International Conference on Complex, Intelligent and Software Intensive Systems

A Secure RFID Authentication Protocol with Low Communication Cost

Mohammad Shahriar Rahman[1] , Masakazu Soshi[2], Atsuko Miyaji[1]
[1] School of Information Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, Japan 923-1292
Email: {mohammad, miyaji}@jaist.ac.jp
[2] School of Information Sciences, Hiroshima City University
3-4-1, Ozuka-higashi Asa-Minami-Ku, Hiroshima, Japan 731-3194
Email: soshi@hiroshima-cu.ac.jp

***Abstract*—Gene Tsudik proposed a Trivial RFID Authentication Protocol (YA-TRAP*), where a valid tag can become incapacitated after exceeding the prestored threshold value and is thus vulnerable to DoS attack. Our scheme solves the problem by allowing a tag to refresh its prestored threshold value. Moreover, our scheme is forward secure and provides reader authentication, resistance against timing, replay, tracking attacks. We show the use of aggregate hash functions in our complete scheme to reduce the reader to server communication cost. The reader uses partial authentication to keep the rougue tags out of the aggregate function.**

I. Introduction

Radio-Frequency IDentification (RFID) is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags or transponders. An RFID tag is an object that can be applied to or incorporated into a product, animal, or person for the purpose of identification using radio waves. Some tags can be read from several meters away and beyond the line of sight of the reader. RFID tags has opened the door to previously unexplored applications. For example in supply chains as suggested by the EPC Global Inc. [1], to locate people in amusement parks, to combat the counterfeiting of expensive items [2], to trace livestock, to label books in libraries [3], etc.

A goal of researchers in RFID tag development is to see them serve ubiquitously as a replacement for barcodes. This change promises more flexible and intelligent handling of consumer goods and devices. The imminent ubiquity of RFID tags, however, also poses a potentially widespread threat to consumer privacy.

If RFID tags are easily readable, then tagged items will be subject to indiscriminate physical tracking, as will their owners and bearers. Today, RFID is used in enterprise supply chain management to improve the efficiency of inventory tracking and management. RFID has seen a swirl of attention in the past few years. One important reason for this is the effort of large organizations, such as Wal-Mart, Procter and Gamble, and the U.S. Department of Defense, to deploy RFID as a tool for automated oversight of their supply chains [4].

In such an environment, it is required to read and authenticate a large number of tags within a small period of time. A key to safe and secure supply chain is the emphasis on authenticating the objects as well as tracking them efficiently [[5] chapter 12] where unauthorized tracking of RFID tags is viewed as a major privacy threat. Moreover, the computational and communication complexity are two prime factors related to energy consumptions of an RFID system where the tags are highly resource constrained.

Our Contribution: In this paper, we introduce a two-way message authentication protocol where both tag and reader authenticate each other. Compared to the previous RFID authentication protocol named YA-TRAP* [6], our protocol has improved from the efficiency's point of view specially for batch-mode; our protocol satisfies security requirements better and the required computation is kept at a minimum. Moreover, we show the use of aggregate function for the reader to server communication. The reader to server communication cost is reduced through introducing aggregate hash function. Again, the YA-TRAP* protocol has a limitation- where a valid tag becomes non-operative after the tag is read equal to the prestored threshold timestamp value. In our two-way authentication protocol, a reader helps a tag to recover from the non-operative state. Furthermore, our scheme is forward secure and provides reader authentication, resistance against timing, replay, tracking attacks. In the extended version of our scheme, the reader uses partial authentication to keep the rougue tags out of the aggregate function. This increases a tag's computation by one hash function and also the tag to reader communication by b bits. We consider this as a trade-off between security and efficiency.

II. Operating Environment

In an RFID environment, an RFID system consists of three components: tags, reader(s) and server. **Tags** consist of an integrated circuit with a small antenna and are placed on each object that should be identified (e.g. the medicines). Each tag will send its corresponding information when interrogated by a valid reader. **Reader**(s) communicate with a server and with the tags. They are responsible of performing the queries to the tags. They also have computational and storage capabilities. A **server** is a trusted entity that knows and maintains all information about tags, such as their assigned keys. It is assumed to be physically secure and not subject to attacks.

