Secure EPC Gen2 compliant Radio Frequency Identification

Mike Burmester[1], Breno de Medeiros[2], Jorge Munilla[3], and Alberto Peinado[3]

[1] Department of Computer Science
Florida State University, Tallahassee, FL 32306, USA
burmester@cs.fsu.edu
[2] Information Security Consultant Santa Clara, CA 95054, USA
breno@brenodemedeiros.com
[3] Departamento de Ingeniería de Comunicaciones
Universidad de Málaga, Spain
munilla@ic.uma.es, apeinado@ic.uma.es

Abstract. The increased functionality of EPC Class1 Gen2 (EPCGen2) is making this standard a de facto specification for inexpensive tags in the RFID industry. Recently three EPCGen2 compliant protocols that address security issues were proposed in the literature. In this paper we analyze these protocols and show that they are not secure and subject to replay/impersonation and statistical analysis attacks. We then propose an EPCGen2 compliant RFID protocol that uses the numbers drawn from synchronized pseudorandom number generators (RNG) to provide secure tag identification and session unlinkability. This protocol is optimistic and its security reduces to the (cryptographic) pseudorandomness of the RNGs supported by EPCGen2.

Keywords: EPCGen2 compliance, security, identification, unlinkability.

1 Introduction

Radio Frequency Identification (RFID) is a promising new technology that is widely deployed for supply-chain and inventory management, retail operations and more generally for automatic identification. The advantage of RFID over barcode technology is that it is wireless and does not require direct line-of-sight reading. Furthermore, RFID readers can interrogate tags at greater distances, faster and concurrently.

One of the most important advantages of RFID technology is that tags have read/write capability, allowing stored tag information to be altered dynamically. Typically an RFID system consists of tags, one or more readers, and a backend server. The communication channel between the reader and the backend server is assumed to be secure while the wireless channel between the reader and the tag is assumed to be insecure.

To promote the adoption of RFID technology and to support interoperability, EPCGlobal [10] and the International Organization for Standards (ISO) [12] have been actively engaged in defining standards for tags, readers, and the communication protocols. A recently ratified standard is EPC Class 1 Gen 2 (EPCGen2). This defines a platform for the interoperability of RFID protocols, by supporting efficient tag reading, flexible bandwidth use, multiple read/write capabilities and basic reliability guarantees, provided by an on-chip 16-bit Pseudo-random Number Generator (RNG) and a 16-bit Cyclic Redundancy Code (CRC16). EPCGen2 is designed to strike a balance between cost and functionality, with little attention paid to security.

In this paper we are concerned with the security of EPCGen2 compliant protocols. Clearly one has to take into account the additional cost for introducing security into systems with restricted capability. It is important therefore to employ lightweight cryptographic protocols that are compatible with the existing standardized specifications. Several RFID authentication protocols that address security issues using cryptographic mechanisms have been proposed in the literature. Most of these use hash functions [17, 22, 2, 8, 20, 9, 16], which are beyond the capability of most low-cost tags and are not supported by EPCGen2. Some protocols use pseudorandom number generators (RNG) [22, 13, 5, 4, 21, 3], a mechanism that is supported by EPCGen2, but these are not optimized for EPCGen2 compliance. One can also use the RNG supported by EPCGen2 as a pseudorandom function (PRF) (as in [3, 11]) to link challenge-response flows, however it is not clear if such protocols are vulnerable to *related key* attacks [3].

The research literature for RFID security is extensive. We refrain from a detailed review, and refer the reader to a comprehensive repository available online at [1]. Recently three RFID authentication protocols specifically designed for compliance with EPCGen2 have been proposed [7, 18, 19]. These combine the CRC-16 of the EPCGen2 standard with its 16-bit RNG to hash, randomize and link protocol flows, and to prevent cloning, impersonation and denial of service attacks. In this paper we analyze these protocols and show that they do not achieve their security goals. One may argue that, because the EPCGen2 standard supports only a very basic RNG, any RFID protocol that complies with this standard is potentially vulnerable, for example to ciphertext-only attacks that exhaust the range of the components of protocol flows. While this is certainly the case, such attacks may be checked by using additional keying material and by constraining the application (e.g., the life-time of tags). We contend that there is scope for securing low cost devices. Obviously, the level of security may not be sufficient for sensitive applications. However there are many low cost applications where there is no alternative.

The rest of this paper is organized as follows. Section 2 introduces the EPCGen2 standard focusing on security issues. Section 3 analyzes three recently proposed EPCGen2 protocols. In Section 4 we propose a novel EPCGen2 compliant protocol that provides tag identification and session unlinkability. In Section 5 we define a security framework for Radio Frequency Identification, and show that our protocol is secure in this framework.

2 The EPCGen2 standard

EPC Global UHF Class 1 Gen 2, commonly known as the EPCGen2, was approved in 2004, and ratified by ISO as an amendment to the 18000-6 standard in 2006. This standard defines the physical and logical requirements for a passive-backscatter, Interrogator-talks-first (ITF), radio-frequency identification (RFID) system operating in the 860 MHz - 960 MHz frequency range. The EPCGen2 standard defines a protocol with two layers, the physical and the Tag-identification layer, which specify the physical interactions, the operating procedures and commands, and the collision arbitration scheme used to identify a Tag in a multiple-Tag environment.

