An elliptic curve and zero knowledge based forward secure RFID Protocol [⋆]

S. Martínez, M. Valls, C. Roig, F. Giné and J.M. Miret

Escola Politècnica Superior. Universitat de Lleida.
{santi,magda,roig,sisco,miret}@eps.udl.es

Abstract Nowadays, the use of Radio Frequency Identification (RFID) systems in industry and stores, has been increased. Nevertheless, some of this systems have privacy problems that may discourage potential users. Hence, secure and efficient privacy protocols are urgently needed. Previous works in the literature proposed schemes that were proven to be secure, but they had scalability problems. A feasible and scalable protocol to guarantee privacy is presented in this paper. The proposed scheme uses elliptic curve cryptography with the addition of zero knowledge based authentication. An analysis that proves that the system is secure, and even forward secure, is also provided.

1 Introduction

A Radio Frequency Identification (RFID) system allows the remote identification of items that have an RFID tag attached. This is particularly useful in supply chains, stores, etc. It is expected that, in the future, everyday objects will have RFID tags that will enable interesting applications, such as medicines with RFID tags on their package which would allow to link a unique identifier for that package to important information of it, like the caducity or contraindications. Anyway, this kind of services would not be wished by the end user if they entailed serious security problems and, for that reason, several works are directed to solve the vulnerabilities of these systems, in order to make them secure [1,2,5,6].

As can be seen in Figure1, an RFID system consists of three components:

- Tags, that consist of an integrated circuit with a small antenna. Tags use to be placed in each object that should be identified (e.g. the medicines). Each tag will send its identifier (ID) when interrogated.
- Reader(s) that communicate with a database and with the tags. They are responsible of performing the queries to the tags.
- Database with information of the tags and their items (e.g. medicine name, chemical components,...). RFID readers will check the database for identifying an object and for obtaining its associated information.

Depending on the power source of tags, they can be classified as passive, semi-passive or active tags. Passive tags do not have batteries, they derive their power from the signal of the reader.

[⋆] This work is supported by the Generalitat de Catalunya with a Grant FIC, and the projects of the Spanish MCyT TIN2006-15662-C02-02 and TIC2003-09188

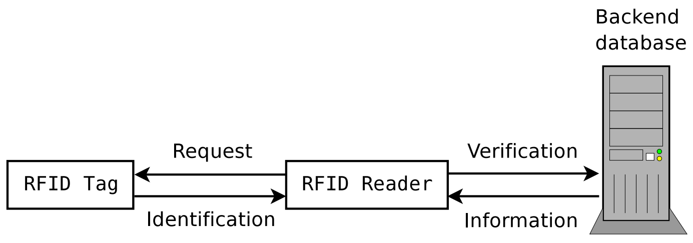

Figure1. RFID system.

In this work, we will focus on passive tags, since they are cheaper and the most broadly used type of tags [7]. However, their simplicity implies important restrictions that have to be respected: the cost (cents of a dollar), the number of logic gates (about 15000) and the transmission rate (520 or 640 bps depending on the band used). Considering that the duration of a read operation should be near a second, the protocol should transmit less than 500 bits.

Furthermore, there are two main privacy problems related to RFID systems: (a) leakage of information of the tags should be avoided, otherwise it would allow clandestine inventorying and, (b) tracking the behavior of its user should be prevented. The solution to the privacy problem implies some form of encryption of the IDs. On the other hand, in order to avoid the tracking problem, tag IDs should be frequently changed. An additional property will ensure that the revelation of tag secret information will not put in danger the security of information previously sent, this desirable property is called forward security.

In this work, we propose a new protocol to guarantee the privacy in the communications established between tags and readers in a RFID system. This protocol is able to solve the privacy problems while taking into account the implementation restrictions associated to current passive tags. The proposed approach is based on elliptic curve cryptography, which allows the use of fewer bits than conventional public key cryptography guaranteeing same security. Unlike previous proposals, this protocol requires reader authentication (instead of only tag authentication), by means of a zero knowledge protocol. We prove that in these conditions, the system is scalable (i.e. the implementation continues to be feasible when the number of tags increases).

