A Key Recovery Attack on Error Correcting Code Based a Lightweight Security Protocol

Imran Erguler

Abstract—One of the interesting types of RFID application is RFID searching which aims to hear a specific RFID tag from a large group of tags, i.e. ability of detecting whether a target RFID tag is nearby. Very recently, a lightweight protocol using error-correcting codes has been proposed by Chen et al. to provide a solution to needs in this field. The authors give a detailed analysis of their protocol in terms of security, privacy, communication overhead, hardware cost and they claim that it is a realizable scheme with fulfilling security and privacy requirements. In this study, however, we investigate security of this protocol and clearly demonstrate its security flaws that completely allow an adversary to exploit the system. In particular, by using linear properties of error correcting coding we firstly describe a tag tracing attack that undermines untraceability property which is one its design objectives. Then along with its implementation details we present a key recovery attack that reduces dramatically search space of a tag's secret key and show that an adversary can compromise it in practical time by only querying this tag for several times. As an illustrative example we retrieve the secret key of the protocol in two hours for the adopted linear block code $\mathcal{C}(47, 24, 11)$ which is one of the suggested codes.

Index Terms—Authentication, error correcting coding, lightweight, privacy, RFID, security

1 Introduction

RFID technology has become prevalent in various fields. Manufacturing, supply chain management and inventory control are some popular examples of these areas where RFID applications are deployed. There are three key elements in a typical RFID system: Tags, one or more readers, and a back-end server. Nevertheless for the sake of clarity in protocol description, reader and server are sometimes treated as a single entity (throughout this study we also consider them as one entity). In a traditional RFID communication protocol, through a wireless environment RFID reader queries RFID tags that are attached to objects and delivers their responses to a back-end server to identify and get related information about the object. As opposed to this classic tag identification scenario, RFID searching, emerged as a new trend in RFID applications, aims to communicate or find out a specific tag among a large group of tags [1]. In this case, RFID application initially knows the target tag and executes a defined protocol to segregate this tag from the others, e.g. this mechanism should be used to check whether an item is nearby. In [2], Chen et al. introduced such a protocol, which we will call CMC due to name of the first author, to provide secure and authenticated communication between the reader and the tags. At the same time, they intended to satisfy computation and hardware constraints of low-cost RFID tags, without sacrificing security and privacy requirements. In fact, this is one of the most challenging issues in design of secure RFID protocols, since proven cryptographic primitives need higher hardware resources and cannot be implemented in small world of these gadgets. In this regard, the authors only use a pseudo-random number generator (PRNG), an applicable one-way hash function like in [3], [4], [5] and an error correcting code technique is adopted to comply with definition of a lightweight protocol [6]. Furthermore, it is asserted that the CMC protocol accomplishes its security and privacy goals, including resistance to tracing, tag impersonating, replay and desynchronization attacks. In this paper, however, we scrutinize security of the CMC protocol and by using properties of linear block codes we describe a tracing attack that exhibits untraceability is not assured depending the on selected code. Also, by launching a key recovery attack we diminish key search space from $\mathcal{O}(2^k|\mathcal{S}_e|)$ to $\mathcal{O}(\max_{n,k}(|S_e|, 2^k))$, where $\mathcal{S}_e$ is the set of all error patterns that can be corrected by the $[n, k]$ linear block code. Although in [2] it is claimed that guessing the key is computationally infeasible for the suggested code examples, by means of our proposed attack it becomes realizable in practical time. In other words we have shown easy applicability of our attacks by performing them against different code examples and full disclosure of the secrets of a tag can be realized in hours.

1.1 RFID Authentication Protocols

The main objective behind deployment of authentication protocols in RFID systems is to provide secure communication between the reader/server and the tags in an authenticated manner and to satisfy privacy of the tag holders, i.e. concealing private information like location, ID etc. The majority of the proposed RFID security protocols like in [7], [8], [9], [10], [11], [12], [13] (readers may refer to [14] for more examples) deal with identification of an RFID tag through some queries and involvement of search mechanisms with different computational com-

I. Erguler is with National Research Institute of Electronics & Cryptology TUBITAK-BILGEM, 41470 Gebze, Kocaeli, Turkey.

plexities. In other words, these schemes aim to solve the problem of recovering ID from responses of an inquired tag without leaking any valuable information to an adversary. Recently, apart from these studies, another application field of RFID that intents to find out a specific ID from a pile of items have been drawn attention from RFID community. This field is also named as secure RFID searching. Nevertheless, only a few studies such as [15], [16], [17], [18], [19], [20] have handled this issue to provide a secure and privacy preserved protocol in this area. As a contribution in this direction, rather recently Chen et al. have presented a lightweight RFID searching protocol that specifically purposes to mitigate computational load at the reader side resulting from process of messages sent by unintended tags. In order to achieve this, the model basically takes advantage of computational simplicity of error correcting codes compared to a hash or a cryptographic primitive computation. It is equivalent to say that the CMC protocol comes up with a solution to computational load of the reader in RFID searching by tailoring error correcting codes to their scheme.

1.2 Error Correcting Codes in Crypto Systems

Error correcting codes have been deployed in different cryptographic schemes to create a trapdoor in the systems. McEliece's public-key cryptosystem is the most widely known example of this field [21]. The clever idea behind this system utilizes the NP hard problem of syndrome decoding in case of the number of errors is not bounded, i.e. its security relies on decoding of random linear codes. Another popular problem, as an NP-hard, related to error correcting code based crypto is learning parity with noise (LPN) which was introduced by Hopper and Blum in [22]. From the perspective of an attacker this problem can be seen as solving numerous equations of the form $z_i = a_i \cdot x \oplus e_i$, to capture secret x, where x and the e_i's are disguised from her and e_i is an error bit which is set to 1 with a probability $\gamma \in [0, \frac{1}{2}]$. Since this mechanism only needs scalar dot product and simple binary operations, it has attracted interest of lightweight crypto community and several protocol designs whose security rests on LPN problem have been proposed. The HB-family authentication protocols [22], [23], [24], [25], [26] are nice pioneer examples that comply with constraints of low-cost RFID tags, although in different studies it was addressed that these schemes cannot fulfill their security objectives [27]. The TCHo public key encryption scheme of J. Aumasson et al. [28] is another case where linear block codes are employed in security systems to provide a trapdoor. The subtlety of this scheme relies on the fact that the encryption of a message is performed like transmitting a message over a noisy channel: One small Linear Feedback Shift Register (LFSR) encodes the message, while a large one randomly initialized along with a source of biased random bits produces the noise. The sparse multiple of the feedback polynomial acts as the trapdoor whose availability converts ciphertext to a noisy message that can be decoded successfully. Also, noisy stream cipher models [29], [30] implicitly add error vectors to encoded messages to boost search space of an attacker, while these errors can be easily corrected at the receiver who shares same secret with the sender.

1.3 Organization

This paper is organized as follows: In Section 2, we describe some notations that used in the rest of the paper and give a brief description of the CMC protocol. Section 3 gives preliminaries and introduces our proposed attacks along with their underlying assumptions. In Section 4, we address complexities of the attacks and present their implementation details. Finally, we conclude our study in Section 5.

2 The CMC Protocol

2.1 Protocol Description

In [2], Chen et al. have presented a lightweight RFID authentication protocol to provide secure and authenticated communication between the reader and the tags. The fundamental purpose of this protocol is identifying a specific tag from a large group of tags such that communication is then proceeded with this tag and responses of others discarded. Since many response messages transmitted by unintended tags are collected at the reader side, the CMC protocol utilizes error correcting codes to filter garbage messages and isolate the target tag from the others, i.e. avoiding any computational burden at the reader side. In order to simplify the description of the protocol, the notations depicted in Table 1 are used in the remaining parts of the study.

