Providing Stronger Authentication at a Low Cost to RFID Tags Operating under the EPCglobal Framework

Pedro Peris-Lopez,
pperis@inf.uc3m.es

Tong-Lee Lim
tllim@i2r.a-star.edu.sg

Tieyan Li
litieyan@i2r.a-star.edu.sg

Abstract

In 2006, EPCglobal and the International Organization for Standards (ISO) ratified the EPC Class-1 Generation-2 (Gen-2) [1] standard and the ISO 18000 standard [2] respectively. These efforts represented major advancements in the direction of universal standardization for low-cost RFID tags. However, a cause for concern is that security issues do not seem to be properly addressed in these standards. In this paper, we propose a new lightweight RFID tag-reader mutual authentication scheme for use under the EPCglobal framework. The scheme is based on previous work by Konidala and Kim [3]. We attempt to mitigate the weaknesses observed in the original scheme, and at the same time, consider other possible adversarial threats, as well as constraints on low-cost RFID tags requirements.

1. Introduction

In this paper, we focus on designing a secure authentication scheme for use under the EPCglobal framework. A number of works, such as [3], [4] and [5], have proposed protocols to enhance the security of the EPC Class-1 Gen-2 standard. Unfortunately, due to weaknesses that have been exposed against them, these protocols fall short of meeting the desired security objectives. From these bad experiences, it seems like enforcing authentication under the EPC Class-1 Gen-2 standard specifications is an almost impossible task and any proposed scheme based on the standard looks doomed to fail. In fact, the proposed infrastructure seems to be too weak to support any real security. However, we contend that by making some small enhancements and without the need to revamp the entire set of specifications, it is still possible to design a reasonably secure authentication scheme for use on low-cost RFID tags.

A tag-reader mutual authentication scheme that uses a specially designed $MixBits$ function is presented in this work. The underlying protocol is similar to that proposed by Konidala and Kim in [3] (we shall refer to their scheme as the tag-reader mutual authentication or **TRMA** scheme), with its observed weaknesses addressed by introducing the $MixBits$ function. Under some rigorous analysis, we show that $MixBits$ increases the security of the scheme by providing stronger resistance against attacks. Furthermore, we show that $MixBits$ requires only a small amount of circuit area, memory size, and power consumption and can be feasibly implemented on low-cost RFID tags.

2. The TRMA Schemes

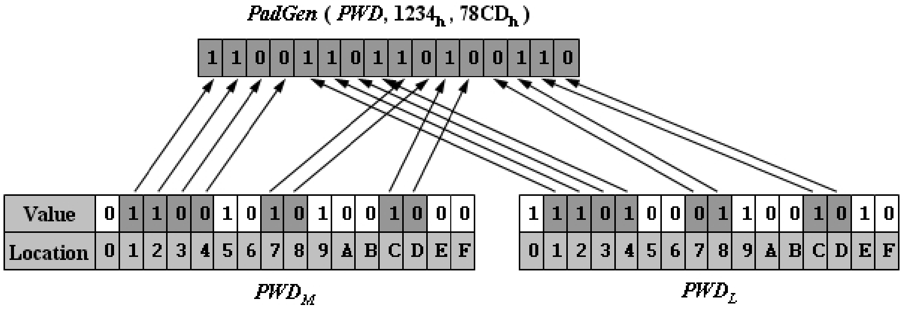
In this section, we briefly describe the original TRMA scheme, as well as the extended TRMA scheme by Konidala *et al.*, and describe the weaknesses observed in them.

2.1. The Original TRMA Scheme

In [3], Konidala and Kim presented a lightweight tag-reader mutual authentication (TRMA) scheme that uses some of the features in a EPC Class-1 Gen-2 tag, as well as a specially designed pad generation function $PadGen$. The $PadGen$ function is used to produce a cover-coding pad to mask the tag's access password before transmission. The function is performed on the tag's 32-bit access password PWD, which is broken up into 2 parts – PWD_M (comprising the 16 most significant bits) and PWD_L (comprising the 16 least significant bits). $PadGen$ takes two 16-bit random numbers R_i^{Tag} (generated by the tag) and R_i^{Rdr} (generated by the reader) as its inputs. Using each of the four hexadecimal digits in R_i^{Tag} (or R_i^{Rdr}) to indicate a bit address within PWD_M or PWD_L, $PadGen$ then selects those bits from PWD_M and PWD_L to form the 16-bit output pad. An example of a pad computation under $PadGen$ is shown in Fig. 1. Under the scheme, each cover-coding pad PAD_i (for $i = 1, 2, 3, 4$) can be expressed as

$$PAD_i \quad = \quad PadGen(PWD,\, R_i^{Tag},\, R_i^{Rdr}) \qquad (1)$$

978-0-7695-3492-3/08 $25.00 © 2008 IEEE
DOI 10.1109/EUC.2008.82



1	1	0	0	1	1	0	1	1	0	1	0	0	1	1	0

Value	0	1	1	0	0	1	0	1	0	1	0	0	1	0	0	0
Location	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

1	1	1	0	1	0	0	0	1	1	0	0	1	0	1	0
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Figure 1. Computing $PadGen(PWD$, 0x1234, 0x78CD$)$ on PWD = 0x6548 E8CA. (0x6548 = $0110\ 0101\ 0100\ 1000_2$ and 0xE8CA = $1110\ 1000\ 1100\ 1010_2$.)

and the authentication responses (otherwise known as the cover-coded passwords in [3] nad [7]) can be expressed as

$$\begin{aligned} CCP_1 &= PWD_M \oplus PAD_1 && (2)\\ CCP_2 &= PWD_L \oplus PAD_2 && (3)\\ CCP_3 &= PWD_M \oplus PAD_3 && (4)\\ CCP_4 &= PWD_L \oplus PAD_4 && (5) \end{aligned}$$

Fig. 2 depicts a single run of the authentication protocol.

