# An RFID Authentication Protocol Suitable for Batch-mode Authentication

Rahman Mohammad Shahriar† Masakazu Soshi‡ Kazumasa Omote†
Atsuko Miyaji†

†Japan Advanced Institute of Science and Technology (JAIST)
1-1, Asahidai, Nomi, Ishikawa 923-1292, Japan
{mohammad, omote, miyaji}@jaist.ac.jp
‡Hiroshima City University
3-4-1 Ozuka-Higashi, Asa-Minami-Ku, Hiroshima, 731-3194, Japan
soshi@hiroshima-cu.ac.jp

**Abstract** Passive Radio Frequency Identification (RFID) tags are nowadays used in stores and industries. These RFID tags are heavily constrained in computational and storage capabilities, Supply-chain, inventory management are the areas where batch-mode authentication of RFID tags is required in a secure and cost effective manner. The reader needs to forward to server a small number of messages corresponding a large number of tags to keep the communication cost as low as possible. Gene Tsudik proposed a trivial RFID authentication protocol named YA-TRAP* [1]. Our proposal improves the efficiency of the aforementioned technique in a cost effective way specially for batch-mode. The tag to reader and reader to server communication costs are reduced maintaining the same level of security. Moreover, a tag is helped by a reader to get operative from a non-operative state.

## 1 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. Today, RFID is used in enterprise supply chain management to improve the efficiency of inventory tracking and management. In both the popular press and academic circles, 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 [2]. 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 [[3] chapter 12] where unauthorized tracking of RFID tags is viewed as a major privacy threat. Moreover, the computation complexity 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 improve the efficiency of a trivial RFID authentication

protocol named YA-TRAP* [1] in a cost effective way specially for batch-mode. The tag to reader and reader to server communication costs are reduced while maintaining the same level of security. The aforementioned protocol has a limitation where a valid tag becomes non-operative after the tag is read equal to the pre-stored threshold timestamp value. We propose a reader-tag bothway authentication protocol which helps a tag recover from the dead state. Moreover, we reduce a tag's computation requirement while communicating with the reader.

The remainder of the paper is organized as follows: section 2 presents the description of the operating environment, section 3 describes the previous work. Next our scheme is proposed achievements are discussd in section 4. Section 5 shows the comparison between the previous work and our scheme. And finally the section 6 includes some concluding remarks.

## 2 Operating Environment

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. The legitimate entities are: tags, readers and servers. A reader is a device querying tags for identification information. It has also computational and storage capabilities. A server is a trusted entity that knows and maintains all information about tags, their assigned keys and any other such information. A server is assumed to be physically secure and not subject to attacks. Multiple readers might be assigned to a single server. A server only engages in communication with its constituent readers. All communication between server and reader is assumed to be over private and authentic channel. Moreover, servers and readers maintain loosely synchronized clocks. 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 reader to tag messages are all in plaintext that means the adversary has full access to the messages.

In batch mode, a reader scans numerous tags, collects replies and sometime later performs their identification and authentication 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: $k_i$, $T_0$, $T_{max_i}$; $k_i$ is a tag-specific value that serves two purposes: (1) tag identifier, and (2) cryptographic key. Thus, its size (in bits) is required to serve as sufficiently strong cryptographic key for the purposes of Message Authentication Code (MAC) computation. 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, per se. 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. The

bit-size of $T_0$ depends on the desired granularity of time and the number of times a tag can be authenticated. $T_{max_i}$ 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 $i$. No synchronization whatsoever is assumed as far as PRNG-s on the tags and either readers or servers. In other words, 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.

# 3 Previous Work

Before going to state the original YA-TRAP* algorithm, we are describing the parameters used here. We continue to use 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. $ET_{r_i}$ 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 $T_{r_i}$.For example, $T_{t_i}$ and $T_{r_i}$ are measured in minutes, whereas, the epoch token might change daily. At any given time, a tag holds its last time-stamp $T_{t_i}$ and the its last epoch token $ET_{t_i}$. When a reader queries a tag (in step 1), it includes $ET_{r_i}$, the current epoch token. The tag calculates theoffset of $ET_{r_i}$ as $\nu$ in step 2.2. Assuming a genuine reader, this offset represents the number of epochs between the last time the tag was successfully queried and $ET_{r_i}$. If $T_{r_i}$ is deemed to be plausible in the first two OR clauses of step 2.3, the tag computes $\nu$ successive iterations of the hash function H() over its prior epoch token $ET_{t_i}$ and checks if the result matches $ET_{r_i}$. In case of a match, the tag concludes that $T_{t_i}$ is not only plausible but is at mostINT time units (e.g., one day) into the future.

