev [18], Ming Chen [17], Ying and Nayak [14] , Mohit [13] in this section.

Table 5: Comparison of the different schemes in terms of communication costs.

| No | Schemes | HF | ECC | PKE | PKD | SKE | SKD | Total cost | Total cost (ms) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | [21] | 35 Thf | 4Teccm | 0TPKe | 0TPKd | 0TSKe | 0TSKd | 35 Thf + 4Teccm | 8.9845 |
| 2 | [11] | 8 Thf | 11Teccm | 0TPKe | 0TPKd | 0TSKe | 0TSKd | 8 Thf + 11Teccm | 24.5044 |
| 3 | [15] | 10 Thf | 6Teccm | 0TPKe | 0TPKd | 0TSKe | 0TSKd | 10 Thf + 6Teccm | 13.5518 |
| 4 | [19] | 9 Thf | 7Teccm | 0TPKe | 0TPKd | 0TSKe | 0TSKd | 9 Thf + 7Teccm | 15.8043 |
| 5 | [20] | 13 Thf | 8Teccm | 0TPKe | 0TPKd | 0TSKe | 0TSKd | 7 Thf + 8Teccm | 17.8379 |
| 6 | [18] | 17 Thf | 0Teccm | 0TPKe | 0TPKd | 0TSKe | 0TSKd | 17 Thf | 0.0391 |
| 7 | [17] | 17 Thf | 0Teccm | 0TPKe | 0TPKd | 0TSKe | 0TSKd | 17 Thf | 0.0391 |
| 8 | [14] | 12 Thf | 0Teccm | 0TPKe | 0TPKd | 2TSKe | 2TSKd | 12 Thf + 2TSKe + 2TSKd | 0.046 |
| 9 | [13] | 20 Thf | 0Teccm | 0TPKe | 0TPKd | 0TSKe | 0TSKd | 20 Thf | 0.046 |
| 10 | SAIFC | 30 Thf | 8Teccm | 0TPKe | 0TPKd | 0TSKe | 0TSKd | 30 Thf + 8Teccm | 17.877 |

*6.1. Computational Cost.* The computational costs of the SAIFC scheme and other schemes [21], [11], [15], [19], [20], [18], [17], [14], [13] are tabulated in Table 5.

For analysis, the following symbols are defined. Thf is the number execution of HF. Teccm is the number execution of an ECC point multiplication operation. TPKe is the number execution of PKE. TPKd is the execution number of PKD. TSKE is the number execution of SKE. TSKd is the number execution of SKD. The time required to calculate the XOR operation is small, and we do not consider this. We use the paper [30] evaluation results for different cryptographic.

Our observations show that protocols by Vasudev et al. [18] and Chen et al. [17], with 0.0391 ms, have a lower cost compared to Ying and Nayak [14] and Mohit et al. [13] protocols which cost 0.046 ms. Wazid et al. [21] and Liu et al. [15], and Kalra and Sood [19] protocols have costs of 8.9845 ms and 13.5518 ms, and 15.8043 ms, respectively. The cost of the SAIFC is slightly higher than the Kumari et al. [20] protocol, and Liu et al. [11] protocol has the highest computation cost.

*6.2. Communication Cost.* A comparative study of the communication costs and total bits of different schemes is presented in Table 6. The obtained results have been measured manually and with the E3C tool [34]. Our observations show that Liu et al. [15] protocol has the lowest communication cost. The next is Kalra and Sood [19] and Kumari et al. [20] protocols with communication costs of 3. Next, Chen et al. [17] and Ying and Nayak [14] protocols are the communication cost. The SAIFC scheme costs more than Wazid, [21], Liu [11], and Vasudev [18] protocols, and Mohit et al. [13] protocol has the highest communication cost. First, Mohit [13] protocol has the least bits, and then, the SAIFC scheme and Kalra and Sood [19] and Kumari et al. [20] protocol are 1760 bits. Next are Ying and Nayak [14] and Liu et al. [15] and Vasudev et al. [18] and Chen et al. [17], and Wazid et al. [21] protocols, 1952 bit and 2272 bit and 2496 bit and 3024 bit and 3392 bit, respectively. Finally, Liu et al. [11] protocol has the most bit.

