Wireless Networks manuscript No.
(will be inserted by the editor)

Low-Cost Authentication Protocol for Passive, Computation Capable RFID Tags

Gul N Khan · Markus Moessner

Received: date / Accepted: date

Abstract Authentication of products and humans is one of the major future applications of RFID technology. None of the recent RFID technology related authentication approaches has been fully convincing. Either these schemes offer a low-level of security or they are vulnerable to Denial-of-Service (DoS) attacks that keep the authentication system from proper functioning. Some schemes raise privacy and security concerns as they reveal confidential information about the RFID tag bearer and allow their world-wide tracking. In this paper, we present a novel cryptographic authentication protocol that fills the security holes imposed by RFID technology. Moreover, it provides significantly lower cost in terms of computational effort and communication than currently proposed protocols such as MAP and YA-TRAP* protocols. We also present the implementation of our cryptographic authentication protocol on a real passive computation capable RFID tag known as WISP. The experimental results show that our protocol has double the rate of successful authentication as comapred to YA-TRAP* and MAP. It also takes 33% less time to authenticate.

Keywords Computation Capable Passive RFID Tag · Cryptographic Authentication Protocol · Low Cost Authentication Protocols · Smart RFID Systems · WISP Tag

Department of Electrical and Computer Engineering, Ryerson University, 350 Victoria Street, Toronto ON M5B2K3, Canada
Tel.: +1 416-979-5000 ext 6084
Fax: +1 416-979-5280
E-mail: gnkhan@ee.ryerson.ca

1 Introduction

Wireless identification systems have been around for about 60 years. The first *identification, friend or foe system* (IFF) was introduced during the Second World War by the Royal Air Force [1]. They equipped planes with secondary radar that amplified the received radar impulse. In this way, it was made possible to distinguish between friendly and enemy planes approaching their airbases. *Radio Frequency IDentification* (RFID) has become a ubiquitous technology in our everyday life and is recognized for its benefits and feared due to privacy and security threats that arise by its use [2]. The term RFID describes a technology that provides contact-free communication between a reader and tag over a radio link. An RFID tag stores a unique ID and possibly additional information about the object to which the tag is attached. As soon as a tag is brought into the reader's proximity, its data can be transferred from the tag to the reader. It is also possible to write data to the tag. Therefore, this system allows an automated identification of objects to which an RFID tag is attached. Moreover, it simplifies data acquisition and management and it is going to replace the traditional barcodes.

The reduction in cost and effort to perform product identification is the most prominent advantage of RFID technology in contrast to barcodes [3]. The technology is also considered to extend the superior identification capabilities of RFID to provide product authentication. The cloning of an RFID tag is one of the problems faced by some RFID applications. RFID technology has a major flaw that inhibits its straight forward extension due to the privacy and security concerns. The wireless channel is highly vulnerable to eavesdropping, which allows a plethora of attacks on an RFID authentication system

as well as on the privacy and security of tag bearers. For example, an eavesdropper reveals the ID of an authentic tag during a cloning attack. This tag can then be cloned by writing its ID to a blank tag. The unique ID can also be used to track the tag bearer leading to a tracking attack. Cryptographic authentication-protocols are promising that provide effective countermeasures to ensure the privacy and security of tag bearers as well as to design a well functioning product authentication system.

In this paper, we present a cryptographic RFID authentication protocol for product authentication. We employ a cipher as the cryptographic core of our protocol as compared to previous authentication approaches that uses hash functions. In contrast to past RFID authentication protocols such as *Yet Another Trivial Authentication Protocol** (YA-TRAP*) that provides reader and tag authentication through timestamps[4], we propose to use a counter at the tag level. The counter is employed to count the number of authentication rounds for which the tag has participated. The counter is considered to behave like a timestamp used by the TRAP Family to provide a weak mutual authentication [5], [4], [6]. However, our protocol does not need to transmit counter values (or timestamps) from the reader to the tag. Therefore, we classify our protocol as a big step forward from the TRAP family protocols such as YA-TRAP [4] and mutual authentication protocol, MAP [6]. Our low-cost authentication protocol, LCAP improves these schemes from the security point of view and makes it resistant to *Denial-of-Service* (DoS) attacks.

LCAP proposal has fewer and deterministic number of computations at the server and tag level. To achieve it, we introduce the idea of key classes. A key class is formed by a set of RFID tags that employ a common pair of encryption keys. The only drawback of our proposal is its slightly higher memory utilization at the tag and server level, which is a reasonable trade-off between security and memory requirements. We demonstrate the implementation of our cryptographic RFID authentication protocol as compared to some recent protocols by using real C1G2 compliant reader and RFID tag such as the Intel WISP [7]. WISP tag belongs to semi-active type of battery less tags with processing capabilities that harvest power to process commands, compute and communicate. WISP has a low power TI MDP430 micro-controller to support communication and computation at the tag level.

This paper is organized as follows. The overall RFID system is introduced in Section 2. In Section 3, we review the past authentication techniques that inspired our protocol proposal. We present our protocol proposal in Section 4. We also analyze the protocol security and compare its performance with some other comparable protocols. Section 5 outlines the implementation of YA-TRAP*, MAP and our proposed protocol utilizing real components such as the Intel WISP tag. Finally, we draw brief conclusions in Section 6.

2 RFID System and Notations

2.1 Operational Environment

The RFID system consists of three entities: tags, reader(s) and a back-end server as shown in Fig. 1. The **tags** are highly resource constraint devices and they are attached to the objects that need to be identified and authenticated. We assume that the tags are passive, which means that they are powered by the RFID reader via RF signals and use backscattering to transmit data. Tags have sufficient memory and computational resources to perform lightweight cipher operations. We also assume that the link between reader and tag is established through the EPCglobal Class 1 Generation 2 (C1G2) Tag protocol [8]. The reader software connects it to a back-end server that stores all the relevant data. The reader to server link is secure and it cannot be attacked by an **adversary**. RFID readers comply with EPCglobal standards and operate within 860-960 MHz range [9]. The entity **reader** is comprised of an RFID reader and an application software that is running on a PC. A **back-end server** stores and maintains the tag database. Moreover, it can perform large computations. An adversary can only attack the reader-tag RF link and the attacking party can be active or passive. In this way, the attacks can range from passive eavesdropping to active tag or reader impersonation.

We further assume a modified batch mode operation. Typically, a reader that employs batch mode to identify and query a selection of tags. Then the reader transmits the tag responses together to the back-end server, which performs authentication and provides more information about the scanned tags. However, systems that comply with EPCglobal standards utilize an EPC (a specialized form of a unique ID) to identify a tag. Therefore, identifying the transmitted EPC is enough to track an RFID tag. In our authentication protocol, a tag transmits only a random number to the reader. This number is used by the reader to address a tag when it wants to authenticate a particular tag. However, another random number is to be generated at the tag level. Moreover, there must not be two tags generating the same random numbers in the readers *field-of-view* (FOV) at the same time. The EPC or tag

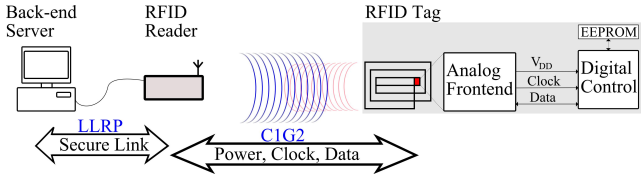

Fig. 1 RFID System

ID is only transmitted encrypted and is sent by a tag only in response to a reader challenge.