**Algorithm 1 (YA-TRAP*)** [1] *Tag* $\leftarrow$ *Reader:* $T_{r_i}^j, R_{r_i}, ET_{r_i}$
[2] $Tag_i$*:*
[2.1] $\delta = T_{r_i}^j - T_{t_i}^j$
[2.2] $\nu = \lfloor T_{r_i}^j / INT \rfloor - \lfloor T_{r_i}^j / INT \rfloor$
[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*

A tag needs to compute 3 keyed hash operations including a PRNG and in addition, needs to compute $\nu$ hashes over $ET_t$. The communication cost of Tag to Reader is 3 messages per Tag. The communication cost of Reader to Server is 5 messages per Tag. That means, for $n$ number of tags, the total message is $5n$. This needs a huge amount of resource for communication in case of batch-mode authentication. Eventhough a tag is valid, it becomes non-operative 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.

The protocol is vulnerable to DoS attacks. DoS resistance in YA-TRAP* is limited by the magnitude of the system-wide INT parameter. Once revealed by the server and distributed to the genuine readers, the current epoch token $ET_r$ is not secret; it can be easily snooped by the adversary. Therefore, the adversary can still 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 withinthe current epoch. Moreover, an adversary can permanently incapacitate a tag by feeding arbitrary future timestamps and making it exceed the stored threshold value $T_{max_i}$.

The scheme can be made forward secure by adding an extra step in tag's operation as shown by the author: [2.4.6] $K_i^{\nu} = H^{\nu}(K_i)$ As a result of this modification, the computation of ephemeral tables by the server has to be changed.

# 4 Our Scheme

We will describe our scheme here.

1.Instead of HMAC, SQUASH, proposed by Shamir [4] is used. This executes in fewest gates and operates in a single block.

2. A tag computes hash of its last updated $T_{t_i}^j$, hash of $R_{r_i}$ and $R_{t_i}$ and makes $H_{auth_i}$ by XOR-ing them. The tag then sends this $H_{auth_i}$ and $R_{t_i}$ to the reader. After receiving responses from the tags, the reader aggregates all the $H_{auth_i}$ by XOR-ing them. The security of aggregate hash functions has been shown in [5]. The reader groups the values it recieved from each tag. After that, it concatenates all the $R_{t_i}$ into one message $R_t$. The reader forwards $H$ and $R_t$ to server. Upon receiving them, the serverlooksup its pre-computed 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.

**Algorithm 2 (Proposed Algorithm)** [1] *Tag* $\leftarrow$ *Reader:* $T_{r_i}^j, R_{r_i}, ET_{r_i}$
[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* $H^{\nu}(ET_{t_i}) \neq ET_{r_i}$*, then*
$H_{id_i} = PRNG_i{}^j$, $H_{id_i'} = PRNG_i{}^{j+1}$, $H_{id_i'} = PRNG_i{}^{j+2}$
[2.4] *else* $T_{t_i}^j = T_{r_i}^j$, $ET_{t_i} = ET_{r_i}$
$H_{id_i} = HASH(T_{t_i}^j)$, $R_{t_i} = PRNG_i^{j+1}$, $H_{id_{i'}} = HASH(R_{t_i}, R_{r_i})$
[2.5] $HASH_{auth_i} := H_{id_i} \oplus H_{id_{i'}}$
[3] *Tag* $\rightarrow$ *Reader:* $HASH_{auth_i}, R_{t_i}$
[4] *Reader:*
[4.1] $H = \bigoplus_{i=1}^{n} HASH_{auth_i}$
[4.2] $R_t = R_{t_1} \parallel R_{t_2} \parallel R_{t_3} \parallel ..... \parallel R_{t_n}$
[5] *Reader* $\rightarrow$ *Server:* $H, R_t$
[6] *Server:*
[6.1] *if* $\bigoplus_{i=1}^{n}(HASH(T_{t_i}) \oplus HASH(R_{r_i}, R_{t_i})) \neq H$
[6.1.1] *MSG=TAG-AUTH-ERROR*
[6.2] *else MSG=TAG-VALID*
[7] *Server* $\rightarrow$ *Reader: MSG*

## 4.1 Achievement of Our scheme

The server keeps the list of $T_{r_i}^j$, corresponding to tag's secret $k_i$. So, it becomes possible for the server to identify if there is any particular rogue tag. SQUASH reduces the number of gates in the Tags.

The cost of Tag to Reader communication includes 2 messages instead of 3 of the original scheme.

The cost of Reader to Server communication is drastically reduced to constant number of messages by introducing aggregate function. For

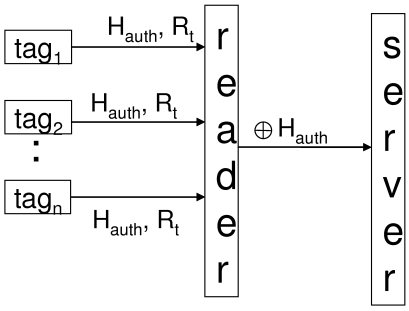


Figure 1: Our Scheme

$n$ number of tags, the communications needs only 2 messages instead of 5n messages compared to the original scheme.

### 4.2 Making a valid non-operative tag into an operative one

As discussed earlier in section 3, a valid tag can become non-operative after reaching the threshold $T_{max_i}$ after being read by a valid reader. Moreover, an adversary can permanently incapacitate a tag by feeding arbitrary future timestamps and making the tag exceed the stored threshold value $T_{max_i}$.