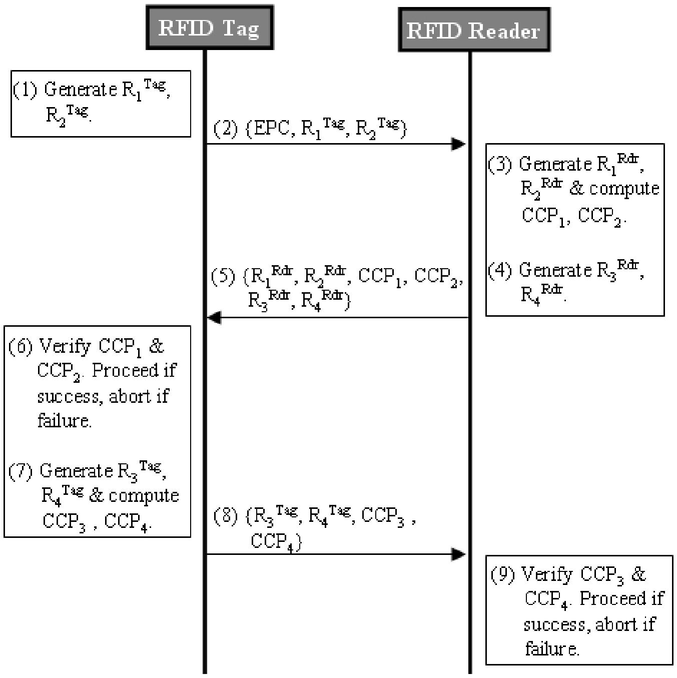

RFID Tag
RFID Reader
(1) Generate R1Tag, R2Tag.
(2) (EPC, R1Tag, R2Tag)
(3) Generate R1Rdr, R2Rdr & compute CCP1, CCP2.
(4) Generate R3Rdr, R4Rdr.
(5) (R1Rdr, R2Rdr, CCP1, CCP2, R3Rdr, R4Rdr)
(6) Verify CCP1 & CCP2. Proceed if success, abort if failure.
(7) Generate R3Tag, R4Tag & compute CCP3 , CCP4.
(8) (R3Tag, R4Tag, CCP3 , CCP4)
(9) Verify CCP3 & CCP4. Proceed if success, abort if failure.

Figure 2. The authentication protocol under the TRMA scheme.

In [6], Lim and Li uncovered two effective attacks against the TRMA scheme, which arises due to the following weaknesses:

- The pseudo-random number inputs to the $PadGen$ function (R_i^{Tag}, R_i^{Rdr} for $i = 1, 2, 3, 4$) are known to an eavesdropper. This allows the eavesdropper to identify which bits of the access password are selected to form the output pad.
- Each bit in every authentication response (CCP_i for $i = 1, 2, 3, 4$) is obtained by xoring bits in the access password, where the location of the bits are known to an eavesdropper. For example: (we use $[A]_b$ to denote the value of bit b in A and $\langle A\rangle_x$ to denote the value of hexadecimal digit x in A)

  $$\begin{aligned} [CCP_1]_1 &= [PWD_M]_1 \oplus [PWD_M]_{\langle R_1^{Tag}\rangle_1}\\ [CCP_1]_4 &= [PWD_M]_4 \oplus [PWD_L]_{\langle R_1^{Tag}\rangle_0} \end{aligned}$$

  Since each CCP_i is sent in clear, this allows the eavesdropper to correlate the bits in the access password.
- Each hexadecimal digit in R_i^{Tag} and R_i^{Rdr} (where $i = 1, 2, 3, 4$) is always mapped to the same bits in CCP_i for a given access password. Hence, the adversary can collect a dictionary of valid authentication responses corresponding to each eavesdropped value for R_i^{Tag} and R_i^{Rdr}.

In the first attack, Lim and Li showed how a passive attacker can perform correlations over the eavesdropped information to recover the access password from only a single eavesdropped authentication session. In the second attack, they showed how a passive attacker can record the random challenges and their corresponding responses eavesdropped from multiple authentication sessions to form a dictionary.