978-0-7695-3575-3/09 $25.00 © 2009 IEEE
DOI 10.1109/CISIS.2009.162

Multiple readers might be assigned to a single server. A server only engages in communication with its constituent readers. All communication between a server and a reader is assumed to be over private and authentic channel. We assume that the adversary can be either passive or active. It can corrupt or attempt to impersonate or incapacitate any entity or track RFID tags. Namely, an adversary succeeds to trace a tag if it has a non-negligible probability to link multiple authentication and/or state update sessions of the same tag. Compromise of a set of tags should not lead to the adversary's ability to track other tags. Furthermore, the possibility of Denial of Service (DoS) attack, i.e., attacks that aim to disable the tags should be in a minimum level. Both reader and server have ample storage and computational capabilities. We assume that an RFID tag has no clock and small amounts of ROM to store a key and non-volatile RAM to store temporary timestamp. With power supplied by reader, a tag can perform a modest amount of computation and change its permanent state information stored in its memory. The messages which are in plaintext, are fully accessible by the adversary.

In batch mode, a reader scans numerous tags, collects the replies, and sometimes, performs their identification and authentication later in bulk. The batch mode is appropriate when circumstances prevent or inhibit contacting the back-end server in real time. An inventory control system, where readers are deployed in a remote warehouse and have no means of contacting a back-end server in real time is such an application.

Each tag $RFID_i$ is initialized with at least the following values: k_i, T_0, T_{max_i}; where k_i is a tag-specific value that serves two purposes: (1) tag identifier, and (2) cryptographic key. Thus, the size (in bits) of k_i is required to serve as sufficiently strong cryptographic key for the purposes of Message Authentication Code (MAC) computation. A new hash SQUASH proposed by Shamir [7] is the underlying hash function since it executes in fewest gates and operates in single block which are very important for resource constrained devices like RFID. In practice, a 128-bit k_i will most probably suffice. T_0 is the initial timestamp assigned to the tag. This value does not have to be a discrete counter. For example, T_0 can be the time-stamp of manufacture. T_0 need not be tag unique; an entire batch of tags can be initialized with the same value. T_{max_i} can be viewed as the highest possible time-stamp. T_{max_i} is a tag specific secret value. This threshold value can be changed in case a tag becomes inactive due to exceeding the value. Each tag is further equipped with a sufficiently strong, uniquely seeded pseudo-random number generator (PRNG). For a tag $RFID_i$, $PRNG_i^j$ denotes the j-th invocation of the (unique) PRNG of tag i. Given a value $PRNG_i^j$, no entity (including a server) can recover k_i or any other information identifying $RFID_i$. Similarly, given two values $PRNG_i^j$ and $PRNG_j^k$, deciding whether $i = j$ must be computationally infeasible.

III. Previous Works

We consider only related works relevant for comparison with our approach, i.e., protocols that emphasize involving secure minimal 2-round reader-tag interaction, aim to reduce tag requirements and computation, and address the communication complexity.

MSW protocols by Molnar, et al. [8] use hierarchical tree based keying. But they have a security flaw shown by Avoine, et al. [9] whereby an adversary who compromises one tag, is able to track/identify other tags that belong to the same families (tree branches) as the compromised tag. Zhu et.al. showed the security of aggregate function for RFID tags [10]. But they do not show the use of the aggregate function in a complete authentication protocol. Moreover, it is not clear how to find out individual rougue tags by the verifier.

We mainly consider YA-TRAP* to compare our work with. Before going to state the original YA-TRAP* algorithm, we are describing the parameters used here. For the sake of simplicity and better understandability, we will continue to use same notations related to most of these parameters in our scheme. T_{r_i}, R_{r_i}, ET_{r_i}: timestamp, random challenge, epoch token sent by the reader to a tag i. ET_{r_i} allows a tag to ascertain that the reader-supplied T_{r_i} is not too far into the future. This token changes over time, but its frequency of change (epoch) is generally much slower than the unit of T_{r_i}. Due to spcae limitation, we omit the detail of the protocol. The following Algorithm (YA-TRAP*) shows authentication between a Tag_i and a reader, and, between a reader and a server at time j.