*6.3. Security Features Comparison.* Our observations show that all protocols are resistant to the replay attack. However, it is vulnerable to Mohit [5] protocol and MITM and Kalra [11] protocol insider attack. All protocols are resistant to stolen-verifier attacks, impersonation attacks, Brute force attacks, and offline password-guessing attacks. However, Kalra and Sood [19] protocol is vulnerable to offline password-guessing attacks. Except for Kalra and Sood [19] protocol, everyone can support device anonymity, mutual authentication, session key agreement, and forward secrecy. In the SAIFC scheme, a timestamp is considered for sending each message, which is checked at the destination with expiration time. For this reason, the SAIFC scheme can be resistant to rainbow attacks. The SAIFC scheme is based on HTTP Protocol and can support fog, OFMC, and CL-ATSE used for security evaluation. Table 7 shows a comparison of security features.

Note: FV1: replay attack; FV2: MITM; FV3: insider attack; FV4: stolen-verifier attack; FV5: impersonation attack; FV6: Brute force attack; FV7: offline password guessing attack; FV8: device anonymity; FV9: mutual authentication; FV10: session key agreement; FV11: forward secrecy; FV12: confidentiality; FV13: RTA; FV14: OFMC; FV15: CL-ATSE; FV16: fog-based; FV17: HTTP-based.

## 7. Simulation Results and Analysis

A feasible demonstration of the SAIFC by the NS3 presents in this section.

*7.1. Simulation Environment and Settings.* Table 8 presents the parameters used in the NS3.

*7.2. SAIFC Simulation Results.* We simulated our SAIFC using the three routing protocols AODV, DSDV, and OLSR. The results of packet delivery show that DSDV protocol performed better, OLSR protocol performed moderately, and AODV protocol performed poorly. Figure 18 shows the packet delivery rate comparison. A comparison of throughput shows that DSDV protocol performed better and AODV protocol performed poorly. Figure 19 shows the throughput comparison. In packet loss, DSDV protocol is higher after AODV and OLSR protocol is placed, respectively. Figure 20 shows the packet loss comparison. OLSR protocol has less delay than DSDV and AODV. Figure 21 shows the end to end delay comparison. According

Table 6: Comparison of the different schemes in terms of communication cost and the number of bits.

| No | Schemes | Number of messages | Total bits |
|---|---|---|---|
| 1 | [21] | 6 | 3392 |
| 2 | [11] | 6 | 8992 |
| 3 | [15] | 2 | 2272 |
| 4 | [19] | 3 | 1760 |
| 5 | [20] | 3 | 1760 |
| 6 | [18] | 6 | 2496 |
| 7 | [17] | 4 | 3024 |
| 8 | [14] | 4 | 1952 |
| 9 | [13] | 9 | 1280 |
| 10 | SAIFC | 7 | 1760 |

Table 7: Comparison of the different schemes in terms of security features.

| Security features | Schemes | | | | | | | | | SAIFC |
|---|---|---|---|---|---|---|---|---|---|---|
| | [21] | [11] | [15] | [19] | [20] | [18] | [17] | [14] | [13] | |
| FV1 | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| FV2 | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | Yes |
| FV3 | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| FV4 | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| FV5 | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| FV6 | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| FV7 | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| FV8 | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| FV9 | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| FV10 | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| FV11 | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| FV12 | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| FV13 | No | No | No | No | No | No | No | No | No | Yes |
| FV14 | Yes | No | No | Yes | Yes | No | No | No | No | Yes |
| FV15 | Yes | No | No | Yes | No | No | No | No | No | Yes |
| FV16 | Yes | No | No | No | No | No | No | No | No | Yes |

Table 8: Simulation parameters.

