Research Article

A Highly Efficient ECC-Based Authentication Protocol for RFID

Hasnae Lamrani Alaoui ¹, **Abdellatif El Ghazi**², **Mustapha Zbakh**¹, **Abdellah Touhafi**³,
and An Braeken³

¹*École Nationale Supérieure d'Informatique et d'Analyse des Systèmes, Morocco*

²*Université Internationale de Rabat, Morocco*

³*Vrije Universiteit Brussel, Belgium*

Correspondence should be addressed to Hasnae Lamrani Alaoui; hassnae.lamrani@gmail.com

Received 4 June 2020; Accepted 31 May 2021; Published 3 July 2021

Academic Editor: Syed K. Islam

Copyright © 2021 Hasnae Lamrani Alaoui et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The availability of safety has always been a challenge in all fields of computing and networking, and the development of reliable security measures for low operating power systems has been a continuous task for researchers. This paper focuses on Radio Frequency Identification (RFID) systems, whose market is tremendously growing due to the increasing need for efficient supply chain and healthcare management. Two protocols are proposed that offer mutual authentication, confidentiality, forward security, anonymity, unlinkability, scalability, and resistance against the most important security attacks. They differ in storage requirements at the server side. Both proposed protocols are very efficient as the number of required elliptic curve multiplications is limited to two at the tag side. Moreover, the schemes are also more effective and secure than other related work presented in literature. In addition, we show how to extend the protocols in order to provide authentication with respect to multiple readers.

1. Introduction

Radio Frequency Identification (RFID) is a technology working on radio frequency and mainly used for the identification and tracking of tags attached to objects. RFID is considered to be one of the key technologies of the fast-growing Internet of Things (IoT) market, and its market value is expected to rise from \$12.08 bn in 2020 to \$16.23 billion by 2029 [1]. In recent years, RFID technology has also been commonly utilized in the healthcare context for applications such as child protection [2], position monitoring of medical assets [3], medical condition monitoring and validation [4], patient tracking and medication management [5, 6], blood transfusion [7], and nursing home safety management [8]. Because health data is an integral aspect of personal privacy information [9], it is vital to secure private medical data and to guarantee anonymity and protection of tracking from access during the RFID authentication procedure. It is also already widespread deployed in a variety of other applications such as smart logistics management [10], safe access control

[11], automated toll payment [12], staff monitoring, and theft prevention [12].

An RFID network consists of a device (including a tag), a reader, and a backend processor, where the reader has a special identification and can access the tag's working zone for read/write operations. As shown in Figure 1, after being initialized with the required security material from the backend server, the tag encrypts the identity details and transfers it to the reader. The reader can then authenticate the obtained data and the tag ID by relying on information available at the backend server, either in online or offline mode. Security and protection issues will be of considerable concern during the whole process for a lot of applications mentioned above. Besides confidentiality, authentication, and integrity, also anonymity and unlinkability are important features to assess in the development of the system in order to avoid tracking attacks.

The RFID device is classified as either passive or active, based on whether the device utilizes tags with an internal power source or is controlled by energy distributed by the RFID readers to relay signals on a continuous basis. Low-

FIGURE 1: An example of an RFID authentication system.

cost RFID devices highlight the challenges of securing systems, as traditional encryption primitives increase the cost of such devices due to their high power consumption. Although the difficulties in the design of low-cost protected RFID systems involve reduced computer size and processing capability, the introduction of a reliable and cost-effective authentication strategy to ensure the security of user information is critical. An RFID network consists of a device (including a tag), a reader, and a backend processor, where the reader has a special identification and can access the tag's working zone for read/write operations. As shown in Figure 1, after being initialized with the required security material from the backend server, the tag encrypts the identity details and transfers it to the reader. The reader can then authenticate the obtained data and the tag ID by relying on information available at the backend server, either in online or offline modus. Security and protection issues will be of considerable concern during the whole process for a lot of applications mentioned above. Besides confidentiality, authentication and integrity, also anonymity and unlinkability are important features to assess in the development of the system in order to avoid tracking attacks.

The RFID device is classified as either passive or active, based on whether the device utilizes tags with an internal power source or is controlled by energy distributed by the RFID readers to relay signals on a continuous basis. Low-cost RFID devices highlight the challenges of securing systems, as traditional encryption primitives increase the cost of such devices due to their high power consumption. Although the difficulties in the design of low-cost protected RFID systems involve reduced computer size and processing capability, the introduction of a reliable and cost-effective

authentication strategy to ensure the security of user information is critical.

Previously, authentication protocols for RFID relied on hash functions and symmetric key cryptography. However, due to the good performance and low-key size specifications of elliptic curve cryptography (ECC), the creation of an effective RFID authentication protocol based on ECC is made possible. ECC has been already used in different authentication and key establishment protocols for IoT-based systems. ECC is a public key or asymmetric key cryptographic solution, offering smaller key sizes and lower computing costs compared to alternatives like RSA or systems relying on the discrete logarithm as trapdoor. In this paper, we propose two highly efficient EC-based authentication protocols for client-server architectures with a small client, e.g., sensor or tag and server, corresponding to the reader. The first protocol needs the storage of a list of acceptable tags and keys on the reader side, while the second protocol includes just the preservation of the list of revoked tags on the reader. The mathematical complexity of both protocols is limited by a maximum of two EC multiplications at the constrained tag side, which results in the best performance compared to other related work.

While both protocols satisfy many required security features, such as anonymity, untraceability, perfect forward secrecy, and mutual authentication, the only disadvantage is that the tag cannot validate the authenticity of the first received message, consisting of a random value from the reader. We therefore suggest to introduce an additional phase at the beginning of the procedure, requiring two additional EC multiplications at the tag side. The other benefit of this additional phase is that the authentication procedure can be

extended to any reader and not only to readers whose public key is prestored in the memory of the tag.

We show that the proposed protocols have a comparable or even stronger security strength, compared to related work. The security of both protocols is proven in an informal way and formal way. For the formal review, the *ProVerif* [13] Model Checker is used. The paper is organized as follows. In Section 2, related work is described in more detail. Section 3 discusses some background on the architecture, attack model, and the EC-based operations. The proposed schemes are explained in Section 4. Section 5 shows the security of the scheme with respect to informal and formal evaluations. In Section 6, the performance of the protocols is discussed and compared with related work. Finally, Section 7 presents the conclusions of the paper.

2. Related Work

Client-server authentication is an essential operation, which is required in many use cases and application domains, for instance, health care, smart grid, and smart home. The literature on client-server authentication schemes can be divided into different areas. First, there are the schemes, where the client represents a user connected to a device or smartcard, enabling multifactor authentication schemes. Next, we can also distinguish the schemes in terms of the type of operations used, which mainly consists of symmetric and public key-based schemes. In our case, we focus on the schemes, where the client is a device with a limited amount of resources, but sufficient to still implement public key-based operations like EC operations. Examples of such clients are for instance a sensor or a tag, which requires authentication to be accepted by the reader.

In [14], a survey is given on the different classes of RFID authentication protocols, with a distinction in fully fledged, simple, lightweight, and ultralightweight protocols. Fully fledged protocols are defined as protocols supporting both symmetric and asymmetric operations and thus are the focus of this work. From the fully fledged protocols, EC-based operations are the most efficient ones and in particular [15–18] are analysed in their survey. Each of these schemes has also compared their strength and efficiency with other schemes in literature. For instance, the scheme of [15] has been compared with [14, 19, 20], the scheme of [16] with [21–25], the scheme of [17] with [14, 20, 26, 27], and the scheme of [18] with [14, 20, 26, 28]. From Table 2 of [14], it can be concluded that [18] is the most complete scheme with respect to security features and resistance against important security threats. However, when having a closer look to [18], we can also conclude that this scheme is not scalable as the reader needs to perform exhaustive search in the whole list of tags to verify the validity of the signature. This feature has not been considered in the survey. Moreover, unless it is considered secure against tracking and location privacy in [14], the attacker can easily follow the tag once the public key of the tag is known.

Recently, there are some new ECC-based schemes proposed. First, there is the scheme of [29], presented in the context of smart grid, which has been designed to satisfy resistance in the Canetti-Krawczyk (CK) security model. This model allows an attacker to also reveal session state-specific

information, session keys, or long-term private keys, which is a pretty strong requirement. In [29], it has also been shown that the EC-based schemes of [30–32] were not resistant in the CK model and suffered also man-in-the-middle attacks and denial of service attacks. The scheme utilized the EC Qu Vanstone (ECQV) certificate mechanism [33]. We want to note that our second proposed scheme is inspired by this protocol but requires less computation as the proposed attack model is less strong than the CK model.

A second recent and interesting scheme is proposed in [34]. It is very efficient with only three EC multiplications at the tag side; however, not scalable as an exhaustive search at the reader side is required to find the Diffie Hellman key in order to decrypt the message and to be able to find the elementary information. Moreover, if the secret key of the reader or tag is known, the identity of the tag can be revealed. The protocol is compared against [35–37], where all of these schemes contain several shortcomings like no resistance against session-specific temporary information attack and time synchronisation attack.

