Cryptanalysis of a Novel Authentication Protocol Conforming to EPC-C1G2 Standard

Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda

Computer Science Department, Carlos III University of Madrid
{pperis, jcesar, jestevez, arturo}@inf.uc3m.es

Abstract. In 2006, the EPC Class-1 Generation-2 (EPC-C1G2) standard was ratified both by EPCglobal and ISO. This standard can be considered as an "universal" specification for low-cost RFID tags. Although it represents a great advance for the establishing of RFID technology, it does not pay due attention to security and, as a consequence, its security level is indeed very low. In 2007, Chien et al. published a mutual authentication protocol conforming to EPC-C1G2 which tried to correct all its security shortcomings. In this article, we point out various major security flaws in Chien et al.'s proposal. We show that none of the protocol objectives are met. Unequivocal identification of tagged items is not guaranteed due to birthday attacks. Furthermore, an attacker can impersonate not only legitimate tags, but also the back-end database. Location privacy is easily jeopardized by a straightforward tracking attack. Finally, we show how a successful auto-desynchronization (DoS attack) can be accomplished in the back-end database despite the security measures taken against it. At the core of all these vulnerabilities lays the abuse of a CRC function.
Keywords— RFID, EPC-C1G2, security, authentication, cryptanalysis

1 Introduction

One of the most relevant standards connected to RFID technology is the EPCglobal Class-1 Gen-2 RFID specification (EPC-C1G2) [6]. EPC-C1G2 tags are passive, so they receive their energy from reader's RF waveform. The very constrained computational and storage capabilities dictates that these tags can not afford the use of traditional cryptographic primitives. Following the standard, tags only support on-chip a 16-bit Pseudo-Random Number Generator (PRNG) and a 16-bit Cyclic Redundancy Code (CRC). Tag memory is insecure, and susceptible to physical attacks. Two 32-bit kill and access passwords are used to permanently disable the tag and to trigger it into secure mode, respectively.

Despite of the great advance that EPC-C1G2 represents in terms of communication compatibility and performance between tags, and the major implications this could have to ease the widespread use of this technology, the security level of this standard is extremely weak. The two most relevant operations for managing tag populations are inventory and access. These two operations present serious security flaws, as described below:

– Inventory command: the private information stored in the tag is compromised by any attacker with access to the radio channel, because the EPC is transmitted in plain text. Additionally, an adversary can easily impersonate a legitimate tag: the attacker can obtain the EPC of any tag by simply eavesdropping the air channel, as this EPC will be emitted by the tag when the reader sends any request. After obtaining this value, the attacker can use it to impersonate the tag. Finally, as tags transmit always a fixed EPC value, this could be associated with its holder allowing an easy tracking of user's movements and behaviors.
– Access command: the security of the access command is extremely weak, so performing a passive attack is very simple. An attacker listening the backward and forward channel (a very realistic assumption when using the air channel) can pick up the random numbers sent by the tag. Next, the attacker will be able to decrypt the ciphertexts sent by the reader by performing an xor (addition modulo 2) with the previous eavesdropped random numbers. So the plaintexts or PINs can be obtained by this quite simple mechanism, which constitutes an important security pitfall.

In spite of the serious security failures of EPC-C1G2 (the case of the *Inventory* and *Access* commands described above are only two examples), this standard could already be considered a great success after having been adopted by many RFID manufacturers [1]. This is the reason why efforts to develop new security features (i.e. authentication or key exchange protocols) compliant with the EPC-C1G2 standard have blossomed lately [3, 9, 10].

2 Chien's et al. Protocol

In [5], Chien et al. propose a mutual authentication protocol for improving not only the security of EPC-C1G2 but also that of all the previous proposals compliant with this standard. Their scheme consists on two phases:

Initialization phase For each tag denoted as T_i, the server randomly selects an initial authentication key K_{i_0} and an initial access key P_{i_0}. These two values, joined with the EPC (EPC_i) are stored in the tag. The authentication and access key will be updated after each successful authentication. For each tag, the server S (back-end database) maintains a record of six values: (1) EPC_i; (2) the old authentication key for this tag (K_{old}), which is initially set to K_{i_0}; (3) P_{old} denotes the old access key for this tag, which is initially set to P_{i_0}; (4) K_{new} denotes the new authentication key, which is initially set to K_{i_0}; (5) P_{new} denotes the new authentication key, which is initially set to P_{i_0}; (6) $Data$ denotes all the information about the tagged objet.