| Parameters | Description |
|---|---|
| OS | Ubuntu-20.04.1 |
| Hardware | Dell 5110, Core i5, 4 GB RAM |
| Tool | NS 3 2.29 |
| No. of $V$ | 30 |
| No. of $R$ | 10 |
| No. of fog | 5 |
| Mobility of $V$ | 20 m/s (no pause) |
| Mobility model | Random |
| Environment area | $300 * 1500$ M |
| Loss model | Two-ray ground loss |
| Transmit power | 7.5 dBm |
| Routing protocol | AODV-DSDV-OLSR |
| MAC | IEEE 802.11 |
| Wireless protocol | 802.11 p |
| Communication range of $R$ to $V$ | 145 M |
| Simulation scenario | Highway |
| Simulation time | 300 seconds |

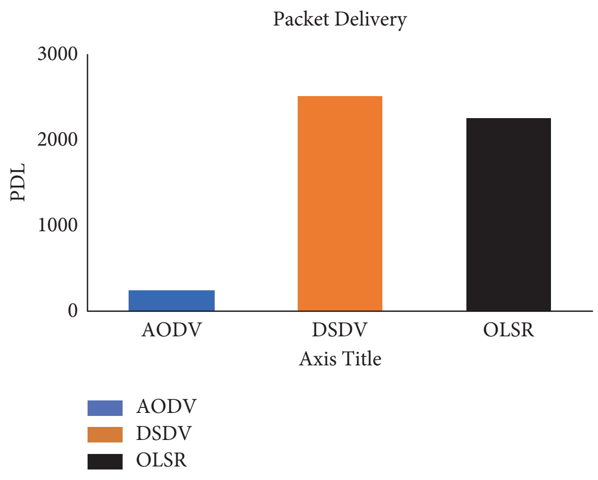


Figure 18: Comparison of packet delivery.

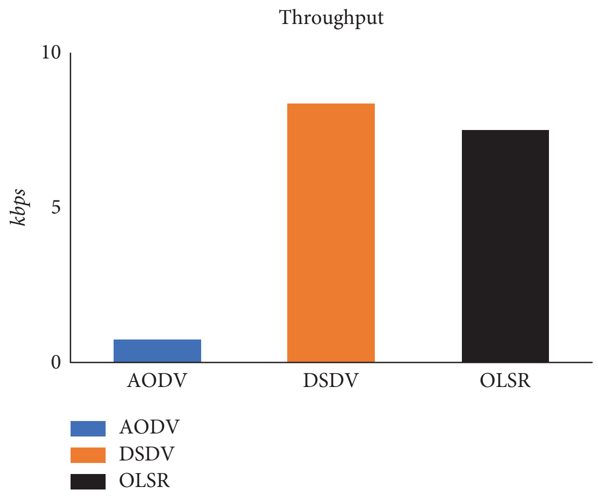


Figure 19: Comparison of throughput.

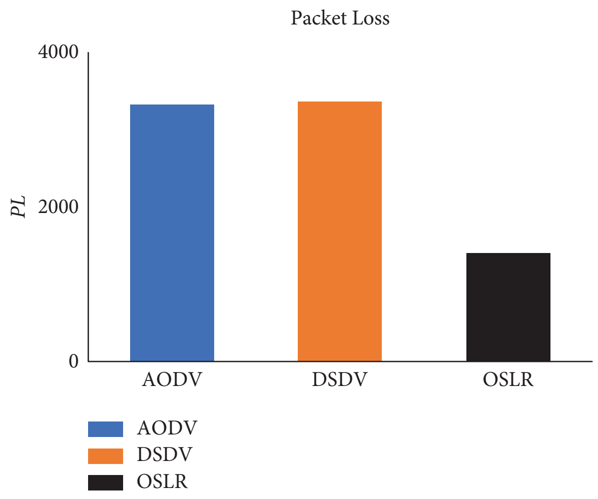


Figure 20: Comparison of packet loss.