The remaining paper is structured as follows. Section 2 outlines some related works and the idea of our solution. Section 3 sketches some preliminaries. In Section 4 the proposed protocol is described. In Section 5 some implementation issues are discussed. A security analysis is given in Section 6. Finally, Section 7 outlines the main conclusions.

2 Related Work

There have been many proposed approaches in the literature during the last years [1,2,3,4,5,6], trying to solve the RFID user privacy problems.

Ohkubo, Suzuki and Kinoshita in [1] proposed a scheme in which tags output the result of applying a hash function to its secret identifier and then change the identifier using a second hash function each time a tag is read, but it has a high computational complexity for the database during the identification, as it musts compute hashes until it finds a coincidence.

This problem is partially solved in the Avoine and Oechslin's scheme [2] using a time-memory trade-off, in which tag identifiers are precalculated and stored. The main problem of this approach relies on its scalability, as the number of stored identifiers grows exponentially when the number of tags increases.

The reason that creates the need of extra resources in these two previous protocols is that any reader can read the tags. These extra, and unexpected, reads will cause the change of the internal secret of the tag in such a manner that the reader will not know what is the expected output value for each tag. This uncertainty is the responsible of the extra computations in the Ohkubo *et al.*'s scheme and the extra memory in the Avoine and Oechslin's one. That makes these two previous protocols secure but not scalable.

To solve these problems we propose a new protocol in which readers must be authenticated and hence, only valid readers will be allowed to read the tags. Our solution is based on elliptic curve cryptography, because it allows the use of fewer bits than conventional public key cryptography, making its implementation more feasible. Due to the intrinsic insecurity of the communication channel between readers and tags, a zero knowledge protocol is used to carry out readers authentication. Such a protocol allows the reader to prove the knowledge of a secret without revealing any information related to it. Thus, the previous authentications between the readers and the tags cannot be reused for an attacker trying to impersonate a valid reader.

3 Preliminaries

As we mentioned earlier, our approach is based on elliptic curve cryptography and zero knowledge authentication, so, in this section we will introduce some preliminaries of these two techniques.

Elliptic Curve Cryptography. [13,14] An elliptic curve E over a field $\mathbb{F}_q$ consists of all the points $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ that satisfy an equation of the form $E(\mathbb{F}_q) : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$, with $a_i \in \mathbb{F}_q$ whose discriminant is non null, together with the point at infinity.

There is a point addition operation whose neutral element is the point at infinity. This set of points under this operation is an abelian group. Therefore, a point $Q \in E(\mathbb{F}_q)$ can be multiplied by a scalar: $e \cdot Q = \underbrace{Q + \ldots + Q}_{e \text{ times}} = P$.
The inverse problem (i.e. given P and Q, finding an e such that $P = e \cdot Q$), called the Elliptic Curve Discrete Logarithm Problem (ECDLP), turns out to be computationally hard to solve.

Zero Knowledge Authentication. [8,15] The purpose of zero knowledge protocols is to prove the knowledge of a secret without revealing it.

In a zero knowledge authentication protocol an entity A (prover) has some secret s whose knowledge has to be proven to an entity B (verifier). Given the fact that only A is able to do this demonstration, B will accept this as a proof of the identity of A. Additionally, the protocol does not reveal any useful information, other than that A has knowledge of s, so neither B nor an eavesdropper will obtain s and, because of that, they will not be able to impersonate A.

The idea is that B asks to A a question (or questions) related to s that only A can respond, but in a manner that the answer does not reveal the secret or even a part of it. In order to do this, A and B exchange messages typically dependant on random numbers. After these messages B either accepts, with certain probability of being correct, or rejects the proof of knowledge of s.

This probability comes from the fact that an attacker can impersonate A if he is able to guess correctly the question.

The protocol presented in this paper takes benefit of the usage of the elliptic curve version of the Schnorr's zero knowledge protocol [8]. Notice that, since security is based on the ECDLP, small keys can be used, which fits better the restrictions that RFID tags present.

4 Protocol description

The proposed protocol, sketched in Figure2 (a), uses the elliptic curve Schnorr's protocol and a secret changing mechanism. It consists on the following phases: (a) setup phase, (b) reader authentication phase, (c) tag identification phase and, (d) tag verification phase.