Initially, the administrator chooses a pseudo random number generator $g()$, a one-way hash function $h()$ and a linear block code $\mathcal{C}(n, k, d_{min})$ with the generator matrix G and the parity check matrix H. Also each tag $\mathcal{T}_i$ is assigned with a unique identifer, n-bit length secret key K_i and k-bit length syndrome s_i. Then $g(), h(), K_i, s_i, G, H$ are written into storage memory of $\mathcal{T}_i$. On the other hand, reader/server stores K_i, s_i of each tag and keeps $G, H, g(), h()$. A step by step protocol definition of the CMC protocol illustrated in Fig. 1 is given below:

- Step 1: $\mathcal{R}$ selects a target tag $\mathcal{T}_i$, and determines the error vector e_R such that its syndrome is equal to s_i.
- Step 2: $\mathcal{R}$ picks a codeword C_R randomly and sums e_R with it to compute $C'_R = C_R \oplus e_R$.
- Step 3: $\mathcal{R}$ generates a random nonce N_R and sends it with C_R to $\mathcal{T}_i$ as a query.
- Step 4: Upon receiving the C_R, $\mathcal{T}_i$ computes $s = C_R \cdot H^T$ and checks whether $s = s_i$.
- Step 5: If $s = s_i$ holds, $\mathcal{T}_i$ randomly selects a codeword $C_i \in \mathcal{C}$ and an error vector e_i and gets

TABLE 1
Notations for the CMC Protocol

$\mathcal{T}_i$	RFID tag
$\mathcal{R}$	RFID reader
s_i	The syndrome pattern of $\mathcal{T}_i$
$g()$	Pseudo random number generator
$h()$	One-way hash function
G	Generator matrix
H	Parity check matrix
C_R, C_i	Codewords generated by $\mathcal{R}$ $\mathcal{T}_i$ respectively
e_R, e_i	Error vectors added by $\mathcal{R}$ and $\mathcal{T}_i$ respectively
N_R, N_i	Random nonce produced by $\mathcal{R}$ and $\mathcal{T}_i$ respectively
K_i	Secret key of $\mathcal{T}_i$
$w()$	Hamming weight
$\|\|$	Concatenation operator
$\oplus$	Bitwise XOR operation

$C'_i = C_i \oplus e_i$, where $w(e_i)$ less than or equal to error correcting capability of the code.

Step 6: $\mathcal{T}_i$ evaluates $M = K_i \oplus C'_i$ and calculates $V_i = g(s_i \oplus N_R \oplus h(N_i \oplus K_i)$

Step 7: If $s \neq s_i$, $\mathcal{T}_i$ sets V_i and C'_i to random values and gets $M = K_i \oplus C'_i$

Step 8: $\mathcal{T}_i$ transmits $\{M, V_i, N_i\}$ tuple to $\mathcal{R}$.

Step 9: $\mathcal{R}$ computes $M \oplus K_i$ and checks whether the result is decodable.

Step 10: If C'_i is decodable, $\mathcal{R}$ checks whether $g(s_i \oplus N_R \oplus h(N_i \oplus K_i) = V_i$ or not.

Step 11: If V_i is verified, $\mathcal{R}$ authenticates the tag and calculates $V_R = g(s \oplus N_i \oplus h(N_R \oplus K_i))$.

Step 12: If C'_i is not decodable or V_i is not verified $\mathcal{R}$ sets V_R to a random value

Step 13: $\mathcal{R}$ sends V_R to $\mathcal{T}_i$ and updates $K_i^{old} = K_i$, $K_i = h(K_i||N_R||s)$ in case of authentication, where K_i is matched key, i.e. K_i^{cur} or K_i^{old}.

Step 14: If $\mathcal{T}_i$ verifies $g(s \oplus N_i \oplus h(N_R \oplus K_i)) = V_R$, it authenticates the reader and updates the secret key $K_i = h(K_i||N_R||s)$.

After completing the authentication process, both of $\mathcal{R}$ and $\mathcal{T}_i$ compute $sk = g(K_i \oplus N_R \oplus N_i)$ to obtain the session key sk which is used encryption of exchanged sensitive data between them.

2.2 Security Claims

Below, we briefly overview the claimed security properties of the CMC protocol with their justifications detailed in [2]. For further information about common attack types on RFID systems, readers may refer to an interesting publication [31].

Anonymity and Untraceability: An adversary who does not have internal secrets of a tag cannot distinguish it from other tags by analyzing the tag responses, because N_i is a randomly produced value and V_i is generated by the pseudo-random number generator. Also, the C'_i is masked with the tag's secret key which is not available to the adversary, i.e. $K_i \oplus C'_i$ appears to random for her and it alleviates success probability of an adversary in linking transmitted messages with a specific tag. Furthermore, every tag replies to a query even if it is not the target tag (e.g. by responding with a random value) to avoid leakage of side channel information [32]. Thus, the protocol satisfies anonymity and untraceability.

Mutual Authentication: Before proceeding to secure communication, the reader and the tag mutually authenticates each other by using the verifier messages V_i and V_R.

Confidentiality: The security of transferred messages after the authentication process relies on guess complexity of the key K_i. In order to derive K_i, one may make exhaustive search over all possible C'_i values for a specific M message due to $M = C'_i \oplus K_i$. Note that $|C'_i| = \sum_{i=0}^{t_C} \binom{n}{i} \times 2^k$, where t_C is error correcting capability of an $[n, k]$ linear block code. For a properly selected code this search space makes the exhaustive search prohibitively expensive and the system achieves secure communication.

Resistance to Desynchronization: An adversary may block the message V_R to prevent update of the secret key at the tag side. In this case the tag might lose synchronization with the reader, because the reader updates the secret key while the tag keeps the current key. Nonetheless, the CMC keeps both of the old and new keys, so the reader can still resynchronize with the tag which missed V_R message in previous session by using the old session key K_i^{old}.

Resistance to Replay Attacks: In replay attacks, the goal of an adversary is to authenticate herself to reader or tag by reusing messages of previous sessions. Notice that the CRC protocol employs a challenge response

