The security of EPC Gen2 compliant RFID protocols

Mike Burmester[1] and Breno de Medeiros[2]

[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

Abstract. The increased functionality of EPC Class1 Gen2 (EPCGen2) is making this standard the de facto specification for inexpensive tags in the RFID industry. EPCGen2 supports only very basic security tools such as a 16-bit Pseudo-Random Number Generator and a 16-bit Cyclic Redundancy Code. Recently two 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 synchronization attacks. We then consider the general issue of supporting security in EPCGen2 compliant protocols and propose two RFID protocols that are secure within the restricted constraints of this standard, and an anonymous RFID mutual authentication protocol with forward secrecy that is compliant with the EPC Class2 Gen2 standard.

Keywords: EPCGen2 compliance, security, anonymity, forward secrecy, unlinkability.

1 Introduction

Radio Frequency Identification (RFID) is a promising new technology that is envisioned to replace barcodes and to be massively deployed for inventory management, supply-chain logistics and retail operations. 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, much faster and concurrently. Perhaps one of the most important advantages of RFID technology is that tags have read/write capability, allowing stored tag information to be altered dynamically. A typical RFID system has three components: tags, one or more readers, and a backend server. The communication channel between the reader and the backend server is (usually) assumed to be secure while the (wireless) channel between the reader and the tag is insecure.

To foster and promote the adoption of RFID technology and to support interoperability, EPCGlobal [13] and the International Organization for Standards

(ISO) [16] have been actively engaged in defining standards for tags, readers, and the communication protocols between them. A recently ratified standard is EPC Class 1 Gen 2 (EPCGen2). This is a communication standard that creates a platform on which to build interoperable RFID protocols. It supports efficient tag reading, flexible bandwidth use, multiple read/write capabilities and basic security guarantees, provided by an on-chip 16-bit Pseudo-Random Number Generator (RNG) and a 16-bit Cyclic Redundancy Code (CRC-16). EPCGen2 is designed to strike a balance between cost and functionality, with less attention paid to security which, arguably, at this stage in the development of RFID technology may be justified.

In this paper we are concerned with the security of EPCGen2 compliant protocols. We recognize that there are situations in which one has to design security into systems with restricted capability so as to promote low-cost widespread use. In such situations it is important to employ protocols that offer the best level of security within the constraints of the specification. Several RFID authentication protocols that address security issues using lightweight cryptographic mechanisms have been proposed in the literature. Most of these use hash functions [21, 20, 25, 15, 1, 3, 11] and [23, 12, 19], which are beyond the capability of most low-cost tags and are not supported by EPCGen2. Some protocols use pseudo-random number generators [25, 17, 5, 4, 24], a mechanism that is supported by EPCGen2, but these are not optimized for EPCGen2 compliance. Other protocols use timestamps (*e.g.* [23]), however these are also not supported by EPCGen2.

The research literature for RFID security is already quite extensive and growing. We refrain from a comprehensive review of the literature, and refer the interested reader to a fairly comprehensive repository available online at [2]. Recently two RFID authentication protocols specifically designed for compliance with EPCGen2 have been proposed [10, 9]. 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 security of EPCGen2 is set to only 16-bits, any RFID protocol is potentially vulnerable, for example to ciphertext-only attacks that exhaust the 16-bit 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 two recently proposed EPCGen2 protocols. Section 4 describes two "trivial" RFID protocols whose security is reduced to the security constraints of EPCGen2, and an anonymous RFID protocol that complies with the EPC Class 2 Gen2 standard. Section 5 considers an extension that captures a kill functionality.

2 The EPCGen2 standard

The EPC Class 1 Generation 2 UHF Air Interface Protocol [13] specifies the operation and functionality of passive RFIDs. It defines the physical and logical requirements for interrogator-talks-first systems that operate in the 860-960 MHz frequency range. Readers (interrogators) transmit information to tags by modulating an RF signal. Tags receive both transmitted information and operating energy from the RF signal. The readers receive information from the tags by transmitting a continuous-wave RF signal, which the tags modulate (backscatter). EPCGen2 deals with the medium access control layer (air interface) and the tag identification layer (physical interactions) of RFID systems. These layers involve RF signaling, managing tag populations, tag singulation (identifying individual tags), collision arbitration (resolving collisions in multi-tag environments), and conformance requirements. A particular attractive feature of EPCGen2 is that it provides for high-speed reading and sortation. The following minimal on-chip tag persistent memory (non-volatile) features are specified:

- Reserved memory that contains a 32-bit kill password (KP) to permanently disable the tag and a 32-bit access password (AP).
- EPC memory that contains: the parameters of a cyclic redundancy code CRC-16 (16 bits), protocol control (PC) bits (16 bits), and an electronic product code EPC that identifies the object to which the tag is (or will be) attached (at least 32 bits).
- TID memory that contains sufficient information to identify to a reader the (custom/optional) features that a tag supports and tag/vendor specific data.
- User memory that allows user-specific data storage.

EPCGen2 also provides for optional user memory and password-protected access control. Two basic mechanisms to protect the tag $\Leftrightarrow$ reader channels are supported:

- A 16-bit Pseudo-Random Number Generator and,
- A 16-bit Cyclic Redundancy Code.

2.1 The Pseudo-Random Number Generator

A pseudo-random number generator (RNG) is a deterministic function that on input a random binary string, called *seed*, outputs a sequence of numbers that are indistinguishable from random numbers. RNGs are constructed from pseudo-random bit generators (RBGs). A RBG stretches a random seed of length k to a pseudo-random binary string of length $\ell \gg k$. This string can be partitioned into blocks of length m to get an m-bit RNG. The length of the random seed must be selected carefully to guarantee that the numbers generated are pseudo-random.