Algorithm 1 (YA-TRAP):*

[1]Tag $\leftarrow$ Reader: $T_{r_i}^j, R_{r_i}^j, ET_{r_i}$

[2] Tag_i:

- [2.1] $\delta = T_{r_i}^j - T_{t_i}^j$
- [2.2] $\nu = \lfloor T_{r_i}^j / INT \rfloor - \lfloor T_{t_i}^j / INT \rfloor$
- [2.3] If $(\delta \leq 0)$ or $T_{r_i}^j > T_{max_i}$ or $H^\nu(ET_{t_i}) \neq ET_{r_i}$
  - [2.3.1] $H_{id_i} = {PRNG_i}^j$
  - [2.3.2] $H_{auth_i} = {PRNG_i}^{j+1}$
  - [2.3.2] $H_{auth_i} = {PRNG_i}^{j+2}$
- [2.4] else $T_{t_i}^j = T_{r_i}^j$, $ET_{t_i} = ET_{r_i}$
  - [2.4.1] $H_{id_i} = HMAC_{k_i}(T_t)$
  - [2.4.2] $R_{t_i} = PRNG_i^{j+1}$
  - [2.4.3] $H_{auth_i} = HMAC_{k_i}(R_{t_i}, R_{r_i})$

[3] Tag $\rightarrow$ Reader: $H_{id_i}, R_{t_i}, H_{auth_i}$

[4] Reader $\rightarrow$ Server: $H_{id_i}, R_{r_i}, R_{t_i}, T_{r_i}^j, H_{auth_i}$

[5] Server:

- [5.1] $s = LOOKUP(HASH_{TABLE_{T_{r_i}}}, H_{id_i})$
- [5.2] $if(s == -1)$
  - [5.2.1] MSG = TAG-ID-ERROR
- [5.3] else if $(HMAC_{K_s}(R_{t_i}, R_{r_i}) \neq H_{auth})$
  - [5.3.1] MSG=TAG-AUTH-ERROR
- [5.4] else MSG=TAG-VALID

[6] Server $\rightarrow$ Reader: MSG

The protocol is vulnerable to DoS attacks. DoS resistance in YA-TRAP* is limited by the magnitude of the system-wide INT parameter. An adversary can incapacitate tags for at most the duration of INT if it queries each victim tag with the current epoch token and the maximum possible T_{r_i} value within the current epoch. A tag needs to compute 4 keyed hash operations and a PRNG. In addition, a tag needs to compute

ν hashes over ET_t. The communication cost of Tag to Reader is 3 messages (2 HMAC, 1 R_{t_i}) per Tag. The communication cost of Reader to Server is 5 messages (2 HMAC, 1 R_{t_i}, 1 R_{r_i}, 1 T_{r_i}) per Tag. That means, for n number of tags, a total of $2n$ HMAC, n R_{t_i}, n R_{r_i} , n T_{r_i} messages are sent to the server. This needs a huge amount of resource for communication in case of batchmode authentication. Moreover, eventhough a tag is valid, it becomes nonoperative when T_{r_i} exceeds T_{max_i}. That means, after being read for several times, when the valid timestamp value T_{r_i} sent by the reader becomes higher than the T_{max_i} stored in a valid tag, this valid tag no longer responds correctly to the reader.

IV. Our Secure and Low Cost Scheme

We mainly divide our scheme into two parts. The protocol we describe below reduces the communication costs between tag to reader and between reader to server. Then in subsection A, we extend the protocol into a more secure one which provides partial authentication of the tags. But the extended version increases a tag's computation by one hash function and also the tag to reader communication by b bits. We use one-time pad and aggregate hash function in our scheme.

1) One-time pad: The one-time pad is a simple, classical form of encryption (See, e.g., [11] for discussion). We briefly recall the underlying idea. If two parties share a secret one-time pad p, namely a random bitstring, then one party may transmit a message m secretly to the other via the ciphertext $p \oplus m$, where $\oplus$ denotes the XOR operation. It is well known that this form of encryption provides information theoretic secrecy. Suppose, for instance, that pads from two different verifier-tag sessions are XORed with a given tag value in order to update it. Then even if the adversary intecepts the pad used in one session, it may be seen that she will learn no information about the updated value. Application of a one-time pad requires only the lightweight computational process of XORing. Indeed, one-time padding results in less communications efficiency than that achievable with standard cryptographic encryption tools like block or stream ciphers. The problem is that, standard cryptographic primitives require more computational power than is available in a low-cost RFID tag. This is the real motivation behind our use of one-time pads.

