YA-SRAP: Yet Another Serverless RFID Authentication Protocol

Sheikh I. Ahamed[1], Farzana Rahman[1], Endadul Hoque[1], Fahim Kawsar[2], and Tatsuo Nakajima[2]

[1]*Ubicomp lab, MSCS Dept., Marquette University, USA*
[2]*Dept. of Computer Science, Waseda University, Japan*

Contact author: sheikh.ahamed@mu.edu

ABSTRACT

Several threats in RFID systems are obstacles to intermingle this technology into human lives. On the other hand ensuring flexible privacy mechanism has been an enormous challenge due to extremely inadequate computational storage of typical RFID tags. So in order to relieve tags from responsibility, privacy protection and security assurance was guaranteed by central server. In this paper, we describe serverless, lightweight, forward secured and untraceable authentication protocol for RFID tags. This authentication protocol safeguards both tag and reader against all major attacks without the intervention of server. Though it is very critical to guarantee untraceability and scalability simultaneously, here we are proposing a scheme to make our protocols more scalable via ownership transfer. To the best of our knowledge this feature is incorporated in the serverless system for the first time.

General Terms

Reliability, Security, Verification.

Keywords

Scalability; Ownership transfer; Serverless; SRAP.

1. INTRODUCTION

RFID (Radio Frequency Identification) is identified as the next revolutionary technology that will be utilized by various ubiquitous services to identify things and people. However, the expansion of RFID technology is limited because of security and privacy concerns. Conventional security primitives cannot be integrated in RFID tags as they have inadequate computation capabilities with extremely limited resources. So security and privacy issues must be addressed before the enormous deployment of RFID tags in omnipresent environment. That is why research community devoted themselves in search of appropriate authentication protocols that will ensure RFID privacy and security without compromising the cost. One goal of authentication is to ensure that only authorized reader can access a tag. As tags are under heavy threats of adversaries, it is also mandatory to make sure that the tag's reply is accurate.

All these security requisites were ensured by a central database so far. This central server based model has drawn much consideration and some of the outcomes are reflected in [13], [3], [12], [2], [9], [1], [10] and [7]. The fundamental architecture of RFID technology involves a tag, a reader (or scanning device), and a back end database. A reader scans a tag (or multiple tags simultaneously) and relays the information to a database. Other than the back end database, not even a reader is able to infer any information from tag's reply as it is encrypted. Database returns tag's data to the reader only after verifying both tag and reader. Here, only the central server can authenticate involved reader and tags. Actually in server based system, central server played an essential role and it was quite easy to check validity of tags or reader, which is very important for privacy protection and security issues. Consequently a malicious reader could hardly obtain precious information from tags in such a system.

The major drawback of central server based system is that the readers always have to be connected to the server, which limits usage of RFID systems in remote locations where connectivity with server cannot be ensured. Besides having a single database makes the whole system more vulnerable to privacy attacks. Central server has knowledge of all tag secrets and tag information. So if the database is collapsed by an adversary, entire user community's privacy is jeopardized. So a serverless RFID system was proposed in [11] noticing the shortcomings of server based system. This paper introduced RFID systems to the world where a gigantic central server is not a conqueror anymore.

In serverless system, because of the absence of server, a reader has to identify legitimate tags all by itself. At the same time, in order to get tag's data reader has to authenticate itself to the tag as legitimate.

In this paper, we tried to find solutions to the following questions: a) how a reader can identify legitimate tags without the help of server? b) how a tag identifies that the communicating reader is legitimate? c) how readers maintain scalability when quantities of tags tend to increase? Here, we propose a low cost serverless authentication protocol that provides solutions to the preceding questions. This protocol can protect user privacy and it is also secured against major attacks. And all these

characteristics are ensured without a back end server which makes our proposal suitable for immense application areas.

1.1 Our Major Contributions

Our major contributions in this paper are the following:

I. We propose a serverless, forward secure, reliable, untraceable mutual authentication protocol which ensures that tag and reader will authenticate each other.

II. Here, a tag will release its data to only an authenticated reader and reader can access only its authorized tags.

III. We are proposing ownership transfer in serverless protocol for the first time. Protocols should be designed so that after ownership transfer, none but new owner (reader) can access the tag. Our protocol provides this facility.

IV. In real life practical and scalable RFID protocols need to be designed as tag's quantities are assumed to be increasing. So we propose a way to make our protocol scalable so that it can keep pace with increasing number of tags.

V. Our protocol is secured against all the major attacks. Moreover we considered security of both tags and readers as both can be attacked by adversaries.

VI. In this paper, we point out some real life application challenges which require secured, low cost serverless RFID systems. We also proposed solutions to these challenges using our proposed protocols.