Each number generated by a RNG is said to be *drawn* by the generator. If the numbers drawn by a RNG (or the bits drawn by a RBG) cannot be predicted given the outcomes of prior draws, then the RNG (RBG) is said to be cryptographically secure.

EPCGen2 does not detail the structure of the implemented 16-bit RNG; however it does specify the minimum security levels that should be supported:

1. **Probability of RN16**: The probability that a pseudo-random number RN16 drawn from the RNG has value r is bounded by:

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

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.

Strength of EPCGen2 Compliant RNGs. The strength of a cryptographic RNG is usually expressed in terms of the average maximum length of the sequences of drawn numbers that are indistinguishable from random sequences. With this in mind, we interpret the requirements above as imposing minimal security requirements on the RNGs specified by EPCGen2:

1. The first "randomness" requirement implies that the value of a drawn number cannot be too biased. This requirement, while certainly satisfied by cryptographically secure RNGs, is actually too weak *by itself* to imply much in the way of pseudo-randomness. For instance, a simple counter that increments from 0 to $2^{16} - 1$ and then cycles back satisfies this condition.
2. The second "collision" requirement needs a detailed examination. A well known approximation for the collision problem is $n = \sqrt{2d \ln(1/(1-p))}$ [22, 18], where n is the number of integers drawn randomly with uniform distribution, d the size of the range of the integers, and p the probability that at least two integers have the same value. From this approximation we see that when n is close to $\sqrt{d}$ then the probability of collision is greater than 50%. When $p = 0.1\%$ and $n = 10,000$ we get that d is approximately 2^{36}. That implies that observing two successive numbers (36-bits) drawn from the RNG of two or more tags still guarantees a fairly unbiased (and random-looking) sequence, at least as far as collisions are concerned.
3. The third EPC requirement for the next-number prediction also needs examination. Let RBG be the pseudo-random bit generator that defines the RNG. For the cryptographic security of RBG, the next-bit prediction should be $p = 0.5 + \varepsilon$, ε negligible. This expression can be used to compute the next-number prediction. Let B_0 be the bit-sequence of prior numbers. The prediction for the next 16-bit number is:

$$P = \prod_{i=0}^{i=15} Prob\left(b_{i+1} | B_0 b_0 \cdots b_i\right) = p^{16} = (0.5 + \varepsilon)^{16},$$

where $b_1, \ldots b_i, b_{i+1}, \ldots$, are the bits drawn by the RBG after B_0. EPCGen2 bounds P by 0.025%. Taking $(0.5 + \varepsilon)^{16} < 0.025\%$ we see that ε is bounded 0.094. This is not sufficiently small to provide cryptographic security for the 16-bit RNG, and suggests that the 0.025% bound should be lowered at least one order.

In conclusion, in particular with respect to Condition 2, a reasonably conservative assumption is that the standard for EPCGen2 guarantees the pseudo-randomness of at least two RN16s (32-bits), and probably more, before there is a need to re-seed the key. If Condition 3 were to be interpreted in the absence of restraints on the number of prior draws, then at least 3-4 RN16s (48-64-bits) drawn from the RNG should be indistinguishable from pseudo-random.

2.2 The 16-bit Cyclic Redundancy Code

CRCs are error-detecting codes that check accidental (non-malicious) errors caused by faults during transmission. In EPCGen2, a CRC-16 is used to protect the information transmitted by both readers and tags. The CRC-16 algorithm maps arbitrary length inputs onto 16-bit outputs as follows: an n-bit input p is first replaced by a binary polynomial $p(x)$ of degree $n-1$, and then reduced modulo a specific polynomial $g(x)$ of degree 16 to a polynomial remainder $r(x) : p(x) = q(x)g(x) + r(x)$. The remainder has degree less than 16 and corresponds to a 16-bit number. For EPCGen2, the polynomial $g(x)$ is the irreducible polynomial: $x^{16} + x^{12} + x^5 + 1$ (over the finite field $GF(2)$ of two elements). CRC-16 will detect burst errors of 16-bits or less, any odd number of errors less than 16, and error patterns of length 2 [13].

CRCs by themselves are not suitable for protecting against intentional (malicious) alteration of data. They do not provide the one-wayness required by message digest codes: they are linear codes whose one-wayness is comparable to xor-sums.

3 Weaknesses of currently proposed EPCGen2 compliant RFID protocols

In this section we consider two recently proposed EPCGen2 compliant protocols: the Duc-Park-Lee-Kim protocol [10] and the Chien-Chen protocol [9]. The first protocol is designed to support untraceability and uncloneability; the second to support the same security features, but also to provide forward secrecy. We shall show that both protocols fall short of their claimed security.

In the protocols below we use the following notation: $\mathcal{S}$ is the backend server, $\mathcal{R}$ the reader, $\mathcal{T}$ the 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 The Duc-Park-Lee-Kim RFID protocol and its weaknesses

In this protocol [10] each tag $\mathcal{T}$ shares two values with the backend server $\mathcal{S}$: a 32-bit key and a 16-bit key. The tag stores these in its non-volatile memory and the server $\mathcal{S}$ stores them in a database DB. The 16-bit key is initialized $K \leftarrow K_{init}$ and updated with each successful authentication of the tag by the server. The 32-bit key is assigned the tag's EPC. The protocol has four passes.