(a) Setup phase. In the proposed scheme, readers have a secret number s, the knowledge of which must be proven to tags for reading.

The authentication process between a reader and a tag needs the generation of the following public parameters:

- A finite field $\mathbb{F}_q$ and an elliptic curve defined over this finite field $E(\mathbb{F}_q)$.
- A generator Q of a cyclic subgroup of points of the elliptic curve where the ECDLP was unfeasible [9,10].
- The public key $P \in E(\mathbb{F}_q)$ of valid readers, computed as $P = sQ$

Each tag needs a secret elliptic curve point $K \in E(\mathbb{F}_q)$, from which it will compute its identifier. The database has the information related to the tags.

(b) Reader authentication phase. In order to avoid impersonation of a valid reader, an authentication phase must take place before tag identification. This phase consists of these three steps:

1. The reader chooses a random value r (the commitment), with $2 \leq r \leq (\#Q - 1)$, computes $W = rQ$ (the witness), and sends W to the tag.
2. The tag chooses a random l-bit-length value c (the challenge) and sends it to the reader.
3. The reader computes $a = r + cs$ (the response) and sends it to the tag.

At this point the tag will accept that the reader is valid if $aQ - cP = W$ (notice that indeed $aQ - cP = (r + cs)Q - c(sQ) = rQ = W$).

(c) Tag identification phase. This process will take place every time the reader wants to read a tag, and it has been successfully authenticated. Every time a tag is read it will send its current pseudo-id, id_j^i, corresponding to tag T_i, at j-th reading. In order to do this, each tag T_i keeps a secret elliptic curve point K_j^i, which will vary over time, belonging to the same curve used for reader authentication. This point will be changed each time the tag is successfully identified to prevent that two reads of the same tag can be related by an adversary.

The process of identification of a tag consists of these three steps:

1. The tag computes its pseudo-id $id_j^i = LastBits(x(K_j^i)\ bxor\ y(K_j^i))$, where $bxor$ is the bitwise xor operator, $LastBits$ takes some of the last bits of its input and $x(K_j^i)$, $y(K_j^i)$ are the abscissa and the ordinate of the secret point. We assume that this computation is sufficiently secure. Section 5.1 will discuss the appropriate number of bits for id_j^i.
2. The tag computes its next secret point $K_{j+1}^i = x(K_j^i) \cdot Q$.
3. The tag stores its new secret point and finally sends id_j^i to the reader.

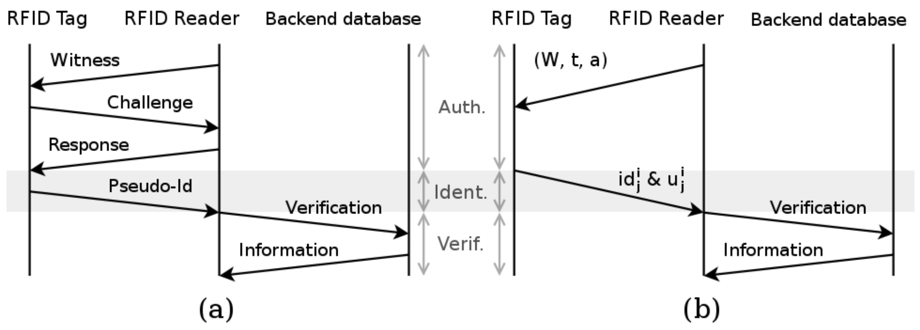

Figure2. (a) Basic and, (b) advanced protocol diagrams.

(d) Tag verification phase. At this point, the reader has already received id_j^i, so it has to access the database in order to verify the identity of the tag, as well as obtaining the information associated with it. For a successful identification the backend database has to store the outputs for all the tags i.e. all the id_j^i for $i \in [1, n]$ where n is the number of tags in the system. It also needs to store the corresponding secret points K_j^i. These values should be stored in a hash table for easy access. So, when a valid reader obtains a pseudo-id, it sends it to the backend database, which searches it in the hash table, changes the corresponding secret point in the same manner that the tag does, removes the old pseudo-id from the hash table and inserts the one that will be obtained in the next reading, finally it sends the product information to the reader.