Another important and recent protocol to be mentioned is given in [38]. Here, the tag identity and corresponding key shared with the reader are updated after each session. In order to avoid synchronisation attacks, there are two values stored at the reader, the last and previous one. Therefore, complete perfect forward security is not satisfied with respect to the second last session key. The scheme also suffers from the session-specific temporary information attacks as if the session data TK_{t1}, TK_{t2} is stored; the identity of the tag and corresponding key can be revealed for the current and next sessions. Consequently, if the attacker blocks the last communication message and the tag wants to restart the process, it uses its updated identity and key, which are now known to the attacker. In addition, the scheme is very weak for denial of service attacks as it consists of four phases and the authentication is only verified in the end. The scheme is compared with [39–42]. Again, each of these schemes contains significant weaknesses compared to [38], like no integrity check and no scalability. The scheme is very efficient due to the fact that the number of EC multiplications at the tag side is limited to three.

Finally, in [43], two ECC-based client-server authentication protocols have been proposed. The first protocol, which is an improved version of [44], is not secure for unlinkability in case of identity guessing, does not offer perfect forward secrecy, and is vulnerable for session temporary attacks. The second protocol, also called hash-based authentication protocol is developed in order to be used in a multireader-based scenario. It is very efficient as only 2 EC multiplications are required at the tag side. However, the scheme suffers from many security issues. First, it does not satisfy unlinkability as the attacker can send challenges at any moment to the tag, which enables the attacker to track a certain tag. Second, it is not secure against malicious insider servers as a server is able to impersonate the tag. It also does not offer perfect forward secrecy and is vulnerable for session temporary attacks.

To conclude, all the proposed EC-based authentication protocols in literature, satisfying at least anonymity, unlinkability, and mutual authentication, require at least three EC

multiplications at the tag side. Moreover, the schemes with three EC multiplications [34, 38] suffer from several attacks, in particular when the attacker is allowed to get access to session information. In particular, it has been explicitly mentioned in [45] that the usage of the more expensive public key-based protocols is only allowed in case protection against a semitrusted TTP is needed, as otherwise all other security features can simply be addressed by a symmetric key-based protocol [46]. As shown in [45], almost all of the recently proposed EC-based protocols, defined for a more general client-server architecture (not limited to an RFID client), suffer from this weakness (e.g., [47, 48]). Therefore, this has been carefully taken into account in our protocol designs. Finally, most of the protocols described in literature are only defined for tags authenticating to one predefined reader, which is not a very practical use case, and do not address scalability from the reader point of view.

3. Background

We first start to give some elementary background on the cryptographic operations. Next, we describe the architecture and corresponding adversary model.

3.1. Cryptographic Operations

3.1.1. Low-Cost Operations. The concatenation and xor operation of two messages M_1 and M_2 is denoted by $M_1 \| M_2$ and $M_1 \oplus M_2$, respectively. We denote the symmetric encryption, like AES128, using a symmetric shared key K of message M by $C = E_K(M)$ and the corresponding decryption by $M = D_K(C)$. Note that in our protocol, we only use the $\oplus$ operation for encryption and decryption, which corresponds with the one-pad scheme, offering perfect security [49].

3.1.2. Hash Functions. For a one-way hash operation, we will use the latest SHA3 standard [50] and in particular the $\text{SHAKE128}(M, d)$ function on message M with variable input size and variable output size d . This function has a proven resistance of at least $\min(d/2, 128)$ on collision attacks and preimage and second preimage attacks independent on the input length and output lengths larger than 256 bits. For ease of notation, we denote $\text{SHAKE128}(M, d)$ as $H(M)$.

3.1.3. Elliptic Curve-Based Operations. ECC allows lightweight public key cryptographic solutions. In order to achieve 128-bit security, a field size of 256 bits for ECC is sufficient, compared to at least 3072 bits for RSA. ECC is based on the algebraic structure of ECs over finite fields F_p . The E25519 curve [51] over the prime field $p = 2^{255} - 19$ enables the fastest implementation for the EC operations and possesses also resistance against the well-known implementation attacks [52]. Compressed EC points, containing only the X coordinate, can be used to represent the complete point. We denote the EC addition of two points by $P_1 + P_2$ and multiplication by rP having $r \in F_q$ with q the order of the generator G . The security of ECC relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP) and the Elliptic Curve

Diffie Hellman Problem (ECDHP), both well-known computational hard problems.

3.1.4. Elliptic Curve Qu Vanstone Mechanisms. The ECQV mechanism [33] allows to define certificates of very small size, compared to other approaches like X.509 certificates. Moreover, the scheme enables also the derivation of a secret key pair, with private key only known by the requested entity and not the trusted third party (TTP).

Denote the secret key pair of the TTP by $(d_{\text{TTP}}, Q_{\text{TTP}})$. The different steps in the derivation of the key pair (d_n, Q_n) for an entity with identity id_n are as follows. First, the entity sends its identity id_n , together with an EC point $R_i = r_i G$, where r_i is a randomly chosen variable. Next, also, the TTP chooses a random r_T and computes $R_T = r_T G$. The certificate is then defined by $\text{Cert}_n = R_i + R_T$. Next, an auxiliary value $r = H(\text{Cert}_n \| \text{id}_n) r_T + d_{\text{TTP}}$ is computed. The pair (r, Cert_n) is sent to the entity id_n . The private key d_n can now be derived as $d_n = H(\text{Cert}_n \| \text{id}_n) r_i + r$. Only if $Q_n = d_n G$ corresponds with $H(\text{Cert}_n \| \text{id}_n) \text{Cert}_n + Q_{\text{TTP}}$ the entity approves the key pair (d_n, Q_n) and stores it in its memory. Note that this last derivation

$$Q_n = H(\text{Cert}_n \| \text{id}_n) \text{Cert}_n + Q_{\text{TTP}} \quad (1)$$

also corresponds to the operation, required to be executed by an outsider who is in the possession of the certificate Cert_n and the identity id_n .

3.2. Architecture. In our setting, we assume a tag (T), reader (R), and trusted third party (TTP). The tag is considered as a low-cost device in which efficiency plays an important role. The reader has to verify the legitimacy of the tag. Therefore, it needs to get access to the list of approved tags or a list of invoked tags. This can be either by storage at its own premises or by contacting a secure authentication service. Note that for the ease of notation, we consider here the first scenario. Finally, there is the trusted third party, who shares this list of tags to the reader during the initialisation process and does not participate in the actual authentication protocol.

3.2.1. Attack Model. The adversary has the following possibilities:

- (A1) The attacker can actively participate in the communication by modifying, replaying, or blocking the transmitted messages
- (A2) The attacker is able to passively monitor the communication and collect transmitted data in order to eventually exploit later on
- (A3) The attacker is able to derive the session-specific temporary information at both reader and tag, e.g., by means of timing attacks. Note that this can be simultaneously at the reader and tag at the same time
- (A4) The attacker can also get access to the long-term key material, stored at both reader and tag. Note

that this can be simultaneously at the reader and tag at the same time

The attack scenarios (A1, A2) can be combined with (A3) or (A4). However, (A3) and (A4) cannot be executed at the same time, which is the main difference compared to the CK security model. Note that this security model is applied in most of the RFID-based authentication protocols.

For each of the above-described attack scenarios, our protocol should still be able to guarantee the following security criteria:

- (C1) Mutual authentication is offered in the protocol to avoid impersonation attacks. Both parties also need to participate in order to avoid key control attacks
- (C2) The anonymity of the tag should be guaranteed
- (C3) It should not be able to link the different requests to a specific tag and thus to track some particular tag
- (C4) Even if session-specific temporary information is leaked (cf (A3)), it should not be possible to impersonate or reveal the identity of the tag. This is called resistance against session-specific temporary information attack
- (C5) Even if the key material is revealed (cf (A4)), it should not be possible to impersonate the tag or to reveal its identity. This corresponds to the perfect forward secrecy feature
- (C6) Resistance against replay attacks should be realised
- (C7) Resistance against desynchronisation attacks should be offered. Even if the attacker blocks some messages, the reader and tag will still be able to successfully execute the protocol. This also results in the availability feature
- (C8) Scalability should be offered in order to find the matching record in the database of the reader in an efficient way, i.e., not through exhaustive search in all of the stored records
- (C9) Resistance against denial of service attacks should be provided as much as possible in order to avoid unnecessary work to be done by one of the entities before noticing a potential attack

4. Proposed Schemes

We first explain the two basic schemes, called Scheme 1.1 and Scheme 1.2, being a protocol with storage of individual key material for each legitimate tag (Scheme 1.1) and a protocol without storage of individual key material (Scheme 1.2). Note that in this last protocol, we can assume that only the list of revoked tags is stored. For both protocols, we discuss the initialisation and the actual authentication phase. We assume that all entities participating in the scheme implement the required EC parameters and the associated EC operations, together with the hash function.

4.1. Basic Schemes

4.1.1. Scheme 1.1: With Storage at Reader Side

(1) *Initialisation Phase*. Each tag receives from the TTP the public key of the reader $Q_r = d_r G$ to which it needs to authenticate to and the tuple (id_n, K_{n_1}, K_{n_2}) , representing a unique identity id_n and two corresponding unique keys (K_{n_1}, K_{n_2}) , respectively. These parameters are securely stored on the tag. The reader receives from the TTP the list of legitimate $(id_n, K_{n_1}, K_{n_2})_n$, which are securely stored in its database (DB). Also, its key pair (d_r, Q_r) , containing private and public key, is securely stored at the reader.

(2) *Authentication Phase*. In the first step, the reader chooses a random value r_r and computes $R_r = r_r G = (x_r, y_r)$, which is sent to the tag. Note that by using the E25519 curve, it is sufficient to send x_r as mentioned before. However, for clarity in notation, we still denote it as the full point R_r .