The (n+1) authentication phase

$R \rightarrow T_i$: N_1 The reader sends a random nonce N_1 as a challenge to the tag.

$T_i \rightarrow R \rightarrow S$: M_1, N_1, N_2

The tag generates a random number N_2, computes $M_1 = CRC(EPC_i||N_1||N_2) \oplus K_{i_n}$, and sends the value back to the reader, which will forward these values to the server.

The server interactively selects an entry $(EPC_i, K_{old}, K_{new}, P_{old}, P_{new}, Data)$ from its database, computes $I_{old} = M_1 \oplus K_{old}$ and $I_{new} = M_1 \oplus K_{new}$, and checks wether any of these two equations hold $I_{old} = CRC(EPC_i||N_1||N_2)$ $I_{new} = CRC(EPC_i||N_1||N_2)$. This is designed to be a way of avoiding desynchronization attacks. The process is repeated until a match is found in the database, thus implying a successful authentication of the tag. If no match is found, a failure message is sent to the reader, and the authentication process is stopped.

$S \rightarrow R$: M_2, $Data$

After a successful authentication, the server computes $M_2 = CRC(EPC_i||N_2) \oplus P_{old}$ or $M_2 = CRC(EPC_i||N_2) \oplus P_{new}$, depending on which value (K_{old}, K_{new}) satisfies the equation in the previous step. It also updates $K_{old} = K_{new}$, $P_{old} = P_{new}$, $K_{new} = PRNG(K_{new})$ and $P_{new} = PRNG(P_{new})$. The server sends M_2, $Data$ to the reader.

$R \rightarrow T_i$: M_2

Upon receiving M_2, the tag verifies whether the equation $M_2 \oplus P_{i_n} = CRC(EPC_i||N_2)$ holds. If so, it updates its keys $K_{i_{n+1}} = PRNG(K_{i_n})$ and $P_{i_{n+1}} = PRNG(P_{i_n})$.

3 Cyclic Redundancy Codes - CRC's

A Cyclic Redundancy Code (CRC) is a checksum algorithm that can be used to detect transmission errors (typically one or two bit flips, or bursts) in a very efficient way. CRCs operate by interpreting input binary sequences as polynomial coefficients, that they divide by a prefixed polynomial in order to obtain a remainder, which, in its binary expression, constitutes the *crc* value.

CRCs are completely linear, so they shouldn't be use in cryptographic applications, as they cannot detect malicious changes by a knowledgeable attacker [2, 12, 13]. Instead, cryptographic primitives such as hash functions or message authentication codes should be used for this purpose.

So computing a *crc* value for a given binary stream is essentially dividing the polynomial associated with this stream by another fixed polynomial (that depends on the particular CRC implementation) and computing a reminder. The stream should be multiplied by x^N (being N the degree of the *crc* polynomial) prior to division. That is, computing the *crc* of a polynomial $i(x)$ is basically finding a remainder $r(x)$ so that,

$$i(x) \cdot x^N = d(x) \cdot p(x) + r(x) \quad with \;\; |r(x)| < |p(x)| \tag{1}$$

3.1 CRCs properties

Due to their linearity, CRCs have some properties that, from the security point of view, one can label as bad. In fact, we will show that one of these "bad" properties (derived from their linear structure) will be enough to successfully attack Chien et al.'s mutual authentication protocol in various ways.

Theorem 1. *For any CRC (independently of its generator polynomial) and for any values a, b, c, and* $d \in F_2[x]$*, it holds:*

$$CRC(a\|b) \oplus CRC(c\|d) = CRC(a \oplus c\|b \oplus d) \tag{2}$$

Proof. ¿From the definition in *Equation 1* above, one can write:

$$CRC(a\|b) = (a \cdot x^N \oplus b) \cdot x^N \oplus d_1(x) \cdot p(x) \tag{3}$$

$$CRC(c\|d) = (c \cdot x^N \oplus d) \cdot x^N \oplus d_2(x) \cdot p(x) \tag{4}$$

for certain polynomials $d_1(x)$ and $d_2(x) \in F_2[x]$. Substituting these values in the left side of *Equation 2* we obtain the following:

$$(a \cdot x^N \oplus b) \cdot x^N \oplus d_1(x) \cdot p(x) \oplus (c \cdot x^N \oplus d) \cdot x^N \oplus d_2(x) \cdot (x) \tag{5}$$

rearranging terms in this expression we get:

$$((a \oplus c) \cdot x^N \oplus (b \oplus d)) \cdot x^N \oplus (d_1(x) \oplus d_2(x)) \cdot p(x) \tag{6}$$

that is the corresponding expression for $CRC(a \oplus c\|b \oplus d)$ (analogously to *Equation 3* and *4*). □