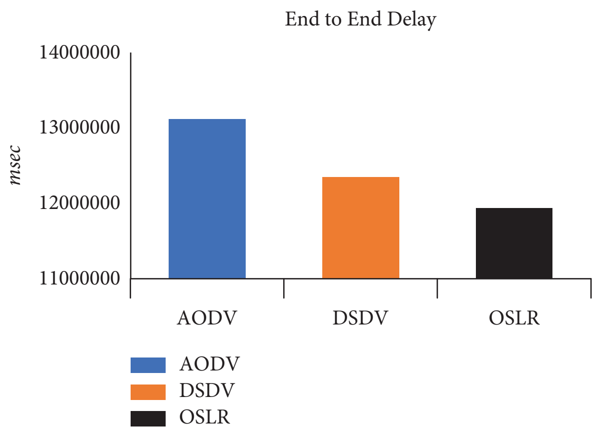


Figure 21: Comparison of end-to-end delay.

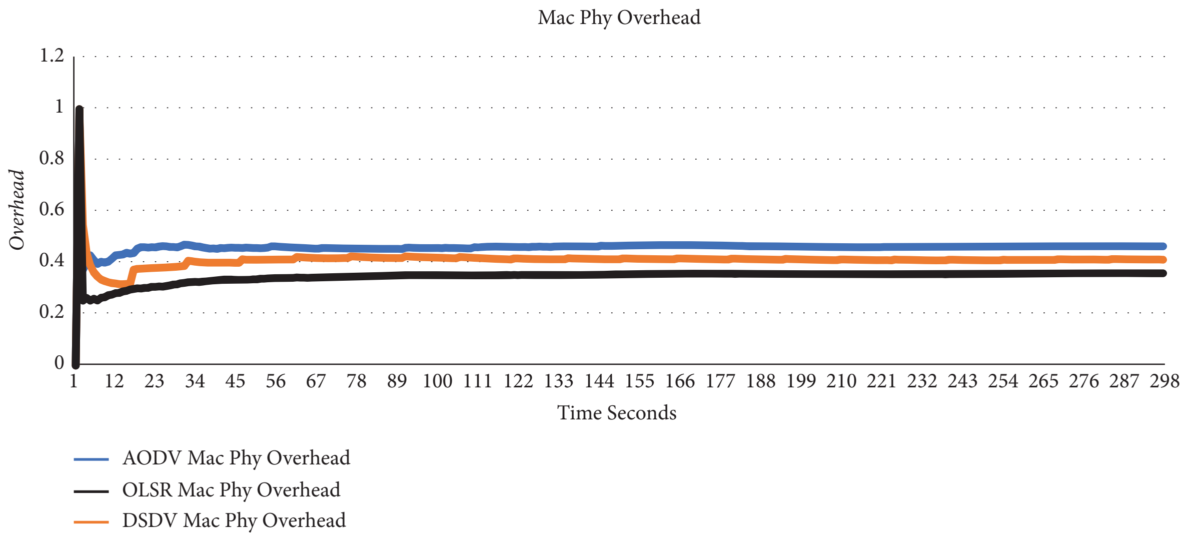


Figure 22: Comparison of MAC/PHY overhead.

to the results, it is impossible to say precisely, which routing protocol works best for the SAIFC scheme.

For this reason, we have measured the overhead of routing protocols. The results show that OLSR, DSDV, and AODV protocols have the lowest overhead and are suitable for the SAIFC OLSR protocol scheme. Figure 22 shows the overhead comparison.

## 8. Conclusion

We have dealt with an important emerging research topic to secure authentication between IOV and fog computing. We propose an HTTP-based secure mutual authentication scheme for IOV-fog-based, which sends data for authentication via a cookie. We used informal and AVISPA for the security analysis SAIFC scheme; the security analysis results show that the SAIFC resists famous attacks. The performance analysis of the SAIFC with other protocols about the number of bits, computation, and communication cost shows that the cost of computation and communication has increased in the SAIFC. We simulated the SAIFC scheme with the tool NS3 and then compared the AODV, DSDV, and OLSR routing protocols. Among the routing protocols compared, OLSR is more efficient. The results show that the SAIFC scheme works well on highways with the OLSR routing protocol and can be used in applications related to the exchange of information in fog-based environments. In future work, the SAIFC scheme can be developed based on the blockchain system, and a lightweight scheme can be reached by reducing communication and computing costs.