2.2. The Extended TRMA Scheme

In [7], Konidala, Kim and Kim presented an extended version of TRMA, which uses both the 32-bit access password and the 32-bit kill password. The extended TRMA scheme uses two rounds of $PadGen$ (instead of a single round in the original TRMA scheme), one nested within the other, to compute each cover-coding pad. The inner round performs $PadGen$ over the access password, while the outer round performs $PadGen$ over the kill password. Instead of (1), the resulting pad would then be expressed as

$$\begin{aligned} PAD_i &= PadGen(KWD, \\ &\quad PadGen(PWD, R_i^{Tag}, R_i^{Rdr}),\ R_i^{Tag}) \end{aligned} \tag{6}$$

where KWD denotes the kill password.

The extended TRMA scheme offers greater resistance against Lim and Li's attacks. It is much more difficult for an adversary to recover the access password under the correlation attack, or to forge a successful authentication under the dictionary attack. However, Peris-Lopez et al. showed how the access password can be disclosed under the assumption of an active attacker [8]. In addition, Lim et al. exposed that the correlation attack can be used against this scheme in a different way – to recover the kill password after eavesdropping over multiple authentication sessions [9].

3. M^3 Authentication Protocol (M^{3}AP)

In this section, we introduce a new lightweight authentication scheme, known as M^{3}AP, to strengthen the security of the EPC Class-1 Gen-2 standard. We design M^{3}AP by extending Konidala and Kim's scheme [3], and make use of a $MixBits$ function to mitigate the security weaknesses found in the original scheme.

3.1. Objectives

With M^{3}AP, we emphasize that the main objective is to design a lightweight authentication protocol that provides mutual authentication between an RFID tag and an RFID reader under the EPCglobal framework, and privacy is not a main focus. Hence, as in the authentication protocol specified under the EPC Class-1 Gen-2 standard and the previously proposed TRMA protocols, we do not make provisions to enforce privacy by protecting the unique EPC but instead, allow the EPC of RFID tags to be transmitted in clear. Inevitably, this poses a problem to application environments whereby the privacy of tags and/or tag users is essential. In such cases, it would be pertinent to include measures for privacy protection. While our current scheme does not enforce privacy, we contend that it would be possible to extend the scheme to provide the necessary protection although this would require additional considerations.

3.2. The Protocol

In this section, we present an improved version of Konidala and Kim's TRMA scheme that seeks to mitigate its security weaknesses. The proposed protocol was designed by taking into account tag restrictions (computational, storage and circuitry) and with minimal modifications to the general framework of the EPC Class-1 Gen-2 specification. The protocol is described as follows:

Assumptions: We assume that the tag is singulated using a probabilistic (i.e. Aloha-based protocol) or deterministic (i.e. binary tree-walking protocol) collision avoidance protocol. At the end of each singulation, a tag is selected to communicate with the reader.

$$\begin{aligned} (1)\ Tag \rightarrow Reader \quad &: \quad EPC,\ R_1^{Tag},\ R_2^{Tag}, \\ &\quad\ R_3^{Tag},\ R_4^{Tag} \end{aligned}$$

The tag backscatters its EPC number. Then, the reader sends the command Req_RN to the tag over four times. Each time, the tag backscatters a new random number (R_i^{Tag} for $i = 1, 2, 3, 4$) and stores it into its memory. These are used as random challenges to the reader. Upon receiving the EPC, the reader uses it to perform an index search to retrieve the access password PWD associated with the tag from the backend database. Once PWD is obtained, the reader will then go on to compute the authentication responses.

$$\begin{aligned} (2)\ Reader \rightarrow Tag \quad &: \quad CCP_1,\ CCP_2,\ R_1^{Rdr}, \\ &\quad\ R_2^{Rdr},\ R_3^{Rdr},\ R_4^{Rdr} \end{aligned}$$

The reader transmits its computed responses, as well as a set of random numbers as authentication challenges to the tag. To obtain the responses CCP_1 and CCP_2, it first computes an intermediate 32-bit vector PWD' from PWD and the received R_i^{Tag}'s using our proposed $MixBits$ function:

$$\begin{aligned} PWD' &= MixBits(PWD \oplus (R_1^{Tag} \,||\, R_2^{Tag}), \\ &\quad R_3^{Tag} \,||\, R_4^{Tag}) \end{aligned} \tag{7}$$

The reader then computes the authentication responses CCP_1 and CCP_2 as follows:

$$\begin{aligned} CCP_1 &= PWD_M \oplus PadGen(\ PWD', \\ &\quad R_1^{Tag} \oplus PWD'_L,\ R_1^{Tag} \oplus PWD'_M) \\ CCP_2 &= PWD_L \oplus PadGen(\ PWD', \\ &\quad R_2^{Tag} \oplus PWD'_L,\ R_2^{Tag} \oplus PWD'_M) \end{aligned}$$

Instead of applying $PadGen$ to the static access password, we apply $PadGen$ on a vector computed from the access password. This vector changes as the random challenges vary. Furthermore, both CCP_1 and CCP_2 depend only on the random challenges generated by the tag. In the original and extended TRMA schemes, CCP_1 and CCP_2 would partially depend on pseudo-random numbers generated by the reader, which presents an avenue for a malicious reader to exploit and reduces the reliability of the responses computed. After computing the responses, the reader then generates four new random numbers (R_i^{Rdr} for $i = 1, 2, 3, 4$) and present them as challenges to the tag. The reader also stores the random challenges, which will be used to verify the tag responses.