2.2 Nomenclature

In this paper we will use the following notations:

$a \oplus b$	XOR operation of a and b
$a \,\|\| \, b$	Conjunction of a and b
$H(a,b)$ *or* $h(a,b)$	Hash function or a cipher over a with the key b
$h^{-1}(a,b)$	Decryption of a with the key b
m_x	A message, exchanged between backend-server, reader and tag
R_x	Random number
T_r	Current timestamp at the reader or backend-server level
T_t	Timestamp of the tag
T_c	Tag's authentication counter
T	Tag
R	Reader
S	Back-end Server
$C1G2$	Class-1 Generation-2 RFID Protocol
DoS	Denial-of-Service
EPC	Electronic Product Code
$LCAP$	Low Cost Authentication Protocol
$LLRP$	Low-level Reader Protocol
MAP	Mutual Authentication Protocol
$PRNG$	Pseudo Random Number Generator

3 Overview and Past Work

Most of the cryptographic authentication protocols employ one or two query-response rounds between reader and tag during the authentication process. The authentication protocols with few authentication rounds have been studied intensively by the scientific community. The hash lock family of authentication protocols was introduced by Weis et al. [10]. The main feature of hash locking schemes is that a tag is either in a locked or in an unlocked state. As long as the tag remains in the locked state, it only responds to the reader's query with a hashed version of its ID called $metaID$. A $metaID$ reveals no information about the real tag ID. A back-end server stores the $metaID$ and the associated key that is used to compute $metaID$. As soon as a reader receives the tag's $metaID$, it can simply query the server for the associated encryption key. The reader hands over the key to the tag that will then transition to an unlocked state and offers its full functionality to the reader.

The original hash lock scheme is vulnerable to almost all the attacks. An attacker can eavesdrop a tag's $metaID$ and key to clone a genuine tag. Moreover, tracking of the tag bearer is possible as the $metaID$ is a unique image of the real ID. However, hash lock schemes can keep the tag's real ID secret. This is important as the ID can contain information about the product or tag bearer. Weis has proposed a randomized version of the hash lock scheme to add some resistance to tracking. In that case, a tag has to be able to compute a hash function and must be equipped with a *pseudo-random number generator* (PRNG). After receiving a reader's query, a tag computes the hash value of its ID concatenated with a random number: $metaID = H(ID||r_t, key)$. In this way, the tag's response seems to be different each time it is queried by a reader and it will not be possible to track a tag. The latest protocol that employs a hash lock scheme was presented by Changqing et al. [11]. Their scheme employs two hash functions and an epoch counter that counts the number of authentication rounds through which a tag has gone. The epoch counter is used to reduce the search space of possible ID-Key pairs as only the pairs of a particular epoch have to be evaluated. A second hash function is applied to generate new keys on hand of the epoch. However, in the case of an electro-magnetically harsh environment or active attacks that can advance the epoch counter, it takes a lot of time to synchronize the reader and tag.

Another popular protocol family belongs to hash chaining schemes and employs two different hash chains on both the tag and reader/server as presented by Ohkubo et al. [12]. The first hash chain is used to create a new secret for every authentication round and the second is used to hide the secret. In terms of privacy and security, this scheme is better than hash locking protocols as it can inhibit tracking attacks successfully. However, they are vulnerable to DoS attacks. An attacker can advance the hash chains of tags or the reader/server and thereby lock the protocol due to de-synchronization of their hash chains. Yong et al. improve on this weakness by employing an additional counter at the tag and at the reader/server level [13]. Their approach can successfully reduce the number of computations at the tag

$Tag \leftarrow Reader : T_r,\ R_r$
$Tag :$
$\delta = T_r - T_t$
if $(\delta \leq 0)$ or $(T_r > T_{max})$ **then**
  $H_{id} = PNRG_i^j$
else
  $T_t = T_r$
  $H_{id} = H(T_i, K_i)$
  $R_t = PNRG_i^{j+1}$
  $H_{auth} = H(R_t, R_r, K_i)$
end if

$Reader \leftarrow Tag : H_{id},\ R_t,\ H_{auth}$

$Server \leftarrow Reader : T_r,\ H_{id},\ R_t,\ R_r,\ H_{auth}$
$Server :$
$s = Lookup(HASHTABLE_{T_r}, H_{id})$
if $s == -1$ **then**
  MSG=TAG-ID-ERROR
else if $(H(R_t, R_r) \neq H_{auth})$ **then**
  MSG=TAG-AUTH-ERROR
else
  MSG=TAG-VALID
end if

$Reader \leftarrow Server$: MSG

Fig. 2 YA-TRAP Methodology

and server level as the hash chain at the server level is only advanced to meet the tag counter. The DoS attacks are still possible, however, their impact is reduced. The main flaw of this scheme is that only one counter at the server takes care of all the tags of an RFID system. The system will run into a self-created DoS attack as tags that have not participated in an authentication round for some time will never be able to meet the counter value at the server level again. Therefore, the hash chain at the server will always remain ahead of the tag's counter whenever a tag lose track of the server's authentication round counter.

The most promising family is the TRAP family that was introduced by Tsudik [4], [5]. It employs timestamps to provide reader and tag authentication. The original scheme, YA-TRAP is explained in Fig. 2 [5]. A tag stores a timestamp of the last authentication round (T_t), a maximum timestamp (T_{max}) and a unique encryption key (K_i). The tag can authenticate a reader by comparing the reader's timestamp (T_r) with its own timestamps, T_t and T_{max}. If a reader sends a timestamp to a tag and the tag replies with random numbers. Then the supplied timestamp is either earlier than the stored one or equal/greater than T_{max}. If the reader's timestamp is within the limits then the tag authenticates the reader, takes over the timestamp and replies with a meaningful message. Their response to a valid reader challenge is composed of three parts: a hash value that allows tag identification, a random number and a hash value for tag authentication. The hash value for tag authentication is computed over the tag's timestamp by using the tag's unique key. At this stage when the timestamps at the server and tag are same, a server can hash its timestamp with all the tag keys and find the key that has produced the hash value H_{id}. The server can identify the tag as the key is uniquely assigned to a particular tag. The second hash value (H_{auth}) that is transmitted by the tag is used for authentication. It contains the random number of the reader (R_r) that was sent to the tag along with the timestamp. This hash value also contains the tags random number (R_t) that is intended to randomize the tag response so that no malicious party can track a tag by sending the same R_r. Challenging the tag with a random number, R_r is commonly used in cryptographic authentication protocols. As the timestamp within H_{id} does not necessarily change, it can be recorded and replayed to the reader. However, the challenge by R_r changes every round and only a tag that knows the right key (key_i) can encode it. The same key is also used for H_{id} to compute the correct H_{auth}. As a server can find the key_i, one simply needs to hash R_r and R_t numbers to confirm a genuine tag.

YA-TRAP has a number of flaws such as its vulnerability to DoS attacks. A malicious party can supply a timestamp to the tag that is far ahead in future. A tag will acknowledge the timestamp and does not respond to a query of an authentic reader as long as the timestamp remains false. YA-TRAP is also susceptible to replay attacks. The reader to tag message can be recorded and replayed to tags that have not participated in the authentication round. The responses of genuine tags to this message can also be recorded. Then these can be replayed (by the attackers) to a reader until the timestamp T_r and random number R_r remains valid. YA-TRAP is computationally intensive and can cause exhaustive database searches. For example, when H_{id} is a random number the server has to try every key and compute n hash functions for n number of tags in the system. The main benefit of the scheme is that the initial reader to tag message is valid for all the tags. Therefore, the communication cost in the reader to server link are very low as T_r and R_r remain same for all the tags.

Later, Tsudik proposed YA-TRAP*, which attempts to improve on DoS attack resistance of YA-TRAP [4]. The scheme introduces an epoch token that is valid for a pre-determined time period. When the time period ex-

pires, a hash chain is used to create a new epoch token. The epoch token is sent along with the reader's timestamp to the tag. The tag advances its own hash chain until it generates the same epoch token. Therefore, it can determine the number of epochs since its last authentication and it will estimate a particular time frame in which the timestamp of the reader should lie. In this way, it authenticates the reader's timestamp and assures that this timestamp is not far ahead in the future. The new scheme improves YA-TRAP substantially, as a tag can validate that a timestamp is within a certain time period. However, an attacker can advance the timestamp to the end of this time period and incapacitate a tag for this time period. Therefore, the protocol is still susceptible to DoS attacks. Moreover, the protocol can introduce high computation at the tag level. For example, when a tag has not participated in an authentication round for a while, it will perform a number of computations to validate the epoch token. Moreover, the communication cost and memory requirements will increase as the epoch token needs to be transferred and stored at the tag.

