Vehicular Communications ••• (••••) ••••••

Contents lists available at ScienceDirect

Vehicular Communications

www.elsevier.com/locate/vehcom

RSEAP2: An enhanced version of RSEAP, an RFID based authentication protocol for vehicular cloud computing

Masoumeh Safkhani [a,*], Carmen Camara [b], Pedro Peris-Lopez [b], Nasour Bagheri [c,d]

[a] Computer Engineering Department, Shahid Rajaee Teacher Training University, Tehran 16788-15811, Iran
[b] Computer Science Department, Carlos III University of Madrid, Madrid 28911, Spain
[c] Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran 16788-15811, Iran
[d] School of Computer Science (SCS), Institute for Research in Fundamental Sciences (IPM), Farmanieh Campus, P.O. Box: 19538 - 33511, Tehran, Iran

ARTICLE INFO

Article history:
Received 27 May 2020
Received in revised form 16 August 2020
Accepted 17 October 2020
Available online xxxx

Keywords:
Vehicular cloud computing
Authentication
Elliptic curve based cryptography
Security analysis
Tag/reader impersonation
Distance bounding attacks

ABSTRACT

RSEAP is a recently proposed RFID based authentication protocol for vehicular cloud computing whose authors claimed to be secure and efficient. In this article, we challenge these claims. More precisely, we show that RSEAP does not provide the desired security, and it is possible to conduct both tag and reader impersonation attacks efficiently. Besides, despite the use of timestamps, we show how this protocol also suffers from a range of relay attacks. The complexity of any of the proposed attacks is negligible while the success probability is maximum (i.e., the adversary's success probability is '1' since all the proposed attacks are deterministic). To improve the security of RSEAP scheme, we suggest the required patches for fixing the security vulnerabilities mentioned above. We show that the improved protocol, called RSEAP2, is more efficient (computation and communication costs) than the original RSEAP, while provides a higher security level. The security of RSEAP2 is evaluated informally and also formally using the Scyther tool, which is a well-known and automated tool to assess the security of cryptographic protocols. Additionally, we have formally verified the security of the proposed scheme under the Real-or-Random oracle model.

1. Introduction

Vehicular cloud computing (VCC) is a new paradigm for transferring information in a typical network of vehicles that attracted many researchers' interest. With this technology, vehicles can interact, collaborate, and share their resources which can be used to sense the environment, process data, or propagate the results [26]. VCC is a sort of intelligent transportation system and has been developed to overcome the drawbacks of its predecessor, i.e. vehicular Ad-Hoc networks (VANET), and is an extension of mobile cloud computing [2]. In a typical VCC, vehicles are equipped with communication sensing capacities, and VCC employs cloud infrastructures, Internet of Things (IoT), vehicular networking, and vehicle resources technologies to provide real-time computational facilities for the constrained devices that are equipped on the vehicles. Thanks to these technologies, it is possible to offer many services for drivers and passengers, and many of them could be based on mobile applications. Examples of facilities that are provided by VCC could be travel planning, avoiding accidents, traffic management, finding nearest roadside infrastructures and messaging, Internet access, and infotainment for vehicle occupants [8,22]. In the context of low cost-devices, an RFID based VCC ecosystem is depicted in Fig. 1, including for example vehicle to vehicle (V2V) communication, vehicles to road-infrastructure (V2I) communication, vehicles-to-sensors (V2S) communication and also communications with cloud infrastructures.

1.1. Related works

Despite many advantages of VCC, it has its challenges among which security and privacy are a major concern [39]. This matter is even more, as instant location-based services (LBS) are becoming very popular and widespread. Nevertheless, to provide a service based on the user's interest and location, we need some information that could compromise his privacy. This is the case for the other potentials of VCC, such as its connection to healthcare and infotainment [8]. Also, to provide services without a high cost, many equipped devices on the edge side of VCC should be low-cost equipment such as RFID tags or mobile devices that transfer sensitive information over a public channel. Given that many of

* Corresponding author.
E-mail addresses: Safkhani@srttu.edu (M. Safkhani), macamara@pa.uc3m.es (C. Camara), pperis@inf.uc3m.es (P. Peris-Lopez), Nbagheri@sru.ac.ir (N. Bagheri).

https://doi.org/10.1016/j.vehcom.2020.100311
2214-2096/

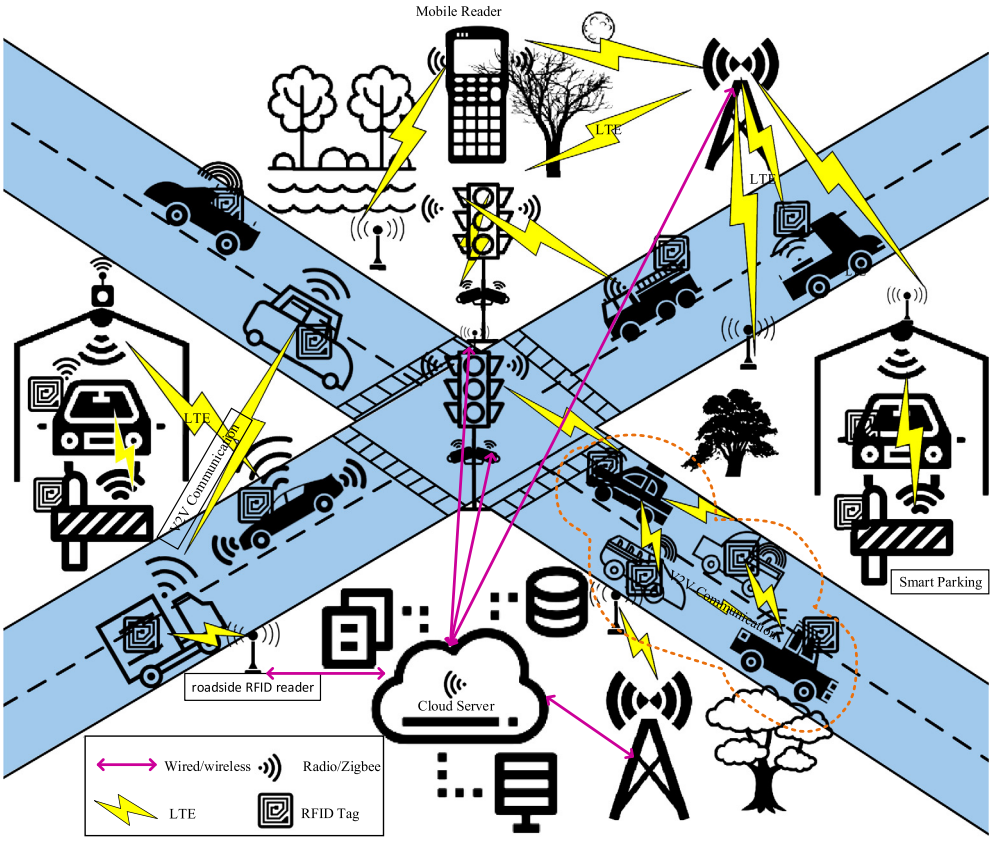

Fig. 1. An RFID based vehicular cloud computing ecosystem (thanks to https://thenounproject.com for the icons in all pictures of this paper).

them are constrained devices, they cannot support on-board common security protocols such as TLS to provide the desired security for the transferred data. Many researchers recently have tried to tackle the security challenges of mobile cloud computing, which might be directly used in VCC by proposing efficient solutions. To mention some, we highlight Yan et al. [41], Tsai and Lo [35], He et al. [16], Sharma et al. [31], Wang et al. [37], Liu et al. [25], Jiant et al. [19], Shi et al. [32] and Choi et al. [10] schemes. For instance, Tsai and Lo [35] introduced an authentication scheme for distributed mobile cloud computing services. The scheme builds on bilinear pairing cryptosystem, dynamic nonce generation and a trusted smartcard generator service. Nevertheless, He et al. [16] pointed out the security weaknesses of Tasai and Lo's scheme and also proposed a privacy-aware authentication scheme for mobile cloud computing services. Vijayakuma et al. [36] proposed a dual authentication and key management protocol to be used in vehicular Ad-Hoc networks. Later, Tan et al. [34] stated that their protocol suffers from replay attacks. However, Azees [6] showed that the Vijayakumar et al.'s protocol is secure against replay attacks by providing appropriate and reasoned explanations.

Some other works concentrated on designing a specific solution for VCC. For example, Wazid et al. [38] recently proposed an authenticated key management protocol for VCC. However, later Saleem et al. [30] showed how it does not provide acceptable security by introducing several impersonation attacks. Jiang et al. [19] introduced an integrated authentication and key agreement (AKA) protocol for VCC. Their proposed scheme is a three-factor identity-based key establishment protocol. Jiang et al. [20] also recently proposed a biometric-based three-factor authentication and key agreement for VCC. Among the last solutions could be SFVCC which is a chaotic map-based security protocol for vehicular cloud computing [27], proposed by Mishra et al. In this research, the authors proposed a mutual authentication protocol for VCC to ensure security and anonymity for the users. In this direction, Kumar et al. also recently presented an RFID based mutual authentication and key agreement protocol for VCC [22], which is called RSEAP. In this protocol, which could also be considered as a scheme for mobile cloud computing, all the messages are passed over a public channel. Hence, to ensure security and privacy of the protocol's entities, the authors employed elliptic curve-based cryptography, one-way hash functions, and also timestamps in each transaction. Based on their security analysis and computational comparison results, RSEAP is more efficient or comparable with their predecessors' protocols. In this article, we evaluate the security of this scheme in more depth, and it is the first third-party security analysis to the best of our knowledge. Apart from well-known attacks such as impersonation attacks, we also evaluate the security of the protocol against relay attacks which are critical in the context of mobile computing and VCC. Our security analysis concerning relay attacks demonstrates dangerous pitfalls on the security of RSEAP scheme despite the merits of this contribution. It worth noting that relay attacks belong to distance bounding attacks in which an adversary aims to cheat a verifier into thinking that the prover is located within a valid physical vicinity (neighbourhood area) while he is not. We urge the reader to consult [5] where Avoine et al. presented a survey about distance-bounding principles, protocols and threats.

1.2. Our contribution

The main contribution of this paper is twofold:

- First, to the best of our knowledge, we present the first third-party security analysis of RSEAP, an RFID based authentication protocol for vehicular cloud computing. Our security analysis demonstrates important security pitfalls in this protocol,

Table 1
Used notations.

Notations			
S	The server	T_i	The i-th RFID tag
R_j	The j-th RFID reader	$\mathcal{A}$	The adversary
U	a client, which could be the tag or the reader	ID_T	The tags identification number
g	A generator point of a large group G	x_S	The server's ECC encryption private key
pw_T	The T_i password	sn_i	A serial number for T_i, generated by S
$h(.)$	One-way hash function	$\oplus$	Bitwise XOR operation
$P+Q$	Addition of two points on the elliptic curve E, results in another point from the curve	$a.P$	Multiplying a point P on the elliptic curve E by natural number (scalar) a, results in another point on the curve
$\|\|$	Concatenation	$A \stackrel{?}{=} B$	Determine whether A and B are equal
SK_{ST}/SK_{TS}	The shared session key between T_i and S	TS_R	A timestamp which is used in the registration phase of the protocol
TS_{LA}	A timestamp which is used in the login and authentication phase of the protocol	$\mathbb{F}_q$	A finite field with q elements $\{0,1,\ldots,q-1\}$ where q is a prime number
ΔT	A threshold for time	$\|X\|$	The cardinality of the set X
$\mathbb{F}_q^*$	The set of integers $\{1,\ldots,q-1\}$, i.e. $\mathbb{F}_q/\{0\}$	$\mathbb{F}_{2^m}$	The finite field with 2^m elements, also known as $GF(2^m)$ (Galois field)

including a very efficient and effective relay attack with an adversary's success probability of '1'.