2) Aggregate Function: An aggregate function follows that if we are able to compress the size of all hash functions $HASH_i$, then the communication complexity between the reader and the server can be reduced accordingly. This leaves an interesting research problem - is it possible to aggregate tags' attestations so that the size of the resulting aggregate attestation (i.e., H) is approximate to that of the original case (i.e., non-aggregate model). That is, given $H_{id_i} = HASH(R_{t_i}, R_{r_i})$, where HASH is a cryptographic one-way hash function, and R_{t_i} and R_{r_i} are random challenges of tag and reader respectively, we ask whether there exists an efficient polynomial time algorithm such that on input $H_{id_i} = HASH(R_{t_i}, R_{r_i})$, it outputs an aggregate of hash functions such that the size of $\oplus(H_{id_1}, H_{id_2}, ..., H_{id_n})$ is approximate to an individual H_{id_i}; moreover, the validity of individual attestations can be checked efficiently given $\oplus(H_{id_1}, H_{id_2}, ..., H_{id_n})$. We derive an aggregate function from [10] as a tuple of probabilistic polynomial time algorithms (HASH,Aggregate,Verify) such that: The authentication algorithm HASH takes as input random numbers R_{t_i} and R_{r_i} and outputs an attestation H_{id}; The aggregate function Aggregate takes as input $H_{id_1}, H_{id_2}, ..., H_{id_n}$ and outputs a new attestation H; The verification algorithm verify takes as input (R_{t_1}, R_{r_1}), (R_{t_2}, R_{r_2}),..., (R_{t_n}, R_{r_n}) and an attestation H, outputs a MSG with Accept denoting acceptance or Error denoting rejection.

As we have seen previously, the data on a genuine tag can be easily scanned and copied by a malicious RFID reader and the copied data can be embedded onto a fake tag. Malicious readers may also try to corrupt and snoop on genuine tags. These threats are nullified by incorporating a RFID reader authentication by a tag. To authenticate itself to a tag, a reader sends $Hash(T_{r_i}^{j-1}, T_{max_i})$. If the received hash value matches with the computed hash value, then it proceeds. Otherwise it rejects by generating pseudo-random numbers(PRNG). PRNGs are generated as becuase they are required for the respective times taken by hash function computations and the rejection as close as possible. This is needed to thwart obvious timing attacks by malicious readers aimed at distinguishing among the two cases. The random number generated must be indistinguishable from $Hash(T_{r_i}^{j-1}, T_{max_i})$. That means, the adversary has to face the decision problem of distinguising the $Hash(T_{r_i}^{j-1}, T_{max_i})$ from a random value. This indistinguishability feature is required to protect against narrowing attack [6] which leads into tracing the tag by an adversary. The use of PRNGs to obfuscate the tag identity was first introduced in [12].

The server keeps the table of valid tags and the last issued timestamp T_{r_i} for each of the tags. So, it can easily findout a valid tag when it becomes after exceeding the threshold T_{max_i}. The new timestamp threshold value $T_{max_{i_{new}}}$ is stored in memory erasing the existing T_{max_i} only after the reader authenticates itself to the tag. However, the reader generates $T_{max_{i_{new}}}$ only when the tag makes sure that itself is a dead tag (unexpected output from a valid tag can be an indicator). The reader sends the value T_r by XOR-ing the T_{max_i} and $T_{max_{i_{new}}}$. Generating $T_{max_{i_{new}}}$ works as one-time padding as it is freshly computed everytime a tag requires to update its T_{max_i}; and the value T_{r_i} also cannot be revealed by the adversary as the adversary cannot distinguish the tag's actual response from a PRNG. Before XOR-ing, the reader must make sure that the value of $T_{max_{i_{new}}}$ is strictly greater than T_{max_i}. Eventhough exceeding the T_{max_i} value is considered to be a feature of YA-TRAP*, we take into account a different scenario: when an item is brought from the store, the reader/server may want to incapacitate the attached RFID tag, for example. In this case, our protocol only needs to send an arbitrary future timestamp value to the tag to incapacitate. On the other hand, in YA-TRAP*, the reader/server needs to send several ET_{r_i} values to eventually reach T_{max_i}-which is not

suitable in cases like the example above.

After the reader authentication, a tag compares whether the received $T_{r_i}^j$ is greater than T_{max_i}. If so, it computes $T_{max_{i_{new}}}$ and verifies whether the $T_{max_{i_{new}}}$ is strictly greater than T_{max_i}. A tag then refreshes its T_{max_i} value. Then the tag computes hash of R_{r_i} and R_{t_i}.

After receiving responses from the tags, the reader chekcs the R_{r_i} whether it matches with any other previous value.

As the tags responses are collected over multiple time intervals, the reader marks the responses according to the T_{r_i} values used.