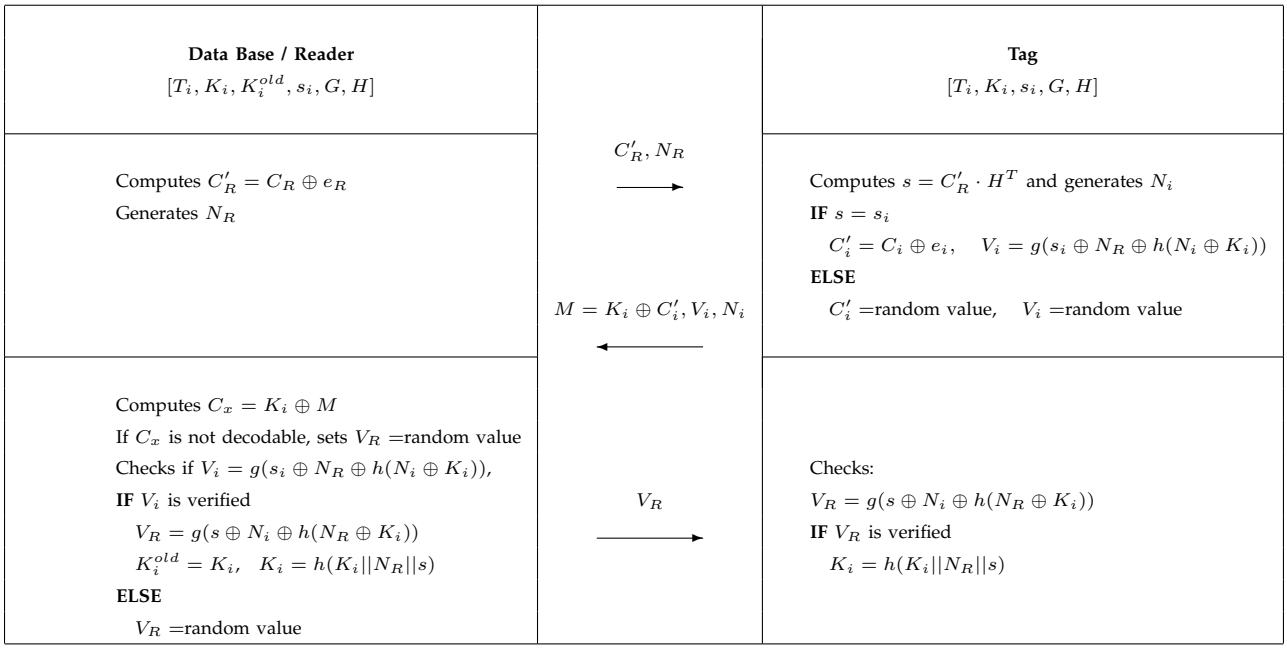

Fig. 1. The CMC mutual authentication protocol. All variables are $n-$bit length except s_i which is padded before XORing.

mechanism. Also, in message flow of the CMC protocol each party is involved in ensuring freshness of the session by generating fresh random nonces N_R, N_i that are used in computation of V_i, V_R, K_i. Thus, the CMC protocol provides resistance against replay attacks.

Forward Secrecy: An adversary cannot acquire previous session keys by analyzing the present secrets, since secret key is updated using one-way hash function at the end of each successful session. Hence, the protocol provides forward secrecy.

3 ATTACKS

Before we elaborate on our attacking details, we will firstly give preliminaries that are used in the remaining parts of the study and state a set of reasonable assumptions about the adversary model. Then, we present a tracking attack which demonstrates that the CMC protocol cannot guarantee untraceability which is one of the objectives of the CMC protocol. Last but not least, we show that a more efficient attack is possible as a key recovery attack which leads to disclosure of a tag's secret key in reasonable time. In fact this attack breaches all security claims of the scheme, because whole security is based on secrecy of this key.

3.1 Preliminaries

Definition 1. *Used linear block code of the CMC protocol is defined as $\mathcal{C}(n, k, d_{min})$, where n, k are lengths of the codeword and the message respectively such that 2^k code words form a k-dimensional vector subspace of the n-tuples over $GF(2)$. Also, d_{min} stands for the minimum distance of the linear block code $\mathcal{C}$ defined as $d_{min} \triangleq \min_{v,y}\{d(v, y) : v, y \in \mathcal{C}$ and $v \neq y\}$, where $d(v, y)$ is the hamming distance of the binary codewords v and y.*

Definition 2. *The error correction capability of the code $\mathcal{C}(n, k, d_{min})$ is defined as $t_C \triangleq \lfloor (d_{min} - 1)/2 \rfloor$, i.e. the code can correct up to t_C-bit errors.*

Definition 3. *The error detection capability of the code $\mathcal{C}(n, k, d_{min})$ is defined as $t_D \triangleq d_{min} - 1$, i.e. the code can detect up to t_D-bit errors.*

Definition 4. *A bounded distance decoder is defined as the decoder that corrects all error patterns e such that $w(e) \leq t_C$ and results in a* decoding failure *or a* decoding error *in case of $w(e) > t_C$.*

Note that when a bounded distance decoder detects an uncorrectable error, it declares a *decoding failure*. On the other hand in *decoding error* case, some errors of weight more than t_C may be decoded to a wrong codeword, if there exists a codeword C_y such that $d(e, C_y) \leq t_C$.

Definition 5. *The set of all possible error patterns whose Hamming weight is less than or equal to t_C is denoted by $\mathcal{S}_e$, i.e. $\forall e_i \in \mathcal{S}_e, w(e_i) \leq t_C$.*

Remark 1. *It is obvious that*

$$|\mathcal{S}_e| = \sum_{i=0}^{t_C} \binom{n}{i}. \quad (1)$$

Definition 6. *$\mathcal{S}_s$ denotes the set of syndromes that is produced by $\mathcal{S}_s = \mathcal{S}_e H^T$.*

Remark 2. *For an error pattern e_i, if $e_i H^T \notin \mathcal{S}_s$, a bounded distance decoder gives* decoding failure.

Definition 7. *$\mathcal{M}$ denotes the set of collected M messages from a tag through different sessions such that each of them is computed with the same secret key, i.e. $\mathcal{M} = \{M_1, M_2, \cdots\}$.*

Proposition 1. *Let $M_1 = K_i \oplus C'_x$ and $M_2 = K_i \oplus C'_y$ be two transmitted messages by the tag $\mathcal{T}_i$ with using the same secret key K_i, where $C'_x = C_x \oplus e_x$, $C'_y = C_y \oplus e_y$ are the noisy codewords and C_x and C_y are codewords in the linear block code $\mathcal{C}(n, k, d_{min})$. Then $M_1 \oplus M_2$ will be an error detectable pattern for the $\mathcal{C}(n, k, d_{min})$ code as long as $w(e_x \oplus e_y) \neq 0$.*

Proof: Note that $M_1 \oplus M_2 = C_x \oplus C_y \oplus e_y \oplus e_z$. Since summation of any two codewords always yields to a codeword, e.g. $C_z = C_x \oplus C_y$, we can express $M_1 \oplus M_2 = C_z \oplus e_x \oplus e_y$. Thus, if we show $w(e_x \oplus e_y) \leq t_D$, we prove detection of the error pattern $e_x \oplus e_y$ and we complete the proof. We know that both of $w(e_x)$ and $w(e_y)$ are equal or less than t_C, because they are correctable error patterns. Hence, we can write

$$\begin{array}{rcl} w(e_x) & \leq & \frac{d_{min}-1}{2} \\ w(e_y) & \leq & \frac{d_{min}-1}{2} \end{array} \quad (2)$$

From the triangle inequality, $w(e_x \oplus e_y) \leq w(e_x) \oplus w(e_y)$, so $w(e_x \oplus e_y) \leq d_{min} - 1$. If we replace $d_{min} - 1$ with t_D, we obtain the expression in Definition 3 and it is shown that $e_x \oplus e_y$ is a detectable error pattern for code $\mathcal{C}(n, k, d_{min})$. Note that $w(e_x \oplus e_y) \neq 0$ is necessary, because the syndrome $(M_1 \oplus M_2)H^T = 0$ is obtained otherwise, i.e. error is not detected.