- Given that RSEAP has some merits, e.g., it outperforms previous related protocols in terms of computation and communication costs, we have kept its core and patched its security flaws. We have assessed the security of this enhanced protocol, called RSEAP2, from informal and formal perspectives. For the formal analysis, we have used the Scyther tool, which is an automated tool to evaluate the security of a cryptographic protocol formally. The security and performance analyses of RSEAP2 show how this proposal is a more efficient and more secure scheme compared with its predecessor, i.e. RSEAP.

1.3. Organization

The remainder of this paper is structured as follows: In section 2, we provide the required preliminaries, including notations, system model, and description of RSEAP scheme. In section 3, we present the security analysis of RSEAP and show several efficient and powerful attacks against this proposal. To patch RSEAP, in section 4, we introduce RSEAP2. Security analysis of RSEAP2 is described in section 5. In section 6, we compare RSEAP and RSEAP2 in terms of performance and security. Finally, we conclude the paper in section 7.

2. Preliminaries

In this section, we introduce notations, the system model, elliptic curve-based cryptography, and also provide a brief description of RSEAP protocol.

2.1. Notations

The listed notations, used in this paper, are represented in Table 1.

2.2. System model

It is worth noting that in this paper, we evaluate RSEAP and enhance its security and efficiency, instead of starting from scratch and design a new model. Our motivation is the merits that RSEAP has [22]. Hence, we follow the same system model, but, we revise its adversarial model to make it more realistic. More precisely, we consider two types of adversaries. The first kind is a **local adversary**, who can just eavesdrop messages, impersonate a protocol's entity or modify the transferred messages if a valid tag (resp. reader) is in the vicinity of a legitimate reader (resp. server) and communications occur over a public channel. This adversary can control the channel between T_i, R_j, and S and can store, block or modify any transferred message between them or initiate a new session with T_i, R_j or S. The second type of adversary is a **distance bounding adversary** which is aiming to make a relay attack. This adversary is a man-in-the-middle that uses one or more relay devices to trick two remote devices into thinking they are close enough. The aim of the adversary, as a prover, in a relay attack is to cheat a legitimate verifier/reader into believing that for example, an honest RFID tag is within the reader vicinity area when it is not the case. [28]. Hence, besides the basic attacks that were considered in RSEAP, our model also includes *relay attacks*. For the details of our proposed system model in RSEAP2, the interested reader is referred to [22, Sec. 3].

2.3. Elliptic curve cryptography

Elliptic Curve Cryptography (ECC) is a sort of public-key cryptography based on a mathematical-group and defined over an elliptic curve. A benefit of ECC could be its smaller key size compared to other public-key systems, which is a critical factor for constrained environments [29]. For example, the security level of ECC-256, which provides 128 bits security level, equates to the security level of RSA-3072 [22].

An elliptic curve E is defined formally as follows [17]:

Definition 1. An elliptic curve E is the set of solutions to a Weierstrass equation $E: y^2 = x^3 + ax + b$, along with a distinguished point at infinity which is denoted by $\mathcal{O}$, where the constants a and b must satisfy $4a^3 + 27b^2 \ mod \ q \neq 0$. Assuming P and Q are two points on E, the addition law on E is defined as follows. Let L the line which connects P and Q, or the tangent line to E at P when $P = Q$. Then the intersection of E and L consists of three points P, Q, and R. Denoting $R = (a, b)$, the sum of P and Q is defined to be the reflection $R' = (a, -b)$ of R across the x-axis and it is denoted by $P + Q$. Similarly, the multiplication of a point P by an integer a is performed as repeated addition as $a.P = \underbrace{P + P + \ldots + P}_{a}$.

For cryptographic purposes, we need to focus on elliptic curves whose points have coordinates in a finite field. Let q be a large prime number. An elliptic curve $E_{\mathbb{F}_q}$ over the finite field $\mathbb{F}_q$ is defined as the set of all $(\lambda, \mu) \in \mathbb{F}_q \times \mathbb{F}_q$ such that $\lambda^2 = \mu^3 + a\mu + b$, where $a, b \in \mathbb{F}_q$ and $4a^3 + 27b^2 \ mod \ q \neq 0$. Then $G = \{(\lambda, \mu) \in E_{\mathbb{F}_q} \cup \mathcal{O}, +\}$ is a group. It is also possible to define an elliptic curve over the finite field $\mathbb{F}_{2^m}$.

If there is an element $g \in G$ that its different orders can generate all elements of the group, G is called a cyclic group and

g is called a generator. In general, the order of any $g \in E_{\mathbb{F}_q}$ is denoted as the smallest positive number n such that $n.g = \mathcal{O}$. Assuming that n is enough large, given any natural value $a \in \mathbb{F}_q$ and $g = \{(\lambda, \mu) \in E_{\mathbb{F}_q}\}$ of order n, it is easy to calculate $y = a.g$. However, given y, $E_{\mathbb{F}_q}$ and g, it is computationally infeasible to determine a, which is known as Elliptic Curve Discrete Logarithm Problem (ECDLP). Similarly, for $a, b \in \mathbb{F}_q$, given $a.g$, $b.g$, $E_{\mathbb{F}_q}$ and g, it is computationally infeasible to determine $a.b.g$, which is known as Elliptic Curve Computational Diffie-Hellman Problem (EC-CDHP) [22,29].

2.4. RSEAP protocol

In this section, we provide a brief description of RSEAP [22]. In this protocol, the tag T_i and the cloud database server S communicate via the reader R_j, to establish a session key SK_{ST}. It includes two phases. The first phase is the tag enrollment or initialization phase, where the tag communicates with S over a secure channel to share the required information. The second phase of the protocol is the login and authentication phase, which is used to do mutual authentication and share the session key $SK_{ST} = SK_{TS}$. This phase of the scheme occurs over a public channel.

In the initialization phase of RSEAP, the server S chooses an elliptic curve $E(\mathbb{F}_q)$ over $\mathbb{F}_q^*$ and a generator g over G. It also selects $x_s \in \mathbb{F}_q$ as its secret key and its public key will be $x_s.g$. Any tag T_i which aims to register with S, inputs its ID_T and pw_T, generates a random value $R_T \in \mathbb{F}_q^*$, computes $PWT = h(ID_T \| pw_T \| R_T)$ and sends the tuple $M_{R1} = \{PWT, ID_T, TS_{R1}\}$ to S. Once S received M_1, verifies the timestamp, i.e. $TS_{R2} - TS_{R1} \stackrel{?}{\leq} \Delta T$ at the first. Next, it generates $sn_i \in \mathbb{F}_q^*$ and sets it as the T_i's serial number, computes $X_T = h(sn_i \| ID_T \| x_S)$, $A_T = h(PWT \| X_T \| ID_T)$, $B_T = X_T \oplus PWT$, and stores sn_i corresponding to ID_T. It then sends tuple $M_{R2} = \{A_T, B_T, sn_i, g, x_s.g, G, h(.)\}$ to T_i. The tag T_i stores $\{A_T, B_T, sn_i, g, x_s.g, G, h(.)\}$.

In the login and authentication phase of the protocol, see Fig. 2, the already registered tag T_i, which wants to communicate with S, logs on by its (ID_T, pw_T, R_T), computes $PWT = h(ID_T \| pw_T \| R_T)$, $X_T^* = B_T \oplus PWT$ and $A_T^* = h(PWT \| X_T^* \| ID_T)$. Then it verifies whether $A_T^* \stackrel{?}{=} A_T$. Assuming the verification passed correctly, T_i generates $a \in \mathbb{F}_q^*$, computes $a.g$ and $W_1 = h(X_T^* \| a.g \| ID_T)$ and sends $M_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA1}\}$ to the reader R_j. The reader checks the timestamp, i.e. $TS_{LA2} - TS_{LA1} \stackrel{?}{\leq} \Delta T$, and then sends $M_2 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA3}\}$ to S. Once S received M_2, it verifies the timestamp, i.e. $TS_{LA2} - TS_{LA3} \stackrel{?}{\leq} \Delta T$, and then computes $ID_T^* = ID_T \oplus a.x_s.g \oplus a.x_s.g$, retrieves related sn_i, computes $X_T^* = h(sn_i \| ID_T^* \| x_s)$ and $W_1 = h(X_T^* \| a.g \| ID_T)$ and checks $W_1^* \stackrel{?}{=} W_1$. Next, it generates $b \in \mathbb{F}_q^*$, computes $SK_{ST} = h(ID_T^* \| X_T \| bag \| x_s.a.g \| sn_i \| TS_{LA5})$ and $W_2 = h(ID_T^* \| SK_{ST} \| b.g \| TS_{LA5})$ and sends $M_3 = \{W_2, b.g, TS_{TL5}\}$ to R_j, where SK_{ST} is the shared session key. R_j just verifies the timestamp, i.e. $TS_{LA6} - TS_{LA5} \stackrel{?}{\leq} \Delta T$, and then sends $M_4 = \{M_3, TS_{LA7}\}$ to T_i. Similarly, T_i checks the timestamp, i.e. $TS_{LA8} - TS_{LA7} \stackrel{?}{\leq} \Delta T$, and then computes $SK_{TS} = h(ID_T \| X_T^* \| a.b.g \| a.x_s.g \| sn_i \| TS_{LA5})$ and verifies $W_2^* \stackrel{?}{=} h(ID_T \| SK_{TS} \| b.g \| TS_{LA5})$, to set SK_{TS} as the session key.

3. Security analysis of RSEAP

In this section, we evaluate the security of RSEAP. Given that the login and authentication phase happens over the public channel, the transferred messages between protocol's parties are accessible by the adversary. Our security analysis is based on the observations described below:

1. The exchanged message M_1 does not guarantee the integrity of TS_{LA1}. Hence, any active man in the middle adversary can change the TS_{LA1} fraction of the transferred $M_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA1}\}$ without being detected.
2. The transferred message M_4 does not guarantee the integrity of TS_{LA7}. Consequently, an active man in the middle adversary can change the TS_{LA7} fraction of the exchanged $M_4 = \{M_3, TS_{LA7}\}$ without being detected.
3. R_j has no shared key, neither with T_i nor with S, and cannot be authenticated by them.

Following the observations mentioned above, we propose several attacks against RSEAP: 1) tag impersonation attack; 2) several relay attacks; 3) reader impersonation attack; and 4) finally a distance fraud attack.

3.1. Tag impersonation attack

Based on the first observation, the adversary can impersonate the tag T_i to the reader and server. More precisely, the adversary eavesdrops a message $M_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA1}\}$ which is transferred from T_i to R_j, over a public channel, in a session between T_i and R_j/S. Hence after, the adversary can impersonate T_i by determining the current timestamp, e.g. T'_{LA1}, and sending $M'_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, T'_{LA1}\}$ to R_j. Given that neither R_j nor S can detect the T'_{LA1} integrity, the adversary will be authenticated as a legitimate T_i and the server establishes the session key. Although the adversary will not be able to extract the shared session key, however, it could impersonate T_i successfully, which compromises the claimed security by the protocol designers [22, Sec. 5.3].