Then, the tag also chooses a random value r_n and computes $R_n = (r_n + K_{n_2})G$. Next, it derives $H((r_n + K_{n_2})(Q_r + R_r)) = (hr_1, hr_2)$ in order to compute $A_1 = K_{n_1} \oplus hr_1$. Finally, the tag computes $H(id_n \| K_{n_1} \| K_{n_2} \| R_r \| R_n \| hr_2) = (h_1 \| h_2)$ and sends the message R_n, A_1, h_2 to the reader.

Upon receiving this message, with its private key d_r and the temporary local variable r_r , the reader first computes $(d_r + r_r)R_n = (hr_1, hr_2)$ in order to retrieve $K_{n_1} = A_1 \oplus hr_1$ and looks it up in the DB containing the list of legitimate tags. If it does not exist, the protocol is aborted. Otherwise, it retrieves the corresponding identity-related material of that tag id_n and the second key K_{n_2} and computes $H(id_n \| K_{n_1} \| K_{n_2} \| R_r \| R_n \| hr_2) = (h_1 \| h_2)$. If the second part of the output corresponds with the received h_2 , the reader approves the request and forwards h_1 to the tag.

The tag compares the stored h_1 value with the received one, and in case it is similar, the authentication is approved, otherwise the tag aborts the process. The different steps in this protocol are illustrated in Table 1.

4.1.2. Scheme 1.2: Without Storage at Reader Side

(1) *Initialisation Phase*. The tag first derives its private and public key by means of the ECQV protocol. As a result, it receives its identity id_n , certificate $Cert_n$, and key pair (d_n, Q_n) with $Q_n = d_n G = H(Cert_n \| id_n) Cert_n + Q_{TTP}$ as shown in Equation (1). The values $(id_n, Cert_n, (d_n, Q_n), Q_r)$ are stored on the tag.

The reader receives from the TTP the list of revoked tags containing $(id_{rn}, Cert_{rn})_{rn}$.

(2) *Authentication Phase*. The first step is the same as in Scheme 1.1. The reader chooses a random value r_r and computes $R_r = r_r G$. This value R_r is sent to the tag.

Then, the tag also chooses a random value r_n and computes $R_n = (r_n + d_n)G$. Next, it derives a Diffie Hellmann

TABLE 1: Steps and computations in the proposed authentication with storage.

Reader- $(d_r, Q_r), DB : (id_n, K_{n_1}, K_{n_2})_n$	Tag- $(id_n, K_{n_1}, K_{n_2}, Q_r)$
Choose $r_r, R_r = r_r G = (x_r, y_r)$	Choose $r_n, R_n = (r_n + K_{n_2})G$ $H((r_n + K_{n_2})(Q_r + R_r)) = (hr_1, hr_2)$ $A_1 = K_{n_1} \oplus hr_1$ $H(id_n \ K_{n_1} \ K_{n_2} \ R_r \ R_n \ hr_2) = (h_1 \ h_2)$
	$\xleftarrow{R_n, A_1, h_2}$
$(d_r + r_r)R_n = (hr_1, hr_2)$ $K_{n_1} = A_1 \oplus hr_1$ If not K_{n_1} in DB , stop, else $H(id_n \ K_{n_1} \ K_{n_2} \ R_r \ R_n \ hr_2) = (h_1 \ h_2)$ If h_2 not correct, stop, else	
	$\xrightarrow{h_1}$
	If h_1 correct OK, else not

TABLE 2: Steps and computations in the proposed authentication without storage.

Reader- $(d_r, Q_r), DB : (id_{rn}, Cert_{rn})_{rn}$	Tag- $(id_n, Cert_n, (d_n, Q_n), Q_r)$
Choose $r_r, R_r = r_r G$	Choose $r_n, R_n = (r_n + d_n)G$ Choose $r, C = (id_n \ Cert_n \ r) \oplus (r_n + d_n)(Q_r + R_r)$ $H(id_n \ Cert_n \ R_r \ R_n \ r) = (h_1 \ h_2)$ $s_n = (r_n + d_n) - d_n h_2$
	$\xleftarrow{R_n, C, s_n}$
$(id_n \ Cert_n \ r) = C \oplus (d_r + r_r)R_n$ If id_n in DB , stop, else $H(id_n \ Cert_n \ R_r \ R_n \ r) = (h_1 \ h_2)$ If not $s_n G = R_n - h_2 Q_n$, stop, else	
	$\xrightarrow{h_1}$
	If h_1 correct OK, else not

TABLE 3: Results in the different attack scenarios.

Characteristic	Scheme 1.1	Scheme 1.2
[A1] Active attacker—send queries	Send(0, R) = $\{R_r\}$, send(1, T) = $\{R_n, A_1, h_2\}$, send(2, R) = $\{h_1\}$	Send(0, R) = $\{R_r\}$, send(1, T) = $\{C, R_n, s_n\}$, send(2, R) = $\{h_1\}$
[A2] Passive attacker—execute queries	Execute(send(0, R)) = $\{R_r\}$, execute(send(1, T)) = $\{R_n, A_1, h_2\}$, execute(send(2, R)) = $\{h_1\}$	Execute(send(0, R)) = $\{R_r\}$, execute(send(1, T)) = $\{C, R_n, s_n\}$, execute(send(2, R)) = $\{h_1\}$
[A3] Session-specific temporary information—SSReveal queries	SSReveal(T) = $\{r_n\}$, SSReveal(R) = $\{r_r\}$	SSReveal(T) = $\{r_n, r\}$, SSReveal(R) = $\{r_r\}$
[A4] Long-term key material—corrupt queries	Corrupt(T) = $\{id_n, K_{n_1}, K_{n_2}\}$, corrupt(R) = $\{d_r, (id_n, K_{n_1}, K_{n_2})_n\}$	Corrupt(T) = $\{d_n, id_n, Cert_n\}$, corrupt(R) = $\{d_r\}$

key by computing $K = (r_n + d_n)(Q_r + R_r) = (x_k, y_k)$, which is used to encrypt the identity-related information, i.e., $C = E_K(id_n \| Cert_n \| r) = (id_n \| Cert_n \| r) \oplus K$, with r a randomly chosen

value. Then, it computes $H(id_n \| Cert_n \| R_r \| R_n \| r) = (h_1 \| h_2)$ and the corresponding signature $s_n = (r_n + d_n) - d_n h_2$. The message C, R_n, s_n is sent to the reader.

TABLE 4: Informal security analysis of both protocols.

Characteristic	Scheme 1.1	Scheme 1.2
[C1] Mutual authentication	In order to be able to construct a valid outcome in the protocol, the attacker needs to construct a legitimate h_2 (in the case of the tag) or h_1 (in the case of the reader). To do so, the attacker should know at the same time either $\{id_n, K_{n_1}, K_{n_2}, r_n\}$ or $\{id_n, K_{n_1}, K_{n_2}, r_r, d_r\}$. This is not possible, since both sets contain outcomes of both SSReveal and corrupt (see Table 3), which is not considered in the attack model. In addition, the hash function is assumed to be resistant for collision and preimage attacks.	In order to construct a valid signature s_n by the tag or valid response h_1 by the reader, the attacker needs to possess at the same time either $\{d_n, r_n\}$ or $\{d_r, r_r\}$ due to the ECDLP as these values require the computation of the session key $K = (d_n + r_n)(Q_r + R_r) = (d_r + r_r)R_n$ used to encrypt the message $(id_n, Cert_n, r)$. Note that this key is not stored as a separate variable in the protocol in order to avoid exploitation in a session-specific temporary information attack. Also, here, these sets contain outcomes of both SSReveal and corrupt (see Table 3), which is a contradiction in the attack model.
[C2] Anonymity	The identity id_n is considered as secret key material and is only included in the hash function. Due to the one-way property of the hash function, its value cannot be revealed. It can even not be guessed, since that allows the knowledge of K_{n_1} , which is hidden in A_1 relying on the strength of the ECDLP.	To find the identity of the tag, the message C should be decrypted. Therefore, either $\{d_n, r_n\}$ or $\{d_r, r_r\}$ is required due to the ECDHP. However, as mentioned before, this is in contradiction with the attack model, cf Table 3. If also the identity and certificate are guessed, the validity cannot be verified as it requires the knowledge of a random value r , which is included in the encrypted message C.
[C3] Unlinkability	As explained in the criteria of anonymity, the identity of the tag cannot be revealed, but the best to be leaked is K_{n_1} from A_1 , which would threaten the unlinkability feature. However, this is not possible as it requires the knowledge of either $\{r_n, K_{n_2}\}$ or $\{d_r, r_r\}$, which is not possible, cf Table 3.	Identity-related information is packed in the parameters $\{C, s_n, h_1\}$. From C, nothing can be leaked as explained before. From the second one s_n , an attacker can try to verify the signature with a collected public key of the tag (similar to dictionary attack). However, now, the attacker will not be successful as it first needs to find h_2 , which includes the additional random value r , to be able to verify the equation $s_n G = R_n - h_2 Q_n$. On the other hand, if $\{id_n, Cert_n\}$ is leaked, similar to a dictionary attack, together with the session data r , the validity of h_1 can be verified and thus, the tag can be tracked.
[C4] Session-specific temporary information	Given $SSReveal(T)=r_n$, it still does not result in making a fake message containing A_1, h_2, h_1 as also corresponding secret keys K_{n_1}, K_{n_2} of the tag need to be known. Also, with additional $SSReveal(R)=r_r$, no fake messages can be constructed.	With $SSReveal(T) = \{r_n, r\}$ and $SSReveal(R) = \{r_r\}$, no fake messages containing legitimate C, s_n, h_1 parameters can be made without knowledge of valid credentials of the tag as it requires the additional usage of either the private key d_n of the tag or the private key d_r of the reader due to the ECDLP.
[C5] Perfect forward secrecy	When the corrupt queries of tag and reader are executed, we are still not able to create a fake message containing A_1, h_2, h_1 as also the session-specific information r_n or r_r should be known to derive the last concatenated value in the hash operation due to the ECDHP. In addition, the identity cannot be tracked as in the computation of A_1 , temporary session information is included.	Executing the corrupt queries on both reader and tag does not result in additional gain for the attacker as still the temporary session information is required to encrypt the ciphertext for deriving the identity of the tag and for constructing valid hash outcomes. This follows from the ECDHP.
[C6] Replay attacks	This attack is avoided thanks to the usage of random numbers, both at the side of the reader and the tag. Even if the attacker replays the random value of the reader in the beginning of the protocol, the tag still chooses another random value, such that the h_1 value also changes. Also, the message of the tag cannot be replayed as it relies on the usage of the random R_r sent in the beginning by the reader. Due to the large size p of the space (field size of the curve) in which the random variables are defined, the probability to obtain a collision is following the birthday paradox smaller than $n^2/2p$, with n the number of collected variables.	The same reasoning for the protection against replay attack holds for this protocol. Again, random values are used by both reader and tag and should be unique in each call of the protocol. The probability of not being unique is smaller than $n^2/2l$, with l the space size in which the randoms are defined and thus equal to the field size.
[C7] Desynchronisation attack	Since the identity and key material of the tag remain fixed, there is no desynchronisation possible. The only potential problem might be if the tag has a built-in protection to	The similar reasoning as in Scheme 1.1 holds due to the fixed key material of the tag and results in protection against desynchronisation attacks.