A number of recent protocols aim on improving YA-TRAP and YA-TRAP*. For example, the schemes of Chatmon et al. and Lei et al. try to improve the DoS attack resistance of YA-TRAP by allowing a second reader-to-tag message [14], [15]. The idea of these two protocols is that a tag always responds with a meaningful message to the initial reader-to-tag message, even when the supplied timestamp is wrong. The second message can then be used by the server to update the tag's timestamp to the actual value. However, two rounds introduce higher communication cost and authentication round time. Rahman et al. presented another approach that uses an aggregate function (XOR) to combine several tag responses and thereby reduce communication between the reader and server[16]. However, when the tag responses are combined by XOR, the server cannot identify a wrong tag response among the other tag responses. Instead of allowing the server to query the reader for all the tag responses, they further introduce authentication tokens. These tokens can be used by the reader to sort out wrong tag responses before all the responses are aggregated. However, the computation of tokens is computationally intensive and has to be performed for each tag before an authentication round can take place. Moreover, transfer of tokens from the server to reader introduces higher cost communication cost.

Recently, a significant improvement in the YA-TRAP family protocols is presented by Moessner and Khan [6]. They have presented a mutual authentication protocol (MAP) that can be easily embedded in the C1G2 protocol standard. MAP improves on YA-TRAP* by overcoming its weakness against DoS attacks. In the case of mutual authentication, both reader and tag challenge each other for authenticating each other. To improve the security features, MAP mainly relies on two key tables that are stored at the tag level. The key tables also aid in avoiding the problems of de-synchronization, which happens in the case of evolving key schedule. MAP also employs the XTEA cipher instead of a one-way hash function [17]. The higher security for MAP is achieved on the expense of higher communication cost and resource utilization due to an additional round used for mutual authentication and the key tables.

Key Class	Tag ID		Tag Counter	Key Pair	
k_c	ID_h	ID_l	$T_{c_{stored}}$	Key_{c_1}	Key_{c_2}
1	0	0	0	$0x01\ldots$	$0x01\ldots$
1	1	0	0	$0x01\ldots$	$0x01\ldots$
1	2	1	0	$0x01\ldots$	$0x01\ldots$
1	3	1	0	$0x01\ldots$	$0x01\ldots$
⋮	⋮	⋮	⋮	⋮	⋮
2	0	1	0	$0x21\ldots$	$0x01\ldots$
2	1	1	0	$0x21\ldots$	$0x01\ldots$
2	2	2	0	$0x21\ldots$	$0x01\ldots$
2	3	2	0	$0x21\ldots$	$0x01\ldots$
⋮	⋮	⋮	⋮	⋮	⋮

unique ID_h in key class 2

Fig. 3 Key Class - Server Database Lookup Table

4 Low Cost Authentication Protocol

The main problem of YA-TRAP family is the timestamp update at the tag level. Although the epoch token can validate the timestamp but a tag cannot assure that a message has originated from a trusted reader. Therefore, it is possible that a tag acknowledges false timestamps and becomes incapacitated. We believe that real mutual authentication is needed to transfer such critical data like the timestamp from an RFID reader to tag. For example, the protocol of Lee et al. provides real mutual authentication [18]. However, a real mutual authentication process needs two messages from the reader to tag and introduces high communication cost. Therefore, we propose a state of the art *Low Cost Authentication Protocol* (LCAP) that uses a counter at the tag in place of writing timestamps to the tag. The counter is incremented during each authentication round and transferred in an encrypted form. As we employ a cipher and not a hash function, the server can reveal the counter value and store it in the database. A tag always responds with the same intrinsic informa-

tion and does not distinguish between the authentic and malicious readers. The malicious parties can attack the tag by querying it and then advance the tag's counter. However, it is impractical (or almost impossible) to increment the counter to overflow and reset. We compare the tag's previous counter value with its new value and if the counter at the tag level has advanced then a tag is assumed to be authentic.

We also introduce key classes to reduce the computation load of tag and server. The key classes also avoid exhaustive database searches . The database at the server associates key classes and tag related data, which are depicted in Fig. 3. The key class is identified by a key class number, k_c. Several tags are uniquely assigned to a key class. All the tags in a class use the same pair of keys (Key_{c1}, Key_{c2}) to encrypt their messages. The tag's ID is divided into an upper and a lower half. The upper part of tag's ID is unique within a key class that will be described later. Moreover, the server database associates the tag's ID with the authentication counter, $T_{c_{stored}}$ at the tag level. As soon as a tag communicates its key class number (k_c) to the server, the server will fetch the right keys and then decrypts the tag's message. In this way, there will be no exhaustive key searches at the server level. Moreover, the number of computations at the server level is deterministic.

4.1 Proposed Protocol Details

Our low cost authentication protocol assumes that each tag can store a counter, two encryption keys and its key class. Moreover, a tag is equipped with a *Pseudo Random Number Generator* (PRNG) and can compute a cipher. Initially, the counter is set to zero. We also assume that the random numbers have half the width of a tag's ID and its authentication counter, while the tag's ID matches the input/output width of the employed cipher. For example, the width of the tag's ID and the authentication counter to be 64 bits and all the other variables have a width of 32 bits. Different steps of the working of our protocol are listed below.

Start Authentication: The protocol is initiated by the reader that sends message, m_1 to the tag as shown in Fig. 4. The message, m_1 contains a random number, R_r that is generated by the reader and valid for each tag that participates in the ongoing authentication round. For the next round, a new challenge will be computed by the reader.

Ciphering at the Tag Level: After receiving m_1, the tag computes a random number R_t and monotonically increments its authentication counter T_c. Then it cover-codes its ID and the authentication counter and enciphers both values with the key pair that is associated to its key class.

$$h_1 = h((R_r||R_t) \oplus ID, Key_{c_1}) \tag{1}$$

As Equation 1 indicates, the tag's ID is cover-coded with the random numbers of tag and the reader by employing the XOR function. This is done in such a way that the reader's random number (R_r) covers the upper part, and the tag's random number (R_t) covers the lower part of the tag's ID. The result of this operations is encrypted to h_1 by using the key, key_{c_1}. The second encryption of h_2 is done using key, key_{c_2} as given in Equation 2.

$$h_2 = h((R_t||R_t) \oplus ID \oplus T_c, Key_{c_2}) \tag{2}$$

Equation 2 contains the authentication counter, which is cover-coded by the tag's random number (R_t) and the tag's ID.

Tag to Reader/Server Communication: After the cipher computation is completed, the tag transmits the encrypted data and its key class to the reader. The reader concatenates its random number (R_r) with the tag message, m_2 (given in Equation 3) before it forwards it to the back-end server in a single reader-to-server message.

$$m_2 = k_c||h_1||h_2 \tag{3}$$

Server Level Decryption and Authentication: To identify and authenticate a tag, the server fetches the key pair using the tag's key class and decrypts h_1 and h_2. Then it can reveal the upper part of the tag ID with an XOR operation of $h^{-1}(h_1, Key_{c_1})$ and $R_r||0h00000000$. As the upper part of the ID is unique within key class k_c, the server can look-up the lower part of the ID and the last value of the tag's authentication counter, $T_{c_{stored}}$. At this point the tag is identified and the server is enabled to reveal the tag's random number from h_1 with another XOR operation. At this stage, the server is also able to extract the tag's authentication counter, T_c by an additional XOR operation with $h^{-1}(h_2, Key_{c_2})$. The tag gets authenticated, when its authentication counter (T_c) is ahead of the stored authentication counter ($T_{c_{stored}}$). If this is the case, the stored authentication counter is also updated.

4.2 Resistance to Various Attacks

The presented authentication protocol offers a high resistance to various attacks as well as it has higher level of security. This is mainly due to its resistance to eavesdropping.