3.2. Relay attack against the tag and reader

The adversary can use the first and the second observations to make a relay attack against the tag and the reader that are not in an acceptable distance from each other (out of the vicinity area). As described in Fig. 3, the attacker follows the steps outlined below:

1. Assuming the legitimate T_i, which is not in the acceptable vicinity of the legitimate R_j, sends its request as $M_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA1}\}$ to R_j.
2. Using the fake reader R_f, which is in the T_i's vicinity, the adversary relays this message to the phoney tag T_f, which is in the R_j's vicinity.
3. T_f replaces the TS_{LA1} of M_1, e.g. by T'_{LA1}, and sends $M'_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, T'_{LA1}\}$ to R_j.
4. R_j and S will authenticate T_i. In response to the T_f query R_j sends $M_4 = \{M_3, TS_{LA7}\}$ to T_i, where $M_3 = \{W_2, b.g, TS_{LA5}\}$, $SK_{ST} = h(ID_T^* \| X_T \| bag \| x_s.a.g \| sn_i \| TS_{LA5})$ and $W_2 = h(ID_T^* \| SK_{ST} \| b.g \| TS_{LA5})$.
5. T_f relays M_4 to R_f.
6. R_f replaces TS_{LA7} fraction of M_4 by an acceptable value, e.g. T'_{LA7}, and sends M'_4 to T_i.
7. T_i verifies the timestamp, i.e. $TS_{LA8} - T'_{LA7} \stackrel{?}{\leq} \Delta T$ which it is, and then computes

   $$SK_{TS} = h(ID_T \| X_T^* \| a.b.g \| a.x_s.g \| sn_i \| TS_{LA5}),$$

   verifies $W_2^* \stackrel{?}{=} h(ID_T \| SK_{TS} \| b.g \| TS_{LA5})$ and sets SK_{TS} as the session key.

Based on the above attack, T_i and S will successfully share the session key $SK_{TS} = h(ID_T \| X_T^* \| a.b.g \| a.x_s.g \| sn_i \| TS_{LA5})$, believing

Tag T_i	**Reader** R_j	**Cloud database server** S
Logs on by (ID_T, pw_T, R_T), computes $PWT = h(ID_T \| pw_T \| R_T)$, $X_T^* = B_T \oplus PWT$ and $A_T^* = h(PWT \| X_T^* \| ID_T)$. If $A_T^* \stackrel{?}{=} A_T$, generates $a \in \mathbb{F}_q^*$, computes $W_1 = h(X_T^* \| a.g \| ID_T)$ and sends $M_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA1}\}$ to R_j		
$\xrightarrow{M_1}$		
	If $TS_{LA2} - TS_{LA1} \stackrel{?}{\leq} \Delta T$, sends $M_2 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA3}\}$ to S	
	$\xrightarrow{M_2}$	
		if $TS_{LA2} - TS_{LA3} \stackrel{?}{\leq} \Delta T$, computes $ID_T^* = ID_T \oplus a.x_s.g \oplus a.x_s.g$, retrieves sn_i, computes $X_T^* = h(sn_i \| ID_T^* \| x_s)$ and $W_1 = h(X_T^* \| a.g \| ID_T)$. If $W_1^* \stackrel{?}{=} W_1$ it generates $b \in \mathbb{F}_q^*$, computes $SK_{ST} = h(ID_T^* \| X_T \| bag \| x_s.a.g \| sn_i \| TS_{LA5})$ and $W_2 = h(ID_T^* \| SK_{ST} \| b.g \| TS_{LA5})$ and sends $M_3 = \{W_2, b.g, TS_{TL5}\}$ to R_j
		$\xleftarrow{M_3}$
	If $TS_{LA6} - TS_{LA5} \stackrel{?}{\leq} \Delta T$ sends $M_4 = \{M_3, TS_{LA7}\}$ to T_i	
$\xleftarrow{M_4}$		
If $TS_{LA8} - TS_{LA7} \stackrel{?}{\leq} \Delta T$ computes $SK_{TS} = h(ID_T \| X_T^* \| a.b.g \| a.x_s.g \| sn_i \| TS_{LA5})$ and verifies it by $W_2^* \stackrel{?}{=} h(ID_T \| SK_{TS} \| b.g \| TS_{LA5})$.		

Fig. 2. The login and authentication phase of RSEAP, over a public channel.

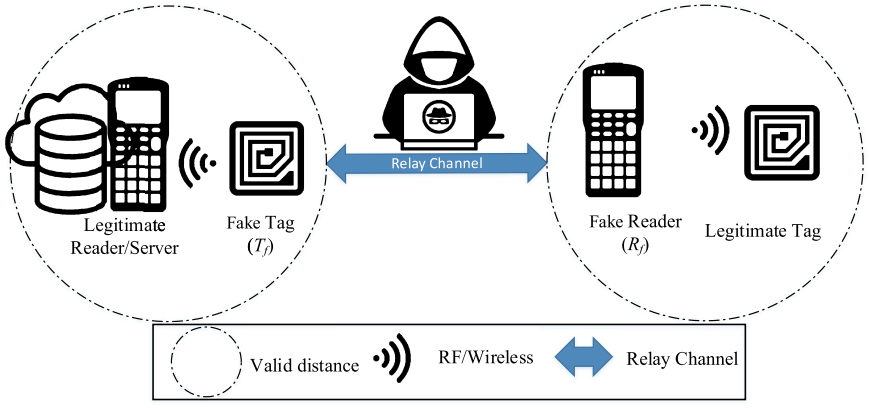

Fig. 3. Representation of the relay attack between the tag and the reader.

they are in the acceptable vicinity of each other while they are not. Hence, the adversary successfully and efficiently made a relay attack. Therefore, after this attack, an adversary can also transfer the encrypted messages between T_i and R_j/S. For example, if this protocol is used for communication between a car and a remote key, the attacker could quickly start that mentioned car.

3.3. Relay attack against the server, reader and tag

Given that protocol designers considered transmission among the readers and the database server to be also over a public channel [22, Sec. 3.2], the adversary can thus make almost similar relay attack against the server, the reader and the tag as well. More precisely, following Fig. 4, we consider a general case where the

reader, the tag and the server are not within an acceptable distance from each other, and the adversary aims to perform both a successful authentication and key establishment using a relay attack. The attack procedure is as follows:

1. The legitimate tag T_i, which is not in the acceptable vicinity of the legitimate R_j, sends its login request $M_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA1}\}$ to R_j.
2. Using a fake reader R_{f1}, which is in the T_i's neighbourhood, the adversary receives this message and relays this message to the phoney tag T_f, which is in the valid vicinity of the legitimate R_j.
3. T_f modifies the TS_{LA1} of M_1, e.g. T'_{LA1}, and sends $M'_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, T'_{LA1}\}$ to R_j.
4. R_j verifies TS_{LA1} and sends $M_2 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA3}\}$ to S.
5. The fake server S_f, located in the R_j's neighbourhood, could be the same device that also plays the role of T_f. It receives M_2 and relays it to the fake reader R_{f2}, which is in the S vicinity.
6. R_{f2} modifies the TS_{LA3} of M_2, e.g. T'_{LA3}, and sends $M'_2 = \{ID_T \oplus a.x_s.g, a.g, W_1, T'_{LA3}\}$ to S.
7. S verifies T'_{LA3}, authenticates the tag T_i, and sends $M_3 = \{W_2, b.g, TS_{LA5}\}$ to R_j, where

   $SK_{ST} = h(ID^*_T \| X_T \| bag \| x_s.a.g \| sn_i \| TS_{LA5})$ and

   $W_2 = h(ID^*_T \| SK_{ST} \| b.g \| TS_{LA5})$.
8. R_{f2} receives M_3 and relays it to S_f.
9. S_f modifies the TS_{LA5} of M_3, e.g. T'_{LA5}, sends $M'_3 = \{W_2, b.g, T'_{LA5}\}$ to R_j and relays $M_3 = \{W_2, b.g, TS_{LA5}\}$ to R_{f1}.
10. R_j verifies T'_{LA5} and sends $M_4 = \{M'_3, TS_{LA7}\}$ to T_i.
11. T_f receives M_4 and relays it to R_{f1}.
12. Given M_3 and M_4 were relayed by S_f and T_f respectively, R_{f1} modifies the TS_{LA7}, e.g. T'_{LA7}, uses M_3, and sends $M'_4 = \{M_3, T'_{LA7}\}$ to T_i.
13. T_i verifies T'_{LA7}, computes

    $SK_{TS} = h(ID_T \| X^*_T \| a.b.g \| a.x_s.g \| sn_i \| TS_{LA5})$,

    verifies $W^*_2 \stackrel{?}{=} h(ID_T \| SK_{TS} \| b.g \| TS_{LA5})$ and sets SK_{TS} as the session key.

Based on this attack, T_i and S (through R_j) will successfully share the session key $SK_{TS} = h(ID_T \| X^*_T \| a.b.g \| a.x_s.g \| sn_i \| TS_{LA5})$. Besides these entities, wrongly believe they are in the acceptable vicinity of each other while they are not. Since the attack procedure is deterministic, the adversary's success probability of the attack described above is maximum, that is, '1'.

3.4. Reader impersonation attack

In addition to the previous attacks, it should be noted that an adversary can completely bypass R_j. This weakness represents a man-in-the-middle (relay) attack against T_i and S. The attack procedure is as follows:

1. The legitimate tag T_i, which is not in the acceptable vicinity of the legitimate server S, sends its login request $M_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA1}\}$ to R_j.
2. The fake reader R_{f1} relays this message to another phoney reader R_{f2}.
3. Given M_1, R_{f2} computes $M'_2 = \{ID_T \oplus a.x_s.g, a.g, W_1, T'_{LA3}\}$ and sends it to S.
4. S verifies T'_{LA3}, authenticates T_i and sends $M_3 = \{W_2, b.g, TS_{LA5}\}$ to R_j, where

   $SK_{ST} = h(ID^*_T \| X_T \| bag \| x_s.a.g \| sn_i \| TS_{LA5})$ and

   $W_2 = h(ID^*_T \| SK_{ST} \| b.g \| TS_{LA5})$.
5. R_{f2} receives M_3 and relays it to R_{f1}.
6. Given M_3, R_{f1} determines the validity of TS_{LA7}, and sends $M_4 = \{M_3, TS_{LA7}\}$ to T_i.
7. T_i verifies TS_{LA7}, computes

   $SK_{TS} = h(ID_T \| X^*_T \| a.b.g \| a.x_s.g \| sn_i \| TS_{LA5})$,

   verifies $W^*_2 \stackrel{?}{=} h(ID_T \| SK_{TS} \| b.g \| TS_{LA5})$, and sets SK_{TS} as the session key.

Once completed the attack, T_i and S successfully share SK_{TS}, without any form of impact on the legitimate reader (R_j) supplanted. This attack can be catalogued as a reader impersonation attack against T_i and S. The adversary only has to follow the described steps. The success is guaranteed as it is a deterministic procedure. Therefore, the adversary's success probability of this attack is maximum, that is, '1'.