The system comprises Interrogators, also known as Readers, and Tags. Below we briefly summarize the EPCGen2 requirements.

1. Physical Layer
   - Communications are half-duplex, meaning that Interrogators and Tags cannot talk simultaneously.
   - An Interrogator transmits information to a Tag by modulating an RF signal. Tags are passive, meaning that they receive all of their operating energy from the Interrogator's RF waveform, as well as information.
   - An Interrogator receives information from a Tag by transmitting a continuous wave (CW) RF signal to the Tag; the Tag responds only after being directed to do so by an Interrogator, by modulating the reflection coefficient of its antenna, thereby backscattering a weak signal.
2. Tag memory is logically separated into four distinct banks
   - Reserved memory that contains a 32-bit kill password (KP) to permanently disable the tag, and a 32-bit access password (AP) used when the Interrogator wants to write/read the memory.
   - EPC memory that contains the parameters of a CRC16 (16 bits), protocol control (PC) bits (16 bits), and an electronic product code EPC that identifies the Tag (32 bits).
   - TID memory that contains sufficient information to identify to a Reader the (custom/optional) features of the tag and tag/vendor specific data.
   - User memory that allows user-specific data storage
3. Tag-identification layer
   - An Interrogator manages Tag populations using three basic operations: *Select* (the operation of choosing a Tag population), *Inventory* (the operation of identifying Tags) and *Access* (the operation of reading from and/or writing to a Tag).
   - The Interrogator begins an inventory round by transmitting a Query command in one of four sessions. An inventory operates in only one session at a time, and the Interrogator inventories Tags within that session.
   - A random-slotted collision algorithm is used. The Interrogator sends a parameter Q, that is an integer in the range $(0, 15)$; the Tags load a random Q-bit number into a slot counter. Tags decrement this slot counter when they receive a Query, and reply to the Interrogator when

their counter reaches zero. When the Interrogator detects the reply of a Tag, it requests its PC, EPC, and CRC16.
   – Link cover-coding can be used to obscure information during Reader to Tag transmissions. To cover-code data (or a password), an Interrogator first requests a random number from the Tag. Then, the Interrogator performs a bit-wise XOR of the data with this random number, and transmits the result (cover coded or ciphertext) to the Tag.
4. Hardware requirements
   – A 16-bit Pseudo-Random number generator (RNG).
   – A 16-bit Cyclic Redundancy Code.

2.1 The Pseudo-Random Number Generator

A pseudorandom number generator (RNG) is a deterministic function that outputs a sequence of numbers that are indistinguishable from random numbers by using as input a random binary string, called *seed*. The length of the random seed must be selected carefully to guarantee that the numbers generated are pseudorandom. The state of the RNG changes each time that a new random number is drawn. Although EPCGen2 does not specify any structure for the RNG, it defines the following randomness criteria.

1. **Probability of RN16**: The probability that a pseudorandom number RN16 drawn from the RNG has value RN is bounded by:

$$0.8/2^{16} < Prob(RN16 = RN) < 1.25/2^{16}.$$

2. **Drawing identical sequences**: For a tag population of up to 10,000 tags, the probability that any two or more tags simultaneously draw the same sequence of RN16s is $< 0.1\%$, regardless of when the tags are energized.
3. **Next-number prediction**: A RN16 drawn from a tag's RNG is not predictable with probability better than 0.025%, given the outcomes of all prior draws.

We refer the reader to the discussion in [3] regarding the strength of EPCGen2 compliant RNGs.

2.2 The 16-bit Cyclic Redundancy Code

Cyclic Redundancy Codes (CRC) are error-detecting codes that check accidental (non-malicious) errors caused by faults during transmission. To compute the CRC of a bit string $B = (B_0, B_1, \ldots, B_{m-1})$ we first represent it by a polynomial $B(x) = B_0 + B_1 x + \cdots + B_{m-1} x^{m-1}$ over the finite field $GF(2)$, and then compute its remainder: $CRC(B(x)) = (B(x) \cdot x^n) \bmod g(x)$, for an appropriate generator polynomial $g(x)$ of degree n.

EPCGen2 uses the CRC-CCITT generator: $x^{16}+x^{12}+x^5+1$, and XORs a fixed bit pattern to the bitstream to be checked. EPCGen2 specifies the Cyclic Redundancy Code CRC16 which, for a 16-bit number B is defined by:

$$CRC(B) = [\, B(x)\cdot x^{16} + \sum_{i=16}^{31} x^i \,] \bmod g(x) = B(x)x^{16} \bmod g(x) \,+\, CRC(0),$$

where $CRC(0) = \sum_{16}^{31} x^i \bmod g(x)$ is a fixed polynomial. Since the modulo $g(x)$ operator is a homomorphism, CRC16 inherits strong linearity aspects. More specifically, if P, Q are 16-bit numbers, then

$$CRC(P(x)+Q(x)) = CRC(P(x)) + CRC(Q(x)) + CRC(0). \qquad (1)$$

It follows that the CRC16 of a sequence of numbers can be computed from the CRC16s of the numbers. Consequently CRC16 by itself will not protect data against intentional (malicious) alteration. Its functionality is to support strong error detection particularly with respect to burst errors, not security.