The remainder of the paper is as follows. Next section presents relevant related work on RFID privacy protection. Some major security requirements for RFID protocols are reflected in section 3. Section 4.1 provides some preliminaries for the rest of the paper. Section 4.2 and section 4.3 provides authentication protocols together with their security analysis. In section 4.4 we propose some additional features of our protocols. Some real life challenges and solution are discussed in section 5. And finally in section 6 some concluding remarks are reported.

2. RELATED WORK

Numbers of techniques have been proposed for ensuring RFID security and the assortment of authentication protocols is quite extensive. Moreover, to the best of our knowledge serverless authentication protocols were discussed only in [11]. Therefore here we shall refrain from a prevalent review and focus on those works that are most directly related to our contribution. Further interested readers may go through [8], [4] and [5].

Back end database played an essential role in most early works on RFID security. Researchers came up with highly secure authentication protocols but authentication was done mostly by the back end server rather than the reader itself. Among them Seo et al. [9] proposed a scalable and untraceable authentication protocol based on hash function. The most significant contribution of this paper is scalability and forward secrecy but the drawback is that ownership transfer requires external intervention. Seo et al. also proposed another authentication protocol [10] that ensures high scalability and ownership transfer. The protocol is based on Proxy and Universal Re-encryption which allows the back end server to get tag identifier only after a simple decryption that requires a constant time. This makes it one of the highest scalable authentication protocols. And it suffers problem of traceability, *Denial-of-Service* (DoS) attack and swapping.

YA-TRAP [12] is a famous authentication protocol that places little burden on the back end server and uses monotonically increasing timestamp which makes it secure against tracking but unsecure against DoS attack. Here, tags update their timestamp based on a value provided by the reader. At the same time each tag stores T_{max},which if reached, a tag does not answer to the reader's queries. Hence an adversary can send to the tag a large enough timestamp so that it goes beyond T_{max} which results in DoS attack. Although the solution to this attack was proposed in [1], it still lacks forward secrecy.

Another hash chain based RFID identification protocol is RIPP-FS [3]. Here Mauro et al. proposed that each tag shares a private symmetric key with the server. After each successful authentication tag and server both updates the symmetric key to maintain synchronization. One of the key features of RIPP-FS protocol is that the reader is free of "on-the-fly" computation while a tag reading is performed. It is also resilient to a specific DoS attack where the adversary attempts to exhaust the hash chain. The main flaw of this protocol is the formation of an infinite hash chain.

Another lightweight protocol is OSK [7]. Ohkubo, Suzuki and Kinoshita proposed that only two hash function H and G is sufficient to provides indistinguishability and forward secrecy, where H is a one way hash function and G has random oracle. According to this protocol a tag is initialized with a shared secret s_i and the back end server maintains a list of tags (id, s_i). Tag updates its secret key after each query according to the following formula $s_{i+1} = H(s_i)$. And in response to the query from a reader, tag replies $a_i = G(s_i)$. The server on the other hand uses this a_i to identify the tag by performing a brute force search through the list of tags. OSK does not ensure high scalability. In [1], Avoine and Oechslin modified OSK which removed the scalability problem. Another problem of OSK is that a malicious reader may easily desynchronize a tag which results in DoS attack.

In [11], Chiu et al. proposed a serverless authentication protocol. In this protocol reader maintains an access list L_i which is used for tag authentication purpose. And each tag has a secret t which is not shared with anyone. Reader and tag both know $f(r, t)$, where r is reader identifier. Here in

response to the query from a reader, tag replies with some of the bits of $h(f(r,t) \parallel n_i \parallel n_j))$ where n_i and n_j are two random numbers generated by the reader and the tag respectively and $h(.)$ is a one way hash function. Since only a legitimate tag can generate $h(f(r,t) \parallel n_i \parallel n_j))$, it works as tag's certificate to the reader. At the same time tag queries reader with a question string. Only a legitimate reader replies with valid answer string which introduces the reader as an authorized reader to the tag. Tag releases its data only after realizing that the reader is legitimate. But here again the reader has to do a lot of computation to find out id of the required tag. But their protocol 2 is not purely and strongly anonymous as they return tag id by performing XOR operation with hash value for authentication. Moreover, they didn't propose any technique for ownership transfer.

3. SECURITY REQUIREMENTS

RFID technology may bring spontaneous risks because of the proliferation of RFID tags. Number of research literatures has dealt with several privacy and security issues of RFID. In section 2 we have also noticed that some of them do not guarantee firm security. So here we point out the security goals that should be guaranteed by a protocol:

***Forward secrecy*:** An adversary compromising a tag will not be able to identify the previous outputs of the tag.

***DoS resiliency*:** *Denial-of-Service* (DoS) attack means an authorized entity is prevented from accessing its authorized entities. In order to ensure successful communication between a reader and its authorized tags, it should be guaranteed that an adversary cannot desynchronize them.