TABLE 4: Continued.

Characteristic	Scheme 1.1	Scheme 1.2
[C8] Scalability	avoid denial-of-service attacks to limit the number of unsuccessful authentication requests by a fixed amount. This scheme is linearly scalable with the amount of sensors stored at the reader side, since the reader has to go linearly over the different records of its database in order to find the matching record.	Since the reader only stores the list of revoked sensors, the system is highly scalable with a complexity limited to the size of the revoked sensors and not the list of legitimate sensors, which is typically considered to be much smaller.
[C9] Denial-of-service attack	The validity of the second and third message can be immediately verified. Only for the first message the tag needs to wait upon the final response in order to decide the legitimacy of the authentication request. However, from the side of the reader, there is no problem as each request can be immediately verified and no sessions need to be stored waiting for approval.	The same protection against denial-of-service attacks holds for Scheme 1.2 as also, here, the first message is the only one which cannot be immediately verified for its legitimacy and this message is sent from the reader to the tag.

After receiving this message, the reader derives $K = (d_r + r_r)R_n$ in order to retrieve $(id_n \| Cert_n \| r) = D_K(C) = C \oplus K$. Next, it checks the validity of the signature by computing $H(id_n \| Cert_n \| R_r \| R_n \| r) = (h_1 \| h_2)$ and checking if the equality $s_n G = R_n - h_2 Q_n$ holds. If so, the reader sends h_1 as a confirmation message to the tag. Else, the protocol is aborted.

If the tag receives the same value h_1 as stored in its memory, the authentication process is successfully executed. The different steps in this protocol are illustrated in Table 2.

4.2. Extension. In the current two versions, the tag receives a random value from the reader but is not able to verify the validity of the request. This will not cause denial-of-service attacks as it is possible to build in a security guideline to block the number of potential requests to a certain threshold.

However, in order to avoid such potential problems, it is possible to replace the first message in both protocols by a message containing the signature of the TTP on the random value generated by the reader. In this way, any reader with a valid signature can send a request to the tag and the tag does not need to store the corresponding public key of the reader.

Let R_r be the random value generated by the reader. The reader requests a signature of the TTP on the pair (R_r, Q_r) , which can be easily realised by the Schnorr signature scheme as also done in Scheme 1.2. Therefore, the TTP chooses a random value r_1 and computes $R_1 = r_1 G$. The corresponding signature $s_r = r_1 - H(R_r \| Q_r \| R_1 \| T_i) d_{TTP}$, with T_i being the current active time interval, is now sent to the reader. As a consequence, the first transmitted message now consists of $(R_r, Q_r, R_1, T_i, s_r)$. This signature can be easily verified by the tag by checking if the following equality holds: $s_r G = R_1 - H(R_r \| Q_r \| R_1 \| T_i) Q_{TTP}$.

The tag does not need to prestore the public key of the reader Q_r anymore, but only the initialisation timestamp T_0 . It is implemented to accept random values within a certain given time interval from the initial T_0 . There is a trade-off in size of this time interval (cf efficiency) and security as the longer the time interval, the more it can become possible that revoked readers can replay their approved random values. In the case that all readers are trustworthy, the usage of timestamps can even be avoided. Note that for Schemes 1.1 and 1.2, we have assumed a trustworthy reader, as there is only

one reader involved. We will call Schemes 1.1 and 1.2 with the extension part also Schemes 2.1 and 2.2.

5. Security Evaluation

The security will be both informally and formally evaluated.

5.1. Informal Security Evaluation. We now discuss the strength of both protocols with respect to the attack scenarios and the security features defined in Section 3. We also compare the security strength of our scheme with the most relevant related work described in Section 2.

In order to analyze the security, we first summarize the knowledge that the attacker gains following the different attack scenarios [A1-A4] in Table 3, being the results of send, execute, session reveal, and corrupt queries, respectively. We want to note that hr_1, hr_2 are no session-specific temporary information at the tag side as it is possible to compute $K_{n_1} \oplus H((r_n + K_{n_2})(Q_r + R_r))$ and to take the first 128 bits of the operation as A_1 and the bits 129 till 256 as hr_2 , since K_{n_1} is considered to be a variable of length 128 bits.

This knowledge can then be exploited in the different attacks, as explained in Table 4.

It can be concluded from Table 4 that both our Schemes 1.1 and 1.2 satisfy the same security features with respect to the above defined criteria [C1-C9]. However, the main difference from a security point of view between both of them is with respect to the resistance against a curious TTP. In Scheme 1.1, with knowledge of the session-specific variables, thus after SSReveal(T), the TTP is able to verify and track the activity of the tag. The TTP can also impersonate the tag as it knows all the security material. Compared to Scheme 1.2, both attacks are not possible if the private key of the tag has been constructed by means of the ECQV mechanism. Consequently, the TTP can never be in the possession of both session data and secret data, like in Scheme 1.1.

In addition, the main added value from a security point of view of Schemes 2.1 and 2.2, including the implementation of the extension in the beginning of the algorithm, is to immediately detect denial-of-service attacks at the tag side. In the case where many invalid requests are sent, it makes sense to include this extension. In addition, it makes the protocol

FIGURE 2: ProVerif structure.

TABLE 5: Comparison of security strength with related and recent literature with respect to the security criteria [C1-C11] (Y: yes; N: no; P: partially).

Scheme (authors + year)	[C1]	[C2]	[C3]	[C4]	[C5]	[C6]	[C7]	[C8]	[C9]	[C10]	[C11]
Braeken et al. 2018 [29]	Y	Y	Y	Y	Y	Y	Y	Y	P	Y	N
Sowjanya et al. 2020 [34]	Y	Y	Y	N	Y	Y	Y	N	Y	N	N
Dinarvand and Barati 2019 [38]	Y	Y	Y	Y	P	Y	Y	Y	N	N	N
Merabet et al., Scheme 1, 2019 [43]	Y	Y	N	N	N	Y	Y	Y	N	N	N
Merabet et al., Scheme 2, 2019 [43]	Y	Y	N	N	N	Y	Y	Y	N	N	Y
Scheme 1.1	Y	Y	Y	Y	Y	Y	Y	Y	P	N	N
Scheme 1.2	Y	Y	Y	Y	Y	Y	Y	Y	P	Y	N
Scheme 2.1	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
Scheme 2.2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

TABLE 6: Comparison of the computation costs.

Scheme (authors + year)	T_M reader	Time reader	T_M tag	Time tag	Total time
Braeken et al. 2018 [29]	5	189.75 ms	4	1378.64 ms	1568.39 ms
Sowjanya et al. 2020 [34]	5	189.75 ms	3	1033.66 ms	1223.41 ms
Dinarvand et al. 2019 [38]	3	113.85 ms	3	1033.66 ms	1167.51 ms
Merabet et al., Scheme 1, 2019 [43]	3	113.85 ms	3	1033.66 ms	1167.51 ms
Merabet et al., Scheme 2, 2019 [43]	2	75.88 ms	2	689.32 ms	765.20 ms
Scheme 1.1	2	75.88 ms	2	689.32 ms	765.20 ms
Scheme 1.2	4	151.80 ms	2	689.32 ms	841.12 ms
Scheme 2.1	4	151.80 ms	5	1723.30 ms	1875.10 ms
Scheme 2.2	4	151.80 ms	4	1378.64 ms	1430.44 ms

TABLE 7: Comparison of the communication costs.

Scheme (authors + year)	N_S (bits)	N_R (bits)	Total bits	Nr of phases
Braeken et al. 2018 [29]	896	512	1408	3
Sowjanya et al. 2020 [34]	800	512	1312	2
Dinarvand et al. 2019 [38]	768	512	1280	4
Merabet et al., Scheme 1, 2019 [43]	768	512	1280	3
Merabet et al., Scheme 2, 2019 [43]	768	512	1280	3
Scheme 1.1	768	512	1280	3
Scheme 1.2	1024	512	1536	3
Scheme 2.1	768	1568	2336	3
Scheme 2.2	1024	1568	2592	3

independent of one particular reader and allows authentication of the tag to a large number of readers. However, it comes of course with an additional communication and computation cost as will be shown in Section 6.