1. $\mathcal{R} \rightarrow \mathcal{T}$: a query request.
2. $\mathcal{T} \rightarrow \mathcal{R} \rightarrow \mathcal{S}$: a random 16-bit nonce r, $M_1 = CRC(EPC||r) \oplus K$, and $C = CRC(r \oplus M_1)$.
   $\mathcal{S}$ checks C and that: $M_1 \oplus K = CRC(EPC||r)$ for some (K, EPC) in DB. If the checksum fails or if it cannot find a match then it rejects $\mathcal{T}$.
   $\mathcal{S}$ computes $M_2 = CRC(EPC||AP||r) \oplus K$ and updates the key K of $\mathcal{T}$ in DB: $K \leftarrow RNG(K)$.
3. $\mathcal{S} \rightarrow \mathcal{R}$: M_2, and details that identify the object to which the tag is attached, depending on the reader's privileges.
4. $\mathcal{R} \rightarrow \mathcal{T}$: M_2, "end session".
   $\mathcal{T}$ checks that: $M_2 \oplus K = CRC(EPC||AP||r)$. If this is valid then it updates the key K: $K \leftarrow RNG(K)$.

This protocol is subject to a synchronization attack as observed by Chien-Chen in [9]: if the adversary prevents the tag in Pass 4 from receiving an "end session" instruction, the tag will not update its key while the server will have updated the corresponding key in DB. Consequently the server will not be synchronized with the tag and any future attempts by the tag to get authenticated will fail. However there is another important weakness, caused by the linearity of CRC-16. The adversary can easily forge the response (r', M_1', C') of a tag $\mathcal{T}$ in any session by simply using an earlier response (r, M_1, C) obtained by interrogating $\mathcal{T}$ (as a rogue reader) or eavesdropping. Indeed let:

1. r' be a random 16-bit number.
2. $A = CRC(00||r \oplus r')$ and $B = CRC(A \oplus r \oplus r')$.
3. $M_1' = M_1 \oplus A = [CRC(EPC||r) \oplus K] \oplus CRC(00||r \oplus r')$
   $= CRC(EPC||r') \oplus K$.
4. $C' = C \oplus B = CRC(M_1 \oplus r) \oplus CRC(A \oplus r \oplus r') = CRC(M_1 \oplus A \oplus r')$
   $= CRC(M_1' \oplus r')$.

Clearly (r', M_1', C') is valid for *any* query request, and so the adversary will succeed in impersonating the tag $\mathcal{T}$.

3.2 The Chien-Chen RFID protocol and its weaknesses

This protocol [9] is an extension of the Duc-Park-Lee-Kim protocol, designed to address its weaknesses, as well as to offer forward secrecy. Each tag $\mathcal{T}$ stores three values in non volatile memory: a 32-bit EPC, a 16-bit key K and a 16-bit access key P. For each tag the backend server $\mathcal{S}$ stores six values in a database DB: the

tag's EPC, two 16-bit keys K_{old}, K_{new} , two 16-bit access keys P_{old}, P_{new} and $DATA$. The values of the key K and the access key P that the tag stores and the corresponding values of the keys K_{old}, K_{new} and P_{old}, P_{new} that the server stores are updated with each successful tag authentication so as to preserve synchronization. The protocol is described below.

1. $\mathcal{R} \rightarrow \mathcal{T}$: a random nonce N_1.
2. $\mathcal{T} \rightarrow \mathcal{R} \rightarrow \mathcal{S}$: a random nonce N_2, and $M_1 = CRC(EPC||N1||N2) \oplus K$.
   $\mathcal{S}$ checks that $M_1 = CRC(EPC||N_1||N_2) \oplus K_j$ for some (K_j, EPC), $j \in \{new, old\}$, in DB.
   If $\mathcal{S}$ cannot find a match then it rejects $\mathcal{T}$ and sends a "failure" message to $\mathcal{R}$.
   Else it updates the keys of $\mathcal{T}$: $K_{old} \leftarrow K_{new} \leftarrow RNG(K_{new})$, $P_{old} \leftarrow P_{new} \leftarrow RNG(P_{new})$ in DB, and computes $M_2 = CRC(EPC||N_2) \oplus P_j$, $j \in \{new, old\}$, using the j-value in M_1.
3. $\mathcal{S} \rightarrow \mathcal{R}$: M_2 and $DATA$, with product information.
4. $\mathcal{R} \rightarrow \mathcal{T}$: M_2.
   $\mathcal{T}$ checks that $M_2 = CRC(EPC||N_2) \oplus P$.
   If this is valid it updates its keys: $K \leftarrow RNG(K)$, $P \leftarrow RNG(P)$.

There are several weaknesses with this protocol. One weakness concerns the synchronization of keys in Pass 4: the protocol does not protect tags against repeated synchronization attacks. Indeed, the first time the adversary prevents the tag from getting its confirmation in Pass 4, the tag will not update (K, P), while the server will have updated the corresponding values of the tag in DB: $K_{old} \leftarrow K_{new}$, $K_{new} \leftarrow RNG(K_{new})$ and $P_{old} \leftarrow P_{new}$, $P_{new} \leftarrow RNG(P_{new})$. Then, if the value of (K, P) stored by the tag prior to the attack was (K_{new}, P_{new}), after the attack it will be (K_{old}, P_{old}). Suppose the attack is repeated. The server will accept the tag's response in Pass 2 because the tag uses the value (K_{old}, P_{old}) in DB. But this will be discarded by the server when it updates its keys. However the tag will not update its keys in Pass 4 if it is prevented from getting a confirmation M_2. It follows that the adversary will succeed in desynchronizing the tag after the second attempt. Desynchronization will also result from two successive reading failures by a tag.