***Synchronization*:** Attacker should not able to update the key used by the tag or the reader to secure the communication.

***Privacy protection*:** A tag cannot be distinguished by an adversary without tampering it and realizing all its stored information.

Anti-tracking: It is tough for an adversary to track a tag if it does not have any information about the tag. But the adversary can track a tag, if the tag replies with a constant response each time it is queried. So protocols should be designed such that a tag neither reveals its id nor replies with constant response.

Anti-cloning: In order to clone a tag, an adversary needs to know the secret key shared by the tag with its authorized reader. So, to be secured against cloning attack, protocols should never reveal the shared secret key.

***Key secrecy*:** Unless the tag is tampered, an adversary cannot identify the secret key used by the reader or the tag in secure communication.

***Anonymity*:** Guaranteeing anonymity means an adversary will never be able to comprehend the id of a tag by listening to the communication between the tag and the reader.

Not susceptible to replay attack: Security must be ensured against replay attacks so that an adversary cannot impersonate a legitimate tag by replaying an eavesdropped message.

Lightweight: Protocols should be lightweight, i.e. they should be free from heavyweight computation.

4. AUTHENTICATION PROTOCOLS

We present two slightly different authentication protocols. In both protocols the tag identifier is not passed on to the reader. But the tag sends certifying information to the reader in such a way that only the authorized reader is able to find out the identifier of the tag.

4.1 Notation and Assumption

We refer an RFID reader as R. Each R has a unique identifier r and a contact list $\mathcal{L}$. We will describe the contents of $\mathcal{L}$ later. R obtains r and $\mathcal{L}$ from a trusted center, TC, after authenticating itself. The TC is a trusted party who deploys all the RFID tags and authorizes any RFID reader. For the sake of simplicity we assume that R and TC communicate through a secure channel. On the other hand, each RFID tag T contains a unique identifier id and a unique secret t in its nonvolatile memory.

Subscripts are used to describe a particular R or T and their respective variables. Thus a particular RFID reader i will be R_i with an identifier r_i and contact list $\mathcal{L}_i$ stored in its nonvolatile memory. An RFID tag j is T_j having a secret t_j. The contact list $\mathcal{L}_i$ contains information about the tags which R_i has access to. And the information about each tag comprises a seed and the id of the tag. If R_i is authorized to access tags $T_1, \cdots, T_n$, $\mathcal{L}_i$ will take the following shape after authenticating itself to TC,

$$\mathcal{L}_i = \begin{Bmatrix} seed_1 : id_1 \\ \cdots \quad : \cdots \\ seed_n : id_n \end{Bmatrix}$$

where, for any tag T_j and $1 \le j \le n$, $seed_j$ is a seed used by R_i to communicate with T_j and $id_j$is T_j's identifier. $seed_j$ is initialized by $seed_j = f(r_i, t_j) = h(r_i \parallel t_j)$ where $h(.)$ is a one way hash function and $\parallel$ represents concatenate. Note that R_i does not know the tag secret t_j. R_i only knows the outcome of the function $f(r_i, t_j)$ as $seed_j$. The initial $seed_j$ is computed by TC and stored in R_i.

On the contrary, the tag T_j will contain only one seed for its only one authorized reader R_i. While deploying the tag T_j by TC, T_j will get $f(r_i, t_j) = h(r_i \parallel t_j)$ as $seed_{T_j}$ from TC. T_j stores $seed_{T_j}$ in its nonvolatile memory. We also assume that the TC cannot be compromised. Moreover our assumptions also include that all the readers once

authenticated by the TC are trusted. And we denote an adversary as ϱ.

All readers and tags have knowledge of a pseudorandom number generator $\mathcal{P}(.)$ and a function $\mathcal{M}(.)$. $\mathcal{P}(.)$ takes a seed as an argument and outputs a pseudorandom number according to its distribution. $\mathcal{M}(.)$ is used by all readers and tags to update the seed of the pseudorandom number generator by passing the current seed as input. We assume $\mathcal{M}(.)$ as an irreversible one way hash function. Therefore a current seed cannot be linked to its previous one.

