On the Improper Use of CRC for Cryptographic Purposes in RFID Mutual Authentication Protocols

Eyad Taqieddin

Multimedia Networking Research Laboratory, Jordan University of Science and Technology, Jordan

***Abstract*—**Mutual authentication is essential to guarantee the confidentiality, integrity, and availability of an RFID system. One area of interest is the design of lightweight mutual authentication protocols that meet the limited computational and energy resources of the tags. These protocols use simple operations such as permutation and cyclic redundancy code for cryptographic purposes. However, these functions are cryptographically weak and are easily broken. In this work, we present a case against the use of these functions for cryptographic purposes, due to their simplicity and linear properties, by analyzing the LPCP protocol. We evaluate the claims of the LPCP resistance to de-synchronization and full disclosure attacks and show that the protocol is weak and can be easily broken by eavesdropping on a few mutual authentication sessions. This weakness stems from the functions themselves as well as the improper use of inputs to these functions. We further offer suggestions that would help in designing more secure protocols.

Keywords: *A*uthentication, cryptography, cyclic redundancy check codes, information security, RFID tags.

1. Introduction

Radio Frequency Identification (RFID) systems are deployed in various industrial and academic applications. They are used in asset management systems, libraries, credit cards, and passports. These systems consist of tags that store unique identifying information about the objects being tracked, one or more readers that communicate wirelessly with the tags, and a backend database to store the information collected from the tags. The information in the database is continuously updated according to the responses from the tags.

Passive RFID tags harvest the incident power from the reader signals and use it to run their circuitry and respond back to the reader. This form of energy harvesting supports a short communication range and limited computational power of the tags.

A critical aspect to be considered with RFID systems is data privacy and integrity. Current RFID implementations have little provision to these necessary features. Thus, the system becomes vulnerable to various attacks that include tag cloning, tag tracking, information disclosure, and denial of service (DoS).

In the past few years, researchers focused on proposing protocols that support mutual authentication between the readers and tags and maintain the privacy of secret information shared between these parties. For successful mutual authentication between the reader and the tag, both parties must possess the same secret encryption keys and use them in the private exchange of random numbers and the tag identifier (ID). A mutual authentication session (hereupon referred to as the session) involves a sequence of challenge-response exchanges using the secret keys and random numbers. The encryption operations must be chosen in line with the limited computational and energy capabilities.

Since the use of conventional ciphers, such as the Data Encryption Standard (DES) and Advanced Encryption Standard (AES), is not viable, several protocols were proposed based on the use of simpler functions. Ultralightweight protocols are based on simple bitwise, addition, and shift operations. These protocols are suitable in terms of their computational complexity but are weak from a cryptographic perspective. On the other hand, lightweight protocols use the same functions as those in ultralightweight protocols along with cyclic redundancy code (CRC) and pseudo-random number generators (PRNG). These two added functions are readily available on the tags that comply with the Electronic Product Code (EPC) Class 1 Generation 2 standard [1].

Unfortunately, although these functions offer limited privacy, they are still being employed in various mutual authentication protocols. Since the CRC function is implemented on the RFID tags, it is widely employed as a possible replacement of more complex cryptographic functions. This poses a serious threat on the overall security of the system and the associated data exchanges.

In this work, we present a case against the use of such functions for mutual authentication. This is done by highlighting the inherent weaknesses of the permutation and CRC functions. To that end, a detailed analysis of a recently proposed protocol called the Ultralightweight RFID authentication Protocol with CRC and Permutation (LPCP) [2] is presented to show that it falls short of providing the claimed security features. Due to the improper usage of the CRC functions and the unique properties of the messages structure, two attacks are presented. The first is a de-synchronization attack in which the key update process is manipulated such that the reader and the tag hold different keys. The second is a full disclosure attack that reveals all the secrets shared between the reader and the tag. Furthermore, we offer suggestions that would help in designing more secure protocols.

The paper is organized as follows: Section 2 presents an overview of earlier attempts for secure ultralightweight and lightweight protocols and their weaknesses. This is followed by an overview and security analysis of the LPCP protocol in Section 3. Finally, the paper is concluded in Section 4.

2. Related Work

The early attempts of ultralightweight cryptography appeared in the Ultralightweight Mutual Authentication Protocol (UMAP) family. It consisted of the Lightweight Mutual Authentication Protocol (LMAP) [3], Minimalist Mutual-

Authentication Protocol (M2AP) [4], and the Efficient t Mutual Authentication Protocol (EMAP) [5]. The main operations in these protocols are the XOR, AND, OR and addition. These protocols were shown to be vulnerable to de-synchronization and full disclosure attacks [6–11].

The Strong Authentication and Strong Integrity (SASI) protocol [12] was proposed as an alternative to overcome these weaknesses. It added the rotation operation to the bitwise operations used in the UMAP family. The protocol was investigated thoroughly and several papers gave detailed attacks that resulted in de-synchronization [13-14]. More importantly, a full disclosure attack with high probability of success by eavesdropping on 217 protocol runs was provided in [15]. A detailed analysis of SASI in [16] presented the de-synchronization, ID disclosure, and full secret values disclosure attacks.

The Gossamer protocol provided enhancements to overcome the vulnerabilities of its predecessors. It still was shown, however, that the de-synchronization problem was not solved. Active attacks that would result in de-synchronization are detailed in [17-18].

One protocol that received attention is the RFID authentication protocol with permutation (RAPP) [19]. This protocol extended the operations used in SASI by adding the permutation operation. The goal behind the permutation operation is to hide any bit relationships that result from bitwise operation. Similar to its predecessors, various attacks appeared and showed its inherent weaknesses. De-synchronization attacks with a reasonable probability of success are given in [20-21]. Moreover, a detailed analysis in [22] gave the steps to run a full disclosure attack using 2^{30} tag queries. This number of queries was hugely reduced down to 192 tag queries in an improved full disclosure attack in [23]. A detailed analysis of these previous attacks is given in [24].

One recent protocol is the Succinct and Lightweight Authentication Protocol (SLAP) [25]. In this protocol, an ultralightweight operation, called conversion, is used as the basis for cryptographic operations. The authors present a detailed security analysis and claim that the protocol is resistant to de-synchronization, replay, and tracing attacks.

SLAP was followed by an even newer protocol called the pseudo-Kasami code based Mutual Authentication Protocol (KMAP) [26] that avoids the use of simple bitwise operations and, instead, uses a primitive operation that enhances the diffusion properties to make the secrets irreversible. In the analysis part, it is claimed that the protocol resists all kinds of attacks.

The work in [27] provided an analysis of SLAP and KMAP and presented a generalized case of a de-synchronization attack that applies to both protocols. The attack shows that the de-synchronization attack is still possible even though the tag and the reader hold copies of the old and new secret values.

As a general observation, most of the protocols that are based on lightweight operations were shown to be susceptible to attacks due to the inherent weaknesses in their cryptographic operations. The work in [28] discusses the typical mistakes that appear in the design of protocols and provides guidelines to be followed in order to design and evaluate the validity of mutual authentication protocols.

Recently, Gao et al. [2] presented an analysis of the RAPP protocol and demonstrated a de-synchronization attack by tampering with the exchanged message. As a solution to the vulnerability, the authors proposed a new protocol called LPCP. This protocol is a lightweight protocol (the authors call it approximate ultra-lightweight) that uses the CRC function along with the operations defined for the RAPP protocol. The authors verified the protocol using the Simple Promela Interpreter and claimed that the LPCP protocol provides confidentiality and is resistant to de-synchronization, tracing, replay, and full disclosure attacks. The work in [29] presented a de-synchronization attack by impersonating a valid reader using eavesdropped messages from earlier sessions. We further analyze the protocol and present two attacks with a success probability equal to one.