□

Proposition 2. *Let R_x be a random string of n bit length, then the probability that R_x is recognized as a decodable word by a $\mathcal{C}(n, k, d)$ linear block code is*

$$\frac{\sum_{i=0}^{t_C} \binom{n}{i} \times 2^k}{2^n}.$$

Proof: The number of all possible n-bit random binary strings is as $|R_x| = 2^n$. On the other hand, we know that all codewords with error patterns can be decoded as long as hamming weight of the error patterns are equal or less than t_C. Hence, there are $\sum_{i=0}^{t_C} \binom{n}{i} \times 2^k$ decodable erroneous codewords and the probability can be calculated by dividing this value by $|R_x|$, i.e.

$$\frac{\sum_{i=0}^{t_C} \binom{n}{i} \times 2^k}{2^n}. \quad (3)$$

□

3.2 Assumptions

In our attacking strategies, we comply with assumptions of the authors in [2] and not stretch them to make our attacks realizable. According to the security analysis of the CMC protocol compromise of a tag is considered, so it is assumed that the generator and parity check matrixes G and H are available to the adversary. Indeed it is a realistic approach, since every tag keeps the same G and H and they are static. An adversary attains e_R and s_i of a target tag $\mathcal{T}_i$ by just eavesdropping a single session between the reader and the tag and computing the syndrome for C'_R. Notice that acquire of e_R is critically important from view of the adversary, because the target tag always responds correctly to the message $C_x \oplus e_R$ which is transmitted by the adversary, where C_x is an arbitrary codeword. In other words, by using e_R the adversary gains opportunities to initiate communication with the target tag. Furthermore, we suppose that the CMC protocol employs a bounded distance decoder, since all error patterns are predetermined (not probabilistic) and only the error patterns whose Hamming weight less than or equal to t_C can be decoded. Last, the designers of the CMC protocol accepts one day limit as a rational upper bound to recover the secret key for an attacker though they assert key recovery needs several years of time effort for some properly selected codes. Nevertheless, in accordance with the assumptions made our proposed key recovery attack extracts the key in terms of hours for the code $\mathcal{C}(47, 24, 11)$ which is one of the recommended linear block codes.

3.3 Tracking Attack

Roughly speaking, an RFID protocol holds untraceability as long as an adversary has a negligible advantage in distinguishing two selected tags by analyzing their messages. Untraceability of an RFID protocol $\mathcal{P}$ is usually examined by a privacy experiment [33], [34] $\mathbf{Exp}_{\mathcal{P}}^{priv}$ which is consisted of the two phases: Learning Phase and Challenge Phase. In learning phase, the adversary $\mathcal{A}$ is able to initiate communication with the reader or tags. On the other hand in challenge phase, she chooses two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ as the challenge candidates and picks one of these tags randomly, represented as $\mathcal{T}_b^*$ for $b \in \{0, 1\}$. Next, $\mathcal{A}$ may again communicate with this tag and the reader to gather some valuable information and make a guess for whether $b = 0$ or $b = 1$. At the end of this experiment if the adversary has a non-negligible advantage in guessing the selected tag correctly, then we can say that the protocol violates untraceability criteria. The success of $\mathcal{A}$ in guessing b is measured by $\mathcal{A}$'s

advantage in identifying the selected tag compared to a random guess [35]. Hence it can be described by:

$$\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},\mathcal{P}}(\tau) = |\Pr[\hat{b} = b] - \frac{1}{2}|, \tag{4}$$

where τ is the security parameter e.g. ID length of the tag. In this respect, a protocol suffers from traceability if $\mathbf{Adv}^{\mathbf{Exp}}_{\mathcal{A},\mathcal{P}}(\tau) > \varepsilon(\tau)$, where $\varepsilon(\tau)$ is a negligible function in τ.

Before proceeding to the details of the attack, we note that by XORing two of the received messages M_i and M_j the attacker obtains a noisy codeword as expressed in Proposition 1. In this way the attacker can obtain an advantage in distinguishing two tags: One of the tags, as the non-target, sends random values for M_i, M_j i.e. $M_i \oplus M_j$ will be recognized by the attacker as a random message with a high probability. On the other hand, $M_i \oplus M_j$ will produce a decodable pattern with a non-negligible possibility in case of the target tag. Hence, we present a tracking attack that takes advantage of this fact and breaks untraceability of the CMC protocol.

According to the CMC protocol, the reader $\mathcal{R}$ chooses its target tag $\mathcal{T}_i$ and initiates the communication by submitting the message C'_R with a random value. Note that an adversary $\mathcal{A}$ who knows the code $\mathcal{C}(n, k, d_{min})$ can easily reveal the e_R and s_i of $\mathcal{T}_i$ by just computing the syndrome for C'_R. Hence, the adversary can impersonate $\mathcal{R}$ to start a session with $\mathcal{T}_i$ by randomly picking a codeword C_x, adding e_R to it and transmitting the result with a random nonce N_R. Notice that $\mathcal{T}_i$ uses the same secret key K_i until a successful session is realized.

Below we present how $\mathcal{A}$ can realize the privacy experiment and achieve the tracking attack against the CMC protocol. In the learning phase, two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ are selected, where s_0 of the tag $\mathcal{T}_0$ is available to $\mathcal{A}$. Later on, she queries the tag about λ times and collects the M messages of each session as $\mathcal{M} = \{M_1, M_2, \cdots, M_\lambda\}$.

Learning Phase

- *$\mathcal{A}$ selects a pair of distinct tags $\mathcal{T}_0$ and $\mathcal{T}_1$.*
- *$\mathcal{A}$ randomly picks a codeword C_x and computes $C'_x = C_x \oplus e_R$, where $e_R H^T = s_0$ as the assigned syndrome of $\mathcal{T}_0$*
- *$\mathcal{A}$ sends C'_x with a randomly generated value N_R to $\mathcal{T}_0$.*
- *$\mathcal{T}_0$ responses with M_1, V_0, N_0 with using its key K_0.*
- *$\mathcal{A}$ stores M and discards other received messages. Then, she ends the session.*
- *$\mathcal{A}$ repeats the previous steps for λ times and obtains different M values as $\mathcal{M} = M_1, M_2, \cdots, M_\lambda$.*

In the challenge phase of the attack, the adversary only the queries the selected tag once and obtains M^*. Then she XORs M^* with each $M_j \in \mathcal{M}$ and checks whether the result as $\hat{C} = M^* \oplus M_j$ is a decodable word or not. Note that there are two cases for $\hat{C}$: In case of $\mathcal{T}^* = \mathcal{T}_1$ it will likely be a random sequence, because M^* is produced by XOR of K^* and a random value r. Thus, as stated in Proposition 2, with a high probability decoding of $\hat{C}$ will result in a decoding failure in that case. On the other hand if $\mathcal{T}^* = \mathcal{T}_0$, as stated in Proposition 1, $\hat{C}$ will be a noisy codeword due to $M^* \oplus M_j = C^* \oplus e^* \oplus C_j \oplus e_j = C_z \oplus e^* \oplus e_j$ for any $M_j \in \mathcal{M}$, where $C_z = C^* \oplus C_j$. As a result, through trials we expect some of $\hat{C}$ will yield a decodable pattern, i.e. $\hat{C}H^T \in \mathcal{S}_s$. Indeed, Algorithm 3.1 seeks such a case to guess the selected tag is $\mathcal{T}_0$. Hence, according to the result of the Algorithm 3.1 as given below, an adversary may distinguish these two cases and makes her guess with a non-negligible probability.