This protocol shares the weakness of the Duc-Park-Lee-Kim protocol resulting from the linearity of CRC-16. In this case, the adversary can forge a response N_2', M_1' to any challenge N_1' of the server by using an earlier protocol flow $(N_1; N_2, M_1)$ obtained by interrogating the tag (as a rogue reader) or eavesdropping. Indeed, let:

1. N_2' be a random nonce.
2. $B_1 = N_1' \oplus N_1$ and $B_2 = N_2' \oplus N_2$.
3. $A = CRC(00||B_1||B_2)$.
4. $M_1' = M_1 \oplus A = [CRC(EPC||N_1||N_2) \oplus K] \oplus [CRC(00||B_1||B_2)]$
   $= CRC(EPC||N_1'||N_2') \oplus K$.

Then N_2', M_1' is a valid response to the challenge N_1'.

4 Secure EPCGen2 protocols

We next consider three "trivial" RFID authentication protocols (TRAPs) that comply with EPCGen2, and whose security is reduced to the minimum levels of statistical behavior of RNGs guaranteed by this standard.

The first protocol TRAP-0, is a toy example to illustrate that the RNG of EPCGen2 cannot be used to securely link protocol flows. We then show how to modify the RNG so that we get security. This will give us TRAP-1. The next protocol, TRAP-2, is an extension that provides anonymity, but not forward secrecy. The last protocol, TRAP-3, supports strong privacy (with forward secrecy).

4.1 A trivial RFID protocol that is EPCGen2 compliant

This protocol uses the 16-bit RNG supported by EPCGen2, seeded with a 16-bit key K. For each tag $\mathcal{T}$ with identifier $id(\mathcal{T})$, the server $\mathcal{S}$ stores in a database DB an entry of the form: $< id(\mathcal{T}), K >$, used to identify the tag. We assume that the server $\mathcal{S}$ and the reader $\mathcal{R}$ are linked by a secure (private and authenticated) channel.

TRAP-0

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N.
   $\mathcal{T}$ computes $L = K \oplus N$ and draws M from $RNG(L)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: $id(\mathcal{T})$, M.
   $\mathcal{S}$ computes $L' = K \oplus N$, where K is the key of $id(\mathcal{T})$, and draws M' from $RNG(L')$.
   If $M' = M$ then the tag $\mathcal{T}$ is authentic. Else it is rejected.
3. $\mathcal{S} \Rightarrow \mathcal{R}$: "end session".

4.2 Analysis of TRAP-0

The security of TRAP-0 is based on the statistical behavior of the RNG of EPCGen2 as specified by the three EPCGen2 constraints in Section 2.1. TRAP-0 has two major weaknesses that result from the fact that a 16-bit RNG is used as a security tool to link the challenge-response flows of a protocol instance. The first concerns *exhaustive-key* attacks: since RNGs are deterministic, an exhaustive search on all possible 2^{16} key (seed) values can be used to determine the key (alternatively, a pre-computed table of RNG entries can be used). One way to prevent such attacks is to use additional keying material (we shall do this in Section 4.4).

A second weakness concerns *related-key* attacks: EPCGen2 does not specify any protection of its RNGs against attacks in which the adversary exploits values drawn from RNGs whose keys are related. We next discuss these attacks, and consider an approach that may be used to deal with them.

The related-key problem

- *Search problem.* Given $RNG(K \oplus N_i)$, N_i, $i = 1, \ldots, t$ and $N \neq N_i$: find $RNG(K \oplus N)$.
- *Decision problem.* Given $RNG(K \oplus N_i)$, N_i, $i = 1, \ldots, t$ and $N, X \neq N_i$: is $X = RNG(K \oplus N)$?

Clearly if the adversary can compute $RNG(K \oplus N)$ for a fresh nonce N, given a history of $RNG(K \oplus N_i)$, N_i, $i = 1, \ldots, t$, obtained by eavesdropping on protocol flows, then the adversary can forge a session with challenge N. Although we have not as yet presented an anonymous TRAP (we shall do this in Section 4.5), if values drawn from a RNG are used as pseudonyms, then the adversary will be able to disambiguate tags if it can solve the related-key decision problem. The related-key problem described above is for passive attacks. In an active attack, the adversary can select the numbers N_i, $i = 1, \ldots, t$, adaptively (but not N or X). Protecting protocols against adaptive attacks is usually much harder.

There are several ways to deal with such attacks. The solution we consider here involves modifying the 16-bit EPCGen2 RNG so that it is hard to link its inputs and outputs.

4.3 Constructing a PRF family from a RNG

We briefly describe a construction, due to Goldreich-Goldwasser-Micali [14]. Let G be an n-bit RNG and K an n-bit number. Denote by $G_0(K)$ the first n-bit number output by G and $G_1(K)$ the next n-bit number. Let $X = X_1, X_2, \ldots, X_t$, $t \geq n$, be a t-bit number and $G_X(K) = G_{X_t}(Z_{t-1})$, where

$$Z_{t-1} = (G_{X_{t-1}}(\cdots(G_{X_1}(G_{X_0}(K)))\cdots).$$

Define the function $f_K : \{0,1\}^t \to \{0,1\}^n$ by $f_K(X) = G_X(K)$. It is shown in [14] that the family $F_n = \{f_K\}_{|K|=n}$ is a pseudo-random function (PRF).