3. Security analysis of LPCP protocol

The message exchange structure and the updates for LPCP are shown in Fig. 1. The tag stores the Index-pseudonym (IDS) along with three keys (K_L, K_M, K_H). The reader maintains two copies of these parameters labeled as $(IDS^{old}, K_L^{old}, K_M^{old} K_H^{old})$ and $(IDS^{new}, K_L^{new}, K_M^{new}, K_H^{new})$. The designation of *new* represents the updated values of the parameters whereas *old* represents the values from the previous session.

The reader initiates a session by sending a Hello message to the designated tag. In return, the tag responds with its IDS. The reader uses the received IDS to retrieve the parameters associated with it from the backend database and then generates a random number R_1 which is used in computing messages α and β. When the tag receives these messages, it extracts R_1 from α and then computes its own version of β. A match indicates that the reader is authentic. The tag then sends γ to the reader which, in turn, uses it to authenticate the tag.

Next, the reader generates a new random number, R_2 and computes messages δ and ζ. Furthermore, the reader checks if the received $IDS = IDS^{new}$ then an update of the keys and IDS value takes place. The tag extracts R_2 from δ and verifies it using ζ. If no anomaly is detected then the tag performs the same update to its parameters.

In this section, a detailed security analysis of the LPCP protocol is given by presenting two attacks. The first is a novel de-synchronization attack in which the reader and tag will hold mismatching keys and will not be able to prove their identities to each other. The second attack leads to the full disclosure of all the secrets shared between the reader and the tag. This allows the attacker to track the tag, impersonate the tag in responding to the reader or vice versa, and to maintain possession of the secrets values, even after the update that takes place at the end of each session.

Preliminaries

To lay the foundation for presenting the attacks, we

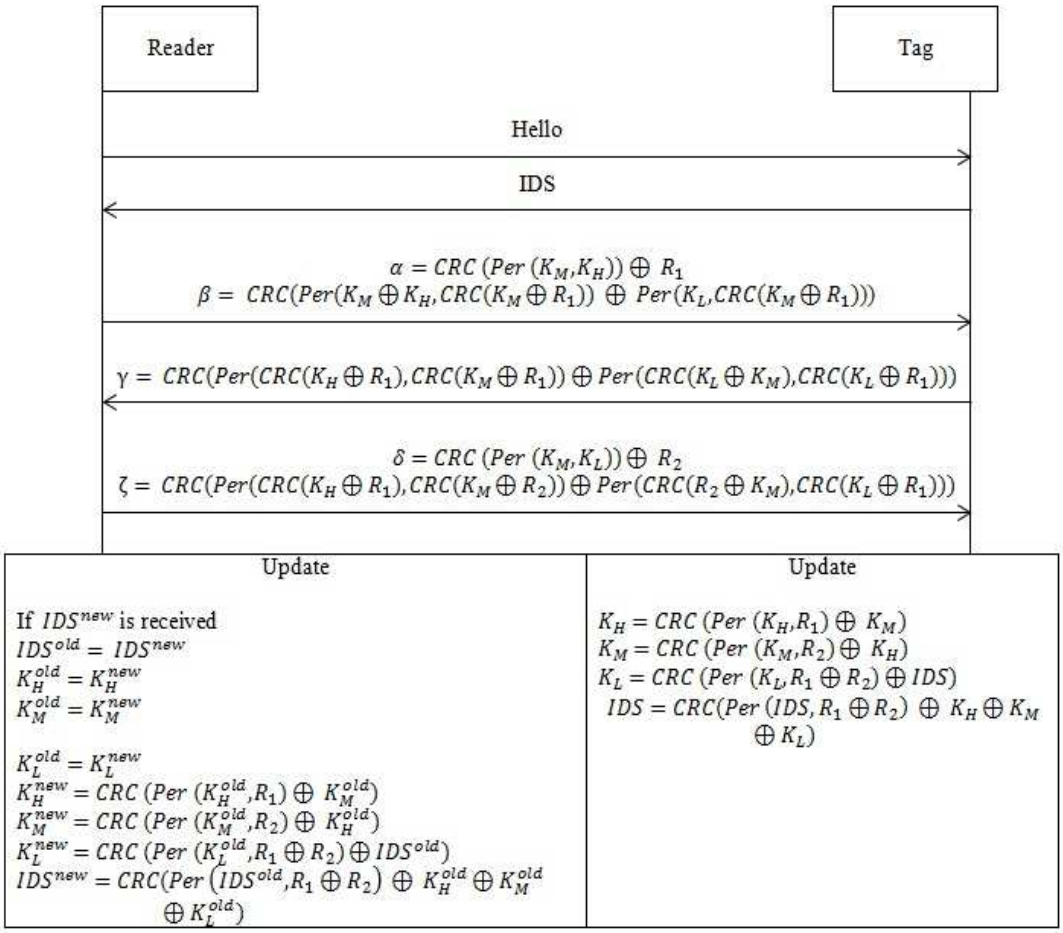

Figure 1. The LPCP message exchange [2]

discuss the two main cryptographic functions used in LPCP: the CRC and permutation (Per) operations. Then we present observations related to the message structure and these functions.

There are various implementations of the 16-bit CRC functions that differ based on the initial value and the polynomial. The CRC function used on EPC compliant tags produces a 16-bit value based on the input value and the generator polynomial $x^{16}+x^{12}+x^5+1$ [1]. For illustration purposes, we adopt the CRC-CCITT (XModem) in which the computation of the CRC is done by initializing a register with $0x0000$ and then clocking the data one bit at a time to be encoded (from MSB to LSB). Once all bits have been clocked, the result in the register is taken as the CRC value. Note, however, that the discussion in this paper would be applicable to any other variant of the 16-bit CRC function.CRC functions are not suitable for cryptographic operations due to their linear properties [30-32]. The cryptanalysis presented provides a further case against adopting these functions for cryptographic purposes.

As for the second function, Per, the definition is as follows [19]:

For two n-bit strings, X and Y, in the form

$X = x_1\, x_2 \ldots x_n,\ x_i \in \{0,1\},\ i = 1, 2, \ldots, n$

$Y = y_1\, y_2 \ldots y_n,\ y_i \in \{0,1\},\ i = 1, 2, \ldots, n$

The Hamming weight of Y, $wt(Y)$, is m $(0 \le m \le n)$ and $y_{k1} = y_{k2} = \ldots = y_{km} = 1$ and $y_{km+1} = y_{km+2} = \ldots = y_{kn} = 0$ where $1 \le k1 < k2 < \ldots < km \le n$ and $1 \le km+1 < km+2 < \ldots < kn \le n$, then

$$Per(X,Y) = x_{k1}\, x_{k2} \ldots x_{km}\, x_{kn}\, x_{kn-1} \ldots x_{km+2} x_{km+1} \quad (1)$$

As an example, assume we have two 8-bit arrays, $X = 10010011$ and $Y = 11101010$ then $Per(X,Y) = 10001101$. The details are shown in Fig. 2.