3 Weaknesses recently proposed EPCGen2 compliant RFID protocols

In this section we consider three recently proposed EPCGen2 compliant protocols: the Chen-Deng mutual authentication protocol [7], the Quinling-Yiju-Yonghua minimalist mutual authentication protocol [18], and the Sun-Ting authentication protocol [19]. We show that these protocols fall short of their claimed security.

In the protocols below we use the following notation: $\mathcal{S}$ is the back end server, $\mathcal{R}$ a Reader, $\mathcal{T}$ a tag. We assume that $\mathcal{S}$ and $\mathcal{R}$ are linked with a secure channel, and for simplicity, only consider the case when the authentication is online.

3.1 Analysis of the Chen-Deng protocol

In the Chen-Deng mutual authentication protocol [7] each tag $\mathcal{T}$ shares three private values with the back end server $\mathcal{S}$: a key K, a nonce N and an EPC identifier. The tag stores these in non-volatile memory and the server stores them in a database DB. The protocol has three passes:

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: query, R_r, a random number, and $P = CRC(N \oplus R_r)$.
   $\mathcal{T}$: Check that P is correct. If it is correct,
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: R_t, a random number, $X = (K \oplus EPC \oplus R_t)$ and
   $Y = CRC(N \oplus X \oplus R_t)$.
   $\mathcal{S}$: Check that X, Y are correct. If they are correct,
3. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: M_{resp}, a response message.

This protocol is clearly subject to a replay attack since the flows from the Reader $\mathcal{R}$ and tag $\mathcal{T}$ use independent randomness (and hence are independent). In fact the adversary needs only one interrogation of $\mathcal{T}$: R_t, $X = (K \oplus EPC \oplus R_t)$ and $Y = CRC(N \oplus X \oplus R_t)$, to impersonate the tag by computing a valid (R_a, X^*, Y^*), for any random number R_a, as: $X^* = X \oplus (R_t \oplus R_a)$, $Y^* = Y$.

3.2 Analysis of the Quinling-Yiju-Yonghua protocol

The Quinling-Yiju-Yonghua protocol is a challenge-response mutual authentication protocol [18]. Each tag $\mathcal{T}$ shares two private 32-bit values with the back end server $\mathcal{S}$: an access password aPW and a tag identifier $TID = TID_h||TID_l$, where TID_h (TID_l) are the high 16-bits (low 16-bits) of TID. $\mathcal{T}$ stores these in non-volatile memory and $\mathcal{S}$ stores them in a database DB. The protocol has three passes.

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: query, and R_r, a 16-bit random number.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: R_t, a 16-bit random number, and $M = (M_l||M_h) \oplus aPW$, where $M_l = CRC(TID_l \oplus R_r \oplus R_t)$ and $M_h = CRC(TID_h \oplus R_r \oplus R_t)$.
   $\mathcal{S}$: Check that M is correct. If so, the tag is accepted as the authorized $\mathcal{T}$,
3. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: $N = (N_l||N_h) \oplus aPW$, where $N_l = CRC(TID_l \oplus R_t)$ and $N_h = CRC(TID_h \oplus R_t)$.
   $\mathcal{T}$: Check that N is correct. If it is, it accepts that $\mathcal{R}$ is an authorized reader.

In this protocol the flows from the tag $\mathcal{T}$ and Reader $\mathcal{R}$ use combined randomness and are dependent. Therefore one cannot use an identical flow for a replay attack. However, because of the strong linearity aspects of CRC16, it is easy for the adversary to modify the protocol flows from an interrogation of $\mathcal{T}$ to get the flow for a replay attack. Suppose that the adversary is given: R_r, R_t and M from a previous successful interrogation; and let R_r^* be the 16-bit random challenge of the Reader for a new interrogation. Then the adversary $\mathcal{A}$ can choose any 16-bit random number, R_a, and compute: $A = CRC(R_r \oplus R_r^* \oplus R_a) \oplus CRC(0)$, and send a valid response to $\mathcal{S}$:

$$R_t^* = R_t \oplus R_a \ , \ M^* = M \oplus (A||A),$$

since $M_l^* = M_l \oplus A$ and $M_h^* = M_h \oplus A$, by Equations (1). Therefore the tag $\mathcal{T}$ can be cloned after an eavesdropped interrogation. Impersonating the Reader is even simpler: $\mathcal{A}$ does not need a previous interrogation. $\mathcal{A}$ sends any value R_r^* to an authorized tag $\mathcal{T}$ to get M^* from $\mathcal{T}$. Then, $\mathcal{A}$ can compute a valid $N^* = M^* \oplus (A'||A')$, where $A' = CRC(R_r^*) \oplus CRC(0)$.

3.3 Analysis of the Sun-Ting Gen2$^+$ protocol

Gen2$^+$ [19] is a four pass mutual authentication protocol. Each tag shares with the back end server $\mathcal{S}$ a random l-word string k ($l \leq 127$) called *keypool*. $\mathcal{S}$ stores the keypool of each tag $\mathcal{T}$ together with its EPC and other identifying data in a database DB. In the protocol $\mathcal{T}$ gets identified by revealing information about its keypool, which $\mathcal{S}$ uses to locate the tag in DB. The keypool of each tag is updated every 14 successful authentications to prevent cloning attacks. We briefly describe the protocol.