Challenge Phase

- *$\mathcal{A}$ takes $\mathcal{T}_0$ and $\mathcal{T}_1$ as its challenge candidates.*
- *$\mathcal{A}$ randomly picks a codeword C_y and computes $C'_y = C_y \oplus e_R$.*
- *$\mathcal{A}$ sends C'_y with a randomly generated value N_R to the selected tag $\mathcal{T}^*_b$.*
- *$\mathcal{A}$ keeps response M^* of $\mathcal{T}^*_b$.*
- *$\mathcal{A}$ executes Algorithm 3.1.*
- *If the algorithm returns true, $\mathcal{A}$ guesses $b = 0$ and decides $\mathcal{T}^*_b = \mathcal{T}_0$. Otherwise guesses $b = 1$, i.e. $\mathcal{T}^*_b = \mathcal{T}_1$.*

Algorithm 3.1: ISSUMDECODABLE($M^*, \mathcal{M}, \mathcal{S}_s$)

for $i \leftarrow 1$ **to** λ
do $\begin{cases} \hat{C} \leftarrow M^* \oplus M_i, M_i \in \mathcal{M} \\ \textbf{if } \hat{C}H^T \in \mathcal{S}_s \\ \quad \textbf{then return } (\textbf{ true }) \end{cases}$
return (**false**)

Taking the described attack into consideration one can see that $\mathcal{A}$ may have non-negligible advantage in guessing the identity of a selected tag. In other words, the CMC is no longer considered to provide untraceability feature. Let us now derive the advantage of the adversary:

$$\begin{aligned} \Pr[\hat{b} = b] &= \sum_{\forall a \in \{0,1\}} \Pr[\hat{b} = b | \mathcal{T}^*_b = \mathcal{T}_a] \times \Pr[\mathcal{T}^*_b = \mathcal{T}_a] \\ &= \Pr[\hat{b} = b | \mathcal{T}^*_b = \mathcal{T}_0] \times \Pr[\mathcal{T}^*_b = \mathcal{T}_0] \\ &\quad + \Pr[\hat{b} = b | \mathcal{T}^*_b = \mathcal{T}_1] \times \Pr[\mathcal{T}^*_b = \mathcal{T}_1]. \end{aligned} \tag{5}$$

It is apparent that both of the marginal probabilities $\Pr[\mathcal{T}^*_b = \mathcal{T}_0]$ and $\Pr[\mathcal{T}^*_b = \mathcal{T}_1]$ are $1/2$, since selection of each challenge candidate has equal chance. Moreover $\Pr[\hat{b} = b | \mathcal{T}^*_b = \mathcal{T}_0]$ is equal to probability of the Algorithm 3.1 returns $true$ in λ trials for $\mathcal{T}^*_b = \mathcal{T}_0$. Let p_1 denote the probability that sum of M^* and any $M_j \in \mathcal{M}$ results in a decodable vector, where M^* is the received M message from the target tag $\mathcal{T}^*_b$. It is equivalent to say that $p_1 = \Pr[(M^* \oplus M_j)H^T \in \mathcal{S}_s]$ for any $M_j \in \mathcal{M}$. Then from binomial distribution probability of hitting at least one

decodable result in λ trials will be $1-(1-p_1)^\lambda$. Thus we obtain:

$$\Pr[\hat{b}=b|\mathcal{T}_b^*=\mathcal{T}_0]=1-(1-p_1)^\lambda.$$

Let p_2 denote probability that addition of M^* with an $M_i \in \mathcal{M}$ yields a non-decodable word such that $M^* = K^* \oplus r$ and r is a random value. It is equivalent to say that $p_2=\Pr[(M^*\oplus M_i)H^T \notin \mathcal{S}_s]$, where $M^*=K^*\oplus r$ and r is an $n-$bit length randomly produced binary string. Since $\Pr[\hat{b}=b|\mathcal{T}_b^*=\mathcal{T}_1]$ is indeed probability of $\Pr[(M^*\oplus M_i)H^T \notin \mathcal{S}_s]\ \forall M_i \in \mathcal{M}$, we evaluate this probability as p_2^λ. Hence, we obtain:

$$\Pr[\hat{b}=b|\mathcal{T}_b^*=\mathcal{T}_1]=p_2^\lambda.$$

If we replace these derived values in (5), we obtain:

$$\Pr[\hat{b}=b]=\frac{1-(1-p_1)^\lambda}{2}+\frac{p_2^\lambda}{2} \tag{6}$$

Remark 3. *Notice that by assuming sum of M^* and any $M_i \in \mathcal{M}$ results in another random binary sequence, where $M^* = K^* \oplus r$ and r is a random value, we can obtain a rough estimation for p_2 by using* (3) *as* $p_2 = 1-\frac{\sum_{i=0}^{t_C}\binom{n}{i}\times 2^k}{2^n}$.

Example 1. *In order to show effectiveness of our proposed attack, we mount it against the CMC protocol with $\mathcal{C}(63,39,9)$ which is one of the suggested code sets by the authors. To compute $\Pr[\hat{b}=b]$, we need p_1, p_2 and λ (see Section 4.1 for evaluation of these values), while $t_C=4$, $n=63$ and $k=39$ for this linear block code. From our experimental computations we heuristically obtained $p_1=0.069$ and also $p_2=0.969$. According to these values with choosing $\lambda=20$, we calculate that $\Pr[\hat{b}=b]=0.646$. Hence, $\mathbf{Adv}_{\mathcal{A},\mathcal{S}}^{\mathbf{Exp}}(\tau) \approx 0.15 > \varepsilon(\tau)$, for some negligible function $\varepsilon(\tau)$. When we carry out the described privacy experiment that run under the configuration shown in Table 2 for 100000 times in about 25 minutes, we observed that the described method guesses correctly in 64208 trials, i.e. with a success rate 0.642. Note that it is very close to our expected value and it validates our derivations.*

Remark 4. *As the probability of a random number is recognized as a valid codeword increases, the p_2 value in* (6) *decreases, i.e. it reduces advantage of an adversary in the proposed attack. Hence, we can say that the proposed attack is effective especially against the codes for which* (3) *has small values. A comparison of success probabilities of the adversary for different codewords is depicted in Table 3. Note that the last column represents the probability of decoding a random sequence as a valid codeword and as can be seen the success probability of adversary increases in case of it decreases, i.e. the results confirm this remark.*

3.4 Key Recovery Attack

In this section, we describe a key recovery attack against the CMC protocol such that an adversary can capture the secret key K_i with $\mathcal{O}(\max_{n,k}(|S_e|,2^k))$ time complexity and $\mathcal{O}(|\mathcal{S}_e|)$ data complexity. For example, as opposed the claims of the authors, the key is recovered in terms of hours if the adopted code is $\mathcal{C}(47,24,11)$. In order to reduce attack complexity we apply a divide-and-conquer approach and the attack is consisted of three steps. In the first stage, the adversary collects many M_i messages by querying the target tag several times. Second stage involves control of whether a special M_i exists in $\mathcal{M}$. In the last stage, a brute force search is performed over the result obtained from the previous phase and checks each candidate key with a known solution for verification. We express each stage of the attack in the following parts.

Operating system	Windows 7 Professional SP1 64 bit
CPU	Intel Core i7-3612QM 2.1 GHz
Programming tool	MATLAB R2012A

TABLE 2
The Configuration Used in Algorithms 3.1 and 3.2

Code	Success Probability	λ	Value in (3)
$\mathcal{C}(47,24,11)$	0.504	4	0.206
$\mathcal{C}(63,39,9)$	0.646	20	0.038
$\mathcal{C}(62,31,12)$	0.856	20	0.003

TABLE 3
Success Probabilities of the Tracking Attack for Different Codes

3.4.1 Collecting Samples

For our proposed attack, an adversary needs occurrence of a special state where the picked error pattern by the target tag is a zero sequence in computation of the M message, i.e. $w(e_i)=0$. The probability of catching such a state is $\frac{1}{|\mathcal{S}_e|}$, so if the adversary obtains about $|\mathcal{S}_e|$ different M messages calculated with the same key, then with a high probability collected messages will contain the required case. Consequently, the adversary queries the target tag for $|\mathcal{S}_e|$ times and keeps responded M messages as $\mathcal{M}=\{M_1,M_2,\ldots,M_{|\mathcal{S}_e|}\}$. As predictable, in addition to those collected messages also a single $\{V_i,N_R,N_i\}$ respective tuple is saved to be used in the verification of the candidate key.