To conclude, we have compared the security of our four protocols with the latest most relevant schemes [29, 34, 38, 43] in literature as discussed in Section 3, considering an attack satisfying the attack scenarios [A1-A4]. We also added two other criteria, as explained above, in order to differentiate our four developed schemes.

(C10) Resistant against curious TTP

(C11) Multireader authentication

The comparison is provided in Table 5. As can be seen, our Scheme 2.2 satisfies all the criteria, while the other Schemes 1.1 and 1.2 still satisfy a nice number of criteria. In particular, with respect to [C1-C9], they clearly outperform the state of the art [34, 38, 43]. Only [29] possesses similar strength as it satisfies resistance in the CK adversary model that is even a stronger requirement, which also attack scenarios [A3] and [A4] can be combined.

Only the second scheme of [43] also offers authentication with respect to multiple readers. With respect to inherent architecture, it is best to compare their proposed scheme with Scheme 2.1 as it also requires the storage of legitimate identities. As can be seen from Table 5, the scheme [43] does not possess sufficient security strength.

5.2. Formal Security Evaluation. Formal verification is a successful approach to provide security assurances by mathematically ascertaining the accuracy of prototypes utilizing a combination of mathematical and logical methods. Such approaches are especially valuable for extracting objective information on the protection and security properties of digital devices.

ProVerif is one of the most common used formal verification tools focusing on security protocols. We refer to [53] for an in-depth survey on the different formal security tools. *ProVerif* is a command line method to automatically evaluate the protection of cryptographic protocols based on the basic description of the protocol by the Prolog rules. In *ProVerif*, an algorithm is implemented that effectively determines whether or not a fact can be proven by the Prolog rules [13,

54–56]. Due to the usage of integration, the state space explosion issue is prevented. Another benefit is that there is no need to restrict the amount of runs of the protocol.

The *ProVerif* structure is shown in Figure 2. *ProVerif* uses the protocol paradigm in the extension of the pi calculus with cryptography as data, analogous to the modified pi calculus. It follows a broad range of cryptographic primitives, defined by rewrite codes or equations. *ProVerif* also uses the protection properties that we want to show as data. Numerous protection features, like anonymity, authentication, and certain empirical equivalence products, can be checked. It instantly converts this knowledge into an internal version of the Horn clauses: the protocol is translated into a collection of Horn clauses, and the protection properties to be proven are translated into the derivability queries for such clauses. *ProVerif* uses a resolution algorithm with free choice to determine whether a fact can be derived from the clauses. Unless the facts are not derivable, the required protection property shall be identified.

Some more background and the *ProVerif* modeling are given in Appendix. Also, the results on the proof verification have been explained there. To summarize, the program reveals for the first two schemes a potential flow on the random value R_r , which has also been described in the informal security analysis and shown to imply no direct security attack as shown in Appendices A and B. The last two schemes do not reveal any security threats, please see Appendices C and D.

6. Performance

We now compare the efficiency of our four schemes with respect to computation and communication costs.

6.1. Computation Cost. Our four protocols and the related work of [29, 34, 38] consist of a combination of xor, hash, and elliptic curve operations. From all these operations, the EC multiplication is the most compute-intensive operation and dominates the total cost of the process. Therefore, we limit our analysis to the comparison of the amount of EC multiplications required by both reader and tag. Denote this number by T_M . To give an estimation of the timings, we implemented an EC multiplication with 256 bits, corresponding to a 128 security level, both on a Zolertia RE-

Mote with ARM Cortex-M3 at 32 MHz and 32 KB RAM using the HW acceleration engine and on a Raspberry Pi 3B Quad Core 1.2 GHz with 1 GB RAM using the Bouncy-Castle library. As a result, we obtained 344.66 ms for the first one and 37.94 ms for the second one, mimicking the tag and reader, respectively.

Table 6 compares the number of EC multiplications T_M at reader and tag side, together with their corresponding timings. As can be concluded from this table, our Schemes 1.1 and 1.2 are the fastest as they both require only 2 EC multiplications. In addition, Scheme 1.1 also only requires 2 EC multiplications at the reader side and thus has the lowest impact with respect to security operations.

6.2. Communication Cost. We now calculate the length of the transmitted messages sent in the protocols, taking into account a security level of 128 bits. This means that the size of the EC points and the outputs of the hash function should be at least 256 bits in order to avoid all types of collision and preimage attacks. We further consider the length of the identities equal to 128 bits in order to avoid exhaustive search attacks. The number of sent bits is denoted by N_S , and the number of received bits by N_R from the point of view of the tag.

Table 7 summarizes the comparison of the performance with respect to communication costs. It can be concluded that our Scheme 1.1 is outperforming the others with respect to number of bits communicated in the channel. Only [38] is able to establish the authentication in two phases. However, it should be noted that in practice, an additional phase should be added as in their implementation, the tag starts the communication and does not receive an approval of the authentication. In order to effectively obtain mutual authentication, at least three phases are needed. It is clear that Schemes 2.1 and 2.2 have the highest communication cost, but they are also the most generic as they are able to handle different readers in the same protocol.

7. Conclusion

This paper introduces four highly effective authentication algorithms for client-server (tag-reader) or RFID system based on ECC. The first scheme (v1.1) requires storage of the identities and related key material of all active tags, while the second scheme (v1.2) only needs the list of revoked tags. An additional phase is proposed on both protocols (v2.1 and v2.2) to enable multireader authentication. The main difference in security strength (between .1 and .2 version) is the additional protection against an honest but curious TTP, offered by the second version.

The proposed schemes were compared with five other recent ECC-based protocols for the computational time, communication costs, storage costs, and protection level they offer. The findings reveal that our proposed schemes have small computing overheads, small storage costs for the code, and better protection rates than other related protocols (see Tables 6 and 7).

7.1. ProVerif for Specifications on the Schemes. In [54], the manual of the *ProVerif* tool includes explanation on the installation, configuration, and programming syntaxes and is provided with rich examples for a good understanding. When writing a file in *ProVerif*, you model what the honest participants are allowed to do and what you give as knowledge to the attacker. Then, *ProVerif* checks any possible interaction between the honest participant and the attacker (provided that the attacker actions are bound by the model, e.g., an attacker cannot break the cryptography). After that, if you have specific properties, you need to think how to model them.

Appendix

A. Applied Pi Calculus Specification Script of Scheme 1.1

Explanations of the pi calculus codes used are as follows:

- (1) The secrets T_{idn} , T_{Kn1} , T_{Kn2} , and dr are declared as a secret to the attacker using the word [private]. ch is the public channel where READER and TAG exchange their messages. The functions h , $mecc$, $concat$, xor , and add represent hash, scalar multiplication, concatenation, xor, and modular addition functions, respectively, and are defined in the library
- (2) Secrecy of T_{idn} , T_{Kn1} , T_{Kn2} , dr , and rn is verified with the queries: *query attacker*(T_{idn}), *query attacker*(T_{Kn1}), *query attacker*(T_{Kn2}), *query attacker*($new\ rr$), and *query attacker*($new\ rn$)
- (3) Mutual authentication between the Tag and Server is modeled by defining four events that are mapped in the TAG and SERVER subprocesses and a set of queries, shown in Figure 3(a)
- (4) The pi calculus is designed for representing concurrent processes that interact using communication channels. First, we should model the protocol, which means READER and TAG tasks, as provided in Figures 3(b) and 3(c)
- (5) Then, the process is defined (see [55]). In our case, we used a parallel composition. Because the processes in *CryptoVerif* must alternate inputs and outputs, input must be followed by computations and output, and output must be followed by replications, parallel compositions, and inputs. The main process must start with an input, a replication, or a parallel composition

In the main process, SERVER and TAG subprocesses are running in parallel, indicating an unlimited number of processes, presented in Figure 3(d).

B. ProVerif Results of Scheme 1.1

It is necessary to consider the difference between the attack derivation and the attack trace in order to interpret the findings correctly. The attack derivation is a description of the

```

free ch:channel.
free T_idn, T_Kn1, T_Kn2, dr: bitstring [private].

fun h (bitstring): bitstring [data].
fun mecc (bitstring, bitstring):bitstring [data].
fun concat (bitstring, bitstring):bitstring.
fun add (bitstring, bitstring):bitstring.
fun xor (bitstring,bitstring): bitstring.
equation forall x:bitstring, y:bitstring; xor(xor(x,y), y)=x.

query attacker(T_idn).
query attacker(T_Kn1).
query attacker(T_Kn2).
query attacker(dr).
query attacker(new rr).
query attacker(new rn).

event beginT(bitstring).
event endT(bitstring).
event beginR(bitstring).
event endR(bitstring).

query x: bitstring; inj-event(beginR(x)) ==> inj-event(beginT(x)).
query x: bitstring; inj-event(endT(x)) ==> inj-event(endR(x)).
query x: bitstring, y:bitstring; inj-event(endR(y)) ==> inj-event(
beginR(x)) && inj-event(beginT(x)).