1. $\mathcal{R} \to \mathcal{T}$: query
   $\mathcal{T}$: Draw a 16-bit pseudorandom number, and use the first 14 bits as 7-bit addresses, a and b, to mark a segment $k[a : b]$ of the keypool, and the last two bits to compute a *check* by XORing the two lsb of the a-th word and the b-th word. If $a \geq b$, the segment $k[a : b]$ contains the words from a to b, otherwise $k[a : b] = k[a : l - 1]||k[0 : b]$.
2. $\mathcal{T} \to \mathcal{R} \Rightarrow \mathcal{S}$: a, b, *check*
   $\mathcal{S}$: First compute *check* for every $k \in DB$, and remove those keypools k with different *check*. Then compute the $CRC(k[a : b])$ of all remaining keypools in the reduced database DB', and finally compute the *central key* ck', whose bits are obtained by taking a majority vote in the corresponding positions of the $CRC(k[a : b])$ in DB' (0 dominates 1).
3. $\mathcal{S} \Rightarrow \mathcal{R} \to \mathcal{T}$: ck'
   $\mathcal{T}$: Compute $ck = CRC(k[a : b])$ for the locally stored keypool and compare it with ck': if their Hamming distance is greater than a threshold t (typically $t = 1$) do not respond. Otherwise, send the locally stored EPC.
4. $\mathcal{T} \to \mathcal{R}$: nothing or EPC
   $\mathcal{S}$: If there is no response from $\mathcal{T}$ then remove from DB' those keypools k for which the Hamming distance of CRC($k[a : b]$) from ck' is less or equal to t, and repeat Step 1.
   If the EPC of one of the tags $\mathcal{T}$ in DB is received, then $\mathcal{T}$ is identified, and $\mathcal{R}$ is considered authentic by the tag.

This protocol is clearly subject to replay attacks because only the tag contributes to the randomness of protocol flows. The adversary $\mathcal{A}$ needs to eavesdrop on only one tag interrogation to get the required protocol flows. The protocol is also subject to a more complex statistical attack in which $\mathcal{A}$ first eavesdrops on a number of tag interrogations and then replays the tag flows to the Reader $\mathcal{R}$, changing adaptively the last challenge. This makes it possible for $\mathcal{A}$ to build up gradually sufficient information about the CRC's of the words in a tag's keypool so as to clone the tag. Below we describe the attack in more detail.

1. $\mathcal{A}$ eavesdrops on $m < 14$ successful interrogations of $\mathcal{T}$ (prior to a keypool update). $\mathcal{A}$ stores for every interrogation the values:

$$([a, b, check]_1, ck'_1), ([a, b, check]_2, ck'_2), \dots, ([a, b, check]_p, ck'_p),$$

   where p is the number of challenges in the interrogation ($p \approx \log(T)/log(4)$, where T is the total number of tags).
2. $\mathcal{A}$ impersonates $\mathcal{T}$ and replays all but one of the challenges in each interrogation. The last challenge is replaced by $[x, x, 00]_p$, $0 \leq x \leq l$. $\mathcal{R}$ responds with x' computed by taking a majority vote on the $CRC(k[x : x])$ for all

keypools k in the reduced DB'. Note that repeating the first $(p-1)$ rounds guarantees that the target tag is always in DB'. $\mathcal{A}$ repeats this step for each one of the l words of the *keypool*.

3. $\mathcal{A}$ analyzes the collected data. Let n be the number of keypools remaining in DB' after the penultimate round $(p-1)$. $\mathcal{A}$ can compute the CRC16 of the word x in the keypool of $\mathcal{T}$, because of the binary structure of ck': e.g., when $n=1$ then $ck' = CRC(x)$ and when $n=2$, ck' is strongly biased with 3/4 of its bits being 0. The case $n=2$ is particularly important because it occurs with high probability ($> 48\%$, for $T=1000$, $l=127$, and $t=1$). Using this information it is now possible to compute the $CRC(w)$ of the word w in the keypool of $\mathcal{T}$.
4. $\mathcal{A}$ now impersonates $\mathcal{R}$ to $\mathcal{T}$ and tries to compute a valid ck' for a given $[a, b, check]$. By exploiting the linearity aspects of CRC16, the CRC16 of an interval $k[a:b] = w_a \cdots w_b$ can be computed from the CRC16s of its words:

$$CRC(k[a:b]) = \bigoplus_{i=a}^{b} CRC^{i-a+1}(w_i) \oplus \bigoplus_{1}^{(b-a-1)} CRC^{i}(0),$$

where CRC^i is CRC iterated i-times. Note also that there is no bound on the number of times that $\mathcal{A}$ can try to compute a valid ck', since the number of challenges in an interrogation is not bounded.

This attack can be modified and enhanced in different ways. For example, $\mathcal{A}$ could use the different tydbit *check*s sent by the tag to guess the values of the lsb of different words, or ask for intervals of different length and combine this with the previous analyzed data. $\mathcal{A}$ could also simplify the attack, by trying to find the CRC of only short block words, and then wait until $\mathcal{T}$ asks for an interval that can be made from these blocks.