In our TRAP protocols we shall use $f_K(X)$ as a RNG: since F_n is a PRF, $f_K(X)$ is secure against attacks that exploit correlated values of X. The first number drawn from $f_K(X)$ will be $G_X(K)$. If a second number has to be drawn then, in the last step of the construction above we take either the second or the third n-bit number output by $G(Z_{t-1})$, depending on whether $X_t = 0$ or $X_t = 1$; for the t-th draw, we take either the t-th output number or the $(t+1)$-th output number of $G(Z_{t-1})$.

4.4 TRAP-1: a trivial EPCGen2 RFID protocol

To deal with exhaustive key attacks and related-key attacks on TRAP-0 we use two 16-bit numbers K_0, K_1 as key, and evaluate $f_{K_0}(K_1 \oplus \cdot)$ on 16-bit number inputs. In particular, for the 16-bit number N, we draw numbers from $f_{K_0}(K_1 \oplus N)$ instead of $f_K(N)$. Observe that if N and $L = f_{K_0}(K_1 \oplus N)$ are given, then there are roughly 2^{16} pairs (K_0', K_1') for which $L = f_{K_0'}(K_1' \oplus N)$, each one being equally likely to be (K_0, K_1). The search range is narrowed as

more values (N_i, L_i) become available: to prevent attacks in which partial information about the seed may be leaked, the tag's key (K_0, K_1) may be updated during the execution of the protocol. We shall do this in our last TRAP protocol (Section 4.7). Finally note that since the value $f_{K_0}(K_1 \oplus N)$ is the output of an EPCGen2 RNG, it is subject to the minimal security requirements of EPCGen2 given in Section 2.

TRAP-1

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N.
   $\mathcal{T}$ computes $L = K_1 \oplus N$, and draws a 16-bit number M from $f_{K_0}(L)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: $id(\mathcal{T})$, M.
   $\mathcal{S}$ computes $L' = K_1 \oplus N$, where $K = K_0||K_1$ is the key of $id(\mathcal{T})$, and draws M' from $f_{K_0}(L')$.
   If $M' = M$ then the tag $\mathcal{T}$ is authentic. Else it is rejected.
3. $\mathcal{S} \Rightarrow \mathcal{R}$: Details of $\mathcal{T}$ (obtained from TID), "end session".

Security is based on the fact that $f_{K_0}(K_1 \oplus \cdot\,)$ is a PRF, since it is generated by a RNG. The numbers drawn from $f_{K_0}(K_1 \oplus N)$ are 16-bit numbers drawn from an EPCGen2 RNG, pseudo-randomly rearranged to thwart related-key attacks. These numbers are bound by the EPC constraints in Section 2.1. Consequently we have:

1. *Robustness against passive attacks.* Suppose that the adversary obtains the values of authenticators M of one or more tags for several sessions. Since these are drawn using independent seeds, the probability of predicting the next authenticator is bounded by the EPCGen2 probability of drawing a number from a RNG.
2. *Robustness against active attacks.* Since the tag's response is linked to the reader's challenge, a replay attack will fail. Furthermore, since the numbers M generated by a tag $\mathcal{T}$ are drawn from its RNG (with key K_0), by the next-number EPCGen2 prediction requirement, the adversary cannot predict their value with probability better than 0.025%.

The security of the TRAP protocols is discussed in a more formal setting in Section 4.6 and the Appendix.

4.5 TRAP-2: a trivial anonymous EPCGen2 RFID protocol

Our next protocol, TRAP-2, extends TRAP-1 to capture anonymity. For this protocol each tag $\mathcal{T}$ stores two numbers, a 16-bit pseudonym P and a 32-bit key $K = K_0||K_1$; the server $\mathcal{S}$ stores in a database DB for each tag an entry of the form: $< id(\mathcal{T}), K >$. Initially P and K are assigned random values.

TRAP-2

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N.
   $\mathcal{T}$ computes $L = (K_1 \oplus P)||N$ and draws two 16-bit numbers M, M_1 from $f_{K_0}(L)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: P, M.
   $\mathcal{T}$ updates $P \leftarrow M_1$.
   $\mathcal{S}$ computes $L' = (K_1 \oplus P)||N$ and draws M' from $f_{K_0}(L')$ for every key $K = K_0||K_1$ in DB. If there is a match $M' = M$ then tag $\mathcal{T}$ is authentic. Else it is rejected.
3. $\mathcal{S} \Rightarrow \mathcal{R}$: Details regarding tag $\mathcal{T}$, and "end session".

TRAP-2 is a simple extension of TRAP-1. Anonymity is assured because the pseudonyms P are pseudo-random, and because they are updated after each interrogation (by authorized or rogue readers). We shall discuss in more detail the security of this, and the other TRAP protocols, in the following section.

4.6 The security of EPCGen2 compliant protocols

EPCGen2 is a standard for Class 1 tags that have restricted memory (particularly non-volatile) and circuit footprint. It provides for high speed reading and sortation of tags. This standard focuses on reliability and efficiency and will only support a very basic level of security, provided by a 16-bit RNG. In our first two protocols, TRAP-1 and TRAP-2 we have used this RNG to secure protocol flows: we added an extra 16-bits of keying material to make it harder to invert the RNG and to prevent related-key attacks. Obviously this will affect the efficiency of the interrogating process: identification will take longer and may lead to reading failures. Also the additional non-volatile memory means that the TRAP protocols can only be used with tags that are at the top end of the EPCGen2 standard.