Corollary 1. *In particular, if in Equation 2 we have* $a = c$*, then,*

$$\begin{aligned} CRC(a\|b) \oplus CRC(a\|d) = CRC(a \oplus a\|b \oplus d) = CRC(0\|b \oplus d) = \\ = CRC(b \oplus d) \end{aligned} \tag{7}$$

because $0 \cdot x^N \equiv 0 \cdot p(x)$

This is the property we will take advantage of for attacking Chien et al.'s protocol (and, for that matter, any other protocol relying in the use of a CRC as a means of concealing secrets). It is important to point out this holds for every CRC implementation, independently of its length and *crc* generator polynomial (CRC-8, CRC-16, CRC-32, CRC-64, etc).

4 Vulnerabilities of Chien's Protocol

In this section we will analyze the most important vulnerabilities of Chien et al.'s protocol.

4.1 Unequivocal identification

The use of RFID tags offers several advantages over barcodes: data can be read automatically, without line of sight, and through a non-conducting material such as cardboard or paper, at a rate of hundreds per second, and from a distance of several meters. But there is a fundamental difference between a barcode technology and RFID. Barcodes use Universal Product Codes (UPC) to identify items. RFID technology replaces UPC with the Electronic Product Code (EPC) that allows the unequivocal identification of tagged items.

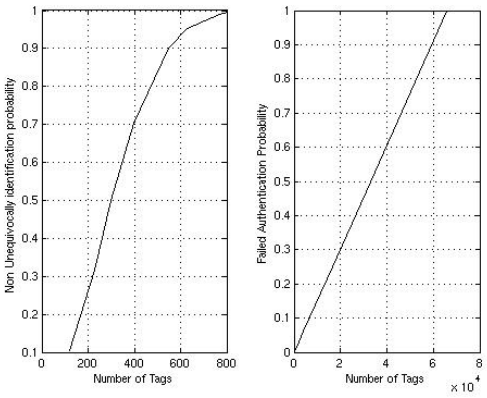
The Tag Data Specification [7] does not provide any specific guidance for using EPCs in UHF Class-1 Generation-2 tags. Due to this fact, in the following we assume that EPCs will be managed in the same way as they were in the EPC-C1G1 standard. So, the EPC is composed of the following fields (identical to those of the General Identifier, GID-96)

- *Header* is set to the fixed hexadecimal value 0x35 (8-bits).
- *General manager* identifies a company, manager or organization (28-bits).
- *Object class* is used by an EPC managing entity to identify class or "type" of thing (24-bits).
- *Serial number* is unique within each object class (36-bits).

Static identifiers (EPC-96) represent valuable information that should be transmitted on the channel guaranteeing its confidentiality and, at the same time, avoiding the tracking of its holders. To solve these two connected problems, researchers have proposed many pseudonym-based solutions. Generally speaking, a pseudonym is a fictitious name that disguises the real EPC-96 value, only allowing authorized parties to link it to its real value. However, just by using pseudonyms only privacy could be guaranteed. For assuring a suitable protection against tracking (location privacy), it is necessary to update pseudonyms in an unpredictable way each time the tag is interrogated. The most commonly used solution in the literature for pseudonym updating consists on repeatedly applying a hash function to the static identifier (i.e $pseudonym_i = hash^i(EPC)$). However, hash functions have not been ratified by the EPC-C1G2 specification, due to the inherent computational limitations of low-cost RFID tags. As we saw in *Section 1*, tags conforming to EPC-C1G2 only support on-board a 16-bit Pseudo-Random Number Generator (PRNG) and a 16-bit Cyclic Redundancy Code (CRC) checksum.

In the inventory command, as described in EPC-C1G2 specification, tags transmit its EPC in plain text. Chien et al. propose that tags transmit $M_1 = CRC(EPC||n_1||n_2)$ instead, where nonce n_1 is generated by the reader and nonce n_2 is generated by the tag. Message M_1, concatenated with these two nonces n_1 and n_2 is sent to the back-end database. This scheme presents a serious security failure, as described in the following.