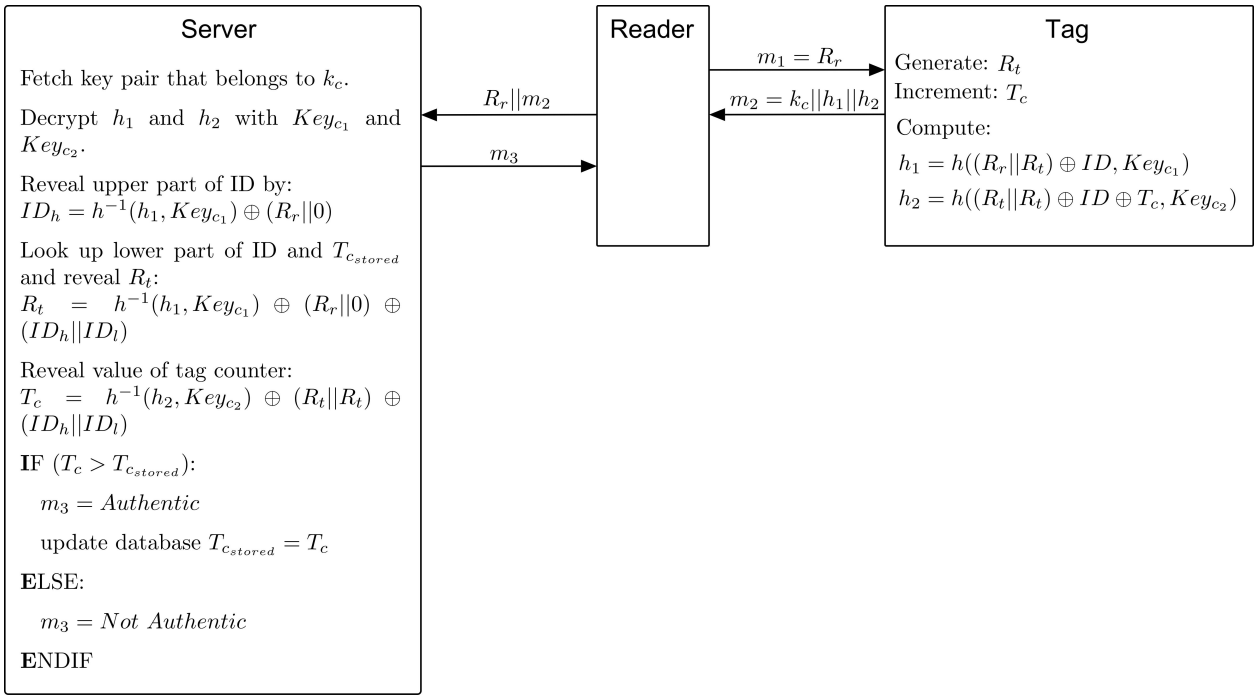

Fig. 4 LCAP: Low Cost Authentication Protocol

Eavesdropping: *Eavesdropping* is the root of all the evils. It is the process where one listens in the wireless link between a reader and the tag. Generally, eavesdropping is undertaken by a malicious party to reveal the confidential data about the tags and their bearers. As we have to rely on hardware that complies with the C1G2 protocol that is wrapped around our protocol, it is impossible to introduce technical measures, e.g., frequency hopping to harden this process for malicious parties. Our LCAP proposal relies on strong ciphers to protect the content of confidential messages that can be eavesdropped. In this way, it is possible to allow any party to hear and record messages. Trusted entities can simply decrypt and read the message content. While a malicious party needs to solve an NP-complete problem to break the cipher for getting hold of confidential information [19].

Cloning Attacks and Forward/Backward Security: Resistance to eavesdropping makes our protocol resistant to *cloning attacks* and provides *forward and backward security*. In the case of a cloning attack, a counterfeiting party tries to clone a genuine tag by writing the eavesdropped tag data on a blank tag. This is not possible in the case of LCAP as the tag's information always remains a secret. Furthermore, the protocol does not reveal much information in plaintext about past or future authentication rounds. The only parameter that remains same for all the authentication rounds is the key class.

Tracking Attacks: The same key class for all the authentication rounds could allow *tracking attacks*. For example, when only one tag is associated to a key class, the key class identifier k_c can become a unique fingerprint of the tag. Therefore, it is necessary that several tags are assigned to a key class. Moreover, the key class should be arbitrarily assigned to tags. Therefore, it is possible to avoid tracking attacks based on the association of products or items (passports, library cards, etc.) with different key classes.

Replay Attacks: *Replay attacks* can happen due to the random numbers generated by the reader and tag and then tag employ these numbers to compute ciphers h_1 and h_2. Replaying message m_1 to a tag will trigger an authentic response from the tag for the next authentication round. Moreover, the tag's authentication counter will also be incremented but this does not affect the tag or the authentication process. The response of a genuine tag to message m_1 can again be recorded and replayed to the reader. However, the reader's random number will change in each authentication round. Therefore, the replayed tag response will not be accepted by the server as the encrypted R_r will not match the recent R_r generated by the reader. Replay attacks can be further blocked by employing high quality random number generators at the reader and tag level that are not

predictable. Moreover, all the random numbers should have a sufficient width (≥ 32 *bits*) so that the recording of all possible tag responses and reader queries will not be viable (or impossible) in practice.

DoS Attacks: Another main advantage of our protocol is its resistance to *DoS attacks*. A malicious party can advance the tag's authentication counter T_c by querying a tag. It is possible for the malicious party to launch a DoS attack if the tag's counter is incremented excessively causing it to overflow or reset. In LCAP, we employ a large size tag counter with a width of at least 32-bits (i.e., ≥ 32 *bit*). A simple increment of the tag counter, T_c will not have any affect (in terms of DoS attacks) as the authenticity of a tag is confirmed when the new value of the counter is greater than its initial value (such as zero).

4.3 Security Analysis

There are three factors that influence the security of our protocol as follows.

- The cipher have to be chosen thoughtfully by taking into account the application. It has to be strong because revealing the keys of one key class can compromise several tags. However, the cipher must be resource preserving and feasible on resource constraint platforms like RFID tags.
- There will be a similar argument for the employed random number generator (PRNG). A low quality PRNG (in terms of its output distribution and length) will compromise the proposed protocol. This is due to the fact that a poor PRNG entity will enable malicious parties to predict the reader's challenges and record the responses of unique tags to these queries. Therefore, it is necessary to employ a high quality PRNG whose implementation is feasible.
- The third factor is the data width of the tag's authentication counter T_c. It must be large enough so that it is not possible for an attacker to increment T_c excessively beyond its range.

As the above three factors affect each other. The output width of the cipher is b bits where the size of random numbers, tag's ID and its authentication counter can be of different widths. We will use these parameters in the comparative implementation of our protocol to show the basic feature and requirements of our protocol. The protocol proposal is intended to be carefully tuned when put into practice. For example, employing random numbers, R_r and R_t in h_1 and h_2 is necessary to avoid replay attacks. However, one can think of a more flexible data alignment to allow a greater data width for the authentication counter T_c. As the server needs only some proof that m_2 is in response to message m_1 and not replayed, we could use part of R_t (e.g. b/4 bits) and increase the width of the authentication counter. In this way, a DoS attack becomes less practicable as it takes significantly more queries and time to make the T_c to overflow. Another security feature would be to include an error detecting/correcting code within h_1 and h_2. In this way, one can defeat the attacker that record a tag response by simply flipping a bit of h_2 and hope that the attacker will increment the tag's authentication counter, which would lead to authentication as long as R_r is valid.

It has to be noted that from the security point of view, our authentication protocol is secure when the above mentioned three factors are considered. It can further be enhanced by adapting it to the intended application and including error correction. Moreover, as LCAP provides tag identification and it can be integrated in higher level systems that employ location based authentication. The protocol can also provide valuable data for intrusion detection systems that protect the authentication system where the difference between the tag's authentication counter and the stored authentication counter is a direct measure for the number of attempted attacks on the system.

4.4 Cost Analysis

We compare our proposed protocol mainly to YA-TRAP, YA-TRAP* [4], [5] and the most recent mutual-authentication protocol (MAP) [6]. These protocols are still the most widely used authentication protocols of the TRAP family. The cost of an authentication protocol can be represented by three factors:

- **Computational Effort:** The number of times the hash or cipher computation takes place at the tag, reader and server level.
- **Memory Requirements:** The amount of storage required for IDs, timestamps, counters, random numbers and other parameters at the tag, reader and server level.
- **Communication Cost:** The amount of bits transferred between the three entities of tag, reader and server.