3.4.2 Recovering the State

In this stage, adversary assumes that at least one $M_j \in \mathcal{M}$ is in the form of $M_j = C_a \oplus K_i$ for any $C_a \in \mathcal{C}(n,k,d_{min})$. That is an all-zero error vector is involved in generation of this M_j. Let Y stand for this M_j. Then XOR of Y with any $M_x = C_b \oplus e_i \oplus K_i$ in the collected data yields a decodable erroneous codeword, since $\forall M_x \in \mathcal{M}$, $Y \oplus M_x = C_a \oplus K_i \oplus C_b \oplus K_i \oplus e_i = C_d \oplus e_i$ and $w(e_i) \leq t_C$.

The following Algorithm 3.2[1] relies on this fact: It takes set of collected messages $\mathcal{M}$ and the set of syndromes $\mathcal{S}_s$. If any $M_j \in \mathcal{M}$ exists such that $M_j = C_a \oplus K_i$ for some codeword C_a, then XOR of M_j with all other messages in $\mathcal{M}$ will result in a decodable word, i.e. $\forall M_x$, $(M_j \oplus M_x)H^T \in \mathcal{S}_s$ and the algorithm returns M_j.

Algorithm 3.2: FINDCORRECTSTATE($\mathcal{M}, \mathcal{S}_s$)

```
for i ← 1 to |M| − α
do { decodable ← true
     for j ← i + 1 to |M|
     do { Ĉ ← M_i ⊕ M_j
          if Ĉ H^T ∉ S_s
            then decodable ← false
          break
     if decodable
       then return (M_i)
       else return (0)
```

3.4.3 Retrieving the Key

From the previous stage of the attack, the adversary gets the special message $M_j = C_a \oplus K_i$. Note that for this given M_j there are 2^k possible solutions for K_i, since number of codewords is 2^k. In this step, adversary executes Algorithm 3.3 as described below to obtain and verify the key K_i. In each iteration of the algorithm a candidate K_x is extracted from the given M_j, then it is used in calculation of $g(s_i \oplus N_R \oplus h(N_i \oplus K_x))$ to verify the key candidate by checking the result with V_j .

Algorithm 3.3: RECOVERKEY($M_j, \mathcal{C}, V_j, N_R, N_i$)

```
for x ← 1 to 2^k
do { K_x ← M_j ⊕ C_x
     V' ← g(s_i ⊕ N_R ⊕ h(N_i ⊕ K_x))
     if V' = V_j
       then return (K_x)
```

4 ANALYSIS OF THE ATTACKS

4.1 Implementation Details

Before giving details of the implementation, we want to emphasize that implementations in this study are not done with intent to get an optimized code in terms of speed, rather we just want to demonstrate that all proposed attacks are applicable and in particular the key can be retrieved in practical time by mounting the presented key recovery attack. In order to realize the tracking attack and the second phase of the key recovery attack, we have implemented the Algorithms 3.1 and 3.2 in MATLAB with configuration depicted in Table 2. Moreover, the last stage of the key recovery attack as the Algorithm 3.3 is implemented in ANSI C and the iterations have been conducted on the configuration given in Table 4.

Operating system	Ubuntu 14.04 LTS 64 bit
CPU	Intel Core i7-3730 3.40 GHz x 8
Compiler	gcc 4.8.2
Cryptographic tool	OpenSSL 1.0.1f

TABLE 4
The Configuration Used in Algorithm 3.3

For the tracking attack, we firstly need p_1 and p_2 values in order to find an optimal λ. This is crucial because as λ increases from some point the success probability of the adversary given in (6) decreases. To evaluate p_1, p_2 we ran the implementation 1000 times in total. For each attempt a challenge M^* is generated from addition of randomly chosen key, codeword and error vector. Also in each attempt the sequence R^* is produced by summing a random binary sequence with the same key. Then, we construct an $\mathcal{M}$ with a size of 1000 and its elements are generated by XOR'ing the same key and randomly selected codeword-error vector pairs. Next, we count $\#\{M_i \in \mathcal{M} : (M^* \oplus M_i)H^T \in \mathcal{S}_s\}$ and $\#\{M_j \in \mathcal{M} : (R^* \oplus M_j)H^T \notin \mathcal{S}_s\}$ separately for each attempt and obtain p_1, p_2 by averaging these values over number of trials. In test of the attack, we conduct the privacy experiment 100000 times in total and in each trial we randomly choose one tag and launch the Algorithm 3.1 for the evaluated λ. At the end, the rate of correct guesses gives us the success probability of the attack.

Since purpose of each algorithm in the key recovery attack is clearly defined, a few things remained to discuss in this part. One of them is that we have carried the last phase of the attack to another platform to utilize efficiency of ANSI C in exhaustive search and benefit from the crypto library of OpenSSL. Moreover to realize $g(s_i \oplus N_R \oplus h(N_i \oplus K_x))$ of Algorithm 3.3, we execute a hash function instead of the pseudo-random function g, because no specific details are given about it and we assume its computational complexity will not be more than a hash function. Thus, we execute the hash function twice within the parameters in the correct order to verify the candidate key. The authors mention that the hash function is a realizable in constrained RFID tag environment. Although we have not implemented a lightweight hash function in our experiment setup, by taking the cycles per byte (cpb) values presented in studies [36] into account we may figure out performance of a lightweight hash function compared to SHA-256. Indeed, in our literature survey we have not met any comparison of software implementation performances for lightweight hash functions with SHA-256.

1. In Algorithm 3.2 to get a correct result, XOR of the candidate with at least α messages must be satisfied. Otherwise, for example, if $\alpha = 1$ is chosen, then the prior to last item likely be returned as the correct state, since only one item is checked. Therefore, α should be set to realistic values such that it is guaranteed that the probability of a false alarm is ignorable, e.g. in our experiments we set $\alpha = 10$.

We have found benchmarks for SHA-256 in 29.3 cpb on Intel Core2 Duo E8400 in study [36] and it is 95 cpb with running on Intel Core(TM) i7 CPU Q 720 for the lightweight hash function PHOTON-80 [5] which is one of the most efficient hash designs. Note that overall performance (single-threaded + multi-threaded) of both platforms are close to each other, so by considering these values one can say that SHA-256 is about 3 times faster than PHOTON-80. Therefore, we have used SHA-256 as the hash function in our experiments and multiply the measured time cost with 3 to estimate time cost for a lightweight hash function.

4.2 Attack Complexities

In this part, we explain required time cost of each attack and analyze their computation complexities. Firstly, the tracking attack needs λ queries of the target tag in its learning phase and λ trials in execution of the Algorithm 3.1 in the challenge phase. The authors in [2] give estimated response times of a tag for a selected linear block code and it is in order of microseconds. Furthermore through our experiments we have observed that required λ has small values, e.g. λ is 4 and 20 for $\mathcal{C}(47, 24, 11)$ and $\mathcal{C}(63, 39, 9)$ respectively. Note that one may calculate success probability given in (6) within attained p_1 and p_2 values over different λ values and cut at some point that suffices in distinguishing the tags. By considering the code $\mathcal{C}(62, 31, 12)$ for example when $\lambda = 10$ an adversary succeeds with a chance of 0.73, while this reaches its maximum value 0.99 in case of $\lambda = 94$. Hence, she may decide $\lambda = 10$ is enough to launch the tracking attack. Thus, whole total time complexity will be in order of milliseconds and it can be ignorable.