In noisy environments, where it may be possible that readers do not receive the pseudo-id, tags can be updated without the updating of the database. For avoiding that, a final message should be sent to the tag indicating that the reception has ended. Obviously, this message is also susceptible of not arriving, but this is a less dangerous problem, since in that case it would be the tag the one that needs to be updated. The only consequence of this fact is that the next reading of a tag would return an old pseudo-id, so the reading operation would have to be repeated until the tag reaches the pseudo-id that is stored in the database.

The advantage of our proposal is precisely that we do not need to compute or to keep long chains of identifiers for each tag, but only the next identifier and the current secret. This makes our system scalable.

In the exposed protocol tags are expected to generate random numbers and they have to send these values before tag identification takes place, so the tags have to send two different messages. A major improvement that may be done is to eliminate this need of a random generator and the extra messages with a modification of the zero knowledge protocol.

All zero knowledge protocols may be done non-interactively [11], this allows the authentication part to be done with only one message. In that case the reader sends to the tag a message with (W, t, a) where $W = rQ$ and $a = r+cs$ as before, but the challenge is computed as $c = H(W|t)$, instead of randomly generated by the tag. H is a hash function and t is a value that cannot be reused (a nonce) to prevent an attacker authenticating successfully by sending a message that he might have eavesdropped before.

Because of the fact that the only restriction for the t value is that it cannot be reused, a good approach is to use timestamps (32 bits should be enough for the majority of systems). Tags should keep a last-heard time value, so when a reader successfully authenticates to a tag, it will be changed to t, and, of course, the tag will only accept the reader as valid if the received t value is strictly greater than its old last-heard time value.

This protocol also admits some extensions, which makes it suitable for being used in several scenarios. The protocol can be tuned to provide the tag the ability to securely send small values v (for example data generated by a sensor) to the reader. This may be achieved with a very simple form of encryption, using the secret of the tag (K) as a key. For example, the additional data can be encrypted using a $bxor$ with the first bits of the ordinate of the secret point: $u_j^i = v_j^i \; bxor \; FirstBits(y(K_j^i))$. Since these first bits were not used in the computation of the pseudo-id (that uses only the last bits), this may be considered a one-time pad. The encrypted data u_j^i can be sent in the same message than the pseudo-id. The enhanced protocol is shown in Figure2 (b).

5 Implementation issues

As it was mentioned in Section 1, strong restrictions in implementation have to be considered as we are dealing with small and cheap devices. In this section, we provide an specific selection for the length of the parameters to be used in the proposed protocol, towards and efficient and feasible implementation.

5.1 Selection of parameters

There are some parameters that need to be accurately selected in order to achieve a good balance between the desired degree of security and the best performance.

The finite field, the elliptic curve and the generator. For the finite field $\mathbb{F}_q$ where $q = p^m$ with p prime, the usage of a binary finite field $\mathbb{F}_{2^m}$ is recommended. In fact, the expected security is similar for prime and binary finite fields but binary field arithmetic can be implemented easier than the prime field one [12].

Now, concerning the degree of the field, m, recall that prime values should be selected for security reasons. We propose the use of the field $\mathbb{F}_{2^{137}}$, since the last solved ECC challenge is the ECC2-109 [1] (which is defined over $\mathbb{F}_{2^{109}}$). Therefore, it is an acceptable level of security for this kind of applications, although bigger fields can be used in sensitive or long term systems.

Finally, concerning the elliptic curve, it should be selected with a big prime factor in its cardinal [9,10], and a point of big order should be chosen as generator.

Length of the challenge: l (the number of bits of c). For the challenge length l there are two considerations: firstly, l must be large enough to make negligible the probability of correctly guessing the challenge, secondly it is proved that the protocol is not zero knowledge for large l values [11]. These two considerations lead to $2^{-l} \approx 0$ and, $q > 2^{2l}$. Thus, for $q = 2^{137}$ we recommend the use of 64 bits, which gives a probability of forgery of 2^{-64}, while maintaining the zero knowledge property.

The hash function H to obtain the challenge. The hash function H must output a l-bits value and should be easy to implement it in the tag circuitry. Its input is the concatenation of the last 128 bits of the abscissa of the witness W and the 32-bits timestamp t, giving an input of 160 bits in length.