We propose a bothway challange-response method where both the tag and reader authenticate each other and then the reader replaces $T_{max_i}$ sending the new value $T_{max_{inew}}$ to the tag. 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 non-operative. The reader sends a '1' and a hash of the last valid timestamp $T_{t_i}$ saved by the tag and $T_{max_i}$. Sending the value '1' indicates the reader's intention to change the $T_{max_i}$. Each tag has its own $T_{max_i}$ stored which is its secret value. An adversary needs to try $2^n$ combinations to find the exact value of $T_{max_i}$ where $n$ is the number of bits in $T_{max_i}$. If the received hash value matches with the computed hash value, the tag sends the response as $H(k_i, T_{max_i})$. Otherwise it generates a pseudo-random number PRNG. The random number generated must be indistinguishable from $H(k_i, T_{max_i})$. That means, the adversary has to face the decision problem of distinguising the $H(k_i, T_{max_i})$ from a random value. This indistinguishability featute is required to protect against narrowing attack [1] which leads into tracing the tag by an adversary. The use of PRNGs to obfuscate the tag identity was first introduced in [6]. The new timestamp threshold value $T_{max_{inew}}$ is stored in memory erasing the existing $T_{max_i}$ only after the reader authenticates itself to the tag. However, the reader generates $T_{max_{inew}}$ only when the tag makes sure that itself is a dead tag. The reader sends the value T by XOR-ing the $T_{max_i}$ and $T_{max_{inew}}$. This bothway authentication is done by using hash functions. And its security is dependent on the onewayness of the hash function. Note that, the tag does not need any extra circuitry for this hash computation. The proposed method is as follows:

**Algorithm 3 (Activating a non-operative tag)**
[1] *Tag* $\leftarrow$ *Reader:* $1, H(T_t, T_{max_i})$
[2] *Tag:*
[2.1] *if* $H(T_t, T_{max_i})$ *is valid*
[2.2] *compute* $MSG{=}H(k_i, T_{max_i})$
[2.3] *else* $MSG{=}\ PRNG_i$
[3] *Tag* $\rightarrow$ *Reader: MSG*
[4] *Reader:*
[4.1] *Generate* $T_{max_{inew}} > T_{max_i}$
[4.2] $T = T_{max_i} \oplus T_{max_{inew}}$
[4.3] *Set* $T_{max_i} = T_{max_{inew}}$
[5] *Tag* $\leftarrow$ *Reader:* $T$
[6] *Tag:*
[6.1] *Extract* $T_{max_{inew}}$ *from* $T$
[6.2] *Set* $T_{max_i} = T_{max_{inew}}$

## 5 Comparison of performance

We compare our work with the YA-TRAP* here. Table 1 shows the comparison of security features like Forward Security(For. Sec.), DoS resistance (DoS rest.), Tag Tracing (Tag Trc.)and whether tag bocomes non-operative (Tag NOp). Our proposed scheme enhances DoS resistance capability compared to YA-TRAP* by helping a tag to get operative from non-operative state. Table 2 is the comparison of cost. It includes tag's computaion(Tag comp), Tag to Reader communication (TtoR comm.) and Reader to Server communication (RtoS comm.) costs.

Table 1: Performance Comparison (Security)

| | For. Sec. | DoS rest. | Tag Trc. | Tag NOp |
|---|---|---|---|---|
| YA-TRAP* | yes | limited | no | yes |
| Our Scheme | *yes* | *enhanced* | *no* | *no* |

Table 2: Performance Comparison (Cost)

| | Tag comp | TtoR comm | RtoS comm |
|---|---|---|---|
| YA-TRAP* | 2 HASH 1 PRNG | 3 msg | 5n msg |
| Our Scheme | 2 HASH 1 PRNG | 2 msg | 2 msg |

## 6 Conclusion

In this paper, we improve the efficiency of an RFID authentication procotol YA-TRAP* from the view point of computational and communication cost. Both the tag to reader and reader to server communication costs are reduced to a constant number of messages which is very useful for batch-mode authentication environment. Moreover, we propose a scheme which provides two fold advantages: it enhances DoS resistance capability and it doesn't allow a tag to be non-operative.

## References


[1] Gene Tsudik. A Family of Dunces: Trivial RFID Identification and Authentication Protocols: Privacy Enhancing Technologies 2007, 45-61

[2] A. Juels. RFID security and privacy: A Research Survey. IEEE Journal on SelectedAreas in Communication, 24(2), February 2006.

[3] Peter H. Cole, Damith C. Ranasinghe. Networked RFID Systems and Lightweight Cryptography Raising Barriers to Product Counterfeiting. Springer, e-ISBN 9783540716419

[4] Adi Shamir. SQUASH - A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags. Fast Software Encryption 2008, 144-157

[5] J. Katz, A.Y. Lindell. Aggregate Message Authentication Codes. CT-RSA 2008,155-169

[6] 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 ' 03), 201-212