```

(a) Parameters and functions

```

(*READER SIDE*)

let READER(dr: bitstring, Qr:bitstring, idn:bitstring,
Kn1:bitstring, Kn2:bitstring, G:bitstring) =
  new rr:bitstring;
  let RR=mecc(rr, G) in
  out(ch, RR);
  in(ch, (mt1:bitstring, mt2:bitstring, mt3:bitstring));
  let (hrr1: bitstring, hrr2:bitstring) = mecc(add(dr,rr), mt1) in
  event beginR(hrr1);
  let Knx= xor(mt2, hrr1) in
  if Knx=Kn1 then
    let (hr1:bitstring, hr2:bitstring)= h(concat(idn,concat(
Knx, concat(Kn2,concat(RR,concat(mt1, hrr2)))))) in
    if hr2=mt3 then
      out(ch, hr1);
      event endR(hr1).

```

(b) Reader side

```

(*TAG SIDE*)

let TAG(T_idn:bitstring, T_Kn1:bitstring, T_Kn2:bitstring,
T_Qr:bitstring, T_G:bitstring)=
  in(ch, mr1:bitstring);
  new rn:bitstring;
  let RN=mecc(add(rn, T_Kn2), T_G) in
  let (hrt1:bitstring, hrt2:bitstring)=h(mecc(add(rn, T_Kn2),mecc(
T_Qr, mr1 ))) in
  let A=xor(T_Kn1, hrt1) in
  let (ht1:bitstring, ht2:bitstring)= h(concat(T_idn, concat(
T_Kn1,concat(T_Kn2, concat(mr1, concat(RN, hrt2)))))) in
  event beginT(ht2);
  out(ch, (RN, A, ht2));
  in(ch, mr2:bitstring);
  if ht1=mr2 then event endT(ht1).

```

(c) Tag side

```

-- Query not attacker(T_idn[]) in process 0.
Completing...
Starting query not attacker(T_idn[])
RESULT not attacker(T_idn[]) is true.
-- Query not attacker(T_Kn1[]) in process 0.
Completing...
Starting query not attacker(T_Kn1[])
RESULT not attacker(T_Kn1[]) is true.
-- Query not attacker(T_Kn2[]) in process 0.
Completing...
Starting query not attacker(T_Kn2[])
RESULT not attacker(T_Kn2[]) is true.
-- Query not attacker(dr[]) in process 0.
Completing...
Starting query not attacker(dr[])
RESULT not attacker(dr[]) is true.

```

(d) Main process

FIGURE 3: Applied Pi calculus specification script of Scheme 1.1.

behavior that the attacker has to take to breach the protection property in the internal representation of *ProVerif*. As this internal interpretation utilizes abstractions, the derivation is not necessarily achievable in reality. For example, it may involve the replication of such acts which, in truth, may never be repeated, for example, since they are not reproduced. The attack trace, on the other hand, refers to the semantics of the applied pi calculus and often corresponds to the functional trace of the considered operation. *ProVerif* is able to show three kinds of results:

- (i) RESULT [Query] is true: the query has been confirmed, there is no threat. In this scenario, *ProVerif* reveals no attack derivation and no attack trace
- (ii) RESULT [Query] is false: the query is incorrect, and *ProVerif* has discovered an intrusion on the required protection property. The attack trace is seen right before the outcome (and the attack derivation is also seen, although you can concentrate on the attack trace because it reflects the real attack).
- (iii) RESULT [Query] cannot be proved: it is a “do not know” reaction. *ProVerif* did not claim that the query is valid, nor could it locate an attack that proved that the question is incorrect. Because the question of checking protocols with an unbounded number of sessions is undecidable, this condition is unavoidable. Still, *ProVerif* offers some supplemen-

tary knowledge that could be helpful in deciding if the question is valid. In fact, *ProVerif* shows an attack bypass

In the verification summary, shown in Figure 4(a), we notice that an attack can derive R_r , which is logic because it is sent in the public channel. But as mentioned in the security analysis, the knowledge of R_r does not lead to a direct attack (see Figures 4(c) and 4(d)). For the rest of the results, there is no threat (see Figures 4(b) and 4(e)).

C. Applied Pi Calculus Specification Script of Scheme 2.1

As explained before, to avoid such potential problems, we replace the first message protocol by a message containing the signature (sr) of the TTP on the random value generated by the reader.

Explanations of the pi calculus codes used are as follows:

- (1) The secrets T_idn , T_Kn1 , T_Kn2 , and dr are declared as a secret to the attacker using the word [private]. ch is the public channel where READER and TAG exchange their messages and h , $mecc$, $concat$, xor , add , and sub represent hash, scalar multiplication, concatenation, xor, modular addition, and modular subtraction functions, respectively


```

Verification summary:
Query not attacker(T_idn[]) is true.
Query not attacker(T_Kn1[]) is true.
Query not attacker(T_Kn2[]) is true.
Query not attacker(dr[]) is true.
Query not attacker(rn[!1 = v]) is false.
Query not attacker(rn[mr1 = v, !1 = v_1]) is true.
Query inj-event(beginR(x_1)) ==> inj-event(beginT(x_1)) is true.
Query inj-event(endT(x_1)) ==> inj-event(endR(x_1)) is true.
Query inj-event(endR(y_1)) ==> inj-event(beginR(x_1)) && inj-event(beginT(x_1)) is true.

```

(a) Verification summary

```

-- Query not attacker(T_idn[]) in process 0.
Completing...
Starting query not attacker(T_idn[])
RESULT not attacker(T_idn[]) is true.
-- Query not attacker(T_Kn1[]) in process 0.
Completing...
Starting query not attacker(T_Kn1[])
RESULT not attacker(T_Kn1[]) is true.
-- Query not attacker(T_Kn2[]) in process 0.
Completing...
Starting query not attacker(T_Kn2[])
RESULT not attacker(T_Kn2[]) is true.
-- Query not attacker(dr[]) in process 0.
Completing...
Starting query not attacker(dr[])
RESULT not attacker(dr[]) is true.

```

(b) Detailed result 1

```

-- Query not attacker(rn[!1 = v]) in process 0.
Completing...
Starting query not attacker(rn[!1 = v])
goal reachable: attacker(rn_1)
Abbreviations:
rr_1 = rn[!1 = @sid]
Derivation:
Abbreviations:
rr_1 = rn[!1 = @sid]
1. The message mecc(rn_1,G[]) may be sent to the attacker at output {15}.
attacker(mecc(rn_1,G[])).
2. By 1, the attacker may know mecc(rn_1,G[]).
Using the function 1-proj-mecc the attacker may obtain rr_1.
attacker(rr_1).
3. By 2, attacker(rr_1).
The goal is reached, represented in the following fact:
attacker(rr_1).
A more detailed output of the traces is available with
set traceDisplay = long.
new dr_1: bitstring creating dr_2 at {1}
new Qr: bitstring creating Qr_1 at {2}
new idn: bitstring creating idn_1 at {3}
new Kn1: bitstring creating Kn1_1 at {4}
new Kn2: bitstring creating Kn2_1 at {5}
new G: bitstring creating G_1 at {6}

```

(c) Detailed result 2

```

new Kn1: bitstring creating Kn1_1 at {4}
new Kn2: bitstring creating Kn2_1 at {5}
new G: bitstring creating G_1 at {6}
new T_idn_1: bitstring creating T_idn_2 at {7}
new T_Kn1_1: bitstring creating T_Kn1_2 at {8}
new T_Kn2_1: bitstring creating T_Kn2_2 at {9}
new T_Qr: bitstring creating T_Qr_1 at {10}
new T_G: bitstring creating T_G_1 at {11}
new rr: bitstring creating rr_1 at {13} in copy a
out(ch, mecc(~M,~M_1)) with ~M = rr_1, ~M_1 = G_1 at {15} in copy a
The attacker has the message ~M = rr_1.
A trace has been found.
The previous trace falsifies the query, because the query is
simple and the trace corresponds to the derivation.
RESULT not attacker(rn[!1 = v]) is false.

```

(d) Detailed result 3

```

-- Query not attacker(rn[mr1 = v, !1 = v_1]) in process 0.
Completing...
Starting query not attacker(rn[mr1 = v, !1 = v_1])
RESULT not attacker(rn[mr1 = v, !1 = v_1]) is true.
-- Query inj-event(beginR(x_1)) ==> inj-event(beginT(x_1)) in process 0.
Completing...
Starting query inj-event(beginR(x_1)) ==> inj-event(beginT(x_1))
RESULT inj-event(beginR(x_1)) ==> inj-event(beginT(x_1)) is true.
-- Query inj-event(endT(x_1)) ==> inj-event(endR(x_1)) in process 0.
Completing...
Starting query inj-event(endT(x_1)) ==> inj-event(endR(x_1))
RESULT inj-event(endT(x_1)) ==> inj-event(endR(x_1)) is true.
-- Query inj-event(endR(y_1)) ==> inj-event(beginR(x_1)) && inj-event(beginT(x_1)) in process 0.
Completing...
Starting query inj-event(endR(y_1)) ==> inj-event(beginR(x_1)) && inj-event(beginT(x_1))
RESULT inj-event(endR(y_1)) ==> inj-event(beginR(x_1)) && inj-event(beginT(x_1)) is true.

```

(e) Detailed result 4

FIGURE 4: ProVerif results of Scheme 1.1.

- (2) Secrecy of T_idn , T_Kn1 , T_Kn2 , dr , and rn is verified with queries: *query attacker(T_idn)*, *query attacker(T_Kn1)*, *query attacker(T_Kn2)*, *query attacker(sr)*, and *query attacker(new rn)*
- (3) Mutual authentication between the Tag and Server is modeled with the definition of four events that are mapped in the *TAG* and *SERVER* subprocesses and a set of queries, shown in Figure 5(a)
- (4) *READER* and *TAG* tasks, Figures 5(b) and 5(c).