Then reader aggregates all the H_{id_i} by XOR-ing them. The security of aggregate hash functions has been shown in [13]. After that, it concatenates all the R_{t_i} into one message R_t. The reader forwards H, R_t to server.

Upon receiving them, the server looksup its database for the marked tags, computes their corresponding hash values and matches their XOR-ed value with the received H. It sends MSG= TAG-$VALID$ back to reader to end the whole process.

The Algorithm 2 below is the secure and low cost authentication protocol:

Algorithm 2 (Low Cost and Secure Authentication Scheme):

[1]$Tag \leftarrow$ Reader: $T_{r_i}^j, R_{r_i}^j, Hash(T_{r_i}^{j-1}, T_{max_i})$
[2] Tag_i:
  [2.1]While $Hash(T_t^j, T_{max_i}) \neq Hash(T_{r_i}^{j-1}, T_{max_i})$,
  $R_{t_i}^j = PRNG_i{}^1$, $H_{id_i} = PRNG_i{}^2$, $k_i^{j+1} = PRNG_i{}^3$
  [2.2] $\delta = T_{r_i}^j - T_{t_i}^j$
  [2.3]While $T_{r_i}^j > T_{max_i}$, then
  $T_{max_{i_{new}}} = T_{r_i}^j \oplus T_{max_i}$,
  if $T_{max_{i_{new}}} > T_{max_i}$, then set $T_{max_i} = T_{max_{i_{new}}}$,
  else Reject
  [2.4] if $(\delta \leq 0)$ then
  $R_{t_i}^j$= $PRNG_i{}^1$, H_{id_i}= $PRNG_i{}^2$, k_i^{j+1}= $PRNG_i{}^3$
  else $T_{t_i}^j$= $T_{r_i}^j$, $R_{t_i}^j$= $PRNG_i$, H_{id_i}= $HASH(R_{t_i}^j, R_{r_i}^j)$
  [2.5] k_i^{j+1}= $H(k_i^j)$
[3] Tag $\rightarrow$ Reader: $H_{id_i}, R_{t_i}^j$
[4] Reader:
  [4.1] If $R_{t_i}^j$ matches with any of the
  previously generated R_{t_j}, then REJECT
  else mark each $H_{id_i}, R_{t_i}^j$
  [4.2] $H = \bigoplus_{i=1}^n H_{id_i}$,
  $R_t = R_{t_1} \parallel R_{t_2} \parallel R_{t_3} \parallel ..... \parallel R_{t_n}$
[5] Reader $\rightarrow$ Server: H, R_t
[6] Server:
  [6.1] lookup accepted T_r according to marked R_{t_i};
  if $\bigoplus_{i=1}^n HASH(R_{r_i}, R_{t_i}) \neq H$,
  then MSG=TAG-AUTH-ERROR
  else MSG=TAG-VALID
  [6.2] update each k_i
[7] Server $\rightarrow$ Reader: MSG

A. Extending the Protocol

The aggregate function enables a server only to find out the anomaly of the resultant XOR operations of the hash values. That means, if the computed aggregate hash value does not match with the received H, the server cannot find out the specific tag for which the result is an oddity.

To alleviate such incident, we use a one-way hash for the partial authentication of a tag. One-wayness means that, having seen the hash value, it is not possible to extract the contents on the hash. For this purpose, we need to add the following step to compute an authencation token(AT) as step [2.5] before renewing the key k_i:

[2.5]$AT_{t_i}^j = Hash(T_{max_i}, k_i^j)$

A tag i has its secret value T_{max_i} and key k_i^j. Generating k_i^j for every read operation also ensures forward security. So, to partially authenticate itself to server, a tag sends the computed $AT_{t_i}^j$ value to the reader. So, the step [3] is rewritten as:

[3]$' : Tag_i \rightarrow Reader : H_{id_i}, R_{t_i}^j, AT_{t_i}^j$

Upon receiving $AT_{t_i}^j$ from a tag i, the reader finds a match with the desired $AT_{s_i}^j$ for tag i, which the reader had received from the server for that particular tag i for the j-th read interaction. This requires us to modify the step [4.1] as [4.1]' like below:

[4.1]$'$ If $R_{t_i}^j$ matches with any of the previously generated R_{t_i}, then REJECT
else if $AT_{t_i}^j \neq AT_{s_i}^j$, then exclude
else mark each $AT_{t_i}^j, H_{id_i}, R_{t_i}^j$

The reader has a table consisting of the expected identification token $AT_{s_i}^j$ values corresponding to each tag (each tag's T_{r_i}). So, the reader checks the hash value of a tag and if a tag is found legitimate, its authentication message is included into the reader's aggregate function.