4 Gen2Sec: a Secure EPCGen2 compliant RFID protocol

We next consider a novel Radio Frequency Identification protocol, Gen2Sec, which only uses the RNG supported by EPCGen2 for security.

4.1 The protocol

In our protocol each tag $\mathcal{T}$ is identified by drawing consecutive numbers from its RNG, say g_{tag}. $\mathcal{T}$ draws three numbers, RN_1, RN_2, RN_3, and sends RN_1 to the server $\mathcal{S}$ as a commitment. If $\mathcal{S}$ shares the RNG of the tag (its current state), and if both RNGs are synchronized, then $\mathcal{S}$ can also draw these same numbers. It can therefore reply to the tag with the challenge RN_2. $\mathcal{T}$ now sends RN_3 as its response. This third step is also used to keep the RNGs of $\mathcal{S}$ and $\mathcal{T}$ synchronized. One more challenge-response round is needed to deal with replay attacks when these are detected (an *alarm* triggers this): $\mathcal{S}$ then draws and sends the next number RN_4 as challenge and $\mathcal{T}$ responds by sending RN_5.

Altogether three numbers are drawn when the adversary is passive and five when the adversary is active. The security of the protocol is based on the fact that the random numbers sent by the tag cannot be predicted by the adversary, and consecutive numbers drawn in each interrogation are pseudorandom. Our protocol *identifies tags* (not Readers) and is provably secure. It offers a degree of *privacy* (session unlinkability), as we shall see in the following section.

We now describe the protocol in detail. Each tag $\mathcal{T}$ shares with the back end server $\mathcal{S}$ an identifier ID_{tag}, its RNG g_{tag} (the state of g_{tag}) and at least one pseudorandom number (this guarantees synchronization). $\mathcal{S}$ stores in a database for each tag a list of seven numbers, ID_{tag} and g_{tag}:

$$DB = \{RN_1^{old}, RN_1^{cur}, RN_1^{next}, RN_2, RN_3, RN_4^{cur}, RN_5^{cur}; ID_{tag}, g_{tag}\}.$$

The lists of DB are doubly indexed by RN_1^{next} and RN_1^{cur} respectively. The tag $\mathcal{T}$ stores in non-volatile memory two pseudorandom numbers, its identifier and (the seed for) g_{tag}:

$$(RN_1, RN_2, ID_{tag}, g_{tag}).$$

To initialize the values of its variables, the tag draws two successive values RN_1, RN_2 from g_{tag}. $\mathcal{S}$ draws six successive numbers from the RNG of each tag and assigns their values to the variable in the tags lists: RN_1^{cur}, RN_2, RN_3, RN_4^{cur}, RN_5^{cur}, RN_1^{next} (in this order). In the protocol $\mathcal{S}$ uses a *timer* and an *alarm* to manage inventories, thwart man-in-the-middle relay attacks and avoid replay attacks, as well as an *update* function in which: $RN_1^{cur} \leftarrow RN_1^{next}$, and the five values $RN_2, RN_3, RN_4^{cur}, RN_5^{cur}, RN_1^{next}$, are updated by drawing new numbers from g_{tag}.

Gen2Sec Protocol

1. $\mathcal{R} \rightarrow \mathcal{T}$: query
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: RN_1
   $\mathcal{S}$: Check in DB
   If $RN_1 = RN_1^{cur}$ for an item in DB then:
      If $RN_1 = RN_1^{old}$ then set $alarm \leftarrow 1$, set $timer$ and broadcast RN_2.
      Else set $RN_1^{old} \leftarrow RN_1$, set $alarm \leftarrow 0$, set $timer$ and broadcast RN_2.
   If $RN_1 = RN_1^{next}$ for an item in DB then $RN^{old} \leftarrow RN_1$, $update$,
      set $alarm \leftarrow 0$, set $timer$ and broadcast RN_2.
3. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: RN_2
   $\mathcal{T}$: Check RN_2.
   If RN_2 is valid then draw five successive numbers from g_{tag} and assign them to the variables RN_3, RN_4, RN_5 (volatile), RN_1, RN_2, and broadcast RN_3.
   $\mathcal{S}$: On timeout abort.
4. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: RN_3
   $\mathcal{S}$: Check RN_3.
   If RN_3 is valid for ID_{tag} then:
      If $alarm = 0$ then $update$ and ACCEPT that $\mathcal{T}$ has identifier ID_{tag}.

Else set $RN_4 \leftarrow RN_4^{cur}$, $RN_5 \leftarrow RN_5^{cur}$, $update$, and broadcast RN_4.
Else abort.

5. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: RN_4
   $\mathcal{T}$: Check RN_4.
   If it is valid then broadcast RN_5.
   $\mathcal{S}$: On timeout abort.
6. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: RN_5
   $\mathcal{S}$: Check RN_5.
   If RN_5 is valid for ID_{tag} then ACCEPT that $\mathcal{T}$ has identifier ID_{tag}.
   Else abort.