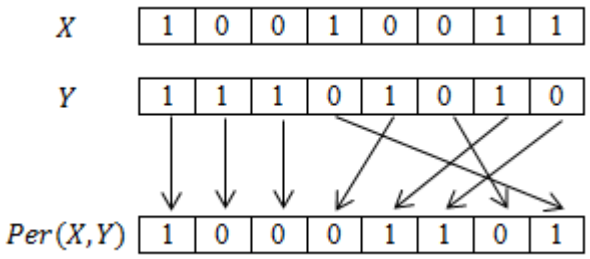

Figure 2. Permutation example

From the example, we see that we start from the leftmost position of Y and scan for the values of 1. Whenever a match is found, the corresponding bit of X is taken. When the rightmost bit is reached, we scan Y in the opposite direction but this time we look for a match with zero and copy the corresponding bit.

Observation 1: The way the CRC function is used in the LPCP protocol makes it a reversible function.

The CRC function may be used with any size of input data. However, with the LPCP protocol, the authors restricted the size of the inputs to 16 bits. This is a serious flaw because there will be 2^{16} possible inputs with 2^{16} possible outputs.

By examining all input combinations and the resulting outputs, we find that the CRC becomes a one-to-one function. In other words, given $y = CRC(x)$ it is possible to find $x = CRC^{-1}(y)$.

Observation 2: Given

$$A = Per(X,Y) = a_1 a_2 a_3 \ldots a_{n-2} a_{n-1} a_n \quad \text{then}$$

$$B = Per(X,\bar{Y}) = a_n a_{n-1} a_{n-2} \ldots a_3 a_2 a_1 \tag{2}$$

Where $\bar{Y}$ is the complement of Y.

To illustrate this observation, we consider a case of 4-bit arrays where $X = x_1\ x_2\ x_3\ x_4$ and we find A and B for all possible values of Y and $\bar{Y}$, as shown in Table 1. Note that each entry in column A is a mirror image of that in column B. This can be extended to any array size, n.

Observation 3: For an n-bit string X with a Hamming weight of X, $wt(x) = m,\ (0 \le m \le n)$

$$A = Per(X, X) = x_{k1}\ x_{k2} \ldots x_{km}\ x_{kn}\ x_{kn-1} \ldots x_{km+2} x_{km+1} \tag{3}$$

Where ,

$x_{k1} = x_{k2} = \ldots = x_{km} = 1$ and $x_{km+1} = x_{km+2} = \ldots = x_{kn} = 0$

Table 1. Possible Permutations of X Based on Y and $\bar{Y}$

Y	$\bar{Y}$	$A = Per(X,Y)$	$B = Per(X,\bar{Y})$
0000	1111	$x_4\ x_3\ x_2\ x_1$	$x_1\ x_2\ x_3\ x_4$
0001	1110	$x_4\ x_3\ x_2\ x_1$	$x_1\ x_2\ x_3\ x_4$
0010	1101	$x_3\ x_4\ x_2\ x_1$	$x_1\ x_2\ x_4\ x_3$
0011	1100	$x_3\ x_4\ x_2\ x_1$	$x_1\ x_2\ x_4\ x_3$
0100	1011	$x_2\ x_4\ x_3\ x_1$	$x_1\ x_3\ x_4\ x_2$
0101	1010	$x_2\ x_4\ x_3\ x_1$	$x_1\ x_3\ x_4\ x_2$
0110	1001	$x_2\ x_3\ x_4\ x_1$	$x_1\ x_4\ x_3\ x_2$
0111	1000	$x_2\ x_3\ x_4\ x_1$	$x_1\ x_4\ x_3\ x_2$
1000	0111	$x_1\ x_4\ x_3\ x_2$	$x_2\ x_3\ x_4\ x_1$
1001	0110	$x_1\ x_4\ x_3\ x_2$	$x_2\ x_3\ x_4\ x_1$
1010	0101	$x_1\ x_3\ x_4\ x_2$	$x_2\ x_4\ x_3\ x_1$
1011	0100	$x_1\ x_3\ x_4\ x_2$	$x_2\ x_4\ x_3\ x_1$
1100	0011	$x_1\ x_2\ x_4\ x_3$	$x_3\ x_4\ x_2\ x_1$
1101	0010	$x_1\ x_2\ x_4\ x_3$	$x_3\ x_4\ x_2\ x_1$
1110	0001	$x_1\ x_2\ x_3\ x_4$	$x_4\ x_3\ x_2\ x_1$
1111	0000	$x_1\ x_2\ x_3\ x_4$	$x_4\ x_3\ x_2\ x_1$

This means that the first m bits of $Per(X, X)$ will be 1's and the remaining $n - m$ bits will be 0's.

To illustrate this point, consider an example where $X = 10001011$ then $Per(X, X) = 11110000$

Finally, we use the theorem below (detailed proof is given in [29]).

Theorem 1: For any CRC and for any n-bit strings a and b, it holds that

$$CRC(a \oplus b) = CRC(a) \oplus CRC(b) \tag{4}$$

De-synchronization attack

The LPCP protocol was originally proposed with the goal of overcoming the threat of a de-synchronization attack. However, it was shown in [29] that the LPCP is still vulnerable to de-synchronization within three sessions by replaying older messages.

In this section, we show a new de-synchronization attack the exploits the weaknesses inherent in the CRC and Per functions. A major enhancement in this attack is that it succeeds in one session only. Furthermore, the updated IDS will be the same for both the tag and the reader which would make it harder to identify the attack.

The premise of the attack is to influence the tag to accept false random numbers instead of those generated by the reader. As such, when the update process takes place, the reader and the tag will use different input values for the update. For this to happen, the attacker does not necessarily need to know the keys.

First, we simplify the equation of β using a property from [22] which states that

$$Per(X,Y) \oplus Per(Z,Y) = Per\ (X \oplus Z, Y) \tag{5}$$

Thus, message β can be presented as

$$\beta = CRC(Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1))) \tag{6}$$

From observation 1, we deduce that

$$T = CRC^{-1}(\beta) = Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1)) \tag{7}$$

For CRC-CCITT (XModem), we know that $CRC(0x84CF) = 0xFFFF$. Thus, we take α sent by the reader and modify it to $\alpha' = \alpha \oplus 0x84CF$. The tag extracts the random number from α' and its value will be $R_1^{'} = R_1 \oplus 0x84CF$. For the attack to succeed, we need to replace β with a new message (we call it β') using the value of $R_1^{'}$.

$$\beta' = CRC(Per(K_M \oplus K_H \oplus K_L,\ CRC(K_M \oplus R_1^{'}))) \tag{8}$$

By considering theorem 1 we find that

$$\begin{aligned} CRC(K_M \oplus R_1^{'}) &= CRC(K_M \oplus R_1 \oplus 0x84CF) \\ &= CRC(K_M \oplus R_1) \oplus CRC(0x84CF) \\ &= CRC(K_M \oplus R_1) \oplus 0xFFFF \end{aligned} \tag{9}$$

This means negating all the bits of $CRC(K_M \oplus R_1)$, thus $CRC(K_M \oplus R_1^{'}) = \overline{CRC(K_M \oplus R_1)}$,

By substituting in (8)

$$\beta' = CRC(Per(K_M \oplus K_H \oplus K_L,\ CRC(K_M \oplus R_1) \oplus 0xFFFF)) \tag{10}$$

From observation 2, we note that $Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1) \oplus 0xFFFF)$ is the mirror image of T. Thus, to find β', we take the mirror image of T (call it T') and then apply it to the CRC function

$$\beta' = CRC(T') \tag{11}$$

The tag receives α' and β' then accepts $R_1^{'}$ as the random number, and generates the message γ'

$$\gamma' = CRC(Per(CRC(K_H \oplus R_1^{'}), CRC(K_M \oplus R_1^{'})) \oplus Per(CRC(K_L \oplus K_M), CRC(K_L \oplus R_1^{'}))) \tag{12}$$

We need to convert γ' to the original γ such that the reader will authenticate the tag without recognizing the mismatched random numbers. For this to happen, the message should be based on the value of R_1.