The EPC Air Interface Protocol [13] covers a wide range of tag types. The Class 1 tags we shall consider in this paper are the most basic passive tags. Class 2 tags are covered by the same specifications, but are allowed to have additional memory. In the last part of this section we consider TRAP-3, a Class 2 mutual authentication RFID protocol that supports strong privacy (with forward secrecy). For this protocol we require that each tag has a 48-bit key, and use a 32-bit RNG. We shall assume the same minimum security level of EPCGen2 as specified in Section 2.1, extended to allow for 32-bit RNGs.

Our TRAP protocols are based on the protocols O-TRAP [5] and O-FRAP [24]) that are secure in the Universal Composability (UC) framework [6–8]. The main feature of these protocols is that their protocol flows are pseudo-random. In the Appendix we describe O-TRAP and show how to adapt it to get a security proof for TRAP-2 in the UC framework, provided a sufficiently large seed for the RNG is used, so as to prevent the adversary from distinguishing its output from random numbers. Similarly, the security proof of O-FRAP can be adapted to get a proof for TRAP-3.

4.7 An EPC Class2 Gen2 compliant protocol that supports strong privacy

Our last protocol is a mutual authentication protocol. This protocol uses a 32-bit RNG and 48-bit keys, to support security. For reliability the transmitted messages are (only) 16-bit numbers. Each tag $\mathcal{T}$ stores a 16-bit number P and a 48-bit key $K = K_0||K_1$, where K_0 is 32-bits long. The server $\mathcal{S}$ stores for each tag $\mathcal{T}$ in a database DB an entry with two 48-bit keys K^{old}, K^{cur}, that is:

$$< id(\mathcal{T}), K^{old}, K^{cur} > .$$

Initially P is assigned a random value, K^{cur}, K are assigned the same random value, and K^{old} is null.

TRAP-3

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N.
   $\mathcal{T}$ computes $L = (K_1 \oplus P)||N$, draws three 32-bit numbers M_1, M_2, M_3 from $f_{K_0}(L)$, parses $M_1 = M_{10}||M_{11}$ and $M_3 = M_{30}||M_{31}$ into 16-bit numbers, and sets $M \leftarrow M_{10}$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: P, M.
   $\mathcal{T}$ updates $P \leftarrow M_{11}$.
   $\mathcal{S}$ computes $L' = (K_1 \oplus P)||N$ and draws M'_1, M'_2, M'_3 from $f_{K_0}(L')$, for every K^j, $j \in \{old, cur\}$ in DB, parses $M'_1 = M'_{10}||M'_{11}$ and $M'_2 = M'_{20}||M'_{21}$, and sets $Q \leftarrow M'_{20}$.
   If for some K^j in DB there is a match $M'_{10} = M$, then $\mathcal{T}$ is authentic: for $j = cur$ it updates $K_1^{old} \leftarrow K^{cur}$, $K_1^{cur} \leftarrow M'_{21}||M'_3$; for $j = old$ it updates $K^{cur} \leftarrow M'_{21}||M'_3$.
   Else $\mathcal{T}$ is rejected.
3. $\mathcal{S} \Rightarrow \mathcal{R}$: Q, details regarding the tag, "end session".
4. $\mathcal{R} \rightarrow \mathcal{T}$: Q.
   $\mathcal{T}$ checks that $Q = M_{20}$; if so, it updates $K \leftarrow M_{21}||M_3$.

The security of TRAP-3. TRAP-3 extends TRAP-2 to capture forward secrecy. To prove forward secrecy we need to show that the adversary cannot (a) desynchronize the updating process of the key K of a tag $\mathcal{T}$ and, (b) link a tag whose key is compromised to earlier protocol flows (obtained by eavesdropping).

For (a) there are two cases to consider. If the adversary is passive then the value of the key K of $\mathcal{T}$ is the same as the value K^{cur} stored in DB for $\mathcal{T}$. If the adversary is active and prevents $\mathcal{T}$ from receiving the confirmation Q, or if a rogue reader interrogates $\mathcal{T}$, then the value of K is K^{old}—even if such attacks are repeated. For (b) observe that if a tag's key K is compromised then the earlier flows $[N; P, M; Q]$ cannot be linked because they are pseudo-random.

Observe that this protocol is optimistic [24]: if the server stores the values P for each tag $\mathcal{T}$, then these can be used to disambiguate the tag $\mathcal{T}$ when the adversary is passive (an eavesdropper) or inactive. Finally, as with the protocols O-TRAP and O-FRAP [5, 24], a compromised tag can be traced back to its last

completed interrogation with an authorized reader, since it will not update its key K until it receives a (valid) confirmation Q.

5 Adding a kill functionality

Our TRAP protocols can be extended to include a kill feature for tags. In this section we show how this is done for TRAP-2. Observe that to disable a particular tag, the server must first authenticate that tag.

5.1 TRAP-2*: a TRAP with kill functionality

EPCGen2 specifies four states that tags may implement: *open*, *ready*, *secure* and *killed*. *Open* is the initial state; ready is a holding state until the tag receives the next message from the reader; secure is a state into which a tag transitions on receiving an authenticator from the reader; *killed* is a state into which a *secured* tag transitions on receiving a kill mandate. A *killed* tag will not respond to any challenge, and cannot be resurrected.