This protocol is *optimistic* in the sense that a tag $\mathcal{T}$ need only use three pseudorandom numbers to get identified when the adversary $\mathcal{A}$ is passive. $\mathcal{T}$ sends a commitment in Pass 1, $\mathcal{S}$ sends a challenge in Pass 2, and $\mathcal{T}$ gets identified in Pass 3. $\mathcal{A}$ may try to impersonate $\mathcal{T}$ by obtaining the flows RN_1, RN_2, RN_3, through an offline man-in-the-middle attack for a discussion on such attacks). However this would cause the Server $\mathcal{S}$ to activate the $alarm$. When this happens an additional interrogation is needed (Pass 5 and Pass 6). If $\mathcal{A}$ attempts to replay the numbers $RN_1, RN_2, RN_3, RN_4, RN_4, RN_5$, $\mathcal{A}$ will fail because in the mean time $\mathcal{S}$ and $\mathcal{T}$ will have updated the locally stored values of the pseudorandom numbers.

In the following section we will discuss the security issues of this protocol in a formal framework.

5 A security framework for RFID

5.1 RFID deployments

A typical RFID deployment involves tags $\mathcal{T}$, Readers $\mathcal{R}$ and a back end Server $\mathcal{S}$. Tags are wireless transponders that typically have no power of their own and respond only when they are in an electromagnetical field, while Readers are transceivers that generate such fields. Readers implement a radio interface to the tags and a high level interface to a back end server. $\mathcal{S}$ is a trusted entity that processes private tag data. Readers do not store locally any private data.

We adopt the Byzantine threat model. All parties including the adversary $\mathcal{A}$ are modeled as a probabilistic Turing machines. $\mathcal{A}$ controls the delivery schedule of all communication channels, and may eavesdrop into, or modify, their contents and may also instantiate new communication channels and directly interact with honest parties. However the channels that link the Server and authorized Readers are assumed to be secure. Readers do not store any private tag information.

5.2 The UC framework

The universal composability (UC) framework specifies a particular approach to security proofs for protocols, and guarantees that proofs that follow that approach remain valid if the protocol is, say composed with other protocols (modularity) and under arbitrary concurrent protocol executions (including with itself). The UC framework defines a *real-world simulation*, an *ideal-world simulation*, an *emulation* $\mathcal{E}$ that translates protocol runs from the real-world to the ideal-world, and an interactive environment $\mathcal{Z}$ that captures whatever is external to the current protocol execution. The components of a UC security formalization are:

1. A *mathematical model* of real protocol executions in which honest parties (the tags and the Server) correctly execute as specified, and adversarial parties under the control of the adversary $\mathcal{A}$ that can deviate from the protocol in an arbitrary way. $\mathcal{A}$ can interact with the environment $\mathcal{Z}$, in arbitrary ways.
2. An *idealized model* of executions, where the security properties of the protocol depend on the behavior of an *ideal functionality* $\mathcal{F}$. $\mathcal{F}$ controls the ideal-model adversary $\widehat{\mathcal{A}}$ so that it reproduces as faithfully as possible the behavior of $\mathcal{A}$.
3. A proof that, for each adversary $\mathcal{A}$ there is a simulator $\mathcal{E}$ that translates real-world runs in the presence of $\mathcal{A}$ into ideal-world protocol runs in the presence of $\widehat{\mathcal{A}}$ such that, no environment $\mathcal{Z}$ can distinguish whether $\mathcal{A}$ is communicating with a instance of the protocol in the real-world or $\widehat{\mathcal{A}}$ is communicating with $\mathcal{F}$ in the ideal-world.

In the UC framework, the context of a protocol execution is captured by a session identifier *sid*. The *sid* is controlled by the environment Z and reflects external aspects of execution. All parties involved in a protocol execution instance share the same *sid*.

Theorem 1. *Gen2Sec guarantees* availability, tag authentication *and* session unlinkability *in the UC framework provided a cryptographically secure RNG is used.*

Proof. We sketch an outline of the proof. First we specify the functionality $\mathcal{F}_{auth}$ of the protocol to capture availability, tag authentication and session unlinkability.

1. Availability requires that the Server and tags be synchronized at all times.
2. Tag authentication requires that the Server can corroborate values produced by the tag in terms of the state of their shared RNG.
3. Session unlinkability requires that: given two tag interrogations $\mathcal{A}$ cannot decide (with probability better than 0.5 + negligible) whether these involve the same tag or not, provided that either the first completed successfully, or an intermediate interrogation of the tag completed successfully.

The functionality $\mathcal{F}_{auth}$ is illustrated in Figure 1. There are four commands: Initiate activates the Server and tags, Send is used to send an output of one party (tag or Server) to the other (Server or tag) and get their response, Repeat is used to repeat interrogations that were not completed (the adversary did not send the required flows), and Impersonate is used to impersonate tags. Observe that in both the protocol and $\mathcal{F}_{auth}$, the receiving party of any message

Functionality $\mathcal{F}_{\mathsf{auth}}$

$\mathcal{F}_{\mathsf{auth}}$ has session identifier *sid* and only admits commands with the same *sid*.

Upon receiving INITIATE **from** *Server*: Generate a unique subsession identification number s_{ser}. Create a new DB, record and send to $\mathcal{A}$: $\mathtt{flow}(s_{ser}, \cdot, Query)$.