Length of the pseudo-id id_j^i. For the pseudo-id id_j^i, computed by the tags, the goal is to prevent that two tags have the same pseudo-id (otherwise, it would present a minor problem when identifying the tags involved in the collision). Because of that, the number of bits must depend on n, the number of tags in the system. It should be greater than $2 \cdot log_2(n)$ to avoid birthday paradox collisions [11]; note that if the length of the pseudo-id is very close to this value then the probability of having two tags with the same pseudo-id is about 50%. So, for instance, in a system with one million tags ($\approx 2^{20}$), the length of the pseudo-id should be 48 bits, that would give a probability of having a collision below 1%.

If a reading operation returns a pseudo-id that is the same for two (or more) tags, one can simply re-read the tag and compare the new result with the next

[1] The Certicom ECC Challenge was introduced by Certicom in 1997 in order to evaluate the difficulty of the elliptic curve discrete logarithm problem.

pseudo-id of the tags previously involved in the collision, then updating only the tag information for the tag that has the two read pseudo-ids.

5.2 Implementation feasibility

Now, we will consider the implementation feasibility of our solution under current standards of implementation for RFID tags. There are four main aspects to consider: the computations to be performed by the tag, the amount of data transfered between tags and readers, the size of the backend database and the scalability of the system. The analysis will be done using the parameters proposed in Section 5.1, the non-interactive version of the zero knowledge protocol and the feature of sending additional data.

Computational complexity. The most costly operations to be performed by the tag are the verification of the identity of the reader and the generation of the next secret point (three point multiplications). This implementation is proved to be feasible and for the field $\mathbb{F}_{2^{137}}$ it may be achieved in less than 10000 gates [12] (current passive tags may have about 15000 gates).

For the hash computation needed in the non-interactive version, a low-cost hash function like the examined in [16] is recommended. The rest of tag computations: comparisons, bitwise xors, etc. are cheaper.

Communications. Now, the length of the messages involved in the protocol will be computed. The first message contains (W, t, a). It is sufficient to include the last 128 bits of the abscissa of the witness, with the condition that the rest of the bits are not needed for the hash. For the timestamp 32 bits are enough. Finally, the answer has 137 bits. So, the first message will have 297 bits.

The second message, sent by the tag contains (id^i_j, u^i_j). In a system with about one million tags, 48 bits pseudo-ids are sufficient. Considering the additional data to be encrypted as having 64 bits, leads to a length of 112 bits. Considering both messages, the total amount of communication will be 409 bits. RFID tags with the minimum transmission rate of 520 bps, will need 0.8 seconds for communication, which fits the bandwidth restrictions for RFID systems.

Database size. The backend database has to store a record with the next expected pseudo-id (48 bits) and the current secret point, which has $137 + 1$ bits (abscissa and one bit for selecting the ordinate), for each tag, maybe with some additional data, such as product type, caducity or similar. These records will be stored in a hash table indexed by the pseudo-id for easy accessing.

Considering only the point and the pseudo-id, but not any additional data not related with the protocol itself, the memory needed by our solution for n tags is of order $O(nlogn)$, because it must store n pseudo-ids each of them with approximately $2log_2n$ bits, and the n fixed length secret points. A system with one million tags will need $1000000 \cdot (48 + 137 + 1)$ bits that is only about 23 MB (perfectly reasonable in current systems).

Scalability. Our protocol does not need to precalculate huge tables, so it avoids all the recalculation times, and does not have any restriction of tag read operations. Additionally we need only 23 MB, for a one million tags system. All this implies that our system is very scalable.

6 Security Analysis

In order to explain the security analysis, it is useful to recall the public and private information of the protocol.

Public information:

- $\mathbb{F}_q$: finite field.
- $E(\mathbb{F}_q)$: elliptic curve.
- $Q \in E(\mathbb{F}_q)$: generator of a cyclic subgroup.
- $P \in \langle Q \rangle$: public key of valid readers.

Secret information:

- $s \in [2, \#Q - 1]$: secret of valid readers.
- $K_j^i \in \langle Q \rangle$: Secret of tag T_i, at moment j.

Secret information per round:

- $r \in [2, \#Q - 1]$: the random commitment (chosen by the reader).