Given

$$G' = CRC^{-1}(\gamma')$$

$$= Per(CRC(K_H \oplus R_1^{'}), CRC(K_M \oplus R_1^{'})) \oplus Per(CRC(K_L \oplus K_M), CRC(K_L \oplus R_1^{'})) \quad (13)$$

Which can be represented as $G' = G_1^{'} \oplus G_2^{'}$, where

$$G_1^{'} = Per(CRC(K_H \oplus R_1^{'}), CRC(K_M \oplus R_1^{'})) \quad (14)$$

$$G_2^{'} = Per(CRC(K_L \oplus K_M), CRC(K_L \oplus R_1^{'})) \quad (15)$$

We need to find the value

$$G = G_1 \oplus G_2 = Per(CRC(K_H \oplus R_1), CRC(K_M \oplus R_1)) \oplus Per(CRC(K_L \oplus K_M), CRC(K_L \oplus R_1)) \quad (16)$$

By examining (14), we see that

$$G_1^{'} = Per(CRC(K_H \oplus R_1^{'}), CRC(K_M \oplus R_1^{'})) = Per(\overline{CRC(K_H \oplus R_1)}, CRC(K_M \oplus R_1) \oplus 0xFFFF) \quad (17)$$

This means that we find G_1 by complementing all the bits of $G_1^{'}$ and then taking their mirror image.

Also, by examining (15), we see that

$$G_2^{'} = Per(CRC(K_L \oplus K_M), CRC(K_L \oplus R_1^{'})) = Per(CRC(K_L \oplus K_M), CRC(K_L \oplus R_1) \oplus 0xFFFF) \quad (18)$$

This indicates that we find G_2 by taking the mirror image of the bits of $G_2^{'}$.

Since the mirror image of both $G_1^{'}$ and $G_2^{'}$ is taken then the relative positions of adjacent bits will be the same. As a result, to find the original value of $G = G_1 \oplus G_2$, we take the mirror image of G' and then complement the result (due to the complement used with G_1).

Finally, the value of γ is found as $\gamma = CRC(G)$ and then sent to the reader.

The reader verifies γ and accepts the tag as authentic, it then generates the random number R_2 and sends the messages δ and ζ. However, we modify δ to δ' (Similar to the case of α and α') so the tag will extract $R_2^{'}$ from it. As a final step, we need to convince the tag to accept the modified random number by sending ζ' instead of ζ.

We follow a similar approach in updating ζ. First we take

$$Z = CRC^{-1}(\zeta) = Per(CRC(K_H \oplus R_1), CRC(K_M \oplus R_2)) \oplus Per(CRC(R_2 \oplus K_M), CRC(K_L \oplus R_1)) \quad (19)$$

Consider that

$$Z_1 = Per(CRC(K_H \oplus R_1), CRC(K_M \oplus R_2)) \quad (20)$$

$$Z_2 = Per(CRC(R_2 \oplus K_M), CRC(K_L \oplus R_1)) \quad (21)$$

We need to modify Z_1 and Z_2 to match with the values of $R_1^{'}$ and $R_2^{'}$. As such,

$$Z_1^{'} = Per(CRC(K_H \oplus R_1^{'}), CRC(K_M \oplus R_2^{'})) \quad (22)$$

$$Z_2^{'} = Per(CRC(R_2^{'} \oplus K_M), CRC(K_L \oplus R_1^{'})) \quad (23)$$

For both (22) and (23), we see that the results are complemented and their mirror image is taken. Thus taking $Z' = Z_1^{'} \oplus Z_2^{'}$ results in eliminating the effect of the complements. What is left is the effect of taking the mirror image of Z (i.e.; Z' is the mirror image of Z)

And $\zeta' = CRC(Z')$

The tag accepts the values of the random numbers. Now, when both parties attempt to run their updates the results will not match.

On the reader side

$$IDS^{new} = CRC(Per(IDS, R_1 \oplus R_2) \oplus K_H \oplus K_M \oplus K_L) \quad (24)$$

$$K_H = CRC\ (Per(K_H, R_1) \oplus K_M) \quad (25)$$

$$K_M = CRC\ (Per(K_M, R_2) \oplus K_H) \quad (26)$$

$$K_L = CRC(Per(K_L, R_1 \oplus R_2) \oplus IDS) \quad (27)$$

On the tag side

$$IDS = CRC(Per(IDS, R_1^{'} \oplus R_2^{'}) \oplus K_H \oplus K_M \oplus K_L) \quad (28)$$

$$K_H = CRC(Per(K_H, R_1^{'}) \oplus K_M) \quad (29)$$

$$K_M = CRC\ (Per(K_M, R_2^{'}) \oplus K_H) \quad (30)$$

$$K_L = CRC(Per(K_L, R_1^{'} \oplus R_2^{'}) \oplus IDS) \quad (31)$$

An interesting point is that that both the reader and the tag will still have similar IDS values because $R_1 \oplus R_2 = R_1^{'} \oplus R_2^{'}$. However, the keys K_H and K_M will differ and the next session will not succeed.

We illustrate this attack by an example with reduced length of 16-bit strings. Table 2 lists the initial values shared between the reader and the tag. We label them with a superscript of R or T to indicate whether it is a reader or a tag value; respectively.

Table 2. Initial Values of Shared Parameters Between the Reader and the Tag

Reader Side		Tag Side	
IDS^R	0xBEAF	IDS^T	0Xbeaf
K_H^R	0xE5AB	K_H^T	0xE5AB
K_M^R	0xA9AF	K_M^T	0xA9AF
K_L^R	0xA9AD	K_L^T	0xA9AD
R_1^R	0x2EA2	R_1^T	Unknown yet
R_2^R	0x5555	R_2^T	Unknown yet

The reader initiates by sending the Hello message and receives $IDS = 0xBEAF$. The reader uses this IDS value to acquire the shared keys from the database and then generates the random number R_1^R. The reader sends

$\alpha = 0xFE8E$ and $\beta = 0x9718$

These messages are modified by the attacker as $\alpha' = \alpha \oplus 0x84CF = 0x7A41$. Also, β' is computed by first taking $T = CRC^{-1}(\beta) = 0x9D4B$

The mirror image of the value of T is taken to find $T' = 0xD2B9$. and $\beta' = CRC(T') = 0x45B7$. The values of α' and β' are sent to the tag, which uses α' to extract $R_1^T = 0xAA6D$. and verifies it by comparing its local value of β' with that received.

Next, the tag generates $\gamma' = 0x1CAE$ and the attacker modifies it by finding $G' = CRC^{-1}(\gamma') = 0xDC62$. From the

discussion above, we need to take the mirror image of the bits and then invert them to find $G = 0xB9C4$ and $\gamma = CRC(G) = 0x3D5D$. The reader uses the received value of γ' and compares it against its local value and then considers the tag authentic. The reader then generates R_2^R and sends $\delta = 0x8FEC6$ and $\zeta = 0x515C$.

The attacker modifies δ to find $\delta' = 0x0x0B29$ and ζ' by first taking

$Z = CRC^{-1}(\zeta) = 0xC668$

The mirror image $Z' = 0x1663$ and its corresponding $\zeta' = CRC(Z') = 0xF510$. The tag extracts $R_2^T = 0xD19A$ from these messages.

Thus, the reader performs its update on its stored values using $R_1^R = 0x2EA2$ and $R_2^R = 0x5555$ while the tag uses the values $R_1^T = 0xAA6D$ and $R_2^R = 0xD19A$.