The required computational resources for YA-TRAP, YA-TRAP*, MAP and our protocol are given in Table 1. For YA-TRAP and YA-TRAP*, the number of computations at the tag level assumes a case in which the timestamp is found to be valid. In the case of YA-TRAP, one has to perform two hash computations for

Table 1 Computational Effort

Protocol	Tag # Hash/Cipher Computations	Tag # rand()	Server # Hash/Cipher Computations
YA-TRAP	2	1	$n+1$
YA-TRAP*	$2+X$	3	$n+2$
MAP	3	2	$n+2$
LCAP (Proposed)	2	1	2

Table 2 Storage Requirements

Protocol	Database Storage	Tag Storage
YA-TRAP	$3bn$	$4b$
YA-TRAP*	$3bn$	$5b$
MAP	$1.5bn + 2bnK$	$1.5b + 2bK$
LCAP (Proposed)	$6.5bn$	$6.5b$

H_{id} and H_{auth}. Furthermore, the random number R_t is to be computed by the tag. For YA-TRAP*, a tag needs at least one more hash computation to evaluate the epoch token. However, as it cannot be assured that a tag participates in each epoch, the number of computations can be significantly higher. This uncertainty is expressed by the factor, X in Table 1.

MAP protocol employs two enciphering and one deciphering operation and has a slightly higher tag level computation as compared to our LCAP approach. Independent from the reader's initial message to the tag, LCAP always employs two cipher computations and requires one random number generation at the tag level. We only consider cipher/hash computations by server, because it is believed that the random number generation is not a computationally intensive operation for the server. Moreover, the depicted hash/cipher computations are the worst case of maximum number of computations that can occur when a single tag is authenticated. In addition to DoS attacks, another problem of YA-TRAP and YA-TRAP* is the key search for H_{id}. This search can become exhaustive and require n computations for n number of tags in the system. We have bypassed this problem by employing key classes in the LCAP scheme, which allows the server to find the encryption keys immediately. Therefore, LCAP approach presented in this paper always need two computations at the server level.

The memory required for the same four protocols is depicted in Table 2. Again a system with n tags are considered and the other parameters are defined as follows.

Table 3 Communication Cost [bit]

Protocol	$R \rightarrow T$	$R \leftarrow T$	$R \rightarrow S$	$R \leftarrow S$
YA-TRAP	$1.5b$	$2.5bn$	$b + 2.5bn$	n
YA-TRAP*	$2.5b$	$2.5bn$	$b + 2.5bn$	$bn + n$
MAP	$bn + 0.5b$	$2.5bn$	$0.5b + 2.5bn$	$bn + n$
LCAP (Proposed)	$0.5b$	$2.5bn$	$0.5b + 2.5bn$	n

- Keys have a length of $2b$ bits.
- A cipher has an input/output width of b bits.
- The timestamps and the authentication token have a width of b bits. The authentication counter T_c is also b bits wide.
- Tag IDs have a width of b bits.
- Random numbers (R_r and R_t) and key class (k_c) have a width of $b/2$ bits.

In the case of YA-TRAP, the tag stores key, recent reader timestamp and a maximum timestamp. Additionally, YA-TRAP* stores the authentication token. This leads to a tag level memory utilization of $4b$ bits for YA-TRAP and $5b$ bits for YA-TRAP*. For both protocols, a server needs to store all the keys, recent timestamp and if necessary the authentication token. In the case of MAP, the storage requirement for the tag and database is much higher as they need to store two key tables having a size of 0.5K each. Our proposed protocol storage at the database and tag level stores the tag ID, two encryption keys, key class and authentication counter that amounts to $6.5b$ bits.

Table 3 provides the communication cost for the same protocols for a scenario where a reader authenticates n tags. This means that for YA-TRAP, the reader sends a timestamp and a random number to the tag, which results in a reader to tag traffic of $1.5b$ bits. For YA-TRAP*, an additional authentication token is also sent to the tag and this figure is increased to $2.5b$ bits. In the case of MAP, two messages are sent from the reader to tag. The first message is common message to all the tags, which is a random number of $b/2$ bits. The second message is for each individual tag of $b/2$ bits each. The overall reader-to-tag communication for MAP will be $(0.5b+bn)$ bits. The other communication values are calculated in the same way, whose details can be found elsewhere [20]. In contrast, our proposed LCAP scheme reduces the reader-to-tag communication to $0.5b$ bits as only one random number, R_r is sent to the tag. The tag-to-reader communication cost for all these protocols remain same. However, server-to-reader communication for LCAP is very low as server only need to send n bits for all the tags whether they are authentic or not. The n bits is comparable to YA-TRAP protocol, which is

not secure against DoS attacks. In conclusion, it has to be noted that our protocol presented here:

- provides a higher level of security
- offers lower communication cost
- is competitive regarding the computational load at the tag level
- employs significantly less computations at the server level

The only drawback is the larger memory consumption at the tag and the server level. We consider this as a reasonable trade-off as our LCAP scheme offers a much higher security.

5 Implementation Details

We present the implementation of our cryptographic authentication protocol, LCAP on a computationally capable RFID tag and a customary RFID reader. Our protocol proposal and two other TRAP family protocols MAP and YA-TRAP* are implemented in a typical RFID environment, which is depicted in Fig. 5. The system consists of a WISP tag [7], a C1G2 compliant RFID reader, and a PC computer that executes reader and back-end server applications. The server application software provides an interface to a MySQL database that stores a tag population of various sizes. It is connected to the reader application software using a customized protocol over TCP. The protocol flow is controlled by the reader application, which can parameterize and command the RFID reader through the Low-Level Reader Protocol (LLRP) [21]. We describe the implementation on the three system entities as follows.

- We employ a passive, computation capable, **RFID tags**. The Wireless Identification and Sensing Platform (WISP) is the first programmable, passive RFID tag that can perform sensing and extensive computations. It was introduced by Intel Research Labs Seattle [22]. The major elements of the architecture are the analog front-end and a TI 16-bit microcontroller (MSP430F2132) with 8K ROM and 0.5K RAM. Furthermore, it carries an external ultra-low power I2C EEPROM with a size of 8K. The microcontroller executes the firmware supplied by Intel, implementing the C1G2 protocol. The hardware and firmware design is described in detail by Yeager et al. [7]. The computational capabilities of WISP are further explored by Chae et al. [23].
- The **reader** is a combination of software application that runs on a PC and a customary RFID reader. The reader software is part of our implementation as it realizes all the protocols being investigated and controls the RFID reader.
- The **server** is composed of a MySQL database that stores the tag data and an interface application, which establishes the connection with the reader software application.

Table 4 Characteristics of Various Ciphers

Cipher	Block Length (bits)	No. of Rounds
TEA	64	32
XTEA	64	32
RC5-32	64	12
RC6-32	128	20
AES-128	128	10

5.1 Selection of a Suitable Cipher

We need a suitable cipher to be computed at the tag and the reader/server level. As outlined earlier, the security of our LCAP proposal relies on a secure cryptographic function. RFID tags are resource constraint platforms and they do not allow the implementation of strong ciphers. Therefore, we have implemented five ciphers, TEA, XTEA, RC5, RC6 and AES, [24],[17],[25],[26],[27] in C and Assembly language for the WISP tag. The adaptable ciphers are designed to meet the properties of XTEA and TEA in terms of data width and key length. For a fair comparison, the data width and key length is selected as 128-bits. The block length and the number of rounds selected for these ciphers are summarized in Table 4.

For each cipher we have evaluated the execution time for key expansion, encryption and decryption. As the runtime of RC5 and RC6 depends on the input data and the key, we perform the experiment 500 times and use random data and keys. The execution time measurement employs a hardware timer/counter module, available at the tag level that has one μs resolution. Moreover, we evaluate tag level memory consumption by examining the compiler/linker output. In our experimental setup, the WISP tag is attached to the debugger (MSP-FET430UIF) that is controlled by IAR *Embedded Workbench* (EW) as shown in Figure 6 . As code design is done with IAR EW, the applications are also debugged by the same tool.