3.5. Distance fraud attack

Finally, it is worth noticing that the role of R_j in RSEAP protocol is to relay messages from T_i to S and vice versa. Expressly, it verifies the transfer time of the received messages and also includes a timestamp in the forwarded messages. Given that this process does not include any secret, based on the third observation, an active adversary can easily emulate it. Hence, it is trivial for an adversary to perform a successful reader impersonation attack. Furthermore, the adversary can program the counterfeited reader such that it accepts the arrival messages even if the timestamps do not satisfy the expected threshold or if the messages include an incorrect timestamp. Given such a fake reader, the adversary can complete a distance-fraud attack (a kind of relay attack) when the tag is not in the acceptable vicinity of the reader. Following Fig. 5, when the counterfeited reader R_f is in the tag's neighbourhood and therefore, outside the server vicinity, the attack procedure is as follows:

1. The legitimate tag T_i, which is not in the acceptable vicinity of the legitimate server S, sends its login request $M_1 = \{ID_T \oplus a.x_s.g, a.g, W_1, TS_{LA1}\}$ to R_j.
2. The fake R_{f1} computes $M'_2 = \{ID_T \oplus a.x_s.g, a.g, W_1, T'_{LA3}\}$ and relays it to S with the help of an adversary or by using enough transmission power. It has to set TS_{LA3} to an acceptable value for S, taking into consideration the actual distance up to S and the relay time.
3. S verifies TS_{LA3}, authenticates T_i and sends $M_3 = \{W_2, b.g, TS_{LA5}\}$ to R_j, where

   $SK_{ST} = h(ID^*_T \| X_T \| bag \| x_s.a.g \| sn_i \| TS_{LA5})$ and

   $W_2 = h(ID^*_T \| SK_{ST} \| b.g \| TS_{LA5})$.
4. The adversary relays M_3 to R_f. Alternatively, R_f has to be equipped with powerful antennas to receive M_3 when it is out of the acceptable vicinity of S.
5. Given M_3, R_f determines the validity of TS_{LA7}, and sends $M_4 = \{M_3, TS_{LA7}\}$ to T_i.
6. T_i checks TS_{LA7}, computes

   $SK_{TS} = h(ID_T \| X^*_T \| a.b.g \| a.x_s.g \| sn_i \| TS_{LA5})$,

   verifies $W^*_2 \stackrel{?}{=} h(ID_T \| SK_{TS} \| b.g \| TS_{LA5})$, and sets SK_{TS} as the session key.

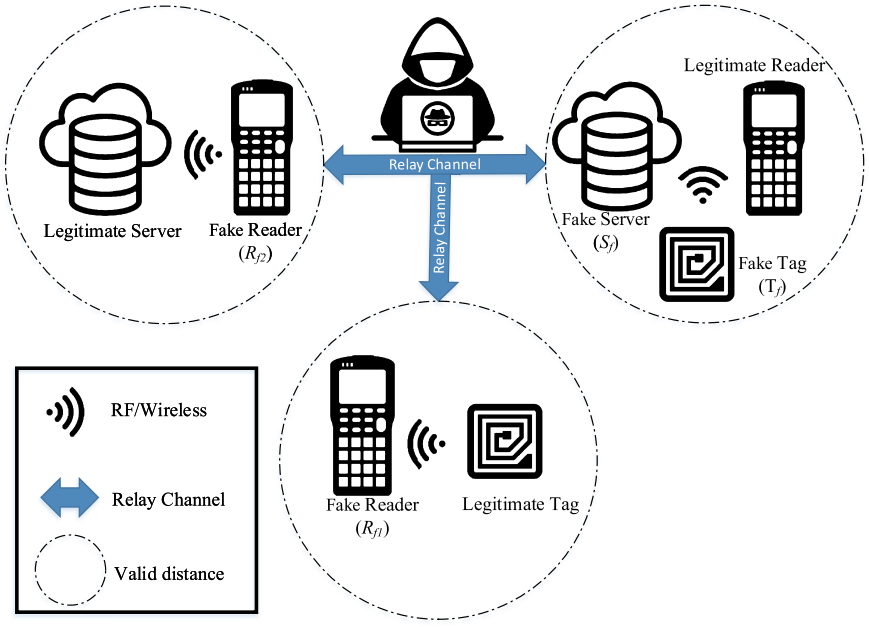

Fig. 4. Representation of the relay attack between the server, the reader and the tag.

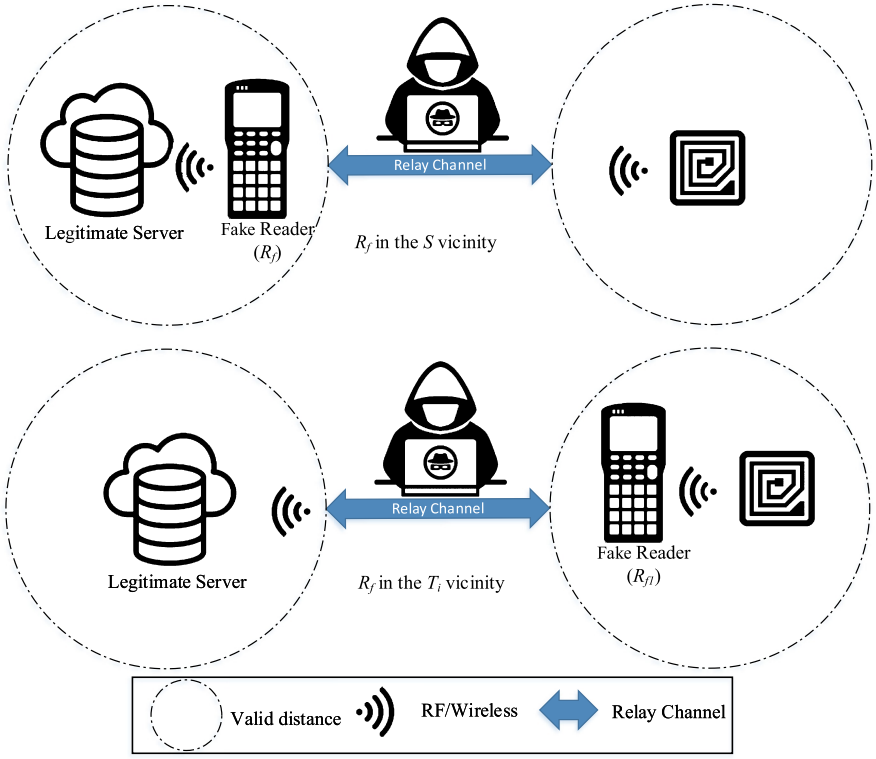

Fig. 5. Representation of the distance-fraud attack (relay attack) between the server and the tag using a counterfeited reader.

The attack procedure for the case where R_f is in the server vicinity is practically the same. It is evident in the above attack that once completed T_i and S share successfully the session key SK_{TS}, and there is no intervention of the legitimate R_j. The success probability of this deterministic attack is also '1'.

4. RSEAP2, the improved version of RSEAP

In this section, following the observations and the attacks previously described in section 3 against RSEAP, we apply the required patches to improve its security. We name the improved version as RSEAP2, considering the initial RSEAP as RSEAP1. Apart from enhancing the security of the new protocol, we design RSEAP2 aiming to reduce the tag's computations (and power consumption) as much as possible.

It is clear at the first that the timestamps' integrity should be guaranteed to evade the relay attacks descried above against RSEAP. Consequently, it could be easy to defeat the tags against impersonation and relay attacks. However, a reader in RSEAP does not share any key with the other entities, and thus it would not be easy to design a secure protocol for which a protocol party does no share any secret value. Hence, in RSEAP, to combat the

proposed attacks, which exploit the reader's weaknesses, we need to assume that any reader in the vehicular cloud has a permanent identifier RID_i and a public key $x_{Ri}.g$ share with S. R_j keeps x_{Ri} as its secret key. This extra step requires a registration phase between R_j and S over a secure channel. As the reader is generally a permanent device, we can assume that this information is set once installed in the environment or used for the first time by an operator. From the tag's point of view, similar to RSEAP, RSEAP2 also includes two phases, i.e. initialization phase and login/authentication phase over secure and public channels respectively. Through the protocol, the message time-duration is checked by validating the attached timestamp to the message. Given that an entity may receive messages from different entities at a once, it should then use different thresholds to validate each message. As described in the protocol, we use ΔT as the base value of the threshold. In this sense, an acceptable threshold of a timestamp sent from T_i to R_j is determined by $t_{TR} \times \Delta T$. Therefore t_{TR} is a factor to set the maximum delay of a transferred message from T_i to R_j. In this way, depending on the situation, it is possible to adjust those parameters to determine the physical coverage of the protocol (neighbourhood area), in which distance attacks are prevented.

The initialization phase of RSEAP2 (see Fig. 6) is identical to that of RSEAP, exclude that to reduce the tag's computation PWT and A_T are receptively computed as $PWT = h(ID_T \| (pw_T \oplus R_T))$ and $A_T = h(PWT \| (X_T \oplus ID_T))$.

In the login and authentication phase of the protocol, see Fig. 7, T_i logs on by its (ID_T, pw_T, R_T), computes $PWT = h(ID_T \| (pw_T \oplus R_T))$, $X_T^* = B_T \oplus PWT$ and $A_T^* = h(PWT \| (X_T^* \oplus ID_T))$. Then it verifies whether $A_T^* \stackrel{?}{=} A_T$. If verification is successful, T_i generates $a \in \mathbb{F}_q^*$, calculates $a.g$ and $W_1 = h((a.g) \oplus (X_T^* \| ID_T) \| TS_{LA1})$, and sends $M_1 = \{(ID_T \| W_1) \oplus a.x_s.g, a.g, TS_{LA1}\}$ to the reader R_j.

The reader checks the timestamp, i.e. $TS_{LA2} - TS_{LA1} \stackrel{?}{\leq} t_{TR} \times \Delta T$, generates $b \in \mathbb{F}_q^*$, computes $b.g$, and then sends $M_2 = \{M_1, TS_{LA3}, b.g, (h((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g)) \| RID_i) \oplus b.x_s.g\}$ to S.

Once S received M_2, it verifies the timestamps, i.e. $TS_{LA4} - TS_{LA1} \stackrel{?}{\leq} t_{TS} \times \Delta T$ and $TS_{LA4} - TS_{LA3} \stackrel{?}{\leq} t_{RS} \times \Delta T$. Next S extracts $h^*((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g)) \| RID_i^* = (h((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g)) \| RID_i) \oplus b.x_s.g \oplus b.x_s.g$, retrieves $x_{Ri}.g$ from the database and evaluates $h((TS_{LA21} \| TS_{LA21}) \oplus (x_{Ri}.g))$ to authenticate R_j. After the success in the authentication of R_j, S extracts $ID_T^* \| W_1^* = (ID_T \| W_1) \oplus a.x_s.g \oplus a.x_s.g$, retrieves the related sn_i using ID_T^*, computes $X_T^* = h(sn_i \| ID_T^* \| x_S)$ and $W_1^* = h(((a.g) \oplus (X_T^* \| ID_T^*)) \| TS_{LA1})$, and verifies $W_1^* \stackrel{?}{=} W_1$ to authenticate T_i. Next, it generates $c \in \mathbb{F}_q^*$, calculates $SK_{ST} = h((ID_T^* \oplus X_T) \| (a.b.c.g \oplus x_s.a.g) \| (sn_i \oplus (TS_{LA1} \| TS_{LA5})))$ and $W_2 = h(ID_T^* \| SK_{ST})$, and sends $M_3 = \{W_2 \oplus h(RID_i \| b.c.g), cg, TS_{LA5}, h(RID_i \| TS_{LA5} \| b.c.g)\}$ to R_j, where SK_{ST} is the shared session key.