(3) Tag : Verify CCP_1 and CCP_2 .

The tag receives CCP_1 and CCP_2. The access password and the random numbers used in the computation of CCP_1 and CCP_2 are already stored in its memory. Therefore, it has the necessary information to compute CCP'_1 and CCP'_2 in the same way that the reader computed CCP_1 and CCP_2. The tag then compares these values with the values sent by the reader:

1. If $CCP_1 = CCP'_1$ and $CCP_2 = CCP'_2$, then verification is successful. The tag considers the reader to be an authorized entity.
2. Otherwise, verification fails. The tag ends its communication with the reader and returns to arbitrate state.

(4) $Tag \rightarrow Reader$: CCP_3, CCP_4

To authenticate itself, the tag needs to reply to the reader with CCP_3 and CCP_4, which are computed by taking steps similar to those taken by the reader. The tag first computes an intermediate 32-bit vector from its access password:

$$\begin{aligned} PWD'' &= MixBits(PWD' \oplus (R_1^{Rdr} \mid\mid R_2^{Rdr}), \\ &\quad R_3^{Rdr} \mid\mid R_4^{Rdr}) \end{aligned} \tag{8}$$

Thereafter, the tag computes CCP_3 and CCP_4 as follows (note that $PadGen$ is now computed over the new intermediate vector PWD''):

$$\begin{aligned} CCP_3 &= PWD_M \oplus PadGen(PWD'', \\ &\quad R_1^{Rdr} \oplus PWD''_L,\ R_1^{Rdr} \oplus PWD''_M) \\ CCP_4 &= PWD_L \oplus PadGen(PWD'', \\ &\quad R_2^{Rdr} \oplus PWD''_L,\ R_2^{Rdr} \oplus PWD''_M) \end{aligned}$$

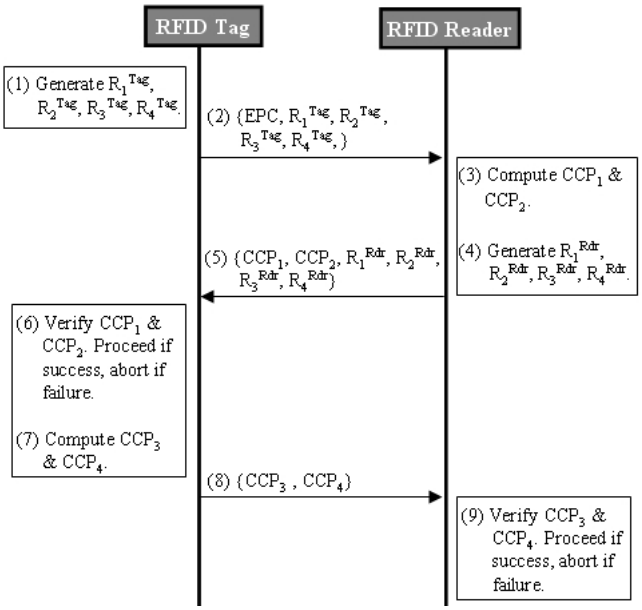
The tag then sends the two authentication responses (CCP_3, CCP_4) to the reader.

(5) $Reader$: Verify CCP_3 and CCP_4 .

The reader receives the responses CCP_3 and CCP_4 from the tag, computes CCP'_3 and CCP'_4 based on the information known to it, and then compares the received values with the computed values:

1. If $CCP_3 = CCP'_3$ and $CCP_4 = CCP'_4$, then verification is successful. The reader considers the tag as an authentic (or genuine) tag.
2. Otherwise, verification fails. The reader will emit an alarm to the back-end database to indicate this event (perhaps to inform the database that a fake tag or a counterfeit product is detected).

Fig. 3 depicts a single run of the authentication protocol.



RFID Tag
RFID Reader
(1) Generate R_1^{Tag}, R_2^{Tag}, R_3^{Tag}, R_4^{Tag}.
(2) {EPC, R_1^{Tag}, R_2^{Tag}, R_3^{Tag}, R_4^{Tag}, }
(3) Compute CCP$_1$ & CCP$_2$.
(4) Generate R_1^{Rdr}, R_2^{Rdr}, R_3^{Rdr}, R_4^{Rdr}.
(5) {CCP$_1$, CCP$_2$, R_1^{Rdr}, R_2^{Rdr}, R_3^{Rdr}, R_4^{Rdr}}
(6) Verify CCP$_1$ & CCP$_2$. Proceed if success, abort if failure.
(7) Compute CCP$_3$ & CCP$_4$.
(8) {CCP$_3$, CCP$_4$}
(9) Verify CCP$_3$ & CCP$_4$. Proceed if success, abort if failure.

Figure 3. The authentication protocol under the M^{3}AP protocol.

4. The $MixBits$ Function

RFID tags (e.g. EPCGlobal Class I or Class II tags) are devices with severe limitations (in terms of computational, storage and circuitry requirements). Due to these hard restrictions, the use of standard cryptographic primitives lie beyond their capabilities. Hence, the design of a secure lightweight $MixBits$ function for our M^{3}AP protocol becomes a thought-provoking challenge. We contend that the basic requirements of this function should be as follows:

1. Only efficient operations that are easily implemented in hardware should be used. For example, rotations may be included, but multiplications should be excluded due to its high cost.
2. A highly non-linear function that provides a negligible relationship between the inputs and the outputs, should be used.