The tag sends a one-way hash value of the random numbers to the reader to authenticate itself. As the reader is not capable of too much computaions like computing hash values of a huge number of tags, it forwards the hash values for authentication to the server.

The server also needs to do some computation after it matches the aggregated value received from the reader. After updating each key k_i^j of the corrsponding tags to k_i^{j+1}, the server computes their respective authentication token values $(AT_{s_i}^{j+1})$ for the next $(j+1)$-th read operation. This step is added to the protocol as step [6.3]:

[6.3]compute $AT_{t_i}^{j+1} = Hash(T_{max_i}, k_i^{j+1})$

Furthermore, the server to reader communication is required to include the newly generated $AT_{s_i}^{j+1}$ values. So, step [7] can be modified like:

[7]$'$ Server $\rightarrow$ Reader: MSG, all $AT_{s_i}^{j+1}$

One significant point here to note that, the extended version of the protocol increases communication cost for both the tag to reader and server to reader interactions. However, sacrificing a part of communication cost strengthens the security. We consider this tradeoff between cost and security to be a feature of our protocol. The partial authentication by the reader helps filter out malicious tags. This erases inclusion of any rougue tag in the aggregate function, which leads into no anomaly when the server verifies the aggregate value.

We emphasize that, for a batch mode environment where validating a number of tags in a very small amount of time,

Algorithm 2 is suitable - where the communication cost is the least. In such a batch mode environment, tags are authenticated in a bulk, hence it is enough to verify the authentication of the whole batch together in a least possible time. Even though the extended protocol is not much effective for batch mode environment where validating a number of tags in a small amount of time is required, it can be used in the settings where the server is not readily available. In such situations, the reader can partially authenticate the tags. Moreover, the extended version is also suitable for small number of tags or even for individual tag authentications.

V. Security Analysis

• *Forward Security*: The forward-security property means that even if the adversary obtains the current secret key, she still cannot derive the keys used for past time periods. To ensure this, we have used a forward-secure message authentication scheme which is key-evolving. For each valid read operation, a tag uses the current key k_i for creation and verification of authentication tags. At the end of each valid read operation, k_i is updated by a one-way hash function H and previous k_i is deleted. An attacker breaking in gets the current key. But given the current key k_i it is still not possible to derive any of the previous keys. Moreover, due to the one-wayness of the hash function used in $AT_{t_i}^j = Hash(T_{max_i}, k_i^j)$, the adversary cannot compute k_i^j or any other previous keys if she manages to get k_i^{j+1}.

• *Timing Attack*: Our protocol is immune to crude timing attacks that aim to determine the tag's state or its T_t value. From the timing perspective, steps 2.1, 2.4 and 2.5 in the algorithm are indistinguishable since PRNG and HASH are assumed to execute in the same time. PRNGs are generated as becuase they are required for the respective times taken by hash function computations and the rejection as close as possible. This is needed to thwart obvious timing attacks by malicious readers aimed at distinguishing among the two cases.

• *Tag Tracking*: Tracking a tag means that it is computationally feasible to infer from the interactions with a tag information about the identity of the tag or link multiple authentication sessions of the same tag. Success or failure of a tag-reader interaction is not observable by the adversary in the environments where the interaction concludes without some sort of publicly visible effect. In our protocol, from a tag's response $PRNG_i$, no entity (including the server) can recover k_i or any other information identifying that particular tag. This is due to the reason that, the PRNG values are indistinguishable from the normal replies (Hash values) and the hash function used is one-way (given the hash value, it is not computationally feasible to derive the contents of the hash function).

It is also not possible for an adversary to track a tag due to the use of one-time padding. As the adversary cannot distinguish a normal response from a PRNG, she cannot track a tag even the tag becomes incapacitated by exceeding the T_{max_i} value. So when the valid reader forwards a $T_{r_i} > T_{max_i}$ to an incapacitated tag i, the adversary cannot find out the $T_{max_{i_{new}}}$ from the value she has seen during the session. As the $T_{max_{i_{new}}}$ is freshly generated and the adversary cannot know for which particular value of T_{r_i} the stored T_{max_i} is going to be refreshed, the information theoretic security of one-time padding provides the secrecy of the $T_{max_{i_{new}}}$.