The new values after the session is completed are shown in Table 3. Note that although both tags have updated to the same IDS value, there is a mismatch between the values of K_H and K_M held by the tag and those held by the reader. This indicates that no future sessions would succeed in mutual authentication.

This attack can be easily extended to cover the case of EPC compliant tags of 96-bit length. The only difference in that all strings would be divided into six 16-bit substrings to be processed as shown above. This can be done in one protocol run, similar to the 16-bit example.

Table 3. New Values of Shared Parameters Between Reader & Tag

Reader Side		Tag Side	
IDS^R	0x7A22	IDS^T	0x7A22
K_H^R	0x170C	K_H^T	0x45C1
K_M^R	0x221E	K_M^T	0xE29A
K_L^R	0xB959	K_L^T	0xB959
R_1^R	0x2EA2	R_1^T	0xAA6D
R_2^R	0x5555	R_2^T	0xD19A

Full disclosure attack

The full disclosure attack is more elaborate with the goal of finding all the secret information shared between the reader and the tag. The attacker deliberately forces the current session not to be completed by blocking the message γ from reaching the reader. In such a case, no update will take place (thus the same secret information will be used). However, per the protocol specification, a new value for R_1 will be generated for each subsequent session.

The following notation is used in presenting the attack

X^i	Bit string X sent in session i, where $X \in \{\alpha, \beta, T, \gamma, \delta, \xi\}$
$[X]_j$	The j[th] bit of string X
R_1^i and R_2^i	Random numbers generated for session i
$[0]_j$	An n-bit array of all 0's except at bit position j

We run the attack in two steps. In the first step, we capture messages from several sessions for offline analysis. The second step involves an active attack phase where the messages are manipulated in order to reveal more information about the secret keys.

Step 1: Message capture and offline analysis

By examining (6) and (7), we see that for any two sessions i and j, the values of α^i and β^i will differ from α^j and β^j. With all secret keys constant, this change depends solely on the values R_1^i and R_1^j.

$$\alpha = CRC(Per(K_M, K_H)) \oplus R_1 \quad (32)$$

$$\beta = CRC(Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1))) \quad (33)$$

One fact about the permutation operation is that $wt(Per(X,Y)) = wt(Per(X,W))$. This is because we only change the positions of the bits of X without any substitution. The same can be applied to (33) where

$$wt(Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^i))) = .$$
$$wt(Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^j))) \quad (34)$$

Consider the scenario where the attacker has captured the tuples $\langle \alpha^1, \beta^1 \rangle$ and $\langle \alpha^i, \beta^i \rangle$, for any $i \geq 2$. Thus, we have:

$$\alpha^1 = CRC\ (Per(K_M, K_H)) \oplus R_1^1 \quad (35)$$

$$\beta^1 = CRC(Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^1))) \quad (36)$$

$$\alpha^i = CRC\ (Per(K_M, K_H)) \oplus R_1^i \quad (37)$$

$$\beta^i = CRC(Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^i))) \quad (38)$$

By taking

$$\alpha^1 \oplus \alpha^i$$
$$= CRC(Per(K_M, K_H)) \oplus R_1^1 \oplus CRC\ (Per(K_M, K_H)) \oplus R_1^i$$
$$= R_1^1 \oplus R_1^i = \Delta^i \rightarrow R_1^i = R_1^1 \oplus \Delta^i \quad (39)$$

And substituting R_1^i in (38), we have

$$\beta^i = CRC(Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^1 \oplus \Delta^i)))$$
$$= CRC(Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^1) \oplus CRC(\Delta^i))) \quad (40)$$

And since the effect of the outer CRC operation in β can be reversed based on observation 1, we get

$$T^1 = CRC^{-1}(\beta^1)$$
$$= Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^1)) \quad (41)$$

$$T^i = CRC^{-1}(\beta^i)$$
$$= Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^1) \oplus CRC(\Delta^i)) \quad (42)$$

The interpretation of this difference between (41) and (42) is that the bits of $K_M \oplus K_H \oplus K_L$ are permuted by $CRC(K_M \oplus R_1^1) \oplus CRC(\Delta)$ instead of just $CRC(K_M \oplus R_1^1)$. When $[CRC(\Delta)]_j = 0$, $j = 1,2,\ldots,n$, then $[CRC(K_M \oplus R_1^1) \oplus CRC(\Delta)]_j = [CRC(K_M \oplus R_1^1)]_j$ and the permutation will not be affected. However, when $[CRC(\Delta)]_j = 1$, the result of $[CRC(K_M \oplus R_1^1) \oplus CRC(\Delta)]_j$ will be inverted and it will

affect the result of the permutation operation. This bit inversion is very useful in reversing the result of the permutation to find its two inputs. The logic behind this is that if the bit was originally 1 and its corresponding bit, $[K_M \oplus K_H \oplus K_L]_j$, was placed at position h in the result of the permutation, then when we invert the bit to 0 its corresponding bit would be placed at position k, where $k > h$. The same logic applies if we invert the bit from 0 to 1 the bit would be moved from position h to position k with $h > k$.

We employ this idea to find the bits of $K_M \oplus K_H \oplus K_L$ and $CRC(K_M \oplus R_1^1)$. By considering the leftmost bit $[K_M \oplus K_H \oplus K_L]_0$, there are two possible positions for this bit to occupy in T, $[T]_0$ if $[CRC(K_M \oplus R_1^1)]_0 = 1$ or $[T]_n$ if $[CRC(K_M \oplus R_1^1)]_0 = 0$. Thus, if we invert bit $[CRC(K_M \oplus R_1^1)]_0$ then either $[T]_0$ will appear as $[T]_n$ (meaning that $[CRC(K_M \oplus R_1^1)]_0$ was originally 1) or $[T]_n$ will appear as $[T]_0$ ($[CRC(K_M \oplus R_1^1)]_0$ was originally 0). From this, we have determined the value of $[CRC(K_M \oplus R_1^1)]_0$ by the change of position and also we have determined the value of $[K_M \oplus K_H \oplus K_L]_0$ because that is the value of the bit $[T]_i$ that got affected by the inversion.

Once we know the value at the position 0, we can proceed iteratively from position 1 up to position $n-1$. At the end of this step, we will have several possible values of $K_M \oplus K_H \oplus K_L$ and $K_M \oplus R_1^1$. In themselves, they do not expose the values of the individual keys but are used in the second step of the attack.

To illustrate the steps of the offline analysis, we use a reduced size example with the same initial values from Table 2. The goal is to find $K = K_M \oplus K_H \oplus K_L$ and $C = CRC(K_M \oplus R_1^1)$ based on the differences between session 1 and its subsequent sessions.

Initially, the attacker captures α^1 and β^1 which are created based on the keys and the random number R_1^1. Message γ^1 is stored by the attacker and blocked form the reader to force it to generate a new random number, R_1^2, and its corresponding α^2 and β^2. The process is repeated several times until we have a sufficient set of α^i and β^i messages. Table 4 lists the values of α^i and β^i for 7 protocol sessions generated by implementing the protocol in C code.

To find $[K]_0$, we consider all cases for which $[CRC(\Delta^i)]_0 = 1$. This applies to sessions 2 and 7. By comparing T^1 with T^2 and T^7, the attacker can extract some information regarding the unknown bits.