- (5) In the main process, *SERVER* and *TAG* subprocesses are running in parallel, indicating an unlimited number of processes, shown in Figure 5(d)

D. ProVerif Results of Scheme 2.1

After signing the reader's random R_r by TTP, the protocol became more secure as the results show that there are no safety issues as mentioned in Figures 6(a) and 6(b).

```

free ch:channel.
free T_idn, T_Kn1, T_Kn2, dr: bitstring [private].

fun h (bitstring): bitstring [data].
fun mecc (bitstring, bitstring):bitstring [data].
fun concat (bitstring, bitstring):bitstring.
fun add (bitstring, bitstring):bitstring.
fun sub (bitstring, bitstring):bitstring.
fun xor (bitstring,bitstring): bitstring.
equation forall x:bitstring, y:bitstring; xor(xor(x,y), y)=x.

query attacker(T_idn).
query attacker(T_Kn1).
query attacker(T_Kn2).
query attacker(dr).
query attacker(new sr).

event beginT(bitstring).
event endT(bitstring).
event beginR(bitstring).
event endR(bitstring).

query x: bitstring; inj-event(beginR(x)) ==> inj-event(beginT(x)).
query x: bitstring; inj-event(endT(x)) ==> inj-event(endR(x)).
query x: bitstring, y:bitstring; inj-event(endR(y)) ==> inj-event(beginR(x)) && inj-event(beginT(x)).

```

(a) Parameters and functions

```

(*READER SIDE*)

let READER(dr: bitstring, Qr:bitstring, idn:bitstring, Kn1:bitstring,
Kn2:bitstring, G:bitstring, Ti:bitstring, r1:bitstring, Dttp:bitstring) =
  new rr:bitstring;
  let RR=mecc(rr, G) in
  let R1=mecc(r1, G) in
  new sr:bitstring;

  let sr=sub(r1, mecc( h(concat(RR, concat(Qr, concat(R1, Ti))))), Dttp)
  ) in

  out(ch, (RR, Qr, R1, Ti, sr));
  in(ch, (mt1:bitstring, mt2:bitstring, mt3:bitstring));
  let (hrr1: bitstring, hrr2:bitstring) = mecc(add(dr,rr), mt1) in
  event beginR(hrr1);
  let Knx= xor(mt2, hrr1) in

  if Knx=Kn1 then
    let (hr1:bitstring, hr2:bitstring)= h(concat(idn,concat(Knx,
concat(Kn2,concat(RR,concat(mt1, hrr2)))))) in
    if hr2=mt3 then
      out(ch, hr1);
      event endR(hr1).

```

(b) Reader side

```

(*TAG SIDE*)

let TAG(T_idn:bitstring, T_Kn1:bitstring, T_Kn2:bitstring,
T_Qr:bitstring, T_G:bitstring)=
  in(ch, mr1:bitstring);
  new rn:bitstring;
  let RN=mecc(add(rn, T_Kn2), T_G) in
  let (hrt1:bitstring, hrt2:bitstring)=h(mecc(add(rn, T_Kn2),mecc(
T_Qr, mr1 ))) in
  let A=xor(T_Kn1, hrt1) in
  let (ht1:bitstring, ht2:bitstring)= h(concat(T_idn, concat(
T_Kn1,concat(T_Kn2, concat(mr1, concat(RN, hrt2)))))) in
  event beginT(ht2);
  out(ch, (RN, A, ht2));
  in(ch, mr2:bitstring);
  if ht1=mr2 then event endT(ht1).

```

(c) Tag side

```

(*START PROCESS*)

process
  new Ti:bitstring;
  new r1:bitstring;
  new Dttp:bitstring;
  new dr: bitstring;
  new Qr:bitstring;
  new idn:bitstring;
  new Kn1:bitstring;
  new Kn2:bitstring;
  new G:bitstring;
  new T_idn:bitstring;
  new T_Kn1:bitstring;
  new T_Kn2:bitstring;
  new T_Qr:bitstring;
  new T_G:bitstring;
  new Qttp:bitstring;

  ((!READER(dr, Qr, idn, Kn1, Kn2, G, Ti, r1, Dttp)) | (!TAG(T_idn,
T_Kn1, T_Kn2, T_Qr, T_G, Qttp)))

```

(d) Main process

FIGURE 5: Applied Pi calculus specification script of Scheme 2.1.

```

... Query not attacker(T_idn[]) in process 0.
Completing...
Starting query not attacker(T_idn[])
isSUT not attacker(T_idn[]) is true.
... Query not attacker(T_Kn1[]) in process 0.
Completing...
Starting query not attacker(T_Kn1[])
isSUT not attacker(T_Kn1[]) is true.
... Query not attacker(T_Kn2[]) in process 0.
Completing...
Starting query not attacker(T_Kn2[])
isSUT not attacker(T_Kn2[]) is true.
... Query not attacker(dr[]) in process 0.
Completing...
Starting query not attacker(dr[])
isSUT not attacker(dr[]) is true.
... Query not attacker(sr[1 = v]) in process 0.
Completing...
Starting query not attacker(sr[1 = v])
isSUT not attacker(sr[1 = v]) is true.
... Query inj-event(beginR(x_1)) ==> inj-event(beginT(x_1)) in process 0.
Completing...
Starting query inj-event(beginR(x_1)) ==> inj-event(beginT(x_1))
isSUT inj-event(beginR(x_1)) ==> inj-event(beginT(x_1)) is true.
... Query inj-event(endT(x_1)) ==> inj-event(endR(x_1)) in process 0.
Completing...
Starting query inj-event(endT(x_1)) ==> inj-event(endR(x_1))
isSUT inj-event(endT(x_1)) ==> inj-event(endR(x_1)) is true.
... Query inj-event(endR(y_1)) ==> inj-event(beginR(x_1)) && inj-event(beginT(x_1)) in process 0.
Completing...
Starting query inj-event(endR(y_1)) ==> inj-event(beginR(x_1)) && inj-event(beginT(x_1))
isSUT inj-event(endR(y_1)) ==> inj-event(beginR(x_1)) && inj-event(beginT(x_1)) is true.

```

(a) Detailed result

```

Verification summary:
Query not attacker(T_idn[]) is true.
Query not attacker(T_Kn1[]) is true.
Query not attacker(T_Kn2[]) is true.
Query not attacker(dr[]) is true.
Query not attacker(sr[1 = v]) is true.
Query inj-event(beginR(x_1)) ==> inj-event(beginT(x_1)) is true.
Query inj-event(endT(x_1)) ==> inj-event(endR(x_1)) is true.
Query inj-event(endR(y_1)) ==> inj-event(beginR(x_1)) && inj-event(beginT(x_1)) is true.

```

(b) Verification summary

FIGURE 6: ProVerif results of Scheme 2.1.

Abbreviations

RFID:	Radio Frequency Identification
ECC:	Elliptic curve cryptography
ECQV:	EC Qu Vanstone
CK:	Canetti-Krawczyk
TTP:	Rusted third party
ECDLP:	Elliptic Curve Discrete Logarithm Problem
ECDHP:	Elliptic Curve Diffie Hellman Problem
IoT:	Internet of Things.