4.2 Authentication Protocol 1

(1) $R_i \rightarrow T_j$: $request$
(2) T_j : $n_j = \mathcal{P}\left(seed_{T_j}\right)$
(3) $R_i \leftarrow T_j$: n_j
(4) R_i : $n_i = rand$
(5) for all m from 1 to n
//run through list $\mathcal{L}_i$
(6) Let $n_m = \mathcal{P}(seed_m)$
(7) if ($n_m == n_j$) then
(8) Let $s = \mathcal{M}(seed_m)$
(9) $n_i = \mathcal{P}(s)$
(10) $seed_m = \mathcal{M}(s)$
(11) $R_i \rightarrow T_j : n_i$
(12) T_j : Let $k = \mathcal{M}\left(seed_{T_j}\right)$
(13) Let $a = \mathcal{P}(k)$
(14) if $(a == n_i)$ then
(15) $seed_{T_j} = \mathcal{M}(k)$

4.2.1 Protocol Description

At the beginning the reader R_i transmits a $request$ to the tag T_j. To authenticate itself to R_i, T_j uses $seed_{T_j}$ to generate n_j which is pseudorandom number. Now T_j transmits n_j to the reader. Only a legitimate tag can accurately generate n_j. At the reader side, n_i is initialized with a $rand$ number. For each tag T_m, R_i generates next pseudorandom number as well as compares each of the generated number with n_j received from T_j. Note that '//' denotes in inline comments. Each pseudorandom number is based on the corresponding $seed_m$ of tag T_m which the reader obtains from its $\mathcal{L}_i$. R_i authenticates T_j if there is a match with n_j. Only then the next pseudorandom number for that tag is computed after updating the corresponding seed. The updated seed is again updated and stored in the list $\mathcal{L}_i$. Finally the reader R_i transmits n_i to the tag T_j. If a match is found, then n_i will be the produced next pseudorandom number. Otherwise, n_i contains $rand$. Now T_j generates next pseudorandom number with the updated $seed_{T_j}$ and compares the number with n_i sent by R_i. T_j authenticates R_i only if a match occurs. And this match causes T_j to update the already updated seed again and store in $seed_{T_j}$. In this way both parties have same seed after a successful authentication which guarantees synchronization.

4.2.2 Problem of Protocol 1

This protocol has some major security problems that are discussed below:

Tracking: Here, ϱ tries to track T_j over time. ϱ succeeds if it is able to distinguish T_j from other RFID tags. ϱ usually performs this attack by repeatedly querying T_j. Those queries will yield consistent reply. This consistent reply becomes a signature of T_j. In protocol 1, if ϱ queries T_j contiguously for at least two times, T_j will reply with the same answer. We assume that no successful authentication is done in the mean time. As T_j will use the same seed to produce pseudorandom number, the generated numbers will be same. Therefore ϱ will be able to track T_j. Thus protocol 1 is not protected against tracking.

***De-synchronization*:** Protocol 1 suffers from de-synchronization problem which is caused by replay attack. An adversary ϱ is able to observe all the interaction between R_i and T_j. In other words, ϱ can eavesdrop any challenge-response. Now, by querying T_j, ϱ gets a reply with valid n_j. Next time whenever R_i tries to access T_j, ϱ impersonates T_j and replies with the learned valid T_j to attack R_i. As n_j is valid, R_i finds a match and updates the seed in its list. Whereas T_j is totally unaware of the authentication between R_i and ϱ. Hence, the seeds become de-synchronized between R_i and T_j. As a result, R_i can never authenticate T_j in future transactions.

4.3 Authentication protocol 2

(1) $R_i \rightarrow T_j$: $request, rand_i$
(2) T_j : $n_j = \mathcal{P}\left(seed_{T_j} \oplus \left(rand_i \parallel rand_j\right)\right)$
(3) $R_i \leftarrow T_j$: $n_j, rand_j$
(4) R_i : $n_i = rand$
(5) for all m from 1 to n
//run through list $\mathcal{L}_i$
(6) Let $n_m = \mathcal{P}\left(seed_m \oplus \left(rand_i \parallel rand_j\right)\right)$
(7) if ($n_m == n_j$) then
(8) Let $s = \mathcal{M}(seed_m)$
(9) $n_i = \mathcal{P}(s)$
(10) $seed_m = \mathcal{M}(s)$
(11) $R_i \rightarrow T_j : n_i$
(12) T_j : Let $k = \mathcal{M}\left(seed_{T_j}\right)$
(13) Let $a = \mathcal{P}(k)$
(14) if $(a == n_i)$ then
(15) $seed_{T_j} = \mathcal{M}(k)$