3. Temporal requirements will limit the number of operations a tag may compute. The read speed of a tag conforming to Gen-2 is about twice that of Gen-1, with average read rates of around 450 tags per second.

We obtained possible candidates for $MixBits$ through evolving compositions of extremely light operands by means of genetic programming. (We refer the interested reader to [11] where a detailed description of the methodology used to obtain our function is explained.) Several experiments were then conducted on the candidates to pick a highly non-linear function. At the end of the experiments, the following function was selected for $MixBits$:

```
Z = MixBits(X,Y)
----------------------------
Z = X;
for(i=0; i<32; i++) {
   Z = (Z<<1) + ((Z+Y)>>1);}
----------------------------
```

where addition is carried out modulo 2^{32}, $<<$ denotes bitwise left shift and $>>$ denotes bitwise right shift.

5. Analysis of the $MixBits$ Function

Linear cryptanalysis, commonly used for block cipher cryptanalysis, was employed to study how the output of this function can be approximated by a linear function. In order to obtain a linear bias, the following experiment was carried out: two 32-bit masks $(A,\ B)$ were randomly picked, and two consecutive outputs $(Z_i,\ Z_{i+1})$ were generated from them. With these two masks, the equality $A * Z_i = B * Z_{i+1}$ is evaluated ($*$ denotes scalar product, with a mod 2 operation carried out after the addition). This process is repeated 2^n times, from which we obtain the number of successes (m). The bias is then defined as:

$$BIAS = \frac{1}{2^{-log_2(|\frac{m}{2^n}-\frac{1}{2}|)}} \tag{9}$$

Several pairs of different masks A and B, were randomly picked and tested. For each mask pair, 2^{25} 32-bit outputs were generated, and the expression $A * Z_i = B * Z_{i+1}$ was evaluated over them. To obtain these outputs, the X and Y variables were initialized to random values in the beginning and as the experiment runs, the X variable remains unchanged (here, we attempt to consider a disadvantageous scenario) while the Y variable is set to a new random value each time a new output is computed. From the above experiment we can deduce that the bias of the $MixBits$ function is bounded by:

$$BIAS = \frac{1}{2^{11.13}} \tag{10}$$

Table 1. Serial Correlation Test

	$Z = MixBits(X, Y)$		
Experiment	**Bit**	**Byte**	**4-Byte**
Z_i	0.000024	-0.000089	0.000279
$Z_i \oplus Z_{i+1}$	-0.000045	0.000026	0.000174
$Z_i - Z_{i+1}$	-0.000059	-0.000009	-0.000196

Table 2. Bit-Byte Prediction Tests for Randomness (Adapted from [12])

	$Z = MixBits(X, Y)$		
Test	Z_i	$Z_i \oplus Z_{i+1}$	$Z_i - Z_{i+1}$
Bit Prediction Test A	0.0032	0.8446	0.8453
Bit Prediction Test B	0.1284	0.7925	0.7928
Bit Prediction Test C	0.4094	0.9735	0.9729
Bit Prediction Test D	0.7448	0.9690	0.9687
Bit Prediction Test E	0.2975	0.6758	0.6717
Byte Prediction Test A	0.3049	0.6919	0.6970
Byte Prediction Test B	0.8854	0.8551	0.8522
Byte Prediction Test C	0.8549	0.8246	0.8209
Byte Prediction Test D	0.1717	0.9493	0.9483
Byte Repetition Test	0.7289	0.0684	0.0685

The serial correlation coefficients (at bit, byte and 4-byte level) were also studied to measure the extent to which a new intermediate output Z_i depended upon the previous value Z_{i-1}. To obtain a sizeable test sample, 2^{24} Z outputs were computed. As in the above experiment, the X and Y variables are randomly initialized at the beginning, and the Y variable is set to a new random value each time a new output is computed. Further analyses on the XOR $(Z_i \oplus Z_{i+1})$ and the difference $(Z_i - Z_{i+1})$ between two consecutive outputs were performed. The results are summarized in Table 1.

In addition, we evaluate how an attacker might predict an output if previous outputs are known. The bit-byte prediction tests [12] used to evaluate the randomness of the Konton2 stream cipher were employed for this purpose. Eight algorithms were used to predict the value of each bit (resp. byte) from the beginning to the end of the sequence. For a perfectly random sequence, the probability of success of any of the algorithms should be 1/2 (resp. $1/2^8$). The number of successes is counted, and a chi-squared statistic with 1 degree of freedom computed. Table 2 shows the results.

From our analyses, we find that $MixBits$ has very good properties. Indeed, our analysis shows that the output of $MixBits$ cannot be predicted significantly better than a pure random guess if the adversary does not have any knowledge of the secret access password. At this point, two of the three initial requirements are fulfilled: only efficient operations are used, and the function is highly non-linear. An estimate of the gate count for $MixBits$ can be easily obtained. Six logic gates are needed for each bit added in parallel[1]. The registers will be implemented by means of flip-flops, each of which requires 8 gates. Furthermore, two 32-bit registers are needed – one to store the output Z and another for the intermediate results. Hence, a total of around 700 logic gates are needed to implement $MixBits$.