In TRAP-2* each tag $\mathcal{T}$ stores a 16-bit number P and a 32-bit key $K = K_0||K_1$, as in TRAP-2, but also stores an additional 32-bit *kill-key* KP. The server $\mathcal{S}$ stores in a database DB, for each tag $\mathcal{T}$, an entry of the form:

$$< id(\mathcal{T}), K, KP, EPC, state\_tag, kill\_mandate >,$$

where *state_tag* specifies the state of the tag and *kill_mandate* is either *null* or *kill*. Initially K and P are assigned random values, *state_tag* is *open* and *kill_mandate* is *null*.

*TRAP-2**

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N.
   $\mathcal{T}$ updates its state to *ready*, computes $L = (K_1 \oplus P)||N$ and draws two 16-bit numbers M, M_1 from $f_{K_0}(L)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: P, M.
   $\mathcal{T}$ updates $P \leftarrow M_1$.
   $\mathcal{S}$ computes $L' = (K_1 \oplus P)||N$ and draws two numbers M', M'_1 from $f_{K_0}(L')$ for every key K in DB.
   If there is a match $M' = M$ then $\mathcal{T}$ is authentic: $\mathcal{S}$ updates $\mathcal{T}$'s *state_tag* in DB to *secure*. Else the tag is rejected.
   If the value of *kill_mandate* of a *secure* tag $\mathcal{T}$ is *kill*, then the server $\mathcal{S}$ parses $KP = KP_0||KP_1$ into 16-bit numbers and draws a 16-bit number Q from $f_{KP_0}((KP_1 \oplus P)||N)$.
3. If the value of *kill_mandate* of a *secure* tag $\mathcal{T}$ in DB is *null* then:
   $\mathcal{S} \Rightarrow \mathcal{R}$: Details regarding the tag, and "end session".
4. If the value of *kill_mandate* of a *secure* tag $\mathcal{T}$ in DB is *kill* then:
   $\mathcal{S} \Rightarrow \mathcal{R}$: Q, and "end session".
   $\mathcal{R} \rightarrow \mathcal{T}$: Q.

$\mathcal{T}$: If the value of *state_tag* is *ready* then $\mathcal{T}$ parses $KP = KP_0||KP_1$ and draws a number Q' from $f_{KP_0}((KP_1 \oplus P)||N)$.
If $Q' = Q$ then $\mathcal{T}$ transitions to a *killed* state.

5.2 Security analysis of TRAP-2*

We only discuss the security of the kill functionality. As noted earlier, non-corrupted tags that receive a kill mandate will always assume a *killed* state. However a tag may be prevented from receiving the kill mandate Q by the adversary. In such cases even though the value of *state_tag* in the database DB is *killed*, the tag is not *killed*. However the tag is, for all practical purposes, disabled: each time it attempts to get identified by an authorized reader, it will only receive a kill mandate.

Concluding remarks

The EPCGen2 standard for Class 1 tags focuses on reliability and efficiency and supports only a very basic security level. Designing EPCGen2 compliant RFID protocols that are secure is particularly challenging because the only security tool that is available in this standard is a 16-bit RNG.

In this paper we have shown that two recently proposed EPCGen2 compliant RFID protocols fail to provide adequate security and are subject to impersonation attacks and synchronization attacks. We proposed two basic RFID authentication protocols, TRAP-1 and TRAP-2 that are EPCGen2 compliant, whose security is reduced to the minimal security levels supported by this standard, and have shown how to add a kill functionality. Finally we proposed a mutual authentication RFID protocol that provides strong anonymity and that complies with the EPC Class2 Gen2 standard.

Acknowledgement

The authors thank the anonymous reviewers for helpful comments and suggestions.

References

1. Ateniese, G., Camenisch, J., and de Medeiros, B. Untraceable RFID tags via insubvertible encryption. In *Proc. ACM Conf. on Computer and Communication Security (ACM CCS 2005)* (2005), ACM Press, 92–101.
2. Avoine, G. http://lasecwww.epfl.ch/ gavoine/rfid/.
3. 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.

4. Burmester, M., de Medeiros, B., and Motta, R. Robust, Anonymous RFID Authentication with Constant Key-Lookup. In *Proc. ACM Symposium on Information, Computer and Communication Security (ASIACCS'08),* (2008), ACM Press, 283–291.
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. *Studies in Secure Multiparty Computation and Application.* PhD thesis, Weizmann Institute of Science, Rehovot 76100, Israel, June 1995.
7. Canetti, R. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology 13:1* (2000), 143–202.
8. 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, 136–145.
9. Chien, H.-Y., and Chen, C.-H. Mutual authentication protocol for rfid conforming to EPC class 1 generation 2 standards. *Comput. Stand. Interfaces 29*, 2 (2007), 254–259.
10. Dang Nguyen Duc, Jaemin Park, H. L., and Kim, K. In *Enhancing Security of EPCglobal Gen-2 RFID Tag against Traceability and Cloning* (2006), Symposium on Cryptography and Information Security, SCIS 2006.
11. 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.
12. 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.
13. EPC Global. EPC Tag Data Standards, vs. 1.3. http://www.epcglobalinc.org/standards/EPCglobal_Tag_Data_Standard_TDS_Version_1.3.pdf
14. Goldreich, O., Goldwasser, S., and Micali, S. How to construct pseudorandom functions. *Journal of the ACM 33*, 4 (1986).
15. Henrici, D., and Müller, P. M. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications* (2004), 149–153.
16. ISO/IEC. Standard # 18000 – RFID Air Interface Standard. http://www.hightechaid.com/standards/18000.htm
17. 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, 149–164.
18. Menezes, A., van Oorschot, P., and Vanstone, S. *Handbook of Applied Cryptography.* CRC Press, 1996.
19. 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.
20. Ohkubo, M., Suzuki, K., and Kinoshita, S. Cryptographic approach to "privacy-friendly" tags. In *Proc. RFID Privacy Workshop* (2003).
21. Sharma, S. E., Weiss, S. A., and Engels, D. W. RFID systems and security and privacy implications. In *Proc. of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 20002)* (2003), vol. 2523 of *LNCS*, Springer, 454–469.