An EPC has the first 8-bits of the header fixed, while the remainder 88-bits are variable. So, there are 2^{88} possible identifiers. However, tags support on-board a 16-bit CRC (ISO/IEC 13239, p(x)=$x^{16}+x^{12}+x^5+1$, Preset=0xFFFF, Residue=0x1D0F). So, the 2^{88} possible EPC values collapse in only 2^{16} values



(a) Fig. 1.A (b) Fig. 1.B

Fig. 1: Non Unequivocal Identification and Failed Authentication

when the CRC is applied to the EPC ($M_1 = CRC(EPC\ ||n_1||n_2)$).

Weakness 1: Chien et al's protocol does not guarantee the unequivocal identification of tagged items, which is an essential property in authentication protocols.

We have simulated a population of N tags. For each tag the values of EPC, K, and P were randomly initialized. These values will be stored both in the tag and at the back-end database. Upon initialization, we simulate the reading of these N tags. For each reading, the following process is repeated:

(1) Reading of tag_x
(2) M_1 = $\text{CRC}(\text{EPC}_x \ ||\ n_1 \ ||\ n_2) \oplus K_x$
(3) Send M_1, n_1, n_2 to the back-end database
(4) for(x'==1,x'< N,x'++)
    M_1'= $\text{CRC}(\text{EPC}_{x'} \ ||\ n_1 \ ||\ n_2) \oplus K_{x'}$,
    if ((x' != x) && (M_1' == M_1)) collision++;
(5) if collision>0 "Failed unequivocal identification"

The above process is repeated T times ($T = 10000$) in order to obtain an estimation of the non-unequivocal identification probability (P_{NUI}). We have simulated the above experiment with eight different values ($N = 118$, 226, 301, 397, 549, 626, 769, 800). The values obtained are summarized in *Fig. 1A*, and perfectly fit with the values obtained for the birthday paradox with a group of N tags and $d = 2^{16}$ boxes:

$$p(N;d) = \begin{cases} 1 - \prod_{k=1}^{N-1}(1 - \frac{k}{d}) & N \leq d \\ 1 & N > d \end{cases} \quad (8)$$

These results are hardly surprising, since each time a tag (tag_x) is read, we search if the equality $M_1 = CRC(EPC_x||n_1||n_2) \oplus K_x == CRC(EPC_{x'}||n_1||n_2) \oplus K_{x'} = M_1'$ holds for $x' \neq x$. As specified in EPC-C1G2, the CRC is 16-bits length. The kill and access passwords are 32-bits in length. However, only one half (MSB or LSB) is included in each message. Chien et al. state neither the length of the authentication key (k_x) nor the length of the access key (P_x). From the above, we can assume that these two keys are 16-bit length. Moreover, the keys are xored with a CRC of 16-bit length, so the previous assumption seems consistent with their usage. Finally, if we assume that the $CRC()$ and K_i are uniformly distributed (which may well not be the case, specially in the case of CRCs), the probability that at least two of N randomly selected tags have the same index ($M_1 = M_1'$), is exactly that given by the birthday paradox with N individuals and $d = 2^{16}$ boxes. Therefore, we have pointed out that tags can not be unequivocally identified under these conditions. For example, with a population of more than 300 tags, we will have at least one non-unequivocal tag identification with a probability over 0.5 ($P_{NUI} > 0.5$). Similarly, even for a relatively low number of tags (600) the probability of non-unequivocal identification rises to more that an 90% (see *Fig. 1.A*).

Finally, even if we relax a lot the requirements of our RFID system for even allowing the existence of collisions in the identification process (still if the collisions probability is low, this seems like a very unusual decision), Chien's protocol could present serious operational problems as the number of tags increase, because the probability of failed authentication rises quickly. For verifying that, we carried out a similar experiment to that described above, although in this case each time a tag is read the absolute number of collisions in the database is computed. The experiment has been simulated with six different values for the tag population ($N = 2^7$, 2^{10}, 2^{12}, 2^{14}, 2^{15}, 2^{16}), and repeated T times ($T = 10000$). The obtained results are displayed in *Fig. 1B*. So, the probability of failed identifications in a population of N tags, is described by the following equation:

$$P_{FI}(N) = \begin{cases} 2^{log_2(N)-16} & N \leq 2^{16} \\ 1 & N > 2^{16} \end{cases} \quad (9)$$

4.2 Tag Impersonation

Each tag shares with the reader some private information: EPC ($EPC_{\mathtt{X}}$), authentication key ($K_{\mathtt{X}}$) and the access key ($P_{\mathtt{X}}$). This information is employed to build messages M_1 and M_2 in order to proof its authenticity. However, a passive attacker eavesdropping the channel (backward and forward channel) will be able to supplant a legitimate tag as described bellow:

Weakness 2: Chien et al's protocol does not guarantee the non-impersonation of legitimate tags.