R_j checks the timestamp, i.e. $TS_{LA5} - TS_{LA3} \stackrel{?}{\leq} t_{SR} \times \Delta T$, and $h(RID_i \| TS_{LA5} \| b.c.g)$ to authenticate S. Subsequent, it extracts W_2 and then sends $M_4 = \{W_2, b.c.g, TS_{LA5}\}$ to T_i.

Similarly, T_i verifies the timestamp, i.e. $TS_{LA7} - TS_{LA1} \stackrel{?}{\leq} t_{ST} \times \Delta T$ and $TS_{LA5} - TS_{LA1} \stackrel{?}{\leq} t_{RT} \times \Delta T$, and then computes $SK_{TS} = h((ID_T \oplus X_T) \| (a.b.c.g \oplus x_s.a.g) \| (sn_i \oplus (TS_{LA1} \| TS_{LA5})))$ and checks $W_2^* \stackrel{?}{=} h(ID_T \| SK_{TS})$. If so, it sets SK_{TS} as the session key.

5. Security analysis of RSEAP2

In this section, we evaluate the security of the login and authentication phase of RSEAP2 against various attacks. The analysis is conducted using both an informal and formal approach. For the latter, we use a popular tool, named as Scyther [11], which is commonly employed to assess the security of cryptographic protocols. We want to emphasize that since the initialization phase is over a secure channel, there are no security concerns linked with this protocol phase.

5.1. Informally security analysis

For the security analysis, we use the system model of RSEAP [22, Sec. 4] and the revised adversarial model. In addition to the attacks considered initially for RSEAP by its designers, we also take into account relay, reader impersonation and server impersonation attacks.

5.1.1. Location privacy (non-traceability)

The tag T_i simply sends $M_1 = \{(ID_T \| W_1) \oplus a.x_s.g, a.g, TS_{LA1}\}$ message, where $W_1 = h((a.g) \oplus (X_T^* \| ID_T) \| TS_{LA1})$. The only fraction of this message that could be used to determine the tag identity is $(ID_T \| W_1) \oplus a.x_s.g$. This token mentioned above is masked by $a.x_s.g$ and randomized by the variables a and TS_{LA1} on each session. Note that all these values are out of the adversary's control. In the worst-case scenario, assuming a collision occurs on the selected a value by T_i, then the adversary could identify its occurrence by monitoring $a.g$ fraction of M_1 and then T_i can be tracked. Nevertheless, the adversary's advantage to find a collision after N sessions of the protocol is $O(\frac{N^2}{|\mathbb{F}_q^*|})$, which is small enough in practice. Besides, M_1 does not disclose any information related to R_j or S.

The reader R_j sends $M_2 = \{M_1, TS_{LA3}, b.g, (h((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g)) \| RID_i) \oplus b.x_s.g\}$ to S, in which $(h((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g)) \| RID_i) \oplus b.x_s.g$ can be used to track the reader whether there is a collision in the $b.g$ fraction. Similarly, the adversary's advantage to detect a collision after N protocol executions is $O(\frac{N^2}{|\mathbb{F}_q^*|})$. Therefore, the adversary's success probability is low.

The server S sends $M_3 = \{W_2 \oplus h(RID_i \| b.c.g), cg, TS_{LA5}, h(RID_i \| TS_{LA5} \| b.c.g)\}$, where $W_2 = h(ID_T^* \| SK_{ST})$. However, each of $h(RID_i \| TS_{LA5} \| b.c.g)$ and $W_2 \oplus h(RID_i \| b.c.g)$ tokens are randomized in each session by both the reader R_j and the server S. Therefore, an adversary does not retrieve information that could help him/her to break the protocol's location privacy.

Lastly the reader R_j sends $M_4 = \{W_2, b.c.g, TS_{LA5}\}$ to T_i. The only target for the adversary in this message might be W_2. Unlikely for the adversary, this token is function of $SK_{ST} = h((ID_T^* \oplus X_T) \| (a.b.c.g \oplus x_s.a.g) \| (sn_i \oplus (TS_{LA1} \| TS_{LA5})))$ which is randomized by T_i, R_j and S on each session.

All in all, RSEAP2 protocol guarantees location privacy of all its entities (i.e., T_i, R_j and S).

5.1.2. Mutual authentication

Assuming that a legitimate tag T_i communicates with an honest server S through a valid reader R_j, and within acceptable time-thresholds, it is clear that the pairs (S, T_i) and (S, R_j) are mutually authenticated. However, in this protocol, we do not need mutual authentication between the reader R_j and the tag T_i. In detail, the source of trust for T_i is S, and R_j is only a gateway to access S. The above assumption is valid since the mutual authentication between the tag and the server implies that S has already authenticated R_j and T_i can trust the reader R_j.

5.1.3. Replay attack

In a replay attack, the adversary aims to use an exchanged message at a time t at a different time t'. Likely, in RSEAP2, any message received at any time out of the threshold time (a predefined factor of ΔT) is rejected. Besides the integrity of timestamps is guaranteed by the usage of the one-way hash function. Hence replay attacks are not feasible against RSEAP2.

Tag T_i		**Cloud database server** S
Inputs its ID_T and pw_T, generates $R_T \in \mathbb{F}_q^*$, computes $PWT = h(ID_T \| (pw_T \oplus R_T))$, and sends $M_{R1} = \{PWT, ID_T, TS_{R1}\}$ to S	$\xrightarrow{M_{R1}}$	
		Inspects $TS_{R2} - TS_{R1} \stackrel{?}{\leq} t_{TS} \times \Delta T$, generates $sn_i \in \mathbb{F}_q^*$ and sets it as the T_i's serial number, computes $X_T = h(sn_i \| ID_T \| x_S)$, $A_T = h(PWT \| (X_T \oplus ID_T))$, $B_T = X_T \oplus PWT$, and stores sn_i corresponding to ID_T. It sends $M_{R2} = \{A_T, B_T, sn_i, g, x_s.g, G, h(.)\}$ to T_i
Stores $\{A_T, B_T, sn_i, g, x_s.g, G, h(.)\}$	$\xleftarrow{M_{R2}}$	

Fig. 6. The initialization phase of RSEAP2, over a secure channel.

Tag T_i		**Reader** R_j		**Cloud database server** S
Logs on by (ID_T, pw_T, R_T), computes $PWT = h(ID_T \| (pw_T \oplus R_T))$, $X_T^* = B_T \oplus PWT$ and verifies A_T^*, generates $a \in \mathbb{F}_q^*$, computes $W_1 = h((a.g) \oplus (X_T^* \| ID_T) \| TS_{LA1})$ and sends $M_1 = \{(ID_T \| W_1) \oplus a.x_s.g, a.g, TS_{LA1}\}$ to R_j	$\xrightarrow{M_1}$	Verifies TS_{LA1}, chooses $b \in \mathbb{F}_q^*$ and sends $M_2 = \{M_1, TS_{LA3}, b.g, (h((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g)) \| RID_i) \oplus b.x_s.g\}$ to S	$\xrightarrow{M_2}$	Verifies TS_{LA1} and TS_{LA3}, extracts $h^*((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g)) \| RID_i^*$, validates $h((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g))$, extracts $ID_T^* \| W_1^*$, verifies $W_1^* = h(((a.g) \oplus (X_T^* \| ID_T^*)) \| TS_{LA1})$, generates $c \in \mathbb{F}_q^*$, computes $SK_{ST} = h((ID_T^* \oplus X_T) \| (a.b.c.g \oplus x_s.a.g) \| (sn_i \oplus (TS_{LA1} \| TS_{LA5})))$ and $W_2 = h(ID_T^* \| SK_{ST})$ and sends $M_3 = \{W_2 \oplus h(RID_i \| b.c.g), cg, TS_{LA5}, h(RID_i \| TS_{LA5} \| b.c.g)\}$ to R_j.
		Verifies TS_{LA5} and $h(RID_i \| TS_{LA5} \| b.c.g)$, extracts W_2 and sends $M_4 = \{W_2, b.c.g, TS_{LA5}\}$ to T_i	$\xleftarrow{M_3}$	
Verifies TS_{LA7} and TS_{LA5}, computes $SK_{TS} = h((ID_T \oplus X_T) \| (a.b.c.g \oplus x_s.a.g) \| (sn_i \oplus (TS_{LA1} \| TS_{LA5})))$ and verifies $W_2^* \stackrel{?}{=} h(ID_T \| SK_{TS})$, to set SK_{TS} as the session key.	$\xleftarrow{M_4}$			

Fig. 7. The login and authentication phase of RSEAP2, over a public channel.

Finally note that if the adversary would extract $x_s.g$ from the $a.x_s.g$ and $a.g$ pair, s/he could break the tag's anonymity. Likely, it is equivalent to solving EC-CDHP, which is known to be a hard problem.

5.1.4. Message authentication

In this protocol, M_1 and M_2 are authenticated by the server S, M_3 is authenticated partially by the reader R_j and is entirely authenticated by the tag T_i. The integrity of all messages is guaranteed by the usage of random numbers and the one-way hash function. Any modification on the transferred message leads to the rejection of the received message by the receiver. For instance, consider $M_1 = \{(ID_T \| W_1) \oplus a.x_s.g, a.g, TS_{LA1}\}$, which should be authenticated by S. The server S first checks $TS_{LA4} - TS_{LA1} \stackrel{?}{\leq} \Delta T$. Hence, if the adversary replays this message at another time, S will reject it. Then, S extracts $ID_T^* \| W_1^* = (ID_T \| W_1) \oplus a.x_s.g \oplus a.x_s.g$, retrieves the related sn_i value using ID_T^*, computes $X_T^* = h(sn_i \| ID_T^* \| x_S)$ and $W_1^* = h(((a.g) \oplus (X_T^* \| ID_T^*)) \| TS_{LA1})$, and finally verifies $W_1^* \stackrel{?}{=} W_1$ to accept the message. It is clear that any modi-

fication in TS_{LA}, $a.g$, or $(ID_T \| W_1) \oplus a.x_s.g$ renders the probability of $W_1^* \stackrel{?}{=} W_1$ to 2^{-n}, where n is the hash length, e.g. 256-bit for SHA-256. We can follow similar reasoning for the other messages included in the protocol. Consequently, RSEAP2 guarantees message authentication between the involved entities.

5.1.5. Tag impersonation attack

Given that the adversary chances of a replay attack are negligible, thanks to the integrity of TS_{LA1}, the only way to impersonate the tag is to generate a valid M_1. However, it is not feasible without guessing or computing a valid $W_1 = h(((a.g) \oplus (X_T \| ID_T)) \| TS_{LA1})$ where TS_{LA1} is the timestamp of the attack time. Also, the adversary does not have X_T and ID_T. Hence, the success probability of the adversary to impersonate the tag is 2^{-n}, where n is the bit length of the hash function. In other words, the replay attack is ineffective.

5.1.6. Reader impersonation attack

The adversary cannot replay messages to take the place of a reader since the integrity of TS_{LA3} is guaranteed in RSEAP2. Hence, the only way to impersonate the R_j in front of S is to generate a valid $\{TS_{LA3}, b.g, (h((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g)) \| RID_i) \oplus b.x_s.g\}$ tuple. However, the adversary does not have RID_i, $x_s.g$ and $x_{Ri}.g$. Even if s/he receives the values $x_s.g$ and $x_{Ri}.g$ somehow, s/he would need to extract b from $b.g$ to determine RID_i. It requires to solve EC-CDHP, which is a hard problem and renders the attack unfeasible. Therefore, it is not practical to impersonate R_j to S in this protocol.