Figure 7 provides the execution time for each cipher implementation using C and assembly language. The memory requirements for each cipher was determined and we found that TEA and XTEA requires the minimum amount of program memory (in the range of

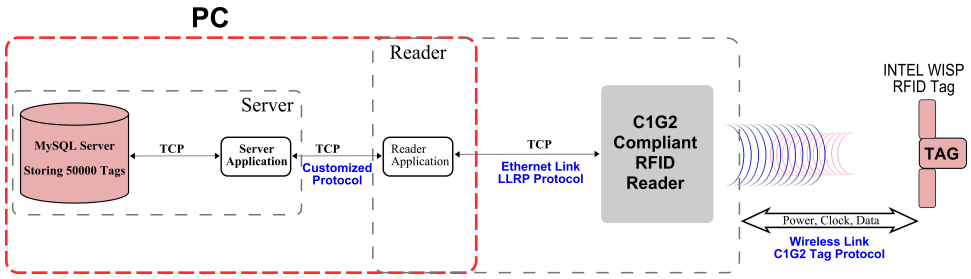

Fig. 5 Experimental RFID Environment

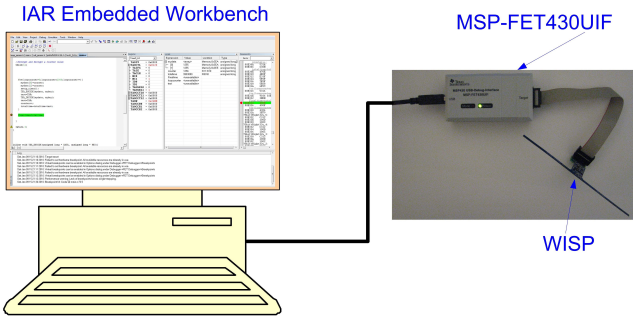

Fig. 6 Experimental Setup

1.1K) as compared to RC5, RC6 and AES-128 ciphers that require 1.7K to 2.7K of program memory. TEA and XTEA are the most resource preserving ciphers. However, it has to be noted that they are significantly weaker than RC5, RC6 or AES. TEA suffers from weak keys, e.g., three different keys can produce the same output for a given input value. Therefore, we have employed XTEA as the cryptographic core for our protocol implementation. XTEA is also employed to generate the epoch token of YA-TRAP*. Contrary to the original XTEA implementation, we have used an 18 round XTEA variant. In this way, we can save energy at the tag level and still defeat the most effective attack on XTEA [28].

5.2 Protocol Implementation

The LCAP implementation involves both the WISP tag and reader/server level coding of the protocol.

Tag Level Protocol Realization

The existing WISP firmware needs to be extended before we can implement our authentication protocol. The WISP firmware was built as a state machine implementing C1G2 protocol to communicate with a compliant RFID reader. The rudimentary implementation provided by Intel has several restrictions. Most states and their corresponding commands defined in the C1G2 specification have not been fully implemented. The WISP tag firmware supports the C1G2 read command to read data from a tag. In response to the read command, the tag returns a preset sequence depending on the compile time options. Writing data to the tag is not possible as the firmware does not provide a write command. Moreover, the tag neither performs error correction nor utilizes any of its external memory resources. In fact, the WISP tag firmware, RFID reader and LLRP library only allows the usage of a single tag and it is not possible to transfer data or to utilize the full resources and capabilities of the WISP tag. The firmware has to be extended substantially to allow any data exchange between the reader and tag. It is also necessary to implement functions that enable the utilization of external EEPROM as the timestamps, epoch tokens and counters need a non-volatile storage.

The authentication protocols utilize BlockWrite command to handle the writing of data on the tag. In this way, the reader's data can be processed immediately and the response to the following Read command can already be placed in RAM. We have implemented the C1G2 BlockWrite command to transfer data to the tag. It provides the following advantages over the standard Write command:

- The command can be directly integrated in the existing firmware without adding any more tag states and C1G2 protocol steps.
- RFID reader does not expect an immediate tag response to a BlockWrite command. A tag has 20ms time to fetch data and perform computations, before it has to acknowledge the command. The RFID reader supplies power to the tag during this time period. In this way, we can assure that the tag will not run short of power and has enough time to perform its computations. Notably, the measurements shown in Table 5 are gathered with a WISP running at 1MHz. The actual protocols execute at 3MHz, which leads to a runtime reduction from around 10ms to 3.33ms. Therefore, 20ms time is sufficient to access the EEPROM to perform some encryptions and to generate a random number.

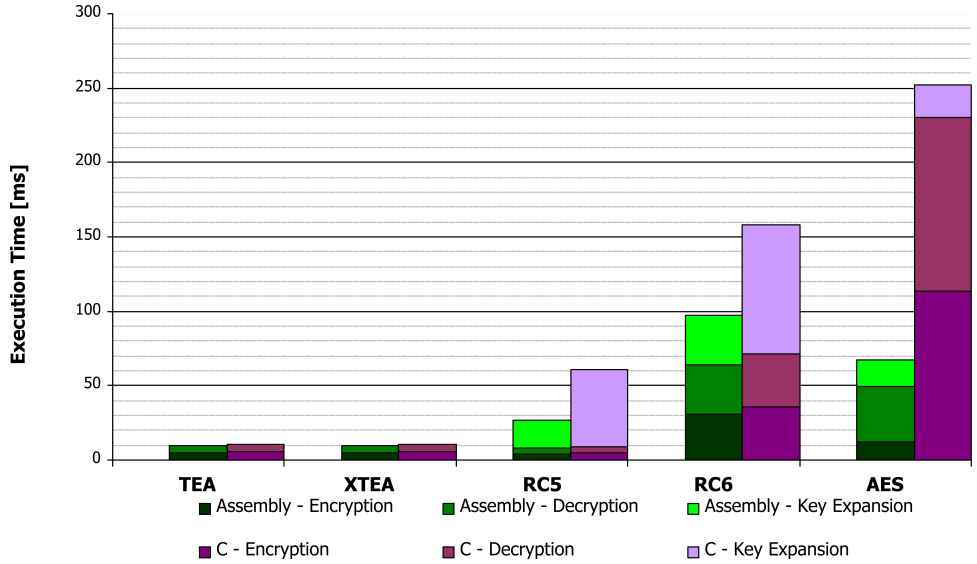

Fig. 7 Execution Time of Ciphers on WISP

The Read command transfers 16-bits of data, due to the limitations of LTKCPP library on the reader side. Our tag level implementation maps the addresses received by a read command directly to the array that holds data at the WISP tag. However, it is observed that all tag addresses cannot be read by the RFID reader due to various factors such as the distance between the reader and tag, and which side of the tag is facing the antenna. As it is impossible to read the tag according to the specification, we decided to read address zero that is often successful. Internally, the read is implemented by incrementing a counter to access the array that contains the information that will be read. The counter is set to zero at the beginning of each authentication round when the first data word is transferred to the tag. The counter turns over to zero when it reaches ten, which is the maximum number of words to be transferred.

Reader/Server Level Protocol Realization

The reader and server software modules are implemented in the same way by employing an object oriented (C++) design. It is possible to realize an implementations that are easy to understand and extend. The main difference between them is the entity that they interface i.e. either an RFID reader or a MySQL database. Moreover, the reader software controls the protocol flow, while the server software has to perform most of the computations. There is no need of any complex computation to be performed at the reader level that would allow the authentication protocol integration directly with the RFID reader. Our implementation of MAP and YA-TRAP* differ from the proposed LCAP protocol as a reader has to query the server in advance to receive the timestamp and the authentication token. The RFID reader can do this over a customized byte oriented protocol over TCP, which we have employed between the server and reader.

5.3 Experimental Results

The protocols are tested assuming a tag population of 50,000 tags. Each test consists of 100 authentication rounds and is repeated by increasing reader-to-tag distance. The reader-to-tag distance is increased until more than 5% of the authentication rounds are successful. After each test, the tag and server database are reset so that one test cannot affect the other. We measure the time needed to transfer data between the reader and tag, number of bytes transferred and the authentication round time.

It is observed that the proposed protocol, LCAP allows about 50% larger reader-to-tag distance than the MAP and YA-TRAP* protocols as the plots of Fig. 8 demonstrates. These results indicate that this is due to the critical power situation on the tag before a tag is read. In the case of LCAP, these plots indicate that writing a tag is successful even at larger distances. It happens as the time to write the tag is significantly increased. All the computations are performed correctly before the response to a BlockWrite command is sent back to the reader. It is also observed that the time needed for writing the tag is linear with the number of bytes to be written.