Public information per round:

- $W = rQ$: the computed witness (sent from reader to tag).
- t: the current timestamp (sent from reader to tag).
- $c = H(W|t)$: the challenge (computed by both parts).
- $a = r + cs$: the response (sent from reader to tag).
- $id_j^i = LastBits(x(K_j^i) \oplus y(K_j^i))$: the pseudo-id (sent from tag to reader).

Typical scenarios assumed in literature [7] consider that in a RFID system two zones may be considered. In the secure zone there are the backend database and the RFID readers. The RFID tags and its communication channels with the readers are not considered secure.

In the following subsections, we will define the basic types of attacks against this protocol, that must be considered. Then, we will also evaluate the security against attacks to the elements of the insecure zone (tags and communication channels between readers and tags).

6.1 Types of Attacks

According to the literature there are five basic types of attacks that we must consider.

Sniffing. In an sniffing attack the attacker eavesdrops the communications between a reader and a tag, trying to obtain useful information.

The only public information is the needed for the protocol setup, the reader's public key, the information sent in the first message (the witness, a timestamp, and the response) and the information sent in the second message (the pseudo-id and, optionally, the encrypted sensor data). So, an sniffing attack is useless due to the use of a zero knowledge protocol in combination with the use of a pseudo-id which cannot be related to the tag's secret and the encryption of the sensor data.

Tracking of the tags. The tag's tracking attack consists on the tracking of the behavior of the owner of a tag.

In this case, the only information to be considered is the pseudo-id (and the optional sensor data), since the rest of the data is sent by the reader. A pseudo-id sniffed at a certain moment, cannot be related with the information obtained before (or after), because a bitwise *xor* is considered a secure ciphering algorithm for small data (the same reasoning applies to the encrypted sensor data).

Spoofing. A spoofing attack is the impersonation of one of the entities of the system. We have to consider two different types of this attack:

Impersonation of a reader:
Due to the use of a zero knowledge authentication protocol the probability of impersonation of a reader is negligible.
Impersonation of a tag:
An attacker willing to impersonate a tag needs the current secret K_j^i of the tag to impersonate. Such a value cannot be obtained from the public information or the communicated during the execution of the protocol.
It should be noted that, if an attacker physically obtains the secret of a tag, but returns the tag to the system without altering it, a reader could read the tag, that will cause the changing of the tag's secret, thus making the obtained tag's secret obsolete. In that case, the attacker should modify the secret if he wants to impersonate the tag. If the attacker impersonates the tag, it is the tag's secret the one that will be obsoleted, since the database will change the expected pseudo-id (this allows a denial of service for that tag).

In both cases, for an spoofing attack the secret of the element to impersonate must be obtained.

Replay attacks. A replay attack consists on an attacker resending information that he had captured before, eavesdropping a previous session.

If an attacker eavesdrops the reading operation of a tag, he will obtain a witness, a timestamp and a response. Since the timestamp must be greater than the last-heard time of any tag, he cannot reuse these values with the tag involved in the communication he has eavesdropped; but, if there were tags in the system whose last-heard time were still lower than the obtained by the attacker, these tags would be vulnerable to a replay attack. So, only the interactive version of the protocol avoids this attack.

On the other hand, a pseudo-id cannot be reused, because the database will wait for the next pseudo-id of any tag, so if an attacker reuses a pseudo-id the reader will not recognize the value as valid.

Denial Of Service. Finally, a denial of service consists on a temporal (or permanent) incapacitation of the system or a part of it.

Since a tag only changes its secret K_j^i when a reader has successfully authenticated, there is no danger of an attacker performing a denial of service attack of multiple read operations.

As said in Section 4(d) an additional message may be needed in some environments, for avoiding the problem of having the database obsoleted.

6.2 Forward security

The property of forward security ensures that the revelation of tag secret information will not put in danger the security of previously sent information.

We will assume that the tag has some kind of sensor, and that the sensor data v_1^i, v_2^i, v_3^i, ..., v_{j-1}^i has been securely sent in previous reading operations. We also assume that the attacker knows all public parameters as before, and has eavesdropped all the u_1^i, u_2^i, u_3^i, ..., u_{j-1}^i, the pseudo-ids and all the other parameters sent during the execution of the protocol.