Considering $T^1 = 9D4B = 1001110101001011$ and $T^2 = E565 = 1110010101100101$, we see that $[T^1]_0 = [T^2]_{15}$ and also $[T^1]_{15} = [T^2]_0$. Thus, it is not possible to determine the effect of the bit flip of $[CRC(\Delta^i)]_0$ because either $[T^1]_0$ has become $[T^2]_{15}$ or $[T^1]_{15}$ has become $[T^2]_0$. However, if we consider Fig. 3, we observe that $[T^1]_0 \neq [T^7]_{15}$ and $[T^1]_{15} = [T^7]_0$. This leads to the conclusion that $[T^1]_{15}$ must have changed its position and became $[T^7]_0$. In other words, the bit which was originally permuted with a $[C]_0 = 0$ to appear as the $[T^1]_{15}$ has now been permuted with a 1 to appear as the $[T^7]_0$.

To confirm this result, assume that the bit was originally permuted with $[C]_0 = 1$, instead of 0. This would mean that the bit should appear as $[T^1]_0$ and when the permuting bit is flipped, $[C]_0 = 1$, it should appear as $[T^7]_{15}$. However, since $[T^1]_0 \neq [T^7]_{15}$, this confirms that the assumption is incorrect and asserts the conclusion that $[K]_0 = [T^1]_{15} = 1$ and $[C]_0 = 0$.

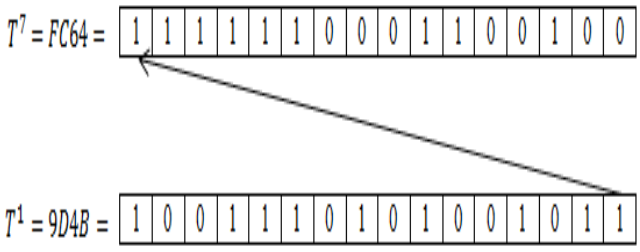

Figure 3. Bit transition after flipping $[CRC(\Delta^i)]_0$

Next, to find the bits $[K]_1$ and $[C]_1$, the attacker considers the cases for which $[CRC(\Delta^i)]_1 = 1$. The sessions of interest are 4, 5, and 6. Not all cases would yield a result but by considering Fig. 4 to examine T^1 and T^4.

Note that $[T^1]_{15}$ is set to 1 and since $[C]_0 \oplus [CRC(\Delta^4)]_0 = [C]_0$, then the permutation of that bit for both strings will not be affected (i.e.; $[T^1]_{15} = [T^4]_{15}$). For illustration purposes, all known bits are shown highlighted.

TABLE 4. Values of Captured Messages α^i and β^i and their Corresponding Δ^i

Session (i)	α^i	β^i	$T^i = CRC^{-1}(\beta^i)$	$\Delta^i = \alpha^1 \oplus \alpha^i$	$CRC(\Delta^i)$

1	0xFE8E	0x9718	0x9D4B	---	
2	0x7EB6	0xD344	0xE565	0x8038	0xAAC3
3	0x76B6	0xAAF7	0xD58D	0x8838	0x256A
4	0x34B6	0x72CB	0x34D7	0xCA38	0x4EC4
5	0x342C	0x59D3	0x965B	0xCAA2	0x6C37
6	0x8579	0x0F91	0x2B2F	0x7BF7	0x4B5B
7	0x8D7E	0x7A8E	0xFC64	0x73F0	0xB215

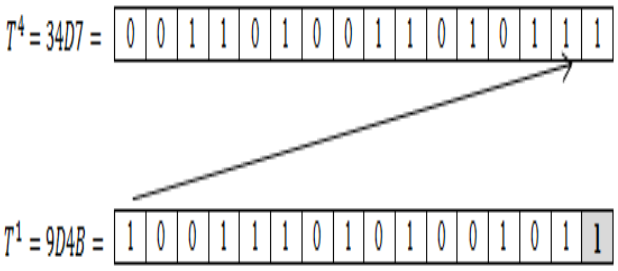

Figure 4. Bit transition after flipping $[CRC(\Delta^i)]_1$

Our interest now is on bit positions 0 and 14. Note that $[T^1]_{14} \neq [T^4]_0$ while $[T^1]_0 = [T^4]_{14}$. This leads to the conclusion that $[T^1]_0$ must have changed its position and became $[T^4]_{14}$. In other words, the bit which was originally permuted with $[C]_1 = 1$ to appear as $[T^1]_0$ has now been permuted with a bit 0 to appear as the $[T^4]_{14}$. This means that the value of the bit $[K]_1 = [T^1]_0 = 1$ and that it was originally permuted with $[C]_1 = 1$.

The same result could be obtained by considering T^1 along with $T^6 = 0010110100101111$.

Up to this point, we know the first two bits of

$K = 1 \quad 1$

$C = 0 \quad 1$

The process is continued to find the bits at position 2. By taking session 2, and knowing that $CRC(\Delta^2) = 0xACC3$, then $[C]_{0,1} \oplus [CRC(\Delta^2)]_{0,1} = 01 \oplus 10 = 11$. Thus, $[K]_{0,1}$ will be permuted with 11 to yield the first 2 bits of T^2, $[T^2]_{0,1}$.

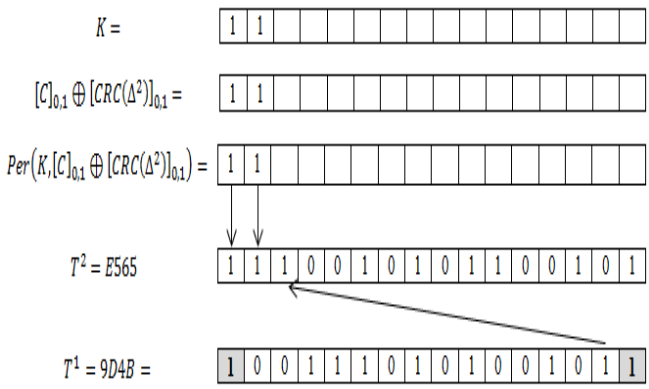

Figure 5. Bit transition after flipping $[CRC(\Delta^i)]_2$

From Fig. 5, $[T^1]_{14} = [T^2]_2$ and $[T^1]_1 \neq [T^2]_{15}$ and we conclude that $[K]_2 = [T^1]_{14} = 1$ and this bit was permuted with $[C]_2 = 0$, giving

$K = 1 \quad 1 \quad 1$

$C = 0 \quad 1 \quad 0$

The same procedure is repeated to find all the remaining bits. Note, however, that in some cases we may not be able to use the procedure to find a conclusive result. For example, when considering position 8, the applicable sessions are 2 and 4. We notice that for none of the sessions produces the case where $[T^1]_h = [T^i]_j$ and $[T^1]_k \neq [T^i]_l$. In such a case, there are two options, either we capture more messages or examine the case when bit $[CRC(\Delta^i)]_8 = 0$. Looking for a case for which the bits on the boundaries are equal and this would give us $[K]_8$.

By studying the case of T^3 in Fig. 6, the boundary bits $[T^3]_5$ and $[T^3]_{13}$ are equal. Regardless of the initial position of these bits, we can safely say that $[K]_8 = [T^3]_5 = [T^3]_{13} = 1$.

Furthermore, from $T^1 = 1001110101001011$, we see that the boundary bits are $[T^1]_4 = 1$ and $[T^1]_{11} = 0$. As such, we know that $[K]_8$ certainly appears as $[T^1]_4$. Hence, $[C]_8 = 1$.

By proceeding with the steps, and getting to position 14, we have two bits only to consider $[T^1]_7 = 0$ and $[T^1]_8 = 1$.