The time spent for reading a tag depends only on the number of bytes read from a maximum distance of 50 cm. At larger distances, the number of failed attempts

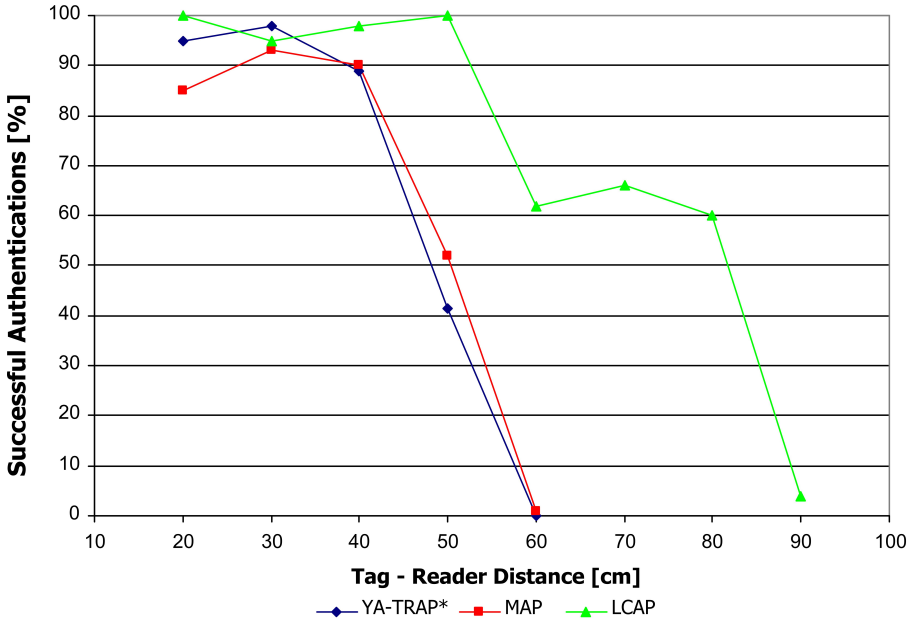

Fig. 8 Successful Authentication Rounds vs. Distance

to read will increase and additional time is required to read. The implementation of the Read command at the tag and the reader application level is similar for both MAP and YA-TRAP* protocols. The reader application tries to read one word from the tag repeatedly (up to 20 times). There is no need to read repeatedly after a read is successful. This is done due to the tag-level counter that is assumed to be vulnerable due to desynchronization. The data provided by Intel indicates that an inventory operation (the tag simply backscatters the EPC) does not even work in 40% of all the cases at a reader to tag distance of 90 cm or more [22]. However, according to the power calculations this has to work with a reader, which is configured at 30dBm signal strength (for a maximum of reader-to-tag distance of 2.9 meters). It is impossible to determine whether an RFID reader or tag is responsible for it.

The implementations for all the three protocols reveal that the only difference between various protocols can be the power situation on the tag before a read occurs. This is mainly determined by the preceding BlockWrite command. Fewer operations are performed during the BlockWrite command and more energy is stored within the tag-level storage capacitor. If a tag drops out of power, it cannot stay in the RAM retention mode and loses its memory contents (or the tag state). Therefore, it will take longer time for a tag to reply a read command as the tag has to be brought back in the reply state. This causes the read operation of an RFID reader to fail. In the case of a successful read, it is still possible that the memory content may have been disturbed.

The main advantage of LCAP is that it employs lower number of operations than YA-TRAP* and the mutual authentication protocol (MAP). Another reason of energy saving is the simpler logic at the tag level. The value received by a tag from the reader has no effect on the number of computations by the tag. The LCAP employs only two cipher operations at the tag level, a counter increment and a random number generation. There is no need to control the operations through if/else-statements to assure that the correct data is encrypted or to validate a timestamp. It is also not necessary to validate the reader's random number by an additional hash chain. It does not matter if the challenge comes from an authentic or a malicious reader. Actually a tag does not need to authenticate the reader as reader only sends a random number to the tag and does not hold any secret. Furthermore, it is sufficient to read/write only a timestamp from/to the EEPROM, while other approaches need to access the EEPROM for epoch tokens. A higher authentication range given in Fig. 8 shows that this simplicity provides a big advantage for LCAP when implemented on WISP like tags. The results of Fig. 8 clearly indicate that LCAP authenticates successfully for a maximum tag to reader distance of 90cm, which is two times of the maximum tag to reader distance for YA-TRAP* and MAP protocols.

This results in another advantage of our protocol i.e. a lower average authentication round time. As compared to MAP and YA-TRAP*, this time is reduced by approximately 200ms even when only 60% of all the rounds are successful as depicted in Figures 8 and 9. One of the reasons of this reduction is due to the lower amount of data sent to the tag. The time reduction also happens due to lower number of computations required by the server. Irrespective of the number of tags and the database size, there is no key search involved in our LCAP proposal and the two deciphering operations at the server are enough to authenticate a tag. There are two hypothetical cases when YA-TRAP* can meet this figure of merit. These cases happen when either the database contains only one tag or the first tag in the database that matches the search criteria. The high round time for our LCAP scheme at 90 cm is only due to higher read time. In Figure 9, we present an average time for an authentication round as the authentication time varies due to an inherit property of the WISP tag used in our experiments. WISP is a semi-active tag and sometimes it needs more time to harvest enough power to compute the parameters for an authentication task. The variation in the authentication round time is insignificant from the performance point of view of an authentication protocol. It is noticed that the variation in authentication time is similar for all the protocols implemented for WISP tags including our LCAP and the past protocols of MAP and YA-TRAP*.

Table 5 provides the communication cost per authentication round for MAP, YA-TRAP* and our LCAP proposal. Due to the overhead of the customary protocol between reader and server, the figures for the reader-server link are slightly higher than it is presented earlier in Table 3 but the communication values for the reader-tag link remain the same. As compared to YA-TRAP* and MAP, our proposed protocol successfully reduces the reader-to-tag communication cost. This has become possible for LCAP as only a random number is transferred to the tag. Moreover, the server-to-reader communication cost is significantly reduced. This is due to the fact that there is only one message from server to reader that consists of a header and result of one byte each. The communication cost will be different when more tags are in the reader's field-of-view but the trend will remain same.

The simplicity of our LCAP proposal is also reflected by the resource consumption at the tag level. As Table 6 indicates, it consumes less ROM, RAM and EEPROM at the tag level. It is in contrast to the theoretic calculation given in Table 2. This is due to the fact that the two keys that are employed by our protocol are stored in the program ROM. Therefore, the EEPROM only stores the authentication counter and not a timestamp and the epoch token, which is required for YA-TRAP*. It is further not surprising that our protocol utilizes less program ROM as the implemented protocol logic is also simpler.

Table 5 Communication Cost [Byte]

Protocol	$R \rightarrow T$	$R \leftarrow T$	$R \rightarrow S$	$R \leftarrow S$
YA-TRAP*	16	20	25	15
MAP	12	20	23	16
LCAP	4	20	25	2
Proposed				

Table 6 Resource Consumption of the investigated Protocols (gathered with IAR EW)

Protocol	ROM [Byte]	RAM [Byte]	EEPROM [Byte]
YA-TRAP*	5366	477	16
MAP	5158	487	324
LCAP	4574	481	8
Proposed			

6 Conclusions

In this paper, we have first presented the benefits of product authentication using RFID. We have reviewed some of the recent authentication protocols and discussed their benefits and drawbacks. We proposed a novel low cost cryptographic RFID authentication protocol (LCAP), which can be integrated into the ubiquitous C1G2 protocol. We have also analyzed its security properties and compared its cost with some other comparable authentication protocols. Our investigation shows that the new LCAP proposal is superior in terms of security, computational effort and communication cost when compared with MAP and YA-TRAP* authentication. The only drawback of our protocol is the slightly higher memory consumption at the tag and server level.

Moreover, we also present a successful implementation of the proposed LCAP scheme, MAP and YA-TRAP* on a passive, computationally capable RFID tag. The experiments carried out by using the real-world components to verify the superior nature of LCAP as compared to some recent comparable approaches. It provides a high level of security, as well as a 50% higher reader-to-tag distance and a significantly lower authentication round time.