• *DoS resistance*: Our protocol also has DoS resistance capability. By feeding an arbitrary future timestamp T_{r_i}, an adversary can incapacitate a tag. As only the server can distinguish a tag i's normal reply from a random reply, and it has the T_{r_i} and T_{max_i} in its database, the server can generate and new threshold timestamp value for the tag to replace the old threshold. As discussed earlier, generating and sending the new threshold value to the tag works as one-time pad. And the adversary is not able to keep the tag incapacitated for a long time (infact, the tag will generate PRNG on the next immediate valid interaction only).

• *Tag Cloning*: Tag cloning means that, the data on a valid tag is scanned and copied by a malicious RFID reader and the copied data is embedded onto a fake tag. Authentication of RFID reader prevents this cloning attack. In our protocol, a tag never generates genuine replies unless it verifies the reader first. This verification thwarts the cloning attack.

• *Replay Attack*: Assuming that the random challanges sent by the reader and the tag are same in two different sessions, an adversary can launch replay attack by snooping the random numbers. In our protocol, the reader matches the random numbers it receives from the tags to make sure that no two random numbers from two defferent sessions with a same tag are equal. This prevents the replay attack.

• *Tag Non-operative*: In our scheme, the server can help an incapacitated tag to become operative by sending a renewed T_{max_i} which is strictly greater than the previous one. This renewing capability also enables a server to willingly incapacitate a tag whenever it wants (consumer goods which are not needed to be authenticated any more after they are sold).

VI. Comparison of Performance

As discussed earlier, MSW protocol has a vulnerability that compromising of a tag makes the other tags of the same family compromised also. This concern does not arise in our protocol since no two tags share any secrets between them. In other words, in our work, it is not possible for an adversary to derive secrets of other tags even if she gets a tag's secrets - which feature is absent in MSW protocol.

Zhu et.al's aggregate function is shown to be secure, but they do not provide any complete protocol to show the use of aggregate function; it also can not find out an individual rouge tag- rather can only find out that the aggregate function has some rogue tag's information by detecting the oddity of the aggregated output. Our protocol uses the aggregate function to reduce the communication cost. Moreover, to indentify a rougue tag, we introduce partial authentication by the reader. This works as a filter to reject any possible fake tags. This partial authentication helps an aggregate function to be correctly verified by the server, hence authenticating

the corresponding tags. But this increases the server-reader communication, which we consider to be a trade off between security and performance.

The comparison of our work with the YA-TRAP* is shown in tables below. For the clearity of comparison, we provide the same environment to the YA-TRAP*, i.e., aggregate function is also applied there. Even if the YA-TRAP* implements aggregate function, our protocol achieves lower communication cost. We compare our extended version with YA-TRAP*. The reason behind is the concern of security. The extended version provides security through partial authentication which is not present in our initial scheme(Algorithm 2). We use the partial authentication to keep the rougue tags out of the aggregate function. The partial authentication requires a tag to compute one more hash function (computing $AT_{t_i}^j$) and contain b more bits ($AT_{t_i}^j$) while communicating with the reader. As we discussed earlier, we consider this as a feature of our scheme; i.e., a trade-off between security and performance. So, to maintain the same level of security while achieving a lower communication cost, we need the same number of tag computations compared with YA-TRAP*.

TABLE I
PERFORMANCE COMPARISON (SECURITY)

	For. Sec.	DoS rest.	Rep Atk.	Tag NOp	Rd. Au.
YA-TRAP*	yes	no	no	yes	no
Our Scheme	yes	yes	yes	no	yes

TABLE II
PERFORMANCE COMPARISON (COST)

	Tag comp	TtoR comm (bits)	RtoS comm (bits)
YA-TRAP*	4 HASH 1 PRNG	3 b	(3n+2)b
Our Scheme	4 HASH 1 PRNG	3 b	(n+1)b

- n = total number of tags.
- b = bit length of HASH, R_{t_i}, R_{r_i}, T_{r_i} (assuming all are equal in bit size)