(16) else
(17) *Reader is not authorized or is an adversary*

4.3.1 Protocol Description

Protocol 2 is improved version of protocol 1. At the beginning, R_i transmits a $request$ and $rand_i$ to Tag T_j. T_j generates n_j by using $seed_{T_j}$, $rand_i$ and a random number $rand_j$ generated by itself. After receiving n_i and $rand_j$, R_i computes $\mathcal{P}\left(seed_m \oplus (rand_i \parallel rand_j)\right)$ for each tag T_m in the list $\mathcal{L}_i$, where $1 \leq m \leq n$. Here '//' denotes inline comment. If R_i finds a match, it changes the value of n_i from $rand$ to a pseudorandom number produced by $\mathcal{P}(.)$ and updates the seed in $\mathcal{L}_i$. Then R_i sends n_i to T_j. If a match does not occur, R_i sends n_i with the value $rand$ and concludes that it is a fake tag. Now, T_j generates the next pseudorandom number and compares it with n_i. If the two numbers are same, T_j updates its seed and concludes R_i as the authorized reader. But in case of a mismatch, T_j decides that the reader is not authorized to access it or the reader is indeed an adversary. Here a major improvement is to be noticed that both the reader and the tag update their seeds only when they are sure about the validity of the opposite party.

4.3.2 Why Protocol 2?

The privacy and security problems of protocol 1 exhibit the necessity of a new protocol that can protect reader or tag against those attacks. Besides overcoming the problems of protocol 1, protocol 2 is secure against other attacks that we will discuss elaborately in section 4.3.3. Here we just explain how protocol 2 is secure against the problems of protocol 1.

Tracking: By incorporating a $rand_i$ and $rand_j$, protocol 2 is secured against tracking. By exploiting the power of eavesdrop-ing, an adversary ϱ can listen all the messages between R_i and T_j. ϱ fails to track T_j by replaying same $rand_i$ learned from any previous challenge-response because T_j replies with different response, due to $rand_j$, each time it is queried.

De-synchronization: In protocol 2 seeds are updated when both the reader and the tag are certain about their validity. After listening to all the messages between R_i and T_j, ϱ queries T_j with a $rand$ and T_j replies with a n_j. Whenever R_i again tries to access T_j, ϱ impersonates T_j by replying with the learned n_j. To make itself legitimate to R_i, ϱ has to use either of the two particular values for $rand$ while communicating with T_j to collect n_j. One of them is correct next $rand_i$ that R_i will generate in the transaction in which ϱ tries to imitate T_j. And the other value is a previous $rand_i$ listened by ϱ before. Unfortunately, guessing of a random number generated by one party is impossible for other party. On the other hand, by using previous $rand_i$, ϱ fails since R_i will not use the same $rand_i$ in future again. Therefore, ϱ attempts in vain to break the synchronization of seeds between a legitimate R_i and T_j.

4.3.3 Security Analysis of Protocol 2

Cloning: Here ϱ queries T_j several times and places its response in a fake tag. Let this fake tag be $\widehat{T_j}$. ϱ wants to counterfeit a legitimate tag and it becomes successful if it can fool a legitimate reader R_i. Under our protocol whenever the adversary queries T_j, it gets a different response because of $rand_i$ and $rand_j$. Now if ϱ places this response in $\widehat{T_j}$ it will never be able to fool an honest R_i. When $\widehat{T_j}$ is queried by honest R_i, $\widehat{T_j}$ will reply with a value that will not match with the one generated by original T_j. This is because R_i will now have a different $rand_i$. Moreover, $\widehat{T_j}$ cannot generate the actual response as it does not know the current seed stored in the tag. Guessing the response will not help either as it has a very low probability. Next we consider the case when ϱ tries to clone a tag by eavesdropping between a tag and reader.

Eavesdropping: Here ϱ eavesdrops the communication between R_i and T_j and later uses these to create a fake reader $\widehat{R_\iota}$ or a fake tag $\widehat{T_j}$. Under our protocol this attack is not possible because after a successful communication between R_i and T_j, both have changed their seeds. ϱ will learn $rand_i$ and $rand_j$ from the whole communication. But the response $\mathcal{P}\left(seed_{T_j} \oplus (rand_i \parallel rand_j)\right)$ of a tag requires random numbers generated by two parties. But ϱ impersonating R_i or T_j cannot control random number generated by the other party. Even knowing correct $rand_i$ and $rand_j$ is useless as ϱ needs correct seed to generate correct response.

Suppose, ϱ impersonates tag T_j which we name $\widehat{T_j}$ and it wants to fool an honest reader R_i with which T_j had communicated recently. Now $\widehat{T_j}$ will not be able to fool R_i as R_i will definitely provide with a different $\widetilde{rand_\iota}$. And $\widehat{T_j}$ cannot generate $\mathcal{P}\left(seed_{T_j} \oplus (\widetilde{rand_\iota} \parallel rand_j)\right)$ as it does not know the $seed_{T_j}$. Even if $\widehat{T_j}$ replays $\mathcal{P}\left(seed_{T_j} \oplus (rand_i \parallel rand_j)\right)$, reader will easily identify that it is a fake tag. Therefore we can say that as every communication involves random number generated by two parties and current seed, ϱ cannot launch a replay attack using pervious values.