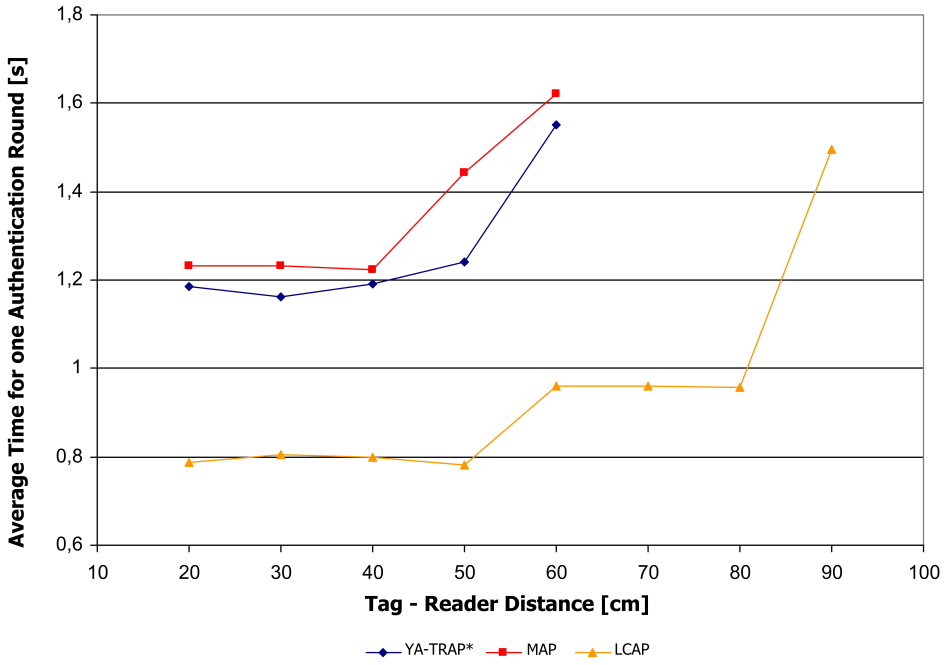

Fig. 9 Average Authentication Round Time

Acknowledgments

The authors acknowledge the financial support of NSERC Canada. Furthermore, we would like to thank INTEL Research Seattle for providing WISP tags and CMC microsystems for the other equipment, which allowed us to implement and test our protocol.

References

1. Lehtonen M., Staake T., Michahelles F., and Fleisch E. (2006). From identification to authentication - a review of RFID product authentication techniques. In *Printed handout of Workshop on RFID Security (RFIDSec-06)*.
2. Juels A. (2006). RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications, 24*(2), pp. 381–394.
3. Lehtonen M., Michahelles F., and Fleisch E. (2007). Trust and security in RFID-based product authentication systems. *IEEE Systems Journal, 1*(2), pp. 129–144.
4. Tsudik G. (2007). A family of dunces: trivial RFID identification and authentication protocols. In *Proceedings 7th International Conference on Privacy Enhancing Technologies*, Springer-Verlag, pp. 45–61.
5. Tsudik G. (2006). YA-TRAP: Yet Another Trivial RFID Authentication Protocol. In *Proceedings 4th annual IEEE international Conference Pervasive Computing and Communications Workshops*, Washington DC, pp. 640–643.
6. Moessner M., and Khan G. (2012). Secure authentication scheme for passive C1G2 RFID tags. *Computer Networks Journal, 56*(1), pp. 273–286.
7. Yaeger D. Y., Sample A. P., Powledge P. S., Mamishev A.V., and Smith J. R. (2008). Design of an RFID-based battery-free programmable sensing platform. *IEEE Transactions on Instrumentation and Measurement, 57*(11), pp. 2608–2615.
8. *EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860 MHz - 960 MHz*, EPCglobal Inc. Specification for RFID Air Interface, Rev. 1.2.0, 2008. [Online]. Available: http://www.epcglobalinc.org/standards/uhfc1g2/
9. Zhang Y., Yang L., and Chen J., Eds. (2010). *RFID and sensor networks: architectures, protocols, security, and integrations*, 1st ed., New York, USA: CRC Press, 2010.
10. Weis S. A., Sarma S., Rivest R., and Engels D. (2003). Security and privacy aspects of low-cost radio frequency identification systems. In *Proceedings Int Conf. on Security in Pervasive Computing*, pp. 201–212. Also in *"'Security in Pervasive Computing"'*, Springer LNCS, Volume 2802/2004, pp. 50–59.
11. Changqing O., Jixiong W., Zhengyan L., and Shengye H. (2008). An enhanced security authentication protocol based on hash-lock for low-cost RFID. In *Proceedings International Conference on Anti-counterfeiting, Security and Identification*, pp. 416–419.
12. Ohkubo M., Suzuki K., and Kinoshita S. (2003). Cryptographic approach to "privacy-friendly" tags. In *Proceedings of RFID Privacy Workshop*, MIT Cambridge MA.
13. Yong G., Lei H., Na-na L., and Tao Z. (2010). An improved forward secure RFID privacy protection scheme. In *Proceedings 2nd International Asia Conference on Informatics in Control, Automation and Robotics*, Wuhan, Vol. 2, pp. 273–276.
14. Chatmon C., van Le T., and Burmester M. (2006). Secure anonymous RFID authentication protocols. Tallahassee, FL, USA, Tech. Rep. TR-060112.
15. Lei H., Song-he J., Tao Z., and Na-na L. (2009). An enhanced 2-pass optimistic anonymous RFID authentication protocol with forward security. In *Proceedings of the 5th International Conference on Wireless communi-*

cations, networking and mobile computing, Beijing, pp. 3692–3695.
16. Rahman M., Soshi M., and Miyaji A. (2009). A secure RFID authentication protocol with low communication cost. In *Proceedings 5th International Conference on Complex, Intelligent and Software Intensive Systems*, Fukuoka Japan, pp. 559–564.
17. Needham R. M., and Wheeler D. J. (1997). TEA extensions. Cambridge University, GB, Tech. Rep., 1997.
18. Lee Y.-C., Kuo W.-C., Hsieh Y.-C., and Chen T.-C. (2009). Security enhancement of the authentication protocol for RFID systems. In *Proceedings 5th International Conference on Information Assurance and Security*, Xi'An China, Vol. 1, pp. 521–524.
19. Schneier B. (1996). *Applied cryptography: protocols, algorithms, and source code in C*, 2nd ed. New York: Wiley.
20. Moessner M. B. (2010). *Secure Authentication Protocols for RFID Systems*, MASc Thesis, Electrical and Computer Engineering, Ryerson University, Toronto, Canada.
21. *Low Level Reader Protocol (LLRP)*, (2007). EPCglobal Inc. Ratified Standard with Approved Fixed Errata, Rev. 1.0.1. [Online]. Available: http://www.epcglobalinc.org/standards/llrp/
22. Sample A., Yeager D., Powledge P., and Smith J. (2007). Design of a passively-powered, programmable sensing platform for UHF RFID systems. In *Proceedings International Conference on RFID*, Grapevine TX, pp. 149–156.
23. Chae H.-J., Yeager D. J., Smith J. R., and. Fu K. (2007).'Maximalist cryptography and computation on the WISP UHF RFID tag. In *Proceedings of the Conference on RFID Security*, Malaga, Spain. [Online]. Available: http://www.cs.umass.edu/ kevinfu/papers/chae-RFIDSec07.pdf
24. Wheeler D., and Needham R. (1995). TEA, a tiny encryption algorithm. In *Proceedings 2nd Int. Workshop on Fast Software Encryption*, Leuven, Belgium, Springer-Verlag, pp. 97–110.
25. Rivest R. (1995). The RC5 encryption algorithm. In *Proceedings of the 2nd Workshop on Fast Software Encryption*, LNCS Vol. 1008 Springer-Verlag, pp. 86–96.
26. Rivest R. L., Robshaw M. J. B., Sidney R., and Yin Y. L. (1998). The RC6 tm block cipher. In *Proceedings of the First Conference on Advanced Encryption Standard*, pp. 16–37.
27. Daemen J., and Rijmen V. (2002). *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag.
28. Lu J. (2009). Related-key rectangle attack on 36 rounds of the XTEA block cipher. *International Journal of Information Security, 8*, pp. 1–11.