Table 1 shows the comparison of security features like Reader Authentication (Rd Au), Forward Security(For Sec), DoS resistance (DoS res), Replay Attack resistance (Rep Atk) and whether tag bocomes non-operative (Tag NOp). Our scheme achieves better performance considering DoS, replay, cloning attack resistances compared to YA-TRAP*. The DoS resistance capability is one of the main achievements of our scheme. In YA-TRAP*, by feeding an arbitrary future timestamp T_{r_i}, an adversary can incapacitate a tag, thus launching a DoS attack. As only the server can distinguish a tag i's normal reply from a random reply, and it has the T_{r_i} and T_{max_i} in its database, the server can generate a new threshold timestamp value for the tag to replace the old threshold. As discussed earlier, generating and sending the new threshold value to the tag works as one-time pad. And the adversary is not able to keep the tag incapacitated for a long time. Moreover, unlike YA-TRAP*, our scheme does not allow a tag to become non-operative. Rather whether a tag will become non-operative or not can be controlled by the reader. One more security feature provided by our scheme is 'reader authentication'. Reader authentication is required to prevent cloning attack.

Table 2 includes tag's computaion(Tag comp), Tag to Reader communication (TtoR comm.) and Reader to Server communication (RtoS comm.) costs. Considering the extended version of our scheme, YA-TRAP* and our scheme requires the same number of computations by a tag; i.e., 4 hash functions and 1 PRNG. Also the tag to reader communication contains $3b$ bits for both schemes. The bit length of reader to server communication is lower in our scheme compared to YA-TRAP*. For n number of tags, our scheme requires $(n+1)b$ bits, whereas YA-TRAP* requires $(3n+2)b$ bits for the reader to server communication. This clearly reduces the cost by a significant amount which is very important for batch-mode communication.

VII. CONCLUSION

In this paper, we have proposed a secure two-way authentication protocol with low communication cost which is more efficient and secure compared to the previous works. Our scheme provides reader authentication to thwart tag cloning and also resists DoS, Replay, Tracking, Timing attacks. We show the use of aggregate hash functions in our complete scheme to reduce reader to server communication cost down to $(n+1)b$ bits for n tags- which is very much suitable for batch mode authentication environment. In the extended version of our scheme, the reader uses partial authentication to keep the rougue tags out of the aggregate function. It also provides forward security and does not allow a tag to become non-operative unless explicitly done by the reader.

REFERENCES

[1] Electronic Product Code Global Inc. http://www.epcglobalinc.org.

[2] Ari Juels, Ravikanth Pappu: Squealing euros: Privacy protection in RFID enabled banknotes. *Financial Cryptography-FC 03*, pp. 103-121, Springer-Verlag (2003).

[3] David Molnar and David Wagner: Privacy and security in library RFID: Issues, practices, and architectures. *Conference on Computer and Communications Security- CCS04*, pp. 210-219, ACM Press (2004).

[4] A. Juels: RFID security and privacy: A Research Survey. *IEEE Journal on Selected Areas in Communication*, Vol. 24, chapter 2,(2006).

[5] Peter H. Cole, Damith C. Ranasinghe: Networked RFID Systems and Lightweight Cryptography Raising Barriers to Product Counterfeiting. *Springer*, e-ISBN 9783540716419(2007).

[6] Gene Tsudik: A Family of Dunces: Trivial RFID Identification and Authentication Protocols:*Privacy Enhancing Technologies- PET*, pp. 45-61, Springer-Verlag(2007).

[7] Adi Shamir: SQUASH - A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags. *Fast Software Encryption- FSE*, pp. 144-157, Springer-Verlag(2008).

[8] D. Molnar, A. Soppera and D. Wagner: A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags, *Workshop in Selected Areas in Cryptography- SAC*, pp. 276-290, Springer-Verlag(2006).

[9] G. Avoine, E. Dysli, and P. Oechslin: Reducing Time Complexity in RFID Systems, *Workshop on Selected Areas in Cryptography- SAC*, pp. 291-306, Springer-Verlag(2006).

[10] H. Zhu and F. Bao: Aggregating Symmetric/Asymmetric Attestations. *IEEE International Conference on RFID*,pp. 105-110, IEEE(2008).

[11] A.J. Menezes, van Oorschot, S.A. Vanstone: *Handbook of Applied Cryptography*, pp. 192-193, CRC press(1997).

[12] S. Weis, S. Sarma, R. Rivest, D. Engels: Security and Privacy Aspects of Low- Cost Radio Frequency Identification Systems. *Security in Pervasive Computing Conference- SPC*, pp. 201-212, Springer-Verlag(2004).

[13] J. Katz, A.Y. Lindell: Aggregate Message Authentication Codes. *Topics in Cryptology CT-RSA*,pp. 155-169, Springer-Verlag(2008).