Physical attack: Physical attack means ϱ can compromise either tag or reader. We will consider each case. We will also assume that once ϱ compromises R_i or T_j it will learn everything about the tag or reader.

A. ϱ compromises R_i: When adversary compromises a reader R_i , adversary will know reader's contact list $\mathcal{L}_i$ and id r_i. It can now impersonate R_i and communicate with T_j. ϱ can counterfeit a tag T_j residing in its contact list $\mathcal{L}_i$, which we will name $\widehat{T_j}$. Adversary will be successful if $\widehat{T_j}$ can fool another legitimate reader R_x. But under our protocol $T_j$is authorized to only R_i. So, $\widehat{T_j}$ cannot fool R_x by learning only $seed_j$.

B. ϱ compromises T_j: In this case, adversary compromises tag T_j and learns $seed_{T_j}$ that it shares with reader R_i. From this information ϱ will want to create a fake tag T_x which will communicate successfully with an honest reader R_i, where T_x resides in contact list $\mathcal{L}_i$. Each RFID tag shares a seed with its authorized reader. So T_x will share a different seed with R_i which is not known by T_j. So even if ϱ knows $seed_{T_j}$, it cannot derive the seed shared between R_i and T_x and therefore ϱ cannot create a fake tag to fool R_i.

Denial of service (DoS): In this case, ϱ does not want to derive any information or tries to impersonate. Its main target is to ensure that a reader cannot access its authorized tags. This is a severe problem where back end database shares a secret key with tags. And the key has to be synchronized for successful communication. Our protocol eliminates need of a back end server. So synchronization between server and tag is not obligatory. Moreover in our protocol both tag and reader updates their shared seed only after becoming certain of the other end's validity. Under our protocol an adversary can never proof himself as legitimate and thus he can never desynchronize tag or reader.

Privacy protection: People carrying various tagged item do not want to hamper their privacy. It means that tagged object should reveal their ID to none but authorized reader, otherwise malicious readers may cause several vulnerabilities to owner's day to day life. Our protocol protects user's privacy strongly. Since under our protocol a tag never send it's id to anyone, not even reader. It sends its reply in disguise so that only an authorized reader can identify a tag.

Anonymity: The problem of leaking information about user possessions occurs if anonymity is not ensured. To ensure id anonymity, a tag should never output it's id directly nor should reply with constant data. Our protocol is totally anonymous in a sense that a tag never replies with its id, not even by encrypting it for security. Rather a tag replies in a special format so that only an authorized reader is able to find out tag's id from its contact list $\mathcal{L}_i$.

Key secrecy: Our protocol ensures key secrecy as both tag and reader uses a shared seed to secure the communication. This seed is updated in both sides only after successful authentication using a one way hash function. So an adversary ϱ can never recover seed listening to the communication between a tag and reader.

Forward secrecy: Forward secrecy means if anyhow an adversary compromises a tag, it will not reveal any data previously transmitted by that tag. It means that if ϱ physically tampers T_j and learns $seed_{T_j}$shared with R_i, ϱ will not be able to trace the data back through past events in which they were involved. Our protocol ensures strong forward secrecy as seed update function $\mathcal{M}(.)$ is an irreversible one way hash function. So ϱ tampering T_j cannot know former outputs based on former seed as it cannot derive previous seed from the current seed .

Lightweight: Our authentication protocol is lightweight and low cost in a sense that they require only random number generator, concatenation and hash function generation capability.

4.4 Additional Features

Ownership transfer: Ownership transfer ensures that an authorized reader renounces the authority of a tag and a new reader gets the authority to access the tag. In other words, a tagged object will continue to be authenticated by only a new authorized reader. However the old authorized reader is no more accredited to access it. Suppose R_i is the current owner of tag T_j. After transferring ownership to another reader R_x, T_j responds to R_x in the same way as it did to R_i. From now on R_i has no rights to access T_j. Ownership transfer is a prominent property that facilitates many RFID applications. As far as we know ownership transfer issue is dealt with only in [6] and [9]. In both of them back end server played a significant role. To the best of our knowledge, we are proposing ownership transfer in serverless system for the first time. Based on our protocol, two methods of ownership transfer are proposed next.