22. Stinson, D. *Cryptography Theory and Practice, Second Edition.* CRC Press, Inc., Boca Raton, 2002.
23. Tsudik, G. YA-TRAP: Yet another trivial RFID authentication protocol. In *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications (PerCom 2006)* (2006), IEEE Press.
24. 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, 242–252.
25. 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 (SPC 2003)* (2003), vol. 2802 of *LNCS*, Springer, 454–469.

Appendix: The security of TRAP-2

We first describe the protocol O-TRAP [5] and show the modifications needed to get TRAP-2. We then state our main result and briefly discuss the security framework.

In O-TRAP, each tag $\mathcal{T}$ stores two n-bit numbers: a pseudonym P and a key K, where n is the security parameter. The server $\mathcal{S}$ has two databases, DB and DB': for each tag $\mathcal{T}$, the server $\mathcal{S}$ stores in DB an entry of the form $< id(\mathcal{T}), K >$, and in DB' and entry of the form $< id(\mathcal{T}), P, K >$. DB is indexed by the (authorized) keys K while DB' is indexed by the pseudonyms P. Initially P and K are assigned random values. Let $H_K(\cdot)$ be a hash function with pseudorandom values.

O-TRAP

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: An n-bit random number N.
   $\mathcal{T}$ computes $M = H_K(N||P)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: P, M.
   $\mathcal{T}$ updates $P \leftarrow M$.
   $\mathcal{S}$ accepts the tag $\mathcal{T}$ as authentic if:
   either there exists an entry $(id(\mathcal{T}), P, K) \in DB'$ with $M = H_K(R_{sys}||P)$,
   or there exists an entry $(id(\mathcal{T}), K) \in DB$ with $M = H_K(R_{sys}||P)$.
   Else the tag is rejected.

We have:

Theorem 1. [5] *O-TRAP guarantees availability, anonymity, and authentication in the Universal Composability (UC) framework* [6–8] *provided the keyed hash function is chosen from a pseudo-random function family* $\{H_K(\cdot)\}_{|K|=n}$.

A key feature of this protocol is that it is *optimistic*, that is its security overhead is minimal when the adversary is passive (an eavesdropper) or inactive, since in this case the server $\mathcal{S}$ needs to do only one key-lookup in DB' to find the pseudonym of $\mathcal{T}$ and then authenticate the tag (we have constant key-lookup [4]). TRAP-2 is not optimistic, however in all other respects is very

similar to O-TRAP. For both protocols the pseudonyms P and authenticators M are pseudo-random. In TRAP-2, M is drawn from $f_{K_0}(K_1 \oplus P||N)$, whereas in O-TRAP it is $H_K(P||N)$. Consequently M is a pseudo-random number determined by: the key K, the pseudonym P, and the challenge P. It follows that one can use the same steps as in the security proof for O-TRAP to get a proof for TRAP-2. We need however to make certain that $f_{K_0}(K_1 \oplus \cdot||\cdot)$ is a PRF, which in our case is guaranteed if the length of the seed is sufficiently long to prevent the adversary from distinguishing its output from random numbers.

The UC-framework. UC security is based on notions of interactive indistinguishability of real from ideal protocol executions. This requires:

1. A mathematical model of real protocol executions, where honest parties are represented by probabilistic polynomial-time Turing machines that correctly execute the protocol as specified, and adversarial parties that can deviate from the protocol in an arbitrary way. The adversarial parties are controlled by a single (PPT) adversary $\mathcal{A}$ that (1) has full knowledge of the state of adversarial parties, (2) can arbitrarily schedule the actions of all parties, both honest and adversarial, and (3) interacts with the environment in arbitrary ways, in particular can eavesdrop on all communications.
2. An idealized model of protocol executions, where the security properties are defined in terms of an *ideal functionality* $\mathcal{F}$, a trusted party that all parties may invoke to guarantee correct execution of particular protocol steps. The ideal-world adversary $\mathcal{S}_\mathcal{A}$ is controlled by the ideal functionality, to reproduce as faithfully as possible the behavior of the real adversary.
3. A proof that no environment can distinguish (with better than negligible probability) real- from ideal-world protocol runs by observing the system behavior.

In the real world the adversary $\mathcal{A}$ interacts with the protocol parties using commands such as REFRESH, START, SEND and END, to cause the beginning of a new server interrogation period, to make available a tag for interrogation, to send a challenge to a tag and get its response, or to send a response to the server, etc. The goal of $\mathcal{A}$ is to prevent an honest tag from (i) getting accepted by the server (availability), (ii) getting authenticated, and (iii) being anonymous. The ideal adversary $S_\mathcal{A}$ has the same goals, but this time the interactions are with the ideal-world functionality $\mathcal{F}$. We get UC-security if no PPT environment $\mathcal{Z}$ can distinguish real- from ideal-world simulations. For more details the reader is refered to [5, 24].