We now address the time complexity of the proposed key recovery attack. In *collecting samples* stage of the attack, an adversary needs $|\mathcal{S}_e|$ queries with tag and storing of $|\mathcal{S}_e|$ samples of M messages. Hence this phase costs $|\mathcal{S}_e| \cdot \mathcal{C}_{time}$ computational time and $|\mathcal{S}_e|$ data samples, where $\mathcal{C}_{time}$ denotes response time of a tag upon receiving the inquiry for the respective linear block code. In second phase of the attack, adversary runs Algorithm 3.2 over the samples set $\mathcal{M}$. Let $E[X]$ represent the expected value of j in Algorithm 3.2, i.e. X is a discrete random variable for number of trials getting the first non-decodable word in addition of any two $M_i, M_j \in \mathcal{M}$. Through our experiments, we realize that only in a few trials a non-decodable result is obtained and the algorithm passes to next candidate. For example $E[X]$ is 2.48, 1.27, 1.08 for the codes $\mathcal{C}(24, 12, 8)$, $\mathcal{C}(47, 24, 11)$ and $\mathcal{C}(63, 39, 9)$ respectively. As can be inferred from these values, in about two attempts algorithm breaks the inner loop and continues with the next candidate. Therefore in this stage for the worst case a total of $E[X] \cdot |\mathcal{S}_e| + |\mathcal{S}_e|$ trials are made and in each attempt one XOR operation + syndrome computation + syndrome check process are done. In syndrome check process whether computed syndrome result resides in $\mathcal{S}_s$ is controlled. Note that the last item $|\mathcal{S}_e|$ in total of trials comes from checking all trials when the candidate is the searched word. Let D denote time cost for these three sub-procedure, then time cost of this phase can be expressed as $D \cdot (E[X] \cdot |\mathcal{S}_e| + |\mathcal{S}_e|)$. For instance we evaluate $D = 3.7ms$ for $\mathcal{C}(47, 24, 11)$. In the last phase of the attack, 2^k possible candidates are searched for a given specific solution. For each candidate we assume that one XOR operation + two hash computation are required. Notice that normally in verification the protocol uses one hash function and one pseudo-random function (PRF). Since a PRF is a computationally complex function, we assume its computational cost is same as a hash function to simplify calculations. Thus, computational time cost of this phase can be expressed as $2F \cdot 2^k$, where F represents a time cost of a computationally complex function such as a hash or a PRF. Time cost for each stage of the attack is summarized in Table 5. Note that complexity of guessing the key is dramatically smaller than the claimed complexities. In contrast to statements of [2] which claims guessing the secret key for $\mathcal{C}(47, 24, 11)$ needs more than one hundred year, in the following example we instantiate the key is extracted in practical time, e.g. about 2 hours.

Phase	Time Cost
Collecting Samples	$\lvert\mathcal{S}_e\rvert \cdot \mathcal{C}_{time}$
Recovering State	$D \cdot (E[X] \cdot \lvert\mathcal{S}_e\rvert + \lvert\mathcal{S}_e\rvert)$
Retrieving Key	$2^k \cdot 2F$

TABLE 5
Time Cost of the Key Recovery Attack

Example 2. *We choose $\mathcal{C}(47, 24, 11)$ linear block code as the example case of our proposed key recovery attack. Note that from* (1) *we get $|S_e| = 2^{20.72}$. For the first phase of the attack we use the given estimated values in [2] to calculate time cost. In this respect response time of a tag for this code is provided as between $166.3 - 1330.2$ μs. In our computations we take mean of this interval as $\mathcal{C}_{time} = 748,25$ μs for time cost of a single query response and obtain $|\mathcal{S}_e| \cdot \mathcal{C}_{time} = 1279$ seconds. Next, we perform the second phase of the attack that run Algorithm 3.2 under the configuration shown in Table 2. The algorithm computes each candidate in about 3.7 milliseconds and we attain the correct state in average 5630 seconds through several trials. In the last step, we make an exhaustive search with size of 2^k for the solution obtained from previous stage. We conduct this procedure by using the crypto library of OpenSSL whose details are shown in Table 4 and assuming used hash function is SHA-256 with running it twice. This process costs 86 seconds in average and to estimate cost of PHOTON-80 we multiply this with 3 and get 258 seconds. As a result, total time cost of our attack now becomes 7167 seconds (depicted in Table 6), i.e. the attack is completed about in two hours.*

Phase	Time in seconds
Collecting Samples	1279
Recovering State	5630
Retrieving Key	258
TOTAL	7167

TABLE 6
Experimentally Time Cost for $\mathcal{C}(47, 24, 11)$

One can see that computational complexity of the proposed attack depends on maximum of $|S_e|$ and $|2^k|$, i.e. attack complexity can be described by $\mathcal{O}(\max_{n,k}(|S_e|, 2^k))$. Comparison of our attack complexities with the claimed values in [2] for different code examples is illustrated in Table 7. As a remark all examined linear block codes are mentioned in [2].

Code	Our Attack	Claim of [2]
$\mathcal{C}(24, 12, 8)$	$\mathcal{O}(2^{12})$	$\mathcal{O}(2^{23.18})$
$\mathcal{C}(47, 24, 11)$	$\mathcal{O}(2^{24})$	$\mathcal{O}(2^{44.72})$
$\mathcal{C}(63, 24, 15)$	$\mathcal{O}(2^{29.23})$	$\mathcal{O}(2^{53.23})$
$\mathcal{C}(63, 39, 9)$	$\mathcal{O}(2^{39})$	$\mathcal{O}(2^{58.28})$
$\mathcal{C}(127, 36, 39)$	$\mathcal{O}(2^{74.26})$	$\mathcal{O}(2^{110.26})$

TABLE 7
Comparison of Computational Complexities in Key Recovery for Different Codes

Remark 5. *Notice that $|S_e|$ is indeed number of correctable error patterns and from coding theory it is well known that $|S_e| \leq 2^{n-k}$. Therefore the real attack complexity will be $\leq \mathcal{O}(\max_{n,k}(2^{n-k}, 2^k))$ which will be very helpful to evaluate a rough estimate of computational complexity for a given linear block code.*

5 Conclusion

In this paper, we have investigated security of the lightweight RFID protocol CMC and presented two attacks namely the tracking and the key recovery attacks. By launching the former we have demonstrated that an adversary may have non-negligible advantage in distinguishing the tags, so it is addressed that the target protocol cannot ensure untraceability property. In addition to this, the latter attack blows up the whole security claims of the protocol and secret key of a selected tag can be extracted in practical time depending on the adopted linear block code. Although use of error correcting codes alleviate load of the reader in computation process and they can be implemented easily at the tag side, linearity characteristic of these codes are exploited in our attack strategy. Indeed apart from examination of RFID security systems, our proposed attack calls attention to the paramount importance of the fact that linear block codes should be carefully deployed in crypto systems, otherwise its advantageous points may weaken the security of the system in favor of an attacker. Finally, we have given implementation details of our attacks and shown that they can be realizable by just using a PC.

References

[1] W. Xie, L. Xie, C. Zhang, Q. Wang, J. Xu, Q. Zhang, and C. Tang, "RFID Seeking: Finding a Lost Tag Rather Than Only Detecting Its Missing," *Journal of Network and Computer Applications*, vol. 42, pp. 135–142, January 2014.

[2] C.-M. Chen, S.-M. Chen, X. Zheng, P.-Y. Chen, and H.-M. Sun, "A Secure RFID Authentication Protocol Adopting Error Correction Code," *The Scientific World Journal*, March 2014.

[3] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "An Efficient Authentication Protocol for RFID Systems Resistant to Active Attacks," in *International Workshop on Security in Ubiquitous Computing Systems – Secubiq 2007*, ser. Lecture Notes in Computer Science, vol. 4809. Taipei, Taiwan: Springer-Verlag, December 2007, pp. 781–794.