A. TC based ownership transfer: TC (Trusted Center) has all the responsibility regarding every type of management. A reader gets its contact list $\mathcal{L}$ from TC using a secure channel at the beginning of its operation. Whenever a reader faces the need to transfer the ownership of a particular tag to other reader, it informs the TC about the change in access policy and ownership information of that tag. Ownership information comprises the identifier and the corresponding seed for the particular tag stored in the reader's list. TC will now authenticate new owner (other reader) and authorize it by updating the contact list of new owner with ownership information. On the other hand, TC will also delete the ownership information of that tag from the old owner's contact list. For example, id_j and current $seed_j$ for tag T_j will suffice as ownership information. Old owner transmits this ownership information to TC at the time of informing about a change in ownership of T_j.

B. Serverless ownership transfer: Previous method is not a feasible solution of transferring ownership as it requires

intervention of TC for every ownership transfer. So we remove the necessity of TC in transferring ownership by introducing serverless ownership transfer. The salient feature of this method is "reader - reader secure communication". At the time of ownership transfer, old owner transmits id_j and current $seed_j$ for the particular tag T_j to new owner and then simply eradicates owner-ship information for that tag from the non-volatile memory of the old owner. Therefore old owner has no valid seed to access T_j while the seed for the new owner and tag T_j still remain synchronized.

Scalability: Scalability means that a reader can find a tag's identifier with constant computational time regardless of the number of tags that is owned by it. So one solution of ensuring scalability can be - each reader will own moderate number of tags which will make the search scalable. If the number of tags in a reader is p, the time complexity of search operation will be $O(p)$, where to make the search scalable p needs to be small enough.

But Juels and Weis proved in [5] that improved randomized hash lock offer strong privacy and security at the cost of poor scalability. In fact the authors in [5] proposed that in case of protocols that protect privacy using symmetric key cryptography, reader or server has to perform exhaustive search to find out a tag's identity. They also advocated the protocols that are more rational but weaker in privacy protection. Hence we entirely comply with their observation and propose a more practical way of ensuring scalability with the help of ownership transfer.

Our proposal is that each reader will have a threshold θ. Here θ is the maximum number of tags' ownership information that can reside in a reader to ensure scalability. When a reader's contact list surpasses threshold θ, the reader called as overloaded reader wishes to reduce its burden. So if the overloaded reader has a co-operative reader (not an adversary) within its radio range and if the co-operative reader has enough memory to accommodate the overloaded reader's load, the overloaded reader will transfer some of its burden to the other one. Therefore by only transferring ownership to a co-operative reader, an overloaded reader's contact list may again become scalable.

5. ILLUSTRATIVE EXAMPLE

The major strength of a server less protocol is support for mobile and outdoor applications where the existences of a dedicated server or a communication channel are often impractical. In this section, we present four such application scenarios and contemplate the effectiveness of our proposed protocol.

a) Container recognition in off-site location*:* Let us consider a case in which a company uses RFID system for employee identification, human authentication while entering into safety regions, document management, product maintenance and etc. All these services are easily ensured with central server based RFID system. But this company faces problem when they have to collect their ordered raw material containers from other companies that belong to the off-site locations. This off site location has no connection with the central server. Normally truck drivers are dispatched to the other companies to collect container deliveries. But it is a very usual case that people employed in this job does not have the capability to ensure that the supplied containers are the correct one that were ordered by his company. Moreover it is not possible to check each container individually because obviously there are enormous numbers of containers. As a result, containers being unchecked, sometimes wrong material are delivered to the warehouse. This causes a loss for the particular manufacturing company. Now this problem can be easily eliminated by using our serverless protocol provided that the containers are tagged objects. The truck driver may have with him his personal PDA, which can act as a reader. Reaching the offsite location this reader can easily authenticate the containers and find out whether they are the ordered containers or not. This can be easily done as under our protocol, readers can authenticate and communicate with tags without the intervention of central server.

b) School children tracking while away from school: School children are often taken at various education tours in different places. Tracking children is already possible with the existing server based RFID systems. But tracking children in picnic spots or places where children are taken on education tours is difficult because of the unavailability of the central database. But our protocol can perform here successfully as it can authenticate and identify any children without the help of central database. Tags are attached to the identity card of the children and the PDAs of the teachers can act as a reader. Then by using our protocol teachers can easily track and identify children or even find missing children.

c) Environmental monitoring: The use of RFID systems in conjunction with highly miniaturized sensors will make it possible to observe diverse environmental phenomena. Environmental scientists perform diverse research on environment by attaching tags with animals and releasing them in the wild again. These attached tags together with our serverless protocol can help scientists on their research. Moreover, sometimes it becomes necessary to regain a tagged animal from the wild for research purpose. In this case our protocol can be very useful as readers can track or locate the tagged animal in the wild without the need of server.