Proof: In order to accomplish this attack, and adversary only needs to listen an iteration between the reader and the legitimate tag.

(1) $R \rightarrow T$: n_1
(2) $T \rightarrow R$: $M_1 = CRC(EPC_{\mathtt{X}}||n_1||n_2) \oplus K_{\mathtt{X}}$, n_2

At this point, the attacker isolates the legitimate tag, preventing its normal operation. He has the following information: M_1, n_1, and n_2. With this, the attacker should be able to build message $M_1' = CRC(EPC||n1'||n2')$ when queried by the reader. Although the attacker does not know the private information stored in the tag (EPC, $K_{\mathtt{X}}$, and $P_{\mathtt{X}}$), message M_1' can be easily computed as described below. *Corollary 1* states:

$$CRC(a||b) \oplus CRC(a||d) = CRC(b \oplus d) \tag{10}$$

As this holds for every a, b, d $\in F_2[x]$, if b and d are the concatenation of some other variables ($b = b1||b2$, $d = d1||d2$), the above expression also holds and can be rewritten as,

$$\begin{aligned} &CRC(a\|b) \oplus CRC(a\|d) = CRC(a\|b1\|b2) \oplus CRC(a\|d1\|d2) = \\ &CRC((b1\|b2) \oplus (d1\|d2)) = CRC(b1 \oplus d1\|b2 \oplus d2) \end{aligned} \tag{11}$$

So the difference between the known value M_1 and the new challenge M_1' is exactly $M_1 \oplus M_1' = CRC(EPC||n_1||n_2) \oplus CRC(EPC||n_1'||n_2')$ and, substituting in *Equation 11*, we got:

$$CRC(EPC||n_1||n_2) \oplus CRC(EPC||n_1'||n_2') = CRC(n_1 \oplus n_1'||n_2 \oplus n_2') \tag{12}$$

Therefore, message M_1' can be obtained by performing an xor between message M_1 and the easily computable value $CRC(n_1 \oplus n_1'||n_2 \oplus n_2')$ (because all nonces are transmitted in clear and the CRC function is public). Therefore, the identity of a legitimate tag could be easily impersonated. An ANSI-C code with the implementation of this attack is available in `http://163.117.149.208/chien/attack2.c`.

4.3 Back-end Database Impersonation

In *Section 4.2* we focus on message M_1 sent by the tag when queried by the reader. In this case we concentrate on message M_2, generated by the back-end database. The attacker should be able to generate this message in order to impersonate a legitimate back-end database.

Weakness 3: Chien et al.'s protocol is vulnerable to back-end database impersonation.

Proof: For the attacker, it is enough to listen an iteration between a legitimate tag and a reader-database in order to exploit this vulnerability:

(1) $R \rightarrow T$: n_1
(2) $T \rightarrow R$: $M_1 = CRC(EPC_{\mathtt{x}}||n_1||n_2) \oplus K_{\mathtt{x}}$, n_2
(3) $R \rightarrow Database$: M_1 , n_1, n_2
(4) $Database \rightarrow R$: $M_2 = CRC(EPC_{\mathtt{x}}||n_2) \oplus P_{\mathtt{x}}$
(5) $R \rightarrow T$: M_2

The attacker has to block or disrupt radio channel to obstruct the correct reception of message 5. The objective of this is to prevent the legitimate tag from updating its keys. At this moment, the attacker could supplant the back-end database without knowing all its private information (the six fields described in *Section 2*). In the next tag reading, the database will receive M_1', n_1', n_2'. The fraudulent database has to compute the message M_2'. But from *Corollary 1*, the following expression can be derived:

$$M_2 \oplus M_2' = CRC(EPC||n2) \oplus CRC(EPC||n2')$$
$$CRC(EPC||n2) \oplus CRC(EPC||n2') = CRC(n2 \oplus n2') \quad (13)$$

So, message M_2' can be obtained by means of an xor between the previous M_2 message listened in the air channel and the easily computable value $CRC(n_2 \oplus n_2')$. Message M_2' will be sent to the tag, which will authenticate the fraudulent back-end database and update its keys. An ANSI-C code with the implementation of this attack is available in `http://163.117.149.208/chien/attack3.c`.