5.1.7. Sever impersonation attack

The adversary would need to compute the token $h(RID_i \| TS_{LA5} \| b.c.g)$ to impersonate the server S in front of R_j. It would require knowledge of RID_i. Besides this token is randomized by $b.c.g$, which is contributed by R_j through sending $b.g$. For the disclosure of b, it would require the solving of an EC-CDHP problem, which is a hard problem. But even more, if the adversary discloses b and adapts it accordingly, yet because of TS_{LA5} in $h(RID_i \| TS_{LA5} \| b.c.g)$, the adversary still needs to know RID_i which is not the case. Therefore, the adversary's advantage for cheating R_j and successfully impersonating S is 2^{-n}. Besides, the impersonation of S in front of R_j is a requirement to impersonate S in front of T_i. Hence, the adversary cannot impersonate efficiently the server S in front of T_i through R_j. The adversary only might attempt to generate a valid message $M_4 = \{W_2, b.c.g, TS_{LA5}\}$, where $W_2 = h(ID_T^* \| SK_{ST})$. Unlikely for the attacker, s/he does not have ID_T^*. Therefore, the adversary's advantage to perform this impersonation attack is minimal (i.e. 2^{-n}).

5.1.8. Provision of key agreement

The shared key between S and T_i is computed as $SK_{ST} = h((ID_T^* \oplus X_T) \| (a.b.c.g \oplus x_s.a.g) \| (sn_i \oplus (TS_{LA1} \| TS_{LA5})))$, where a, b and c are respectively contributed by T_i, R_j and S. Besides ID_T^*, X_T and sn_i are only known by T_i and S. Hence, assuming that S, R_j and T_i are all legitimate entities, S and T_i will successfully share a key (i.e., K_{ST}). On the contrary, if any of them is fraudulent, the key agreement between these parties will fail.

5.1.9. Off-line password guessing attack

The argument for the security against this attack is almost identical to that used in the original RSEAP. In a nutshell, the temporary password of the tag is calculated as $PWT = h(ID_T \| (pw_T \oplus R_T))$. Even if the adversary could guess PWT, s/he would still need the value R_T, which is a random number generated by the tag T_i. Therefore, this attack is unfruitful to the adversary who could not predict R_T.

5.1.10. De-synchronization attack

RSEAP2 is immune against de-synchronization attacks since there is not updating phase of shared parameters once the protocol execution ends. The adversary only might block M_4 message to prevent the tag T_i to set the session key SK_{TS}. Likely, in this situation, since T_i has not received M_4 in the proper time, this entity could re-initiate the login and authentication phase to re-establish the session key. We want to emphasize that the above scenario is different from the impersonation attack presented against RSEAP. In that attack, the adversary successfully impersonates T_i without the presence of the honest T_i. Concerning RSEAP2, we have explained that an adversary cannot take the place of a genuine tag. Furthermore, the tag T_i must initiate the protocol; otherwise, the server S would not accept the request.

5.1.11. Insider attack

In the initialization phase of RSEAP2, T_i sends $M_{R1} = \{PWT, ID_T, TS_{R1}\}$ to S, where $PWT = h(ID_T \| (pw_T \oplus R_T))$. Likely, the chances for an insider attacker to disclose pw_T are almost null (i.e. 2^{-n}).

5.1.12. Relay attack

The critical points in our proposed relay attacks against the original RSEAP were successful impersonation attacks and also the flaw in the integrity of timestamps. Likely, in RSEAP2, we have shown that it is not achievable to impersonate any of the protocol parties. Also, the integrity of any transferred timestamp is guaranteed by the use of a hash function (i.e., $h(.)$). Consequently, it is not feasible to perform a relay attack against RSEAP2.

5.1.13. Man in the middle attack

An adversary should be able to impersonate a protocol entity or perform a message modification without being detected to make a successful man-in-the-middle attack. Nevertheless, in our proposed protocol, the mentioned attack will be unsuccessful due to the following reasons. Firstly, we have previously shown how the adversary's advantage to impersonate the tag (section 5.1.5), the reader (section 5.1.6 or the server (section 5.1.7) is negligible. Secondly, we have illustrated (see

5.2. 5.1.4) that any modification on the transferred message leads to the rejection of the received message by the receiver. Thirdly, in subsections 5.1.3 and 5.1.12, we have shown how an adversary cannot relay a message to deceive about his distance or replay later an old message successfully. Consequently, the proposed protocol is secure against man in the middle attack.

5.3. Formal security evaluation using Scyther tool

Apart from the informal analysis, we have assessed the security of RSEAP from a formal perspective. In the literature, we can find several formal methods that have been developed to evaluate the robustness of a cryptographic protocol. Those methods are either manual such as GNY logic [15] and BAN logic [9] or automated such as Scyther [11] and CryptoVerif [7]. We choose Scyther among them to formally simulate RSEAP2 and to verify its security. Our choice is motivated by the fact that this tool is a widely accepted method for formal verification of cybersecurity schemes and has been used to evaluate the security of many new protocols, including [3,4,14,21].

In Scyther, we model the security protocol using the Security Protocol Description Language (or SPDL programming language), and a graphical analysis shows the possible security threats detected on the analyzed scheme. Regarding the adversary model, this is predefined and based on the Dolev-Yao model. Once described the protocol in SPDL, the Scyther simulator evaluates the

Table 2
Security analysis result of the RSEAP2 scheme with Scyther.

Role	Claim	Status	Comments
Tag	Secret IDT	Ok	No attacks within bounds
	Secret pw	Ok	No attacks within bounds
	Secret RT	Ok	No attacks within bounds
	Secret XT	Ok	No attacks within bounds
	Secret a	Ok	No attacks within bounds
	Niagree	Ok	No attacks within bounds
	Nisynch	Ok	No attacks within bounds
	Alive	Ok	No attacks within bounds
	Weakagree	Ok	No attacks within bounds
Reader	Secret RID_i	Ok	No attacks within bounds
	Secret b	Ok	No attacks within bounds
	Niagree	Ok	No attacks within bounds
	Nisynch	Ok	No attacks within bounds
	Alive	Ok	No attacks within bounds
	Weakagree	Ok	No attacks within bounds
Cloud Server	Secret sn_i	Ok	No attacks within bounds
	Secret c	Ok	No attacks within bounds
	Niagree	Ok	No attacks within bounds
	Nisynch	Ok	No attacks within bounds
	Alive	Ok	No attacks within bounds
	Weakagree	Ok	No attacks within bounds

scheme against the predefined security claims included in the model [14]. For instance, it assumes that the private information used in the protocol is safe from the adversary during the protocol execution [3]. That is, the Scyther tool will not detect any flaw whose success depends on that assumption (e.g., an adversary cannot tamper the device's memory in which private information is stored).

To describe RSEAP2 in SPDL language, all entities, i.e. the tag, the reader and the cloud server, are represented by roles. A role contains all the actions (i.e., computing, sending or receiving) to be performed in the protocol. For each role, we declare some related functions. We also define nonce values and timestamp, which are used in the protocol specification, utilising the *fresh* and *timestamp* declarations, respectively. Next, the macros used to simplify the protocol specification are defined, where *recv* and *send* events are used respectively to denote the receiving and sending of a message. Finally, *claim* events are used in the specifications of each role to model intended security properties. Scyther uses a set of predefined claims including the following ones: *Secret, Alive, Weakagree, Nisynch, Niagree*. The tool automatically verifies all claims and reports the result. Following this specification, Table 2 depicts the results of the security verification of RSEAP2 using the Scyther tool. The obtained results confirm the security of RSEAP2.[1]

Similarly, to the informal analysis, we can conclude that the proposed protocol satisfies its security objectives, and no vulnerabilities are detected.

6. Performance comparison

From the performance point of view, it has already been shown that RSEAP outperforms previous works [22, Sec. 6], i.e. Yan et al. [41], He et al. [16], Sharma et al. [31], Wang et al. [37], Liu et al. [25], Jiant et al. [19], Shi et al. [32], Choi et al. [10]. Hence, for more clarity, we compare RSEAP2 with RSEAP and those new protocols that have been proposed for VCC. From this comparison, we can conclude that if RSEAP2 beats RSEAP, this also implies the overcoming of its predecessors.

RSEAP designers built the protocol based on SHA-1 [13] hash function, which has an output length of 160-bit. Unluckily, recent studies [24,23] showed practical chosen-prefix collision attacks on SHA-1, and the recommendation is to avoid its usage. Consequently, we have opted to use the SHA-2 family [33]. We propose the truncation of its output to 160 bits to maintain the same security level as that offered by RSEAP.

For evaluating the computation time, we take as reference the implementation results in [40] (CPU: Intel(R) Core(TM)2T6570 2.1GHz, Memory:4G OS:Win7 32-bit, Software: Visual C++ 2008, MIRACL C/C++ Library). As SHA-2 consumes 15.8 cycles per bytes [12], it means that its computation requires $T_h^f = 0.0004 \times \frac{15.8}{11.4} = 0.0005$ milliseconds. As a clarification, the value T_h^f corresponds to one call to the compression function (f) of SHA-2. The message-block length of the SHA-2 compression function is 512 bits. We take this fact into account while modified the messages' structure in RSEAP. In detail, we have designed the new protocol optimizing the number of calls to this compression function, especially in the tag's side, which is the most constrained device. Finally, an also considering as reference the results in [40], the time consumed for the calculation of scalar multiplication on ECC-160, denoted by T^{EMP} is 7.3529 millisecond and the time of chaotic map is $T^{CH} = T^{EMP}$ [27]. Depending on the used symmetric encryption scheme, the required time for encryption/decryption T^S of a symmetric scheme varies but the reported time in [40] for AES is $T^S = 0.1303$ milliseconds.

For the performance analysis, we assume that the bit sizes of the hash function output, nonces, timestamps, tag/reader identifiers, a symmetric encryption output block and elliptic curve points are respectively 160, 160, 32, 160, 128 and 320 bits. In Table 3, we summarize the comparison of RSEAP and RSEAP2 in terms of computational and communication costs. We focus this analysis on tags since these are the most constrained devices in the system. In terms of consuming time, as it is illustrated in Fig. 8, there are no significant differences compared to RSEAP–only a slight improvement for RSEAP2. Regarding the bits sent (and received), RSEAP2 is much more efficient than RSEAP, as it is shown in Fig. 9. It implies a considerable reduction in power consumption, which is a fundamental parameter in this sort of devices. Finally, in Table 4, we summarize the security properties offered by RSEAP and RSEAP2, respectively. In conclusion, the new protocol is more efficient and provides a higher security level.

[1] The model of RSEAP2 in SPDL is available at https://lightweightcryptography.com/?p=727.

Table 3
Computation cost (Comp., in millisecond) and communication cost (Comm., in bits) comparison for the tags/users entities in RSEAP2 and related protocols; SM and RM denote the sending and the receiving mode, respectively.