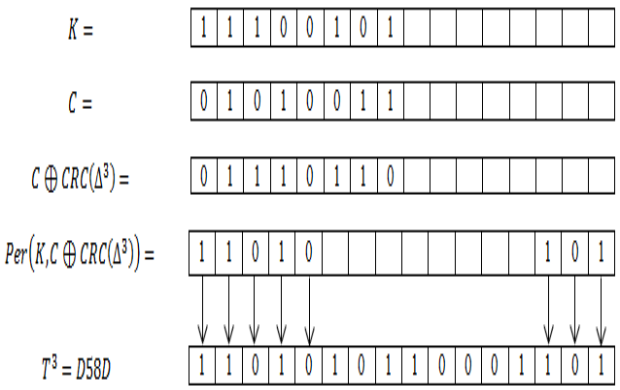

Figure 6. Bit transition after flipping $[CRC(\Delta^i)]_8$

When these bits are swapped there will be no conclusive result to determine which of them is $[K]_{13}$ or $[K]_{14}$.

As a result, we have two possible values for K, denoted by K_1 and K_2 given along with the corresponding possible values of C; respectively.

$K_1 = 11100101101010\boxed{01}$ $\quad$ $K_2 = 11100101101010\boxed{10}$

$C = 01010011101000\boxed{10}$ $\quad$ $C = 01010011101000\boxed{00}$

$C = 01010011101000\boxed{11}$ $\quad$ $C = 01010011101000\boxed{01}$

To summarize, the results of the first step of the full disclosure attack are given in Table 5.

Table 5. Results of First Step of Full Disclosure Attack.

$K_M \oplus K_H \oplus K_L$	$CRC(K_M \oplus R_1^1)$	$K_M \oplus R_1^1$
0xE5A9	0x53A2	0x870D
	0x53A3	0x1A7C
0xE5AA	0x53A0	0xADCE
	0x53A1	0x30BF

Step 2: Active attack by manipulating the random numbers

Knowing $K_M \oplus K_H \oplus K_L$ and $K_M \oplus R_1^1$ does not expose the values of the individual keys. However, we utilize them in the next step of the attack to manipulate α^1 and β^1 that were captured at the beginning. The goal behind this step is to convince the tag to accept a modified value for R_1^1 in a manner that would give more information about the secret values. Since we have four sets of possible values, as shown in Table 5, then this step needs to be repeated four times, once for each set.

The attacker manipulates α by XORing it with $K_M \oplus R_1^1$ as shown in (43)

$$\begin{aligned}\alpha' &= \alpha \oplus K_M \oplus R_1^1 \\ &= CRC(Per(K_M, K_h)) \oplus R_1^1 \oplus K_M \oplus R_1^1 \\ &= CRC(Per(K_M, K_h)) \oplus K_M \end{aligned} \quad (43)$$

When the tag receives α', it extract the value of $R_1^1 = K_M$.

Now, to find the corresponding β' that would be accepted by the tag, we take

$$\beta = CRC(Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^1))) \quad (44)$$

$$\begin{aligned}T &= CRC^{-1}(\beta) \\ &= Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus R_1^1)) \end{aligned} \quad (45)$$

By substituting $R_1^1 = K_M$,

$$\begin{aligned}T' &= Per(K_M \oplus K_H \oplus K_L, CRC(K_M \oplus K_M)) \\ &= Per(K_M \oplus K_H \oplus K_L, CRC(0x0000)) \\ &= Per(K_M \oplus K_H \oplus K_L, 0x0000) \end{aligned} \quad (46)$$

This, as mentioned previously, indicates that T' is the mirror image of $K_M \oplus K_H \oplus K_L$ which is already known. Hence,

$$\beta' = CRC(T') \quad (47)$$

When the tag verifies β', if will accept the value of $R_1^1 = K_M$ and proceed to generate

$$\begin{aligned}\gamma^1 = &CRC(Per(CRC(K_H \oplus R_1^1), CRC(K_M \oplus R_1^1))) \oplus \\ &Per(CRC(K_L \oplus K_M), CRC(K_L \oplus R_1^1)) \end{aligned} \quad (48)$$

By substituting $R_1^1 = K_M$, (9) is reduced to

$$\begin{aligned}\gamma^1 = &CRC(Per(CRC(K_H \oplus K_M), CRC(K_M \oplus K_M))) \oplus \\ &Per(CRC(K_L \oplus K_M), CRC(K_L \oplus K_M)) \end{aligned} \quad (49)$$

Thus,

$$\begin{aligned}\gamma^1 = &CRC(Per(CRC(K_H \oplus K_M), CRC(0x0000))) \oplus \\ &Per(CRC(K_L \oplus K_M), CRC(K_L \oplus K_M)) \end{aligned} \quad (50)$$

From (50), we can find

$$\begin{aligned}G^1 &= CRC^{-1}(\gamma^1) \\ &= Per(CRC(K_H \oplus K_M), 0x0000) \oplus \\ &\quad Per(CRC(K_L \oplus K_M), CRC(K_L \oplus K_M)) \end{aligned} \quad (51)$$

These values are used offline without any further message collection needed. Based on observation 3, we see that $Per(CRC(K_L \oplus K_M), CRC(K_L \oplus K_M))$ will results in a string consisting of m ones and $16-m$ zeros, where m is the Hamming weight of $CRC(K_L \oplus K_M)$. Note that this value is unknown to us but it will be one of 16 possible values $(000\ldots0, 100\ldots0, 110\ldots0, 111\ldots0, 111\ldots1)$. Thus, we take the value of G^1 and run it in a loop for which in each iteration we XOR it with one of the possible 16 values to get the value of $Per(CRC(K_H \oplus K_M), 0x0000)$ for that iteration. By taking the mirror image of these bits, we find the value of $CRC(K_H \oplus K_M)$ which can be easily converted to $K_H \oplus K_M$.

Within the iteration, we run another loop that takes all possible values of K_H ranging from 0x0000 to 0xFFFF. From these values and $K_H \oplus K_M$, we can find the corresponding K_M. And from the known $K_M \oplus K_H \oplus K_L$ we can further extract K_L. Moreover, since we have the value of $K_M \oplus R_1^1$ then we can find the corresponding R_1^1.

The procedure above yields 2^{16} cases. For each case, we compute α and compare it with α^1. If they do not match then we proceed to the next case. However, if a match is found then we verify the result by computing γ and comparing the result with the stored value γ^1. There will only be one matching case which would give all the secret information.

As mentioned earlier, there are four possible cases and when all of them are attempted offline only one solution will result. Moreover, Since this step is done offline, it can be easily computed on a machine in minimal time and complexity.

Once this step is complete, we know K_M, ,K_H, K_L, and R_1^1. To find the last unknown value, we wait for the reader to initiate a new session, j, and passively collect messages a^j and b^j from which we extracts R_1^j and also captures δ^j and ζ^j from which we can easily compute R_2^j. At this stage, we have all the secret values and can update the set of keys after every successful session. Furthermore, we can impersonate a reader or a tag by falsifying messages based on the known secret values in our possession.

The complexity of the attack is very low in terms of the number of messages collected and the time needed for the offline analysis. Even if the EPC tags are used for a size of $n = 96$ bits, the complexity is still low. Since the strings will be divided into six substrings of 16-bits. For the first step of the attack, each substring will be analyzed separately from the others to yield four possibilities. For an attacker to manipulate a and b, he will need to find a combination in

which the concatenation of the six substrings would be accepted by the tag. As such, the attacker will need to run $4^6 = 2^{12}$ cases instead of 4 with the reduced size example. Once that is done, the computation can be performed in which each substring is dealt with separately with a linear increase in the complexity compared to the reduced size example.