4.4 Tracking or private location

Protection against tracking is not guaranteed when tags answer reader queries with the same identifier. In Chien et al.'s protocol, nonces n_1 and n_2 are employed in each session to ensure freshness. With this, it seems that tag's private location is assured, which is not the case as explained bellow:

Weakness 4: Chien et al.'s protocol does not guarantee the location privacy of tags

Proof: The success of this attack depends on preventing the tag key updating. Moreover, if the population of tags is greater that 2^{16}, the normal operation of the protocol will hamper the key update operation (see *Section 4.1*). In the back-end database a pair of keys (new, old) are stored for each tag key. Chien et al. claim that the storage of these two key frustrates DoS attacks. To provide this property, the fact that a tag may use some times the same key (when message M_2 is incorrectly received) to compute message M_1 is considered as a normal operation. In fact, Chien does not specify the maximum number of times that a tag can be authenticated with the same authentication key. Imagine that the reader captures two non-consecutive iterations, upon non-updating key condition (an ANSI-C code with the implementation of this attack is available in `http://163.117.149.208/chien/attack4.c`):

(1) $R \rightarrow T$: $n1$
(2) $T \rightarrow R$: $M_1 = CRC(EPC_{\mathtt{X}}||n_1||n_2) \oplus K^n_{\mathtt{X}}$, n_2
....
(1) $R \rightarrow T$: $n1'$
(2) $T \rightarrow R$: $M'_1 = CRC(EPC_{\mathtt{X}}||n'_1||n'_2) \oplus K^n_{\mathtt{X}}$, n'_2
....

Now, the attacker computes the xor of messages M_1 and M'_1. If messages M_1 and M'_1 came from the same tag, the authentication key $K^n_{\mathtt{X}}$ is canceled when the xor is computed. By means of *Equation 11*, the attacker can verify if answers arise from the same tag:

$$M_1 \oplus M'_1 = CRC(n_1 \oplus n'_1||n_2 \oplus n'_2) \tag{14}$$

A similar attack can be accomplished using messages sent by the database (M_2). Suppose that a croocked reader interrogates a tag: reader sends message M_1 to the database. The database authenticates the tag, and sends back message M_2. M_2 is stored by the reader, and a wrong M_2 message is sent to the tag avoiding its key updating. Then, the above process is repeated obtaining messages M'_1, and M'_2 with nonces n'_1 and n'_2. At this time, the attacker computes the xor of messages M_2 and M'_2 to check if they came from the same tag. From *Corollary 1*, the following equality should hold:

$$M_2 \oplus M'_2 = CRC(n_2 \oplus n'_2) \tag{15}$$

4.5 Back-end database auto-desynchronization

To defend against a DoS attack, Chien et al. propose that the back-end database maintains a pair of keys (new, old) for each tag key. This assumption allows the server to authenticate tags and re-synchronize these each time they suffer a DoS attack. However, the normal operation of the protocol results in a synchronization loss between the database and the tags due to the non-unequivocal identification property described in *Section 4.1*.

Weakness 5: Chien et al.'s protocol is vulnerable to auto-desynchronization attacks.

We have simulated a population of N tags. For each tag the EPC, K, and P values are randomly initialized. These values will be stored both in the tag and in the back-end database. Upon initialization, we simulate the reading of N tags. For each reading, the following process is repeated:

(1) Reading of $\mathtt{tag_X}$
(2) $\mathtt{M} = \mathtt{CRC}(\mathtt{EPC_X} \;||\; \mathtt{n_1} \;||\; \mathtt{n_2}) \oplus \mathtt{K_X}$

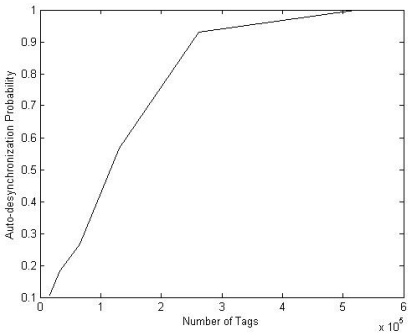


Fig. 2: Auto-desynchronization Probability

```
(3) Send M, n1, n2 to the back-end database
(4) while ((x'<N)&&(output==0))
    {M'= CRC(EPCx, || n1 || n2) ⊕ Kx,
    if (M' == M) autodesyn[x']++; output=1
    x'++}
```