Data Availability

Data can be made available upon request to the contact author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] RFID Forecasts, *Players and Opportunities 2019-2029*, IDTechEx, 2020.
- [2] X. Lin, R. Lu, D. Kwan, and X. S. Shen, "REACT: an RFID-based privacy-preserving children tracking scheme for large amusement parks," *Computer Networks*, vol. 54, no. 15, pp. 2744–2755, 2010.
- [3] D. C. Wyld, "Preventing the 'worst case scenario': combating the lost laptop epidemic with RFID technology," in *Novel Algorithms and Techniques in Telecommunications and Networking*, pp. 29–33, Springer, 2010.
- [4] S. W. Wang, W. H. Chen, C. S. Ong, L. Liu, and Y. W. Chuang, "RFID application in hospitals: a case study on a demonstration RFID project in a Taiwan hospital," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, vol. 8, p. 184a, Kauai, HI, USA, 2006.
- [5] M. M. Pérez, M. Cabrero-Canosa, J. V. Hermida et al., "Application of RFID technology in patient tracking and medication traceability in emergency care," *Journal of Medical Systems*, vol. 36, no. 6, pp. 3983–3993, 2012.
- [6] J. E. Katz and R. E. Rice, "Public views of mobile medical devices and services: a US national survey of consumer sentiments towards RFID healthcare technology," *International Journal of Medical Informatics*, vol. 78, no. 2, pp. 104–114, 2009.
- [7] A. Ohsaga and K. Kondoh, "Bedside medication safety management system using a PDA and RFID tags," in *2013 7th International Symposium on Medical Information and Communication Technology (ISMICT)*, pp. 85–89, Tokyo, Japan, 2013.
- [8] Y. T. Liao, T. L. Chen, T. S. Chen, Z. H. Zhong, and J. H. Hwang, "The application of RFID to healthcare management of nursing house," *Wireless Personal Communications*, vol. 91, no. 3, pp. 1237–1257, 2016.
- [9] K. Fan, W. Jiang, H. Li, and Y. Yang, "Lightweight RFID protocol for medical privacy protection in IoT," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 1656–1665, 2018.
- [10] S. Anandhi, R. Anitha, and V. Sureshkumar, "IoT enabled RFID authentication and secure object tracking system for smart logistics," *Wireless Personal Communications*, vol. 104, no. 2, pp. 543–560, 2019.
- [11] Y. K. Lee, L. Batina, and I. Verbauwhede, "EC-RAC (ECDLP based randomized access control): provably secure RFID authentication protocol," in *2008 IEEE international conference on RFID*, pp. 97–104, Las Vegas, NV, USA, 2008.
- [12] R. Want, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 25–33, 2006.
- [13] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," *csfw*, vol. 1, pp. 82–96, 2001.
- [14] A. Ibrahim and G. Dalkılıç, "Review of different classes of RFID authentication protocols," *Wireless Networks*, vol. 25, no. 3, pp. 961–974, 2019.
- [15] C. Jin, C. Xu, X. Zhang, and J. Zhao, "A secure RFID mutual authentication protocol for healthcare environments using elliptic curve cryptography," *Journal of Medical Systems*, vol. 39, no. 3, p. 24, 2015.
- [16] S. Rostampour, M. E. Namin, and M. Hosseinzadeh, "A novel mutual RFID authentication protocol with low complexity and high security," *International Journal of Modern Education and Computer Science (IJMECS)*, vol. 6, no. 1, pp. 17–24, 2014.
- [17] M. S. Farash, O. Nawaz, K. Mahmood, S. A. Chaudhry, and M. K. Khan, "A provably secure RFID authentication protocol based on elliptic curve for healthcare environments," *Journal of Medical Systems*, vol. 40, no. 7, p. 165, 2016.
- [18] A. Ibrahim and G. Dalkılıç, "An advanced encryption standard powered mutual authentication protocol based on elliptic curve cryptography for RFID, proven on WISP," *Journal of Sensors*, vol. 2017, 10 pages, 2017.
- [19] D. He, N. Kumar, N. Chilamkurti, and J. H. Lee, "Lightweight ECC based RFID authentication integrated with an ID verifier transfer protocol," *Journal of Medical Systems*, vol. 38, no. 10, p. 116, 2014.
- [20] Z. Zhang and Q. Qi, "An efficient RFID authentication protocol to enhance patient medication safety using elliptic curve cryptography," *Journal of Medical Systems*, vol. 38, no. 5, p. 47, 2014.
- [21] Y. Tian, G. Chen, and J. Li, "A new ultralightweight RFID authentication protocol with permutation," *IEEE Communications Letters*, vol. 16, no. 5, pp. 702–705, 2012.
- [22] Z. Qian, C. Chen, I. You, and S. Lu, "ACSP: a novel security protocol against counting attack for UHF RFID systems," *Computers & Mathematics with Applications*, vol. 63, no. 2, pp. 492–500, 2012.
- [23] S. Han, V. Potdar, and E. Chang, "Mutual authentication protocol for RFID tags based on synchronized secret information with monitor," in *International Conference on Computational Science and its Applications*, pp. 227–238, Berlin, Heidelberg, 2007.
- [24] C. Qingling, Z. Yiju, and W. Yonghua, "A minimalist mutual authentication protocol for RFID system & BAN logic analysis," in *2008 ISECS international colloquium on computing, Communication, Control, and Management*, vol. 2, pp. 449–453, Guangzhou, China, 2008.
- [25] C. L. Chen and Y. Y. Deng, "Conformation of EPC class 1 generation 2 standards RFID system with mutual authentication and privacy protection," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 8, pp. 1284–1291, 2009.
- [26] Y. P. Liao and C. M. Hsiao, "A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol," *Ad Hoc Networks*, vol. 18, pp. 133–146, 2014.

- [27] J. S. Chou, "An efficient mutual authentication RFID scheme based on elliptic curve cryptography," *The Journal of Supercomputing*, vol. 70, no. 1, pp. 75–94, 2014.
- [28] J. Chou, "A secure RFID authentication protocol to enhance patient medication safety using elliptic curve cryptography," *The Journal of Supercomputing*, vol. 21, 2014.
- [29] A. Braeken, P. Kumar, and A. Martin, "Efficient and provably secure key agreement for modern smart metering communications," *Energies*, vol. 11, no. 10, p. 2662, 2018.
- [30] V. Odelu, A. K. Das, M. Wazid, and M. Conti, "Provably secure authenticated key agreement scheme for smart grid," *IEEE Transactions on Smart Grid*, vol. 9, pp. 1900–1910, 2016.
- [31] Y. Chen, J. F. Martínez, P. Castillejo, and L. López, "An anonymous authentication and key establish scheme for smart grid: FAuth," *Energies*, vol. 10, no. 9, p. 1354, 2017.
- [32] D. Abbasinezhad-Mood and M. Nikooghadam, "An anonymous ECC-based self-certified key distribution scheme for the smart grid," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 7996–8004, 2018.
- [33] M. Campagna, "SEC 4: Elliptic Curve Qu-Vanstone implicit certificate scheme (ECQV)," in *Institution content-type= institution> Certicom Res</institution>*, Technical Report, Mississauga, ON, Canada, 2013.
- [34] K. Sowjanya, M. Dasgupta, and S. Ray, "An elliptic curve cryptography based enhanced anonymous authentication protocol for wearable health monitoring systems," *International Journal of Information Security*, vol. 19, no. 1, pp. 129–146, 2020.
- [35] Z. Zhao, "An efficient anonymous authentication scheme for wireless body area networks using elliptic curve cryptosystem," *Journal of Medical Systems*, vol. 38, no. 2, p. 13, 2014.
- [36] D. He, S. Zeadally, N. Kumar, and J. H. Lee, "Anonymous authentication for wireless body area networks with provable security," *IEEE Systems Journal*, vol. 11, pp. 2590–2601, 2016.
- [37] X. Li, J. Peng, S. Kumari, F. Wu, M. Karuppiah, and K. K. R. Choo, "An enhanced 1-round authentication protocol for wireless body area networks with user anonymity," *Computers & Electrical Engineering*, vol. 61, pp. 238–249, 2017.
- [38] N. Dinarvand and H. Barati, "An efficient and secure RFID authentication protocol using elliptic curve cryptography," *Wireless Networks*, vol. 25, no. 1, pp. 415–428, 2019.
- [39] Y. P. Liao and C. M. Hsiao, "A secure ECC-based RFID authentication scheme using hybrid protocols," in *Advances in Intelligent Systems and Applications-Volume 2*, pp. 1–13, Springer, 2013.
- [40] Z. Zhao, "A secure RFID authentication protocol for healthcare environments using elliptic curve cryptosystem," *Journal of Medical Systems*, vol. 38, no. 5, p. 46, 2014.
- [41] A. A. Alamr, F. Kausar, and J. S. Kim, "Secure mutual authentication protocol for RFID based on elliptic curve cryptography," in *2016 International Conference on Platform Technology and Service (PlatCon)*, pp. 1–7, Jeju, Korea (South), 2016.
- [42] C. Jin, C. Xu, X. Zhang, and F. Li, "A secure ECC-based RFID mutual authentication protocol to enhance patient medication safety," *Journal of Medical Systems*, vol. 40, no. 1, p. 12, 2016.
- [43] F. Merabet, A. Cherif, M. Belkadi, O. Blazy, E. Conchon, and D. Sauveron, "New efficient M2C and M2M mutual authentication protocols for IoT-based healthcare applications," *Peer-to-Peer Networking and Applications*, vol. 13, no. 2, pp. 439–474, 2020.
- [44] C. Jin, C. Xu, X. Zhang, and F. Li, "A secure ECC based RFID mutual authentication protocol to enhance patient medication," *Journal of Medical Systems*, vol. 40, pp. 1–6, 2016.
- [45] A. Braeken, "Public key versus symmetric key cryptography in client-server authentication protocols," *International Journal of Information Security*, vol. 21, pp. 1–12, 2021.
- [46] A. Braeken, "Highly efficient symmetric key based authentication and key agreement protocol using Keccak," *Sensors*, vol. 20, p. 2160, 2021.
- [47] A. Thakare and Y. G. Kim, "Secure and efficient authentication scheme in IoT environments," *Applied Sciences*, vol. 11, p. 17, 2021.
- [48] D. Rangwani and H. Om, "A secure user authentication protocol based on ECC for cloud computing environment," *Arabian Journal for Science and Engineering*, vol. 46, no. 4, pp. 3865–3888, 2021.
- [49] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.
- [50] T. Hewa, A. Braeken, M. Ylianttila, and M. Liyanage, "Blockchain-based automated certificate revocation for 5G IoT," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020.
- [51] D. J. Bernstein, "Curve25519: new Diffie-Hellman speed records," in *International Workshop on Public Key Cryptography*, pp. 207–228, Springer, 2006.
- [52] D. J. Bernstein and T. Lange, *SafeCurves: Choosing Safe Curves for Elliptic-Curve Cryptography*, 2013, April 2020, <http://safecurves.cr.yp.to>.
- [53] K. Hofer-Schmitz and B. Stojanović, "Towards formal verification of IoT protocols: a review," *Computer Networks*, vol. 174, article 107233, 2020.
- [54] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, *ProVerif 2.00: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, pp. 05–16, 2018.
- [55] B. Blanchet, "Modeling and verifying security protocols with the applied pi calculus and ProVerif," *Foundations and Trends® in Privacy and Security*, vol. 1, pp. 1–135, 2016.
- [56] B. Blanchet, "Automatic verification of security protocols in the symbolic model: the verifier ProVerif," in *Foundations of Security Analysis and Design VII*, pp. 54–87, Springer, Cham, 2013.