## Acronyms

| | |
|---|---|
| AVISPA: | Automated validation of internet security protocols and applications |
| IOT: | Internet of things |
| IOV: | Internet of vehicles |
| RSU: | Roadside unit |
| V2V: | Vehicles to vehicles |
| V2R: | Vehicles to roadside unit |
| R2F: | Roadside unit to fog |
| F2F: | Fog to fog |

| | |
|---|---|
| F2C: | Fog to cloud |
| HLPSL: | High-level protocol specification language |
| OFMC: | On-the-fly model-checker |
| CL-ATSE: | CL-based attack searcher |
| SATMC: | SAT-based model-checker |
| TA4SP: | Tree automata-based protocol analyser |
| AKE: | Authentication and key exchange |
| AKM: | Authenticated key management |
| ECC: | Elliptic curve cryptography |
| EC: | Elliptic curves |
| DL: | Discrete logarithm |
| HF: | Hash function |
| PKE: | Public key encryption |
| PKD: | Public key decryption |
| SKE: | Symmetric key encryption |
| SKD: | Symmetric key decryption |
| RTA: | Rainbow table attack. |

## Data Availability

The data used to support this novel scheme are included within the article.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] S. Sharma and B. Kaushik, "A survey on internet of vehicles: applications, security issues & solutions," *Vehicular Communications*, vol. 20, pp. 100–182, 2019.

[2] Z. Zhang, G. De Luca, B. Archambault, J. Chavez, and B. Rice, "Traffic dataset for dynamic routing algorithm in traffic simulation," *Journal of Artificial Intelligence and Technology*. vol. 2, 2022.

[3] M. Yang, "Research on vehicle automatic driving target perception technology based on improved MSRPN algorithm," *Journal of Computational and Cognitive Engineering*, vol. 1, no. 3, pp. 147–151, 2022.

[4] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine learning techniques," *IEEE Internet of Things Journal*, vol. 4662, no. c, p. 1, 2020.

[5] S. il Hahm, "Reliable real-time operating system for IoT devices," *IEEE Internet of Things Journal*, vol. 4662, no. c, pp. 1–11, 2020.

[6] L. Xu, H. Wang, and T. A. Gulliver, "Outage probability performance analysis and prediction for mobile IoV networks based on ICS-BP neural network," *IEEE Internet of Things Journal*, vol. 4662, no. c, p. 1, 2020.

[7] Y. Salami and V. Khajehvand, "SMAK-IOV: secure mutual authentication scheme and key exchange protocol in fog based IoV," *Journal of Computer and Robotics*, vol. 13, no. 1, pp. 11–20, 2020.

[8] N. Moustafa, B. Turnbull, and K. K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, 2019.

[9] Y. An, F. R. Yu, J. Li, J. Chen, and V. C. M. Leung, "Edge intelligence (EI)-Enabled HTTP anomaly detection framework for the internet of things (IoT)," *IEEE Internet of Things Journal*, vol. 4662, no. c, p. 1, 2020.

[10] H. Hasrouny, C. Bassil, A. E. Samhat, and A. Laouiti, "Group-based authentication in V2V communications," in *Proceedings of the Fifth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pp. 173–177, Beirut, Lebanon, April 2015.

[11] Y. Liu, Y. Wang, and G. Chang, "Efficient privacy-preserving dual authentication and key agreement scheme for secure V2V communications in an IoV paradigm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2740–2749, 2017.

[12] L. Benarous and B. Kadri, "Ensuring privacy and authentication for V2V resource sharing," in *Proceedings of the Seventh IEEE International Conference on Emerging Security Technologies 2017*, pp. 1–6, Canterbury, UK, September 2017.

[13] P. Mohit, R. Amin, and G. P. Biswas, "Design of authentication protocol for wireless sensor network-based smart vehicular system," *Vehicular Communications*, vol. 9, pp. 64–71, 2017.