Upon the reading of the N tags, we reckon the number of times that auto-desynchronization occurred. After the reading of a tag, the authentication and access key are updated both in the database and in the tag. An additionally wrong update, during the reading of a different tag, will cause a synchronization loss for that tag. Therefore, the number of tags whose keys have been updated two or more times constitute the number of de-synchronized tags. So, the probability of auto-desynchronization (P_{ADS}) can be defined as:

$$P_{ADS} = \frac{1}{N}\sum_{x=1}^{N}(autodesyn[x] - 1) \tag{16}$$

The above process is repeated T times ($T = 1000$) in order to obtain an estimation of the P_{ADS}. We have simulated the experiment with different population values (N=2^{14}, 2^{15}, 2^{16} ... 2^{18}), as displayed in *Fig. 2*. The results point out that if we have a population of $N \geq 2^{17}$ tags, the probability of auto-desynchronization is greater than 0.5. This probability increases to 0.93 if the population is $N \geq 2^{18}$.

5 Conclusions

Due to the security faults both of EPC-C1G2 and of the previous proposals conforming to this standard, in 2007 Chien et al. proposed a new mutual authentication protocol that tried to solve these problems. After briefly presenting Chien scheme, the security of this protocol has been analyzed, showing some important security failures: non-unequivocal-identification, identity impersonation

(both of tags and, more importantly, of the back-end database), tracking, and auto-desynchronization.

We have also shown that all security weaknesses are related with the abusive use of the CRC-16 included in the EPC-C1G2 standard. Some of them (non-unequivocal identification and autodesynchronization) could have been solved simply by using larger CRCs (well above the 16-bits proposed in the standard). Settling the rest of the security problems highlighted in this article is much harder, because they are due to the linear properties of CRCs, and will not be solved by changing the CRC length or polynomial. Instead, as classical cryptographic primitives are not supported in low-cost RFID tags, new lightweight cryptographic primitives should be used, taking into account the severe computational and storage limitations of these devices. Indeed, CRCs are designed to detect random errors in messages and not to be resilient against malicious attacks by knowledgeable users. This design pitfall is, however, not at all new: for example, the disaster of SSH v1.5 [8, 11] and WEP [4] protocols were in great part due to a similar CRC abuse.

References

1. Philips and Texas Instruments join forces to accelerate EPC Gen-2 RFID deployment. http://www.nxp.com/news/content/file1171.html, 2005.
2. Anarchriz. CRC and how to reverse it. http://www.woodmann.com/fravia/crctut1.htm, 1999.
3. D. Bailey and A. Juels. Shoehorning security into the EPC standard. Manuscript in submission, 2006.
4. N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Proc. of MobiCom'01*, ACM, pages 180–189, 2001.
5. H.-Y. Chien and C.-H. Chen. Mutual authentication protocol for RFID conforming to EPC Class-1 Generation-2 standards. *Computer Standards & Interfaces, Elsevier Science Publishers*, 29(2):254–259, February 2007.
6. Class-1 Generation-2 UHF air interface protocol standard Version 1.0.9: "Gen-2". http://www.epcglobalinc.org/, 2005.
7. EPC Generation-1 tag data standards Ver. 1.1. http://www.epcglobalinc.org/, 2005.
8. A. Futoransky and E. Kargieman. An attack on CRC-32 integrity checks of encrypted channels using CBC and CFB modes. http://www.coresecurity.com/files/attachments/CRC32.pdf, 1999.
9. A. Juels. Strengthening EPC tags against cloning. Manuscript, 2005.
10. D. Nguyen Duc, J. Park, H. Lee, and K. Kim. Enhancing security of EPCglobal Gen-2 RFID tag against traceability and cloning. In *Proc. of Symposium on Cryptography and Information Security*, 2006.
11. N. Provos and P. Honeyman. Scanssh - scanning the internet for ssh servers. In *In Proc. of USENIX'01*, 2001.
12. M. Stigge, H. Pltz, W. Mller, and J.-P. Redlich. Reversing CRC theory and practice. Technical Report SAR-PR-2006-05, Humboldt-Universitat Berlin, 2006.
13. B. Westerbaan. Reversing CRC. http://blog.w-nz.com/archives/2005/07/15/reversing-crc/, 2005.