4. Conclusion

In this paper, we analyzed the LPCP protocol and presented two serious vulnerabilities that lead to desynchronization and full disclosure of the secret values shared between the reader and the tag. In particular, the use of CRC as a cryptographic function is not a suitable choice due to its linear properties. Furthermore, the permutation operation taken from the RAPP protocol was shown to be of little cryptographic value and easily reversible. The complexity of the attacks is very low in terms of number of captured messages and the offline computation time.

For more secure constructions of protocols, it is recommended that the CRC function is not used. Protocol designers can make use of the PRNG functions available on the tags instead of the CRC. Short inputs of 16-bit length should be avoided to guarantee that no one-to-one relationship between the inputs and outputs is established.

Acknowledgement

This research was supported by the Jordan University of Science and Technology, Research Project No. 259/2016.

References

[1] EPC Radio-Frequency Identity Protocols, Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz–960 MHz, Version 2.0.0, EPC Global, Nov., 2013.

[2] L. Gao, M. Ma, Y. Shu, Y. Wei, "An ultralightweight RFID authentication protocol with CRC and permutation," Journal of Networks and Computer Applications, Vol. 41, No. 1, pp. 37–46, 2014.

[3] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda, "LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags," Workshop on RFID Security (RFIDSec'06), Graz, Austria, July 2006.

[4] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda, "M2AP: A minimalist mutual-authentication protocol for low-cost RFID tags," The 3rd International Conference on Ubiquitous Intelligence and Computing (UIC'06), China, pp. 912-923, 2006.

[5] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda, "EMAP: An efficient mutual authentication protocol for low-cost RFID tags," OTM Federated Conferences and Workshop: IS Workshop (IS'06), France, pp. 352-361, 2006.

[6] M. Bárász, B. Boros, P. Ligeti, K. Lója, D. Nagy, "Breaking LMAP," Workshop of RFID Security (RFIDSec'07), Spain, July 2007.

[7] T. Li, G. Wang, "Security analysis of two ultra-lightweight RFID authentication protocols," The 22nd IFIP TC-11 International Information Security Conference, South Africa, 2007.

[8] M. Bárász, B. Boros, P. Ligeti, K. Lója, D. Nagy, "Passive attack against the M2AP mutual authentication protocol for RFID tags," The 1st International EURASIP Workshop on RFID Technology, Austria 2007.

[9] H.-Y. Chien, C.-W, Huang, "Security of ultra-lightweight RFID authentication protocols and its improvements," ACM SIGOPS Operating Systems Review, Vol. 41, No. 4, pp. 83-86, 2007.

[10] A. Alomair, L. Lazos, R. Poovendran, "Passive attacks on a class of authentication protocols for RFID," International Conference on Information Security and Cryptology (ICISC 2007), South Korea, pp. 102-115, 2007.

[11] T. Li, R. Deng, "Vulnerability analysis of EMAP − an efficient RFID mutual authentication protocol," The 2nd International Conference on Availability, Reliability, and Security (AReS'07), pp. 238-245, Austria, 2007.

[12] H.-Y. Chien, "SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity," IEEE Transactions on Dependable and Secure Computing, Vol. 4, No. 4, pp. 337-340, 2007.

[13] H.-M. Sun, W.-C. Ting, K.-H. Wang, "On the security of Chen's ultralightweight RFID authentication protocol," IEEE Transactions on Dependable and Secure Computing, Vol. 8, No. 2, pp. 315–317, 2011.

[14] T. Cao, E. Bertino, H. Lei, "Security analysis of the SASI protocol," IEEE Transactions on Dependable and Secure Computing, Vol. 6, No. 1, pp. 73–77, 2009.

[15] G. Avoine, X. Carpent, B. Martin, "Strong authentication and strong integrity (SASI) is not that strong," Workshop on RFID Security (RFIDSec'10), Turkey, 2010.

[16] P. D'Arco, A. De Santis, "On ultralightweight RFID authentication protocols," IEEE Transactions on Dependable and Secure Computing, Vol. 8, No. 4, pp. 548-563, 2011.

[17] K.-H. Yeh, N.W. Lo, "Improvement of two lightweight RFID authentication protocols." Information Assurance and Security Letters, Vol. 1, pp. 6–11, 2010.

[18] D. Tagra, M. Rahman, S. Sampalli, "Technique for preventing DoS attacks on RFID systems, "The 18th International Conference on Software, Telecommunications, and Computer Networks (SoftCOM'10), Croatia, pp. 6-10, Sept. 2010.

[19] Y. Tian, G. Chen, J. Li, "A new ultralightweight RFID authentication protocol with permutation," IEEE Communications Letters, Vol. 16, No. 5, pp. 702-705, 2012.

[20] Z. Ahmadian, M. Salmasizadeh, M.R. Aref., "Desynchronization attack on RAPP ultralightweight authentication protocol," Information Processing Letters, Vol. 113, No. 7, pp. 205 – 209, 2013.

[21] D.-Z. Sun, Z. Ahmadian, Y.-J. Wang, M. Salmasizadeh, M.R. Aref, "Analysis and enhancement of desynchronization attack on ultralightweight RFID authentication protocol," Cryptology ePrint Archive (Technical Report) No. 2015/037, 2015.

[22] W. Shao-hui, H. Zhijie, L. Sujuan, C. Dan-wei, "Security analysis of RAPP: An RFID authentication protocol based on permutation," Cryptology ePrint Archive (Technical Report) No. 2012/327, 2012.

[23] N. Bagheri, M. Safkhani, P. Peris-Lopez, J. E. Tapiador, "Weaknesses in a new ultralightweight RFID authentication protocol with permutation-RAPP," Security and Communication Networks, Vol. 7, No. 6, pp. 945–949, 2014.

[24] U. Mujahid, M. Najam-ul-Islam, "Pitfalls in ultralightweight RFID authentication protocol," International Journal of Communication networks and information Security, Vol. 7, No. 3, pp. 169-176, 2015.

[25] L. Hanguang, G. Wen, J. Su, Z. Huang. "SLAP: succinct and lightweight authentication protocol for low-cost RFID system," Wireless Networks, pp. 1-10, 2016.

[26] U. Mujahid, M. Najam-ul-Islam, S. Sarwar. "A new ultralightweight RFID authentication protocol for passive low cost tags: KMAP," Wireless Personal Communications, pp. 1–20, 2016.

[27] M. Safkhani, N. Bagheri. "Generalized desynchronization attack on UMAP: Application to RCIA, KMAP, SLAP and SASI+ protocols," Cryptology ePrint Archive (Technical Report) No. 2016/905, 2016.

[28] G. Avoine, X. Carpent, J. Hernandez-Castro, "Pitfalls in ultralightweight authentication protocol designs," IEEE

Transactions on Mobile Computing, Vol. 15, No. 9, pp. 2317-2332, 2016.
[29] M. Akgün, U. Çağlayan, "On the security of recently proposed RFID protocols," Cryptology ePrint Archive (Technical Report) No. 2013/820, 2013.
[30] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, J. van der Lubbe, "Cryptanalysis of an EPC class-1 generation-2 standard compliant authentication protocol," Engineering Applications and Artificial Intelligence, Vol. 24, No. 6, pp. 1061 – 1069, 2011.
[31] D.C. Ranasinghe, Networked RFID systems and lightweight cryptography, Lightweight Cryptography for Low Cost RFID, Springer, pp. 311–346, 2007.
[32] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda, "Cryptanalysis of a novel authentication protocol conforming to EPC-C1G2 standard," Computer Standards and Interfaces, Vol. 31, No. 2, pp. 372 – 380, 2009.