[14] B. Ying and A. Nayak, "Anonymous and lightweight authentication for secure vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10626–10636, 2017.

[15] J. Liu, Q. Li, R. Sun, X. Du, and M. Guizani, "An efficient anonymous authentication scheme for internet of vehicles," *IEEE International Conference on Communications*, vol. 2018, Article ID 8422447, pp. 16, 2018.

[16] K. Lim and K. M. Tuladhar, "LIDAR: lidar information based dynamic V2V authentication for Roadside infrastructure-less vehicular networks," in *Proceedings of the 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, Las Vegas, NV, USA, January 2019.

[17] C. M. Chen, B. Xiang, Y. Liu, and K. H. Wang, "A secure authentication protocol for internet of vehicles," *IEEE Access*, vol. 7, no. c, pp. 12047–12057, 2019.

[18] H. Vasudev, V. Deshpande, D. Das, and S. K. Das, "A lightweight mutual authentication protocol for V2V communication in internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6709–6717, 2020.

[19] S. Kalra and S. K. Sood, "Secure authentication scheme for IoT and cloud servers," *Pervasive and Mobile Computing*, vol. 24, pp. 210–223, 2015.

[20] S. Kumari, M. Karuppiah, A. Kumar, D. Xiong, L. Fan, and N. Kumar, "A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers," *The Journal of Supercomputing*, vol. 74, pp. 6428–6453, 2017.

[21] M. Wazid, P. Bagga, A. K. Das, S. Shetty, J. J. P. C. Rodrigues, and Y. Park, "Akm-IoV: authenticated key management protocol in fog computing-based internet of vehicles deployment," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8804–8817, 2019.

[22] E. R. Sykes and W. Skoczen, "An improved parallel implementation of RainbowCrack using MPI," *Journal of Computer Science*, vol. 5, no. 3, pp. 536–541, 2014.

[23] Y. Glouche, T. Genet, and E. Houssay, "SPAN--a Security Protocol ANimator for AVISPA--User Manual," *IRISA/Université de Rennes*, vol. 1, p. 20, 2006.

[24] Y. Salami, Y. Ebazadeh, and V. Khajehvand, "Cost-effective secure key exchange scheme in Fog Federation," *Iran J. Comput. Sci.* vol. 4, no. 3, pp. 1–13, 2021.

[25] I. Konnov, *Handbook of Model Checking* Springer International Publishing AG, Cham, Switzerland, 2018.

[26] Y. Salami and V. Khajehvand, "LSKE: lightweight secure key exchange scheme in fog federation," *Complexity*, vol. 2021, Article ID 4667586, 2021.

[27] A. Gotsman, F. Massacci, and M. Pistore, "Towards an independent semantics and verification technology for the hlpsl," *Electronic Notes in Theoretical Computer Science*, vol. 135, pp. 59–77, 2005.

[28] M. Wazid, A. K. Das, N. Kumar, and A. V. Vasilakos, "Design of secure key management and user authentication scheme for fog computing services," *Future Generation Computer Systems*, vol. 91, pp. 475–492, 2019.

[29] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[30] D. Basin, S. Mödersheim, and L. Viganò, "An on-the-fly model-checker for security protocol analysis," *Computer Security -- ESORICS 2003*, vol. 2808, pp. 253–270, 2003.

[31] M. Turuani, "The CL-atse protocol analyser," in *Term Rewriting and Applications*, F. Pfenning, Ed., pp. 277–286, Springer, Berlin Germany, 2006.

[32] A. Biere and D. Kröning, "SAT-based model checking," in *Handbook of Model Checking*, E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds., Springer International Publishing, Berlin Germany, pp. 277–303, 2018.

[33] L. Vigan, "Automated security protocol analysis with the avispa tool 1," *Electronic Notes in Theoretical Computer Science*, vol. 155, no. 1, pp. 61–86, 2006.

[34] Y. Salami, V. Khajehvand, and E. Zeinali, "E3c: a tool for evaluating communication and computation costs in authentication and key exchange protocol," 2022, https://arxiv.org/abs/2212.03308.