An estimate of the temporal requirements can also be carried out. A tag has to spend around 128 clock cycles to compute an output ($Z = MixBits(X, Y)$). Assuming a clock frequency of 100 kHz, a tag can compute around 780 updates per second. Hence, the timing requirements are also fulfilled. To complete the analysis, a comparison with several block/stream ciphers and hash functions was carried out and the results shown in Table 3 (for the price comparison, $MixBits$ is fixed as the reference and every extra 1,000 gates is assumed to increase chip price by $0.01 [13]). We find that $MixBits$ is the most efficient in circuit area and although throughput is not the highest, it is within the requirements of the intended applications (i.e. baggage tracking, electronic toll collection, pallet tracking, etc.).

Light-weight ciphers such as Present or Grain require only 1,570 or 1,294 logic gates respectively. However, this number of gates, even though small, may still exceed the capabilities of tags conforming to the EPC Class-1 Gen-2 specification. Furthermore, where tag price is concerned, slight differences in tag prices can be greatly magnified under an operating environment where large numbers of tags are deployed. Imagine for a company that needs to deploy 500 million tags. A difference of US$0.0059 – 0.0087 (Grain-Present) per tag would amount to US$ 2,950,000 – 4,350,000 of extra costs in total, which is a significant sum. In this case, using the Grain or Present cipher could be rather expensive. In addition, a significant number of logic gates devoted to security would have to be set aside for the 16-bit PRNG since a light-weight PRNG conforming to the EPC Class-1 Gen-2 specification would require around 1,600 gates. While the use of a cipher or hash function will increase the level of security, it will also incur hardware costs. In this work, our main objective is to design a lightweight authentication protocol under the EPCGlobal Framework, and requires balancing tradeoffs between security and hardware restrictions. From our analysis, we find that our proposed $MixBits$ function performs reasonably well and provides an appropriate security level for tags compliant with the EPC Class-1 Gen-2 specification.

6. Analysis of the M^{3}AP Protocol

In this section, we provide a proof sketch to show that our proposed M^{3}AP protocol provides mutual authentication between a tag and a reader. In addition, we also analyze the security of the protocol by examining how the protocol fares against previous attacks exposed on the TRMA schemes, as well as other passive and active attacks.

Table 3. Performance comparison

Cryptographic primitive	Gates Equivalent	Cycles per block	Throughput at 100 KHz (Kbps)	Price (Cents)
Mixbits	700	128	25	K
Block ciphers				
Present [14]	1,570	32	200	K + 0.87
DESL [15]	1,848	144	44.4	K + 1.15
HIGHT [16]	3,048	34	188.2	K + 2.35
AES [17]	3,400	1,032	12.4	K + 2.70
Stream ciphers				
Grain-80 [18]	1,294	1	100	K + 0.59
Grain-80, x16 [18]	3,239	1	1,600	K + 2.54
Trivium [18]	2,599	1	100	K + 1.89
Trivium, x16 [18]	3,185	1	1,600	K + 2.48
Hash functions				
MD5 [19]	8,400	612	20.91	K + 7.7
SHA-1 [19]	8,120	1,274	12.56	K + 7.42
SHA-256 [19]	10,868	1,128	22.69	K + 10.17

6.1. Verification of Mutual Authentication.

Reader-to-Tag Authentication: The first two messages of our proposed scheme allow a legitimate reader that has knowledge of the tag's access password to authenticate itself to the tag. A malicious (or illegitimate) reader does not possess the access password to generate the corresponding responses (CCP_1 and CCP_2). Due to lack of authorization for the illegitimate reader, this information cannot be obtained from the manufacturer (EPC-IS). In addition, the computation of CCP_1 and CCP_2 uses only random challenges from the tag. In the original TRMA scheme, CCP_1 and CCP_2 are computed from random values generated by the tag, as well as random values generated by the reader. However, this provides an avenue for an illegitimate reader to specify the random values in such a way that allows it to circumvent the scheme and forge a successful authentication more easily. By having the reader compute the authentication responses based solely on random challenges generated by the tag and the shared secret (the tag access password), our scheme eliminates such a weakness.

Tag-to-Reader Authentication: The third message of our scheme is for a legitimate tag to authenticate itself to the reader after it has confirmed that the reader is a legitimate one. A fraudulent tag does not possess the access password that is necessary to compute the cover-codes (CCP_3 and CCP_4). In this case, cover-codes only depend on the random numbers picked by the genuine reader and avoids the vulnerability in the original TRMA scheme whereby the attacker has control over the inputs required to compute the authentication response. Hence, without knowledge of the correct access password, a tag impersonation attack cannot

[1] $S = A \oplus [B \oplus C_{ENT}]$ $C_{SAL} = BC_{ENT} + AC_{ENT} + AB$

be successful and authentication would fail.