d) Authenticating smart objects usage at construction site: Several research groups have been investigating applications of smart objects in outdoor working sites where regular tools are augmented for supplementary services. For example: in [14], construction drill machines

are augmented so that usage history can be monitored and usage safety can be ensured by appropriate alerts. Such augmentations have direct implications in the business and logistic processes of the companies since they use performance record of the workers. Our proposed protocol can be applied to such scenarios to authenticate the workers to use the smart tools and to enable secure logging of monitoring data locally which ensures their privacy

6. CONCLUSION

One of the major challenges for RFID technology is to provide benefits without negotiating privacy and security. Many solutions have been recommended but almost as many ways have been found to crack them. While there are several existing methods, none of them provide a complete solution. In this paper we suggested serverless authentication protocols which ensure that both tag and reader are authenticated at the time of communication. Our authentication protocol is lightweight, forward secured and shielded against some major attacks like: tracking, cloning, eavesdropping, physical tampering, and DoS attack. Moreover we also suggested ownership transfer mechanism which facilitates our protocol to be scalable. To the best of our knowledge, this is the first contribution in the literature that enables serverless protocols to perform ownership transfer. We also discussed application of our proposed serverless protocol in some real life examples. The application of our protocol is not limited to these examples only, but it can also be applied to some other real life circumstances.

One future avenue of our work is applying the proposed protocol in real life applications. In section 5, we have provided several outdoor application scenarios where RFID in conjunction with other smart objects are contemplated for effective service provisions. Our proposed authentication protocol can seamlessly integrate into these services to make them more secure and protected. We are currently working on applying our protocol in multiple application scenarios and hope to present some exciting results in near future.

7. REFERENCES

[1] Avoine, G., and Oechslin, P. A Scalable and Provably Secure Hash Based RFID Protocol. In *International Workshop on Pervasive Computing and Communication Security (PerSec '05)*, IEEE, IEEE Computer Society Press, Kauai Island, Hawaii, USA, March 2005, pp. 110–114.

[2] Burmester, M., Le, T. v., and Medeiros, B. d. Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In *Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, IEEE, Baltimore, Maryland, USA, August 2006.

[3] Conti, M., Pietro, R. D., Mancini, L. V., and Spognardi, A. RIPP-FS: an RFID Identification, Privacy Preserving Protocol with Forward Secrecy. In *International Workshop on Pervasive Computing and Communication Security (PerSec '07)*, IEEE, IEEE Computer Society Press, New York, USA, March 2007, pp. 229-234.

[4] Juels, A. RFID Security and Privacy: A Research Survey. *RSA Laboratories*, September 2005.

[5] Juels, A., and Weis, S. Defining Strong Privacy for RFID. *Cryptology ePrint Archive, Report 2006/137*, IACR, April 2006.

[6] Molnar, D., Soppera, A., and Wagner, D. A Scalable, Delegatable Pseudonym Protocol Enabling Owner-ship Transfer of RFID Tags. *In Proceedings of Selected Areas in Cryptography (SAC 2005), 3897*, Springer-Verlag, Kingston, Canada, August 2005, pp. 276-290.

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

[8] Rieback, M., Crispo, B., and Tanenbaum, A. Is Your Cat Infected with a Computer Virus? In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'06)*, Pisa, Italy, IEEE, IEEE Computer Society Press, March 2006.

[9] Seo, Y., and Kim, K. Scalable and Untraceable Authentication Protocol for RFID. In *International Workshop on Security in Ubiquitous Computing Systems (Secubiq '06)*, Springer-Verlag , Seoul, Korea, August 2006.

[10] Seo, Y., Lee, H., and Kim, K. A Lightweight Authentication Protocol Based on Universal Re-encryption of RFID Tags. 2006.

[11] Tan, C. C., Sheng, B., and Li, Q. Severless Search and Authentication Protocols for RFID. In *Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom '07)*, New York, USA, March 2007.

[12] Tsudik, G. YA-TRAP: Yet another Trivial RFID Authentication Protocol. In *International Conference on Pervasive Computing and Communications (PerCom '06)*, IEEE, IEEE Computer Society Press, Pisa, Italy, March 2006.

[13] Vajda, I., and Butty´an, L. Lightweight Authentication Protocols for Low-Cost RFID Tags. In *Second Workshop on Security in Ubiquitous Computing (Ubicomp '03)*, Seattle, WA, USA, October 2003.

[14] G. Kortuem, N. Davies, C. Efstratiou, K. Kinder, M.I. White, R. Hooper, J. Finney, L. Ball, J. Busby, and D. Alford, "Sensor networks or smart artifacts? an exploration of organizational issues of an industrial health and safety monitoring system," *9th International Conference on Ubiquitous Computing (UbiComp 2007), 2007*.