Protocol	Comp.	Time	Comm. (SM)	Comm. (RM)
[19]	$9T_h^f + 5T_S + 3T^{EMP}$	37.4205 (ms)	768	768
[20]	$9T_h^f + T_S + 5T^{EMP}$	22.1935 (ms)	1280	800
[27]	$4T_h^f + 2T^{CH}$	14.7078 (ms)	672	224
RSEAP	$9T_h^f + 3T^{EMP}$	22.0632 (ms)	832	544
RSEAP2	$6T_h^f + 3T^{EMP}$	22.0617(ms)	672	512

Table 4
Security comparison of RSEAP2 versus RSEAP. The used notations are summarized below: A1: Mutual authentication, A2: Replay attack, A3: Message authentication, A4: Tag/User impersonation attack, A5: Reader impersonation attack, A6: Sever impersonation attack, A7: Provision of key agreement, A8: Off-line password guessing attack, A9: Parallel session attack, A10: De-synchronization attack, A11: Insider attack, A12: Relay attack and A13: Man in the middle attack.

Protocol	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
[19]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓
[20]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓
[27]	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓	×	×
RSEA	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓	×	×
RSEAP2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

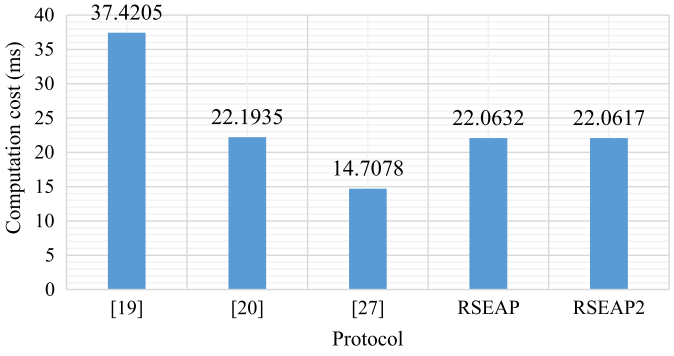

Fig. 8. RSEAP2 versus related protocols, computation comparison.

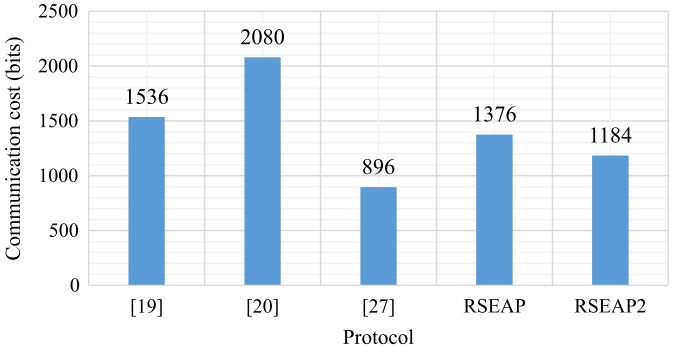

Fig. 9. RSEAP2 versus related protocols, communication comparison.

7. Conclusion

In this contribution, we have analyzed RSEAP [22], a recently proposed RFID based authentication protocol for vehicular cloud computing. We show how an adversary succeeds regarding tag impersonation, reader impersonation, and various relay attacks. The success probabilities for these attacks are maximum, and their complexities are low. In these attacks, we exploit two main weaknesses identified in the original protocol. First, there is a lack of integrity mechanism for the used timestamps. Thanks to this property, the adversary can efficiently and effectively carry out relay and tag impersonation attacks. Secondly, the reader, as a protocol party, does not share any secret with the other protocol parties (i.e., tag or the server). As a consequence of this, the adversary can easily impersonate the reader and can run a distance fraud attack.

We have also proposed a revised version of RSEAP, called RSEAP2. To fix the security faults existing in the original protocol, we guarantee the integrity of all timestamps by proper use of a one-way hash function. Also, in our proposal, the reader shares some secret parameters with the server. Thanks to the meticulous designing of the exchanged messages, the patched protocol (RSEAP2) even has significantly lower communication cost (and implicitly less power consumption) compared to the original scheme (RSEAP). Apart from the performance analysis, we have validated the security of RSEAP2 from both a formal and informal perspective.

Last but not least, we would like to mention that the new SFVCC [27] scheme shares much of the design of RSEAP [22]. Essentially, SFVCC is a variant of RSEAP in which the authors have replaced ECC by chaotic map-based cryptography. Unlikely for their designers, from the security point of view, both protocols share the same two main security pitfalls. First, there is a lack of integrity mechanisms for the used timestamps. Secondly, there is a protocol party (i.e., reader) that does not share any key with the other protocol entities. As a consequence of all the above, all of the attacks against RSEAP would also be applicable against SFVCC.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by Leonardo Grant for Researchers and Cultural Creators, BBVA Foundation (P2019-CARDIOSEC) and by the Comunidad de Madrid (Spain) under the project CYNAMON (P2018/TCS-4566), co-financed by European Structural Funds (ESF and FEDER). Masoumeh Safkhani and Nasoue Bagheri were supported by Shahid Rajaee Teacher Training University (SRTTU).

Appendix A. Formal security analysis of RSEAP2 in real or random oracle model

For completeness and to provide a formal security proof of RSEAP2, we follow the used framework by Abdalla et al. [1,18]. In this framework, the scheme's participants use their permanent password to agree on a common session key SK securely. Then, the involved entities can use the mentioned key to build secure channels for transferring sensitive information.

A protocol's instance could be either an (honest or malicious) client $U \in \mathcal{U}$ or a trusted server $S \in \mathcal{S}$. U holds a permanent password pw_U –the adversary knows it in case of a malicious client. Respectively, S holds a vector $pw_S = \left\langle pw_S[U] \right\rangle_{U \in \mathcal{U}}$, which contains an entry for each client U and $pw_S[U]$ represents a transformation of pw_U. Besides, if two clients U_i and U_j share the same session credentials they are considered partners.

A uniformly random bit b is chosen at the beginning of the experiment and should be guessed by the adversary later. This game determines the adversary's ability to distinguish a real session of the target scheme from a random session. Also, and concerning the exchange of messages, we assume that the adversary ($\mathcal{A}$) controls all the public communications between the scheme's participants and interacts passively or actively with them. To this end, $\mathcal{A}$ has access to the oracle queries listed below [1]:

- Execute(U_i, S, U_j) query models a passive adversary $\mathcal{A}$ who eavesdrops the communication between U_i, S and U_j in an honest protocol execution.
- Send(S/U, m) query models an active adversary who can intercept a message and then even modify it, create a new one, or forward a message to a protocol participant, which could be S or U.
- Reveal(U_i) query models the disclosure of the session key held by the client U_i.
- Test(U_i) query. Assuming that a client U_i and an honest partner set a session key, the answer to this query is conditioned by the value of bit b: 1) if $b = 1$, the answer is the mentioned session key; and 2) if $b = 0$, the query returns a random string of the same size of the session key.

Through the execution of a password-authenticated key agreement protocol $\mathcal{P}$, $\mathcal{A}$ has access to the Execute, Send, and Test queries –and perhaps to the Reveal query. After that, $\mathcal{A}$ outputs its guess bit b_0. The adversary wins the semantic security of the protocol in the Real-or-Random (RoR) game if $b_0 = b$. Note that b is the hidden bit used by the Test query and set at the beginning of the experiment. The advantage of $\mathcal{A}$ to win this game, $Adv^{RoR}_{\mathcal{D},\mathcal{P}}(t, R)$, is defined as follows [1]:

$$Adv^{RoR}_{\mathcal{D},\mathcal{P}}(t, R) = \left((Pr(\mathcal{A} \rightarrow b_0 = 1 : b = 1) - (Pr(\mathcal{A} \rightarrow b_0 = 1 : b = 0))\right)$$

and $\mathcal{P}$ is secure in RoR semantic security model if:

$$Adv^{RoR}_{\mathcal{D},\mathcal{P}}(t, R) < \varepsilon(.)$$

and $\varepsilon(.)$ being some negligible function, where the adversary makes R queries and spend t as the total time.

Following this introduction, we formally evaluate the security of RSEAP2 scheme in the real or random model (RoR).

Theorem 1. *Let q_{exe}, q_{send} and q_{test}, respectively, the number of queries of* Execute, Send *and* Test *oracles in RSEAP2 protocol. Then the adversary's advantage to win RoR game is:*

$$Adv^{RoR}_{\mathcal{D},IP}(t; q_{exe}; q_{test}; q_{send}) \leq 6 \times q \times \varepsilon_h + 4 \times q \times \varepsilon_{ECC}$$

where ε_{ECC} denotes the maximum advantage of solving ECDLP or EC-CDHP by the adversary on each query, ε_h represents the maximum advantage of contradicting indistinguishability of $h(.)$ from a random oracle on each query, and finally, q represents the total amount of queries ($q = q_{exe} + q_{test} + q_{send}$).

Proof. We suppose a tag T_i, a reader R_j and a server S which communicate between them and agree on a session key SK. Besides, we assume that an adversary $\mathcal{A}$ aims to compromise the semantic security of RSEAP2, in the RoR model. Similar to [18], we provide a game-based proof for the above theorem, by defining a set of games $\mathcal{G}$ that start in the random world RW and end in the real world, which represents RSEAP2. We also define $Adv^{RoR-\mathcal{G}_n}_{\mathcal{D},\mathcal{P}}(t, R)$ for each game. Specifically, it determines the advantage of $\mathcal{A}$ to correctly guess the hidden bit b involved in the Test queries, based on the game $\mathcal{G}_n$. In the analysis, we omit trivial advantages (e.g. timestamp) to guess the game since it is possible to adapt the random word to transfer the involved values. Also, we assume that the structure of the transferred messages remains identical in all games.

Game $\mathcal{G}_0$. It defines a random world RW. In such a world, excluding timestamps, any transferred value over the channel is selected at random but of the expected length. Hence, $Adv^{RoR-\mathcal{G}_0}_{\mathcal{D},RW}(t, R) = 0$.

Game $\mathcal{G}_1$. Compared to G_0, in this game we introduce ECC to compute $a.g$, $b.g$ and $c.g$, e.g. T_i generates a random value $a \in \mathbb{F}^*_q$ and computes $a.g$. Given that a, b and c are random, their mapping using g will also be random and this modification does not increase the adversary's advantage. Hence: $Adv^{RoR-\mathcal{G}_1}_{\mathcal{D},RW}(t, R) - Adv^{RoR-\mathcal{G}_0}_{\mathcal{D},RW}(t, R) = 0$.

Game $\mathcal{G}_2$. This game is identical to G_1, excluding that we introduce $h(.)$ through the computations. For example, we compute $W_1 = h((a.g) \oplus (X^*_T \| ID_T) \| TS_{LA1})$ instead of only using a mapping in the previous game. It is also applied to the other transferred values which are computed by $h(.)$ in RSEAP2, *i.e.*, W_1, $h((TS_{LA1} \| TS_{LA3}) \oplus (x_{Ri}.g)) \| RID_i)$, $W_2 = h(ID^*_T \| SK_{ST})$, $h(RID_i \| b.c.g)$ and $h(RID_i \| TS_{LA5} \| b.c.g)$. Note that random values and fresh timestamps influence on those calculations. Also, a secure hash function with a random input is indistinguishable from random oracle up to its indistinguishability bound, determined by ε_h. Hence, this modification increases the adversary's advantage as follows:

$$Adv^{RoR-\mathcal{G}_2}_{\mathcal{D},RW}(t, R) - Adv^{RoR-\mathcal{G}_1}_{\mathcal{D},RW}(t, R) \leq 5 \times q \times \varepsilon_h,$$

where $q = q_{exe} + q_{send} + q_{test}$.