6.2. Resistance against Previous Attacks on TRMA

Resistance against the Correlation Attack. In order to perform the correlation attack described in [6], the adversary first needs to find a correlation between the access password PWD and the $MixBits$ output (PWD' and PWD''). Once this is found, the adversary can make use of the relationships derived in [6] (the relationships between PWD and the output of $PadGen$, where in our new scheme, $PadGen$ is applied to PWD' and PWD'' instead of PWD) to attack the scheme. However, as witnessed in the last section, we have shown that it is highly difficult to obtain any correlation between the input and output of $MixBits$. Hence, we contend that our proposed scheme provides strong resistance against the correlation attack.

Resistance against the Dictionary Attack. In the original TRMA scheme, the value of each bit of the authentication response CCP_i (i=1,2,3,4) is only dependent on the value of a particular hex-digit in R_i^{Tag} or R_i^{Rdr}. For example, the first bit of CCP_1 depends on the first hex-digit of R_1^{Tag}. If the value of a hex-digit in any R_i^{Tag} or R_i^{Rdr} is repeated (i.e. it had the same value in a previous authentication session), then the adversary would be able to successfully predict the value for the corresponding bit in R_i to forge a successful authentication. In our proposed scheme, we find that each bit in any R_i is dependent on all four 16-bit random numbers generated by the tag and the reader. For example, each bit in CCP_1 or CCP_2 is dependent on all four of R_1^{Tag}, R_2^{Tag}, R_1^{Rdr} and R_2^{Rdr}. This is because all of them are involved in the computation of PWD' in $MixBits$. Moreover, the nature of $MixBits$ ensures that the bits of the four random numbers are diffused within PWD'. In order to successfully predict the value of a bit in CCP_1 or CCP_2, the adversary must encounter a situation whereby all four random numbers contain the same values that have appeared together in a previous session. The probability of this occurring is extremely low, since with a total of 64 bits between them, the number of possible combinations amounts to 2^{64}. Hence, the dictionary attack is still possible but becomes extremely difficult. In fact, this attack can be completely prevented if we update or refresh the access password after every authentication session. For example, we can change the access password from PWD to PWD'' at the end of the protocol after both parties are mutually authenticated. The new access password will then be used for the next authentication session, and so on.

Resistance against the Tag Killing Attack. Unlike the extended TRMA scheme, the kill password is not used in our proposed authentication scheme. Furthermore, all messages exchanged during the protocol are independent of the kill password of the communicating tag. Hence, an adversary would not be able to gather any information about the kill password of tags from authentication sessions under our proposed scheme.

6.3. Resistance against Other Attacks

Resistance against Replay Attacks. In a replay attack, the adversary eavesdrops on the messages exchanged between a legitimate reader and a legitimate tag, and replays the authentication responses to masquerade as the reader or the tag. Such an attack would be successful only if all the four random challenges have appeared together and in the right sequence in a previous authentication session. With the legitimate parties generating fresh random challenges for each authentication session, the probability of success for a replay attack would be low.

Resistance against Offline Brute Force Attacks. In an offline brute force attack, an adversary eavesdrops on a single pass (for example, the reader-to-tag authentication) of an authentication session to obtain a set of random challenges and the valid response based on those challenges. Next, the adversary assumes a value for the access password and computes a response based on the collected challenges (by executing the $MixBits$ function, the $PadGen$ function, and other necessary operations). If the computed response matches with the collected response, then the value assumed for the access password was correct. Otherwise, the adversary tries the next probable value for the access password, repeating until a correct match is found. The complexity of this attack is $O(2^l)$, where l is the number of bits in the access password. To offer adequate resistance against such an attack, l should be sufficiently large.

Resistance against Active Brute Force Attacks. Active brute force attacks generally require an adversary to actively take part in the authentication protocol by masquerading as a tag or a reader. A number of scenarios are possible. In the first scenario, an adversary can programme a malicious reader to repeatedly probe a legitimate tag. During each authentication attempt, the reader tries a different value for the access password. This continues until the adversary authenticates successfully to the tag. In another scenario, an adversary can iteratively issue challenges to the legitimate reader and record valid sets of challenges and responses to form a dictionary. Both attacks can be made infeasible with sufficiently large access passwords and random challenges, or the use of password updating.

Resistance against Desynchronization Attacks. Under our proposed scheme, since the access password is constant, there is no threat of desynchronization. However, as discussed earlier, to completely prevent some of the attacks, it would be necessary to update the access password at the end of each successful mutual authentication. In this case, the copies of the access password kept at the tag and the reader (or the back-end database, as in most cases) must be the same at all times, i.e. they must be synchronized. Once any party fails to update its copy of the access password at the end of a successful authentication session, both parties will be de-synchronized. Hence, with password updating, extra measures may need to be taken to ensure that the protocol is robust against desynchronization.

Resistance against Unauthorized Tracking. As mentioned earlier, privacy is not a focus in this work and the current EPCglobal Framework does not seem to address privacy issues. The transmission of the EPC in clear implies that unauthorized tracking of tags is possible. We contend that it is possible to integrate previously proposed methods with our scheme to guard against privacy violation. For example, the EPC can be replaced with a pseudonym (as proposed in [20]) or be relabelled (as in [21]) to prevent tracking of the tag. The EPC can also be protected using masking or RF jamming techniques (e.g. [22]), or through controls provided by an RFID proxy device (e.g. [23]). Naturally, implementing these solutions for privacy protection leads to higher costs incurred on the resulting system.