Upon receiving INITIATE **from** *tag*: if $\mathtt{flow}(s_{ser}, \cdot, Query) \in DB$, then generate a unique subsession identification number s_{tag}, select five random numbers: r_1, r_2, r_3, r_4 and r_5 and assign them to the subsession (s_{ser}, s_{tag}). Set $alarm \leftarrow 0$. Send (tag, s_{tag}) to $\mathcal{A}$.

Upon receiving SEND(s_{ser}, s_{tag}) **from** $\mathcal{A}$:

If $\mathtt{flow}(s_{tag}, s_{ser}, r_5) \in DB$ then ACCEPT(tag) and delete all flows of (s_{ser}, s_{tag}) in DB.

ElseIf $\mathtt{flow}(s_{ser}, s_{tag}, r_4) \in DB$ then record and send to $\mathcal{A}$: $\mathtt{flow}(s_{tag}, s_{ser}, r_5)$.

ElseIf $\mathtt{flow}(s_{tag}, s_{ser}, r_3) \in DB$ then:

If $alarm = 0$ then ACCEPT(tag) and delete all flows of (s_{ser}, s_{tag}) in DB.

Else record and send to $\mathcal{A}$: $\mathtt{flow}(s_{ser}, s_{tag}, r_4)$.

ElseIf $\mathtt{flow}(s_{ser}, s_{tag}, r_2) \in DB$ then record and send to $\mathcal{A}$: $\mathtt{flow}(s_{tag}, s_{ser}, r_3)$.

ElseIf $\mathtt{flow}(s_{tag}, s_{ser}, r_1) \in DB$ then record and send to $\mathcal{A}$: $\mathtt{flow}(s_{ser}, s_{tag}, r_2)$.

ElseIf $\mathtt{flow}(s_{ser}, \cdot, Query) \in DB$ then record and send to $\mathcal{A}$: $\mathtt{flow}(s_{tag}, s_{ser}, r_1)$.

Else ignore.

Upon receiving REPEAT(s_{ser}, s_{tag}) **from** $\mathcal{A}$:

If $\mathtt{flow}(s_{tag}, s_{ser}, r_3) \in DB$ then ignore.

ElseIf $\mathtt{flow}(s_{ser}, s_{tag}, r_2) \in DB$ then delete all flows of (s_{ser}, s_{tag}) in DB.

Set $alarm \leftarrow 1$, record and send to $\mathcal{A}$: $\mathtt{flow}(s_{ser}, \cdot, Query)$.

ElseIf $\mathtt{flow}(s_{tag}, s_{ser}, r_1) \in DB$ then delete all flows of (s_{ser}, s_{tag}) in DB.

Record and send to $\mathcal{A}$: $\mathtt{flow}(s_{ser}, \cdot, Query)$.

Else ignore.

Upon receiving message IMPERSONATE(s_{ser}, tag) **from** $\mathcal{A}$ **:** If $\mathtt{flow}(s_{ser}, \cdot, Query) \in DB$ and *tag* is corrupted then ACCEPT(tag).

Fig. 1. The functionality of Gen2Sec.

or subroutine output is activated next. For more details on security proofs in the UC framework, the reader is referred to [21].

We must show that a real-world adversary $\mathcal{A}$ who can access protocol flows cannot succeed with probability greater than negligible in generating the flows of a "new" interrogation that is accepted by the Server, but *not* accepted in the ideal-world by $\mathcal{F}_{auth}$ (corresponding to an interrogation that is generated in a way not specified by the protocol): if this happens $\mathcal{Z}$ will distinguish real-world from ideal-world executions.

We first emulate real-world actions in the ideal-world. For this purpose we simulate copies $\widehat{\mathcal{A}}$, of the real adversary, $\widehat{Server}$, of the real Server, $\widehat{tag}$, of real tags, and the interactions of the protocol with $\mathcal{Z}$, in particular its invocations of $\mathcal{F}_{auth}$. For our protocol it is straightforward to show that any interrogation in the real-world that is accepted by the Server is also accepted in the ideal-world by the functionality $\mathcal{F}_{auth}$ because:

1. At all times each tag shares at least one number with the Server (availability);
2. If the Server accepts the tag then a fresh flow of numbers must have been used (tag authentication);
3. If for any two interrogations either the first one completed successfully before the second, or an intermediate interrogation completed successfully, then the tag will have updated the values it stores (session unlinkability).

This first property holds because the values of the stored numbers are updated by $\mathcal{T}$ and $\mathcal{S}$ with each successful execution. If the previous execution of the protocol was not disrupted then $RN_1^{cur} = RN_1$ (in this case one *update* is needed); otherwise we may get $RN_1^{next} = RN_1$ (two *updates* are needed). Note that the numbers RN_3, RN_4 and RN_5 are used only once. For the second observe that the adversary (e.g., a rogue tag or reader) cannot guess the protocol flows because these are generated by a PRN. There is of course a small failure probability due to "lucky" guessing. The adversary cannot clone a tag because it cannot get access to the seed of the RNG of the tag (which is never revealed). For the last observe that, if the first interrogations completed successfully, or an intermediate interrogation completed successfully, then the tag will have updated the values it stores. Finally, in the real world all protocol flows involve pseudorandom numbers whereas in the ideal world we have random numbers: the environment $\mathcal{Z}$ cannot distinguish these because is a PPT machine.