[4] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "Quark: A Lightweight Hash," *Journal of Cryptology*, vol. 26, no. 2, pp. 313–339, 2013.

[5] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON Family of Lightweight Hash Functions," in *Advances in Cryptology–CRYPTO 2011*, ser. Lecture Notes in Computer Science, vol. 6841. Springer-Verlag, 2011, pp. 222–239.

[6] 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, December 2007.

[7] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic Approach to "Privacy-Friendly" Tags," in *RFID Privacy Workshop*, MIT, Massachusetts, USA, November 2003.

[8] D. Henrici and P. Müller, "Hash-Based Enhancement of Location Privacy for Radio-Frequency Identification Devices Using Varying Identifiers," in *International Workshop on Pervasive Computing and Communication Security – PerSec 2004*, R. Sandhu and R. Thomas, Eds., IEEE. Orlando, Florida, USA: IEEE Computer Society, March 2004, pp. 149–153.

[9] D. Molnar and D. Wagner, "Privacy and Security in Library RFID: Issues, Practices, and Architectures," in *Conference on Computer and Communications Security – ACM CCS'04*, V. Atluri, B. Pfitzmann, and P. D. McDaniel, Eds., ACM. Washington, DC, USA: ACM Press, October 2004, pp. 210–219.

[10] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "Advances in Ultralightweight Cryptography for Low-cost RFID Tags: Gossamer Protocol," in *Workshop on Information Security Applications – WISA'08*, ser. Lecture Notes in Computer Science, vol. 5379. Jeju Island, Korea: Springer-Verlag, September 2008, pp. 56–68.

[11] B. Alomair, A. Clark, J. Cuellar, and R. Poovendran, "Scalable RFID Systems: a Privacy-Preserving Protocol with Constant-Time Identification," in *the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – DSN'10*, IEEE. Chicago, Illinois, USA: IEEE Computer Society, June 2010.

[12] B. Song and C. J. Mitchell, "Scalable RFID Security Protocols supporting Tag Ownership Transfer," *Computer Communication, Elsevier*, March 2010.

[13] P. Dusart and S. Traoré, "Lightweight Authentication Protocol for Low-Cost RFID Tags," in *Information Security Theory and Practice. Security of Mobile and Cyber-Physical Systems – WISTP 2013*, ser. Lecture Notes in Computer Science, vol. 7886. Heraklion, Greece: Springer-Verlag, May 2013, pp. 129–144.

[14] G. Avoine, "RFID Lounge," http://www.avoine.net/rfid/.

[15] C. C. Tan, B. Sheng, and Q. Li, "Secure and Serverless RFID Authentication and Search Protocols," *Wireless Communications, IEEE Transactions on*, vol. 7, no. 4, pp. 1400–1407, 2008.

[16] M. E. Hoque, F. Rahman, S. I. Ahamed, and J. H. Park, "Enhancing Privacy and Security of RFID System with Serverless Authentication and Search Protocols in Pervasive Environments," *Wireless Personal Communications*, vol. 55, no. 1, pp. 65–79, 2010.

[17] J. Y. Chun, J. Y. Hwang, and D. H. Lee, "RFID Tag Search Protocol Preserving Privacy of Mobile Reader Holders," *IEICE Electronics Express*, vol. 8, no. 2, pp. 50–56, 2011.

[18] Z. Kim, J. Kim, K. Kim, I. Choi, and T. Shon, "Untraceable and Serverless RFID Authentication and Search Protocols," in *Parallel and Distributed Processing with Applications Workshops (ISPAW), 2011 Ninth IEEE International Symposium on*. IEEE, 2011, pp. 278–283.

[19] C.-F. Lee, H.-Y. Chien, and C.-S. Laih, "Server-less RFID Authentication and Searching Protocol with Enhanced Security," *International Journal of Communication Systems*, vol. 25, no. 3, pp. 376–385, 2012.

[20] S. Sundaresan, R. Doss, S. Piramuthu, and W. Zhou, "Secure Tag Search in RFID Systems Using Mobile Readers," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 2, pp. 139–150, 2014.

[21] R. J. McEliece, "A Public Key Cryptosystem Based on Algebraic Coding Theory," *DSN Progress Report*, pp. 42–44, 1978.

[22] N. J. Hopper and M. Blum, "Secure Human Identification Protocols," in *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology – ASIACRYPT'01*, ser. Lecture Notes in Computer Science, vol. 2248. Springer-Verlag, 2001, pp. 52–66.

[23] A. Juels and S. Weis, "Authenticating Pervasive Devices with Human Protocols," in *Advances in Cryptology – CRYPTO'05*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3126, IACR. Santa Barbara, California, USA: Springer, August 2005, pp. 293–308.

[24] J. Bringer, H. Chabanne, and D. Emmanuelle, "HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks," in *IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU 2006*, IEEE. Lyon, France: IEEE Computer Society, June 2006.

[25] S. Piramuthu, "HB and Related Lightweight Authentication Protocols for Secure RFID Tag/Reader Authentication," in *Collaborative Electronic Commerce Technology and Research – CollECTeR 2006*, Basel, Switzerland, June 2006.

[26] J. Munilla and A. Peinado, "HB-MP: A Further Step in The Hb-Family of Lightweight Authentication Protocols," *Computer Networks*, vol. 51, no. 9, pp. 2262–2267, 2007.

[27] Z. Lin and J. S. Song, "An Improvement in HB-Family Lightweight Authentication Protocols for Practical Use of RFID System," *Journal of Advances in Computer Networks*, vol. 1, no. 1, pp. 61–65, January 2013.

[28] J. P. Aumasson, M. Finiasz, W. Meier, and S. Vaudenay, "TCHo: A Hardware-Oriented Trapdoor Cipher," in *Proceedings of the 12th Australasian Conference on Information Security and Privacy – ACISP'07*, ser. Lecture Notes in Computer Science, vol. 4586. Springer, 2007, pp. 184–199.

[29] M. J. Mihaljevic and H. Imai, "An Approach for Stream Ciphers Design Based on Joint Computing over Random and Secret Data," *Computing*, vol. 85, pp. 153–168, June 2009.

[30] O. Kara and I. Erguler, "A New Approach to Keystream Based Cryptosystems," in *SASC–2008. Workshop Record*, 2008, pp. 205–221.

[31] A. Mitrokotsa, M. R. Rieback, and A. S. Tanenbaum, "Classification of RFID Attacks," in *Proceedings of the 2nd International Workshop on RFID Technology – IWRT 2008*, Barcelona, Spain, June 2008.

[32] I. Erguler, E. Anarim, and G. Saldamli, "Unbalanced States Violates RFID Privacy," *Journal of Intelligent Manufacturing*, vol. 25, no. 2, pp. 273–281, 2014.

[33] A. Juels and S. Weis, "Defining Strong Privacy for RFID," in *International Conference on Pervasive Computing and Communications – PerCom 2007*, IEEE. New York City, New York, USA: IEEE Computer Society, March 2007, pp. 342–347.

[34] R. C.-W. Phan, "Cryptanalysis of a New Ultralightweight RFID Authentication Protocol - SASI," *IEEE Transactions on Dependable and Secure Computing*, vol. 99, no. 1, 2008.

[35] I. Coisel and T. Martin, "Untangling RFID Privacy Models," *Journal of Computer Networks and Communications*, vol. 2013, 2013.

[36] E. Fleischmann, C. Forler, and M. Gorski, "Classification of the SHA-3 Candidates," *IACR Cryptology ePrint Archive*, vol. 2008, p. 511, 2008.