7. Conclusions

In this paper, a new authentication protocol, which is named as M^{3}AP and based on the protocol by Konidala *et al.*, is proposed. The security deficiencies were corrected in M^{3}AP with the introduction of the $MixBits$ function. This lightweight function has been obtained by means of Genetic Programming. Its security and performance has been studied in depth. In addition, a security analysis of the whole M^{3}AP protocol has been accomplished and we find that there is greater resistance against attacks. In conclusion, we expect that our M^{3}AP protocol can help to increase the security level for the upcoming Gen-3 specification.

References

[1] EPCglobal, EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz Version 1.0.9.

[2] International Organization for Standards (ISO), ISO/IEC 18000: Radio frequency identification for item management.

[3] D. M. Konidala and K. Kim, "RFID Tag-Reader Mutual Authentication Scheme Utilizing Tag's Access Password", *Auto-ID Labs White Paper WP-HARDWARE-033*, Jan 2007.

[4] D. N. Duc, J. Park, H. Lee, K. Kim, "Enhancing security of EPCglobal GEN-2 RFID tag against traceability and cloning", in *The 2006 Symposium on Cryptography and Information Security*, 2006.

[5] H. Y. Chien, C. H. Chen, "Mutual authentication protocol for RFID conforming to EPC Class 1 Gen 2 standards", in *Computer Standards & Interfaces 29 (2007)*, pp. 254 - 259, 2007.

[6] T. L. Lim, and T. Li, "Addressing the Weakness in a Lightweight RFID Tag-Reader Mutual Authentication Scheme", in *Proc. of IEEE Globecom 2007*, Nov 2007.

[7] D. M. Konidala, Z. Kim, and K. Kim, "A Simple and Cost-effective RFID Tag-Reader Mutual Authentication Scheme", in *Proc. of Int'l Conference on RFID Security (RFIDSec'07)*, pp. 141-152, Jul 2007.

[8] P. Peris-Lopez, T. Li, T. L. Lim, J.C. Hernandez-Castro and J.M. Estevez-Tapiador. "Vulnerability Analysis of a Mutual Authentication Scheme under the EPC Class-1 Generation-2 Standard", in *Proc. of RFIDSec '08*, Jul 2008.

[9] T. L. Lim, T. Li. "Exposing an Effective Denial of Information Attack from the Misuse of EPCglobal Standards in an RFID Authentication Scheme", in *Proc. of IEEE PIMRC*, Sep 2008.

[10] J. R. Koza, "Evolving a computer program to generate random number using the genetic programming paradigm", in *Proc. of the 4th Int'l Conf. on Genetic Algorithms*, 1991.

[11] J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda-Garnacho, B. Ramos-Alvarez, "Wheedham: An automatically designed block cipher by means of genetic programming", in *Proc. of CEC '06*, pp. 192–199, 2006.

[12] David Sexton, Randomness Analysis of Konton2, http://www.geocities.com/da5id65536, 2005.

[13] M. Lehtonen, et al., "Networked RFID Systems and Lightweight Cryptography", in *Chapter from Identification to Authentication - A review of RFID Product Authentication Techniques*, pp. 169-187. Springer, 2007.

[14] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher", in *Proc. of CHES'07*, LNCS vol. 4727, pages 450–466, 2007.

[15] A. Poschmann, G. Leander, K. Schramm, and C. Paar, "New Light-Weight Crypto Algorithms for RFID", in *Proc. of IEEE International Symposium on Circuits and Systems, ISCAS'07*, pages 1843–1846, 2007.

[16] D. Hong, *et al.*, "HIGHT: A New Block Cipher Suitable for Low-Resource Device, in *Proc. of CHES'06*, LNCS vol. 4249, pp. 46–59, 2006.

[17] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES implementation on a grain of sand", in *IEEE Proc. of Information Security*, vol. 152, no.1, pp. 13–20, 2005.

[18] T. Good, and M. Benaissa, "Hardware results for selected stream cipher candidates", in http://www.ecrypt.eu.org/stream/, 2007.

[19] M. Feldhofer and C. Rechberger, "A case against currently used hash functions in RFID protocols", in *Printed handout of Workshop on RFID Security – RFIDSec'06*, 2006.

[20] A. Juels, "Minimalist Cryptography for Low-Cost RFID Tags", in *Proc. of SCN'04*, LNCS 3352, pp. 149-164, 2004.

[21] S. Inoue, and H. Yasuura, "RFID Privacy Using User-Controllable Uniqueness", in *RFID Privacy Workshop*, 2003.

[22] M. R. Rieback, B. Crispo, and A. S. Tanenbaum, "Keep on Blockin' in the Free World: Personal Access Control for Low-Cost RFID Tags", in *Proc. of the 13th Int'l Workshop on Security Protocols*, Apr 2005.

[23] A. Juels, P. Syverson, and D. Bailey, "High-Power Proxies for Enhancing RFID Privacy and Utility", in *Proc. of PET '05*, 2005.