Observe that there are impersonation attacks in the real-world that are not captured in the ideal-world: if a tag updates its RNG while the Server does not (RN_3 was not delivered) then $\mathcal{A}$ can try to impersonate the tag by re-using the flows RN_1 and RN_3. However it will only succeed with negligible probability in guessing RN_5 in response to the Server's query RN_4. Therefore $\mathcal{Z}$ will not see any difference between the successful instances in real-world and ideal-world. □

References

1. Avoine, G. http://lasecwww.epfl.ch/ gavoine/rfid/.
2. Avoine, G., and Oechslin, P. A scalable and provably secure hash based RFID protocol. In *Proc. IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005)* (2005), IEEE Computer Society Press.
3. Burmester, M., and de Medeiros, B. The security of EPC Gen2 compliant RFID protocols. In *ACNS* (2008), S. M. Bellovin, R. Gennaro, A. D. Keromytis, and M. Yung, Eds., vol. 5037 of *Lecture Notes in Computer Science*, pp. 490–506.
4. Burmester, M., de Medeiros, B., and Motta, R. Robust, Anonymous RFID Authentication with Constant Key-Lookup. In *ASIACCS* (2008), M. Abe and V. D. Gligor, Eds., ACM, pp. 283–291. Extended version: *J. Applied Cryptography*, vol. 1(2), 79–90, 2008.
5. Burmester, M., van Le, T., and de Medeiros, B. Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In *Proceedings of the 2nd IEEE/CreateNet International Conference on Security and Privacy in Communication Networks (SECURECOMM 2006)* (2006), IEEE Press.

6. Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS 2001)* (2001), IEEE Press, pp. 136–145.
7. Chen, C.-L., and Deng, Y.-Y. Conformation of EPC Class 1 Generation 2 Standards RFID system with Mutual Authentication and Privacy Protection. *Engineering Applications of Artificial Intelligence*, Elsevier. In Press, Corrected Proof. doi: 10.1016/j.engappai.2008.10.022
8. Dimitriou, T. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Proc. IEEE Intern. Conf. on Security and Privacy in Communication Networks (SECURECOMM 2005)* (2005), IEEE Press.
9. Dimitriou, T. A secure and efficient RFID protocol that can make big brother obsolete. In *Proc. Intern. Conf. on Pervasive Computing and Communications, (PerCom 2006)* (2006), IEEE Press.
10. EPC Global. EPC Tag Data Standards, http://www.epcglobalinc.orgblock
11. Eun Young Choi, D. H. L., and Lim, J. I. Anti-cloning protocol suitable to Epcglobal Class-1 Generation-2 RFID systems. *Computer Standards & Interfaces,* Elsevier. In press, Corrected Proof. doi:10:1016/j.csi.2008.11.002.
12. ISO/IEC. Standard # 18000 – RFID Air Interface Standard. http://www.hightechaid.com/standards/18000.htm.
13. Juels, A. Minimalist cryptography for low-cost RFID tags. In *Proc. Intern. Conf. on Security in Communication Networks (SCN 2004)* (2004), vol. 3352 of *LNCS*, Springer, pp. 149–164.
14. Kim, C. H., Avoine, G., Koeune, F., Standaert, F.-X., and Pereira, O. The Swiss-Knife RFID Distance Bounding Protocol. In *ICISC* (2008), P. J. Lee and J. H. Cheon, Eds., vol. 5461 of *Lecture Notes in Computer Science*, Springer, pp. 98–115.
15. Mangard, S., Popp, T., and Oswald, M. E. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*, Springer - ISBN: 0-387-30857-1. 2007.
16. Molnar, D., Soppera, A., and Wagner, D. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In *Proc. Workshop on Selected Areas in Cryptography (SAC 2005)* (2006), vol. 3897 of *LNCS*, Springer.
17. Ohkubo, M., Suzuki, K., and Kinoshita, S. Cryptographic approach to "privacy-friendly" tags. In *Proc. RFID Privacy Workshop* (2003).
18. Qingling, C., Yiju, Z., and Yonghua, W. A minimalist mutual authentication protocol for rfid system and ban logic analysis. *Computing, Communication, Control and Management, ISECS International Colloquium on 2* (2008), 449–453, doi:10.1109/cccm.2008.305
19. Sun, H.-M., and Ting, W.-C. A Gen2-based RFID authentication protocol for security and privacy. *IEEE Transactions on Mobile Computing 99*, 1 (2009).
20. Tsudik, G. YA-TRAP: Yet another trivial RFID authentication protocol. In *Proc. IEEE Int. Conf. on Pervasive Computing and Communications (PerCom 2006)* (2006), IEEE Press.
21. van Le, T., Burmester, M., and de Medeiros, B. Universally Composable and Forward-secure RFID Authentication and Authenticated Key Exchange. In *Proc. of the ACM Symp. on Information, Computer, and Communications Security* (ASIACCS 2007), (2007), ACM Press, Singapore, pp. 242–252.
22. Weis, S., Sarma, S., Rivest, R., and Engels, D. Security and privacy aspects of low-cost radio frequency identification systems. In *Proc. Intern. Conf. on Security in Pervasive Computing* (2003), vol. 2802 of *LNCS*, Springer, pp. 454–469.