At this time, the adversary physically attacks tag T_i and obtains its current secret point K_j^i. But in order to decrypt the u values, there are needed the previous K_{j-1}^i, K_{j-2}^i, K_{j-3}^i, ..., K_1^i, and they cannot be easily obtained because it implies solving one elliptic curve discrete logarithm for each secret point he wants to obtain.

So, this communication has the property of forward security, i.e. for an attacker trying to decrypt some of the values that he might have eavesdropped, it is useless to obtain the current secret of the tag, because he does not have any way of obtaining the secret keys employed on encrypting them other than solving some discrete logarithms.

7 Conclusions

In this paper, an efficient Radio Frequency Identification protocol has been proposed. It avoids leakage of tag information and tracking of the behavior of its user. Additionally, the system has been proven to be forward secure.

To meet the implementation restrictions of RFID tags, we propose the use of an elliptic curve cryptography protocol with zero knowledge authentication.

The proposed protocol provides a secure channel between tags and readers. This can be useful when tags need to transmit small values from some sensors to readers.

Finally, our approach is scalable on the number of tags in the system, and, a security analysis is provided that proves that the system is secure, and even forward secure.

References

1. M. Ohkubo, K. Suzuki and S. Kinoshita: Cryptographic Approach to “Privacy-Friendly” Tags. RFID Privacy Workshop. November, 2003.

2. G. Avoine and P. Oechslin: A Scalable and Provably Secure Hash-Based RFID Protocol. International Workshop on Pervasive Computing and Communication Security - PerSec 2005. IEEE Computer Society Press. Pp. 110-114. March, 2005.
3. Auto-ID Center: 860MHz-960MHz Class I Radio Frequency Identification Tag Radio Frequency and Logical communication Interface Specification Proposed Recommendation Version 1.0.0. Technical Report MIT-AUTOID-TR-007. 2002.
4. S. Weis, S. Sarma, R. Rivest and D. Engels: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. First International Conference on Security in Pervasive Computing. http://theory.lcs.mit.edu/sweis/spc-rfid.pdf. 2003.
5. S. Kinosita, F. Hoshino, T. Komuro, A. Fujimura and M. Ohkubo: Nonidentifiable Anonymous-ID Scheme for RFID Privacy Protection. CSS 2003 *in Japanese*.
6. A. Juels and R. Pappu: Squealing euros: Privacy protection in RFID-enabled banknotes. In Proceedings of Financial Cryptography - FC'03.
7. A. Juels: RFID security and privacy: A research survey. Manuscript. September, 2005.
8. C. P. Schnorr: Efficient signature generation by smart cards. Journal of Cryptology. Vol. 4. N. 3. Pp. 161-174. January, 1991.
9. J. Miret, R. Moreno, D. Sadornil, J. Tena and M. Valls: An algorithm to compute volcanoes of 2-isogenies of elliptic curves over finite fields. Applied Mathematics and Computation. Vol. 176. N. 2. Pp. 739-750. 2006.
10. S. Martínez, R. Tomàs, C. Roig, M. Valls and R. Moreno: Parallel calculation of volcanoes for cryptographic uses. Proceedings of the 20th IEEE International Parallel and Distributed Symposium. 2006.
11. A. Menezes, P. van Oorschot and S. Vanstone: Handbook of Applied Cryptography. Book. 1996.
12. L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls and I. Verbauwhede: An elliptic curve processor suitable for RFID-tags. Cryptology ePrint Archive, Report 2006/227.
13. V.S. Miller: Use of elliptic curves in cryptography. Advances in Cryptology-CRYPTO'85 (LNCS 218). Pp. 417-426. 1986.
14. N. Koblitz: Elliptic curve cryptosystems. Mathematics of Computation. Vol. 48. Pp. 203-209. 1987.
15. D. Chaum, J-H. Evertse, J. van de Graaf and R. Peralta: Demonstrating possession of a discrete logarithm without revealing it. Advances in Cryptology-CRYPTO'86. Pp. 200-212. 1987.
16. S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk: Cryptographic Hash Function: A Survey. Technical Report 95-09, Department of Computer Science, University of Wollongong, July 1995.