Game $\mathcal{G}_3$. This game is identical to G_2, excluding that we call ECC component to compute $a.x_s.g$, $b.x_s.g$, $a.b.c.g$ and $b.c.g$. Given that $\mathcal{A}$ has access to $a.g$, $b.g$ and $c.g$ from $\mathcal{G}_1$, this modification can compromise the protocol if the adversary can solve ECDLP or EC-CDHP. Likely, the advantage to solve ECDLP or EC-CDHP based on this modification is upper bounded by $4 \times \varepsilon_{ECC}$, on each query. Hence, the $\mathcal{A}$'s advantage to distinguish $\mathcal{G}_3$ from $\mathcal{G}_4$ is as follows:

$$Adv^{RoR-\mathcal{G}_3}_{\mathcal{D},RW}(t, R) - Adv^{RoR-\mathcal{G}_2}_{\mathcal{D},RW}(t, R) \leq 4 \times q \times \varepsilon_{ECC}.$$

Game $\mathcal{G}_4$. This game is identical to G_3, excluding that the session key is calculated using a hash function (i.e., $SK_{TS} = h((ID_T \oplus X_T) \| (a.b.c.g \oplus x_s.a.g) \| (sn_i \oplus (TS_{LA1} \| TS_{LA5}))))$. Since the input values of $h(.)$, to compute SK_{TS}, are randomized by nonces and timestamps, the adversary's advantage to distinguish SK_{TS} from a random string is equivalent to distinguish $h(.)$ from a random oracle. Therefore:

$$Adv^{RoR-\mathcal{G}_4}_{\mathcal{D},RW}(t, R) \leq Adv^{RoR-\mathcal{G}_3}_{\mathcal{D},RW}(t, R) + q.\varepsilon_h.$$

On the other hand, $\mathcal{G}_4$ represents the implementation of RSEAP2. Hence:

$$\begin{aligned} &Adv^{RoR}_{\mathcal{D},RSEAP2}(t; q_{exe}; q_{test}; q_{send}) \\ &= Adv^{RoR}_{\mathcal{D},RSEAP2}(t, R) - Adv^{RoR}_{\mathcal{D},RW}(t, R) \\ &= Adv^{RoR-\mathcal{G}_4}_{\mathcal{D},RW}(t, R) - Adv^{RoR-\mathcal{G}_0}_{\mathcal{D},RW}(t, R) \\ &\leq 6 \times q \times \varepsilon_h + 4 \times q \times \varepsilon_{ECC} \end{aligned}$$

which completes the proof. □

References

[1] M. Abdalla, P. Fouque, D. Pointcheval, Password-based authenticated key exchange in the three-party setting, in: S. Vaudenay (Ed.), Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings, in: Lecture Notes in Computer Science, vol. 3386, Springer, 2005, pp. 65–84.

[2] Z.E. Ahmed, R.A. Saeed, A. Mukherjee, Challenges and opportunities in vehicular cloud computing, in: Cloud Security: Concepts, Methodologies, Tools, and Applications, IGI Global, 2019, pp. 2168–2185.

[3] R. Amin, P. Lohani, M. Ekka, S. Chourasia, S. Vollala, An enhanced anonymity resilience security protocol for vehicular ad-hoc network with scyther simulation, Comput. Electr. Eng. 82 (2020) 106554.

[4] A. Arfaoui, A. Kribeche, S. Senouci, Context-aware anonymous authentication protocols in the internet of things dedicated to e-health applications, Comput. Netw. 159 (2019) 23–36.

[5] G. Avoine, M.A. Bingöl, I. Boureanu, S. Capkun, G.P. Hancke, S. Kardas, C.H. Kim, C. Lauradoux, B. Martin, J. Munilla, A. Peinado, K.B. Rasmussen, D. Singelée, A. Tchamkerten, R. Trujillo-Rasua, S. Vaudenay, Security of distance-bounding: a survey, ACM Comput. Surv. 51 (5) (2019) 94:1–94:33.

[6] M. Azees, Reply to comments on “Dual authentication and key management techniques for secure data transmission in vehicular ad hoc networks”, IEEE Trans. Intell. Transp. Syst. 20 (9) (2019) 3595.

[7] B. Blanchet. CryptoVerif, Computationally sound mechanized prover for cryptographic protocols, in: Dagstuhl Seminar “Formal Protocol Verification Applied”, 2007, p. 117.

[8] A. Boukerche, R.E.D. Grande, Vehicular cloud computing: architectures, applications, and mobility, Comput. Netw. 135 (2018) 171–189.

[9] M. Burrows, M. Abadi, R.M. Needham, A logic of authentication, Proc. R. Soc. Lond. Ser. A, Math. Phys. Sci. 426 (1871) (1989) 233–271.

[10] Y. Choi, D. Lee, J. Kim, J. Jung, J. Nam, D. Won, Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography, Sensors 14 (6) (2014) 10081–10106.

[11] C.J.F. Cremers, The Scyther tool: verification, falsification, and analysis of security protocols, in: Computer Aided Verification, Springer, Berlin, Heidelberg, 2008, pp. 414–418.

[12] W. Dai, Crypto++ 5.6.0 benchmarks, http://www.cryptopp.com/benchmarks.html, 2009.

[13] D. Eastlake, P. Jones, US Secure Hash Algorithm 1 (SHA1), 2001.

[14] M. Eldefrawy, I. Butun, N. Pereira, M. Gidlund, Formal security analysis of LoRaWAN, Comput. Netw. 148 (2019) 328–339.

[15] L. Gong, R. Needham, R. Yahalom, Reasoning about belief in cryptographic protocols, in: IEEE Computer Society Symposium on Research in Security and Privacy, IEEE, 1990, pp. 234–248.

[16] D. He, N. Kumar, M.K. Khan, L. Wang, J. Shen, Efficient privacy-aware authentication scheme for mobile cloud computing services, IEEE Syst. J. 12 (2) (2018) 1621–1631.

[17] J. Hoffstein, J. Pipher, J.H. Silverman, J.H. Silverman, An Introduction to Mathematical Cryptography, Vol. 1, Springer, 2008.

[18] M. Hosseinzadeh, O.H. Ahmed, S.H. Ahmed, C. Trinh, N. Bagheri, S. Kumari, J. Lansky, B. Huynh, An enhanced authentication protocol for RFID systems, IEEE Access 8 (2020) 126977–126987.

[19] Q. Jiang, J. Ni, J. Ma, L. Yang, X. Shen, Integrated authentication and key agreement framework for vehicular cloud computing, IEEE Netw. 32 (3) (2018) 28–35.

[20] Q. Jiang, N. Zhang, J. Ni, J. Ma, X. Ma, K.-K.R. Choo, Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles, IEEE Trans. Veh. Technol. (2020).

[21] Z.A. Khan, Z. Pervez, A.G. Abbasi, Towards a secure service provisioning framework in a smart city environment, Future Gener. Comput. Syst. 77 (2017) 112–135.

[22] V. Kumar, M. Ahmad, D. Mishra, S. Kumari, M.K. Khan RSEAP, RFID based secure and efficient authentication protocol for vehicular cloud computing, Veh. Commun. 22 (2020) 100213.

[23] G. Leurent, T. Peyrin, From collisions to chosen-prefix collisions application to full SHA-1, in: Y. Ishai, V. Rijmen (Eds.), Advances in Cryptology - EUROCRYPT 2019, in: Lecture Notes in Computer Science, vol. 11478, Springer, 2019, pp. 527–555.

[24] G. Leurent, T. Peyrin, SHA-1 is a shambles - first chosen-prefix collision on SHA-1 and application to the PGP web of trust, IACR Cryptol. ePrint Arch. 2020 (2020) 14.

[25] Y. Liu, Y. Wang, G. Chang, Efficient privacy-preserving dual authentication and key agreement scheme for secure V2V communications in an IoV paradigm, IEEE Trans. Intell. Transp. Syst. 18 (10) (2017) 2740–2749.

[26] A. Mehmood, S.H. Ahmed, M. Sarkar, Cyber-physical systems in vehicular communications, in: Handbook of Research on Advanced Trends in Microwave and Communication Engineering, IGI Global, 2017, pp. 477–497.

[27] D. Mishra, V. Kumar, D. Dharminder, S. Rana, SFVCC: chaotic map-based security framework for vehicular cloud computing, IET Intell. Transp. Syst. 14 (8) (April 2020) 241.

[28] J.L.T. Pozo, Computational and symbolic analysis of distance-bounding protocols, PhD thesis, University of Luxembourg, Luxembourg City, Luxembourg, 2019.

[29] S. Rostampour, M. Safkhani, Y. Bendavid, N. Bagheri, ECCbAP: a secure ECC based authentication protocol for IoT edge devices, Pervasive Mob. Comput. (2020) 1–33.

[30] M.A. Saleem, K. Mahmood, S. Kumari, Comment on “AKM-IoV: authenticated key management protocol in fog computing-based Internet of vehicles deployment”, IEEE Int. Things J. 7 (5) (2020) 4671–4675, https://doi.org/10.1109/JIOT.2020.2975207.

[31] M.K. Sharma, R.S. Bali, A. Kaur, Dynamic key based authentication scheme for vehicular cloud computing, in: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), IEEE, 2015, pp. 1059–1064.

[32] W. Shi, P. Gong, A new user authentication protocol for wireless sensor networks using elliptic curves cryptography, Int. J. Distrib. Sens. Netw. 9 (4) (2013) 730831.

[33] S.H. Standard, FIPS Pub 180-2, Secure hash algorithm 2, Natl. Inst. Stand. Technol. 17 (2001) 15.

[34] H. Tan, D. Choi, P. Kim, S.B. Pan, I. Chung, Comments on “Dual authentication and key management techniques for secure data transmission in vehicular ad hoc networks”, IEEE Trans. Intell. Transp. Syst. 19 (7) (2018) 2149–2151.

[35] J. Tsai, N. Lo, A privacy-aware authentication scheme for distributed mobile cloud computing services, IEEE Syst. J. 9 (3) (2015) 805–815.

[36] P. Vijayakumar, M. Azees, A. Kannan, L.J. Deborah, Dual authentication and key management techniques for secure data transmission in vehicular ad hoc networks, IEEE Trans. Intell. Transp. Syst. 17 (4) (2016) 1015–1028.

[37] F. Wang, Y. Xu, H. Zhang, Y. Zhang, L. Zhu, 2FLIP: a two-factor lightweight privacy-preserving authentication scheme for VANET, IEEE Trans. Veh. Technol. 65 (2) (2016) 896–911.

[38] M. Wazid, P. Bagga, A.K. Das, S. Shetty, J.J.P.C. Rodrigues, Y. Park, AKM-IoV: authenticated key management protocol in fog computing-based Internet of vehicles deployment, IEEE Int. Things J. 6 (5) (2019) 8804–8817.

[39] M. Whaiduzzaman, M. Sookhak, A. Gani, R. Buyya, A survey on vehicular cloud computing, J. Netw. Comput. Appl. 40 (2014) 325–344.

[40] L. Xu, F. Wu, Cryptanalysis and improvement of a user authentication scheme preserving uniqueness and anonymity for connected health care, J. Med. Syst. 39 (2) (2015) 10.

[41] G. Yan, D.B. Rawat, B.B. Bista, Towards secure vehicular clouds, in: L. Barolli, F. Xhafa, S. Vitabile, M. Uehara (Eds.), Sixth International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS 2012, Palermo, Italy, July 4-6, 2012, 2012, pp. 370–375.