On Security and Privacy Enhanced Authentication of RFID

By Mohammad Shahriar Rahman

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Atsuko Miyaji

March, 2009

On Security and Privacy Enhanced Authentication of RFID

By Mohammad Shahriar Rahman (710079)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Atsuko Miyaji

and approved by
Professor Atsuko Miyaji
Professor Mineo Kaneko
Professor Kunihiko Hiraishi

February, 2009 (Submitted)

Contents

Chapter 1

Introduction

Cryptography has long been used, practiced and studied to hide secret information. When cryptography is used in RFID tag deployment, concerns rise about the fundamental resources required by an algorithm. The security threats in an RFID tag spans from issues like tracking, denial of service, data authentication, tag and/or reader authentication, communication efficiency, through to a burning issue:- consumer privacy. The range of security threats and their importance varies from application to application.

Cryptographic algorithms are mainly classified into two classes according to how they use key material. If the parties involved in a cryptographic communication are using secret-key or symmetric cryptography then they will share the same key material. If the two parties are using public-key or asymmetric cryptography, then they will use different key material. In this case the key used by the sender of an encrypted message will be publicly available. Considering the diversified functionality, it can be said that either type of cryptography might be used in principle for RFID. However, one significant difference between them is the supporting infrastructure that they require. Such considerations may well make one type of cryptography far more suitable than the other. A second difference, and one that is often cited in RFID tag applications, is that symmetric algorithms tend to require less computational resources than asymmetric algorithms.

1.1 RFID Security and Privacy

Silent physical tracking of consumers and inventorying of their possessions are the risks with RFID implementation. Again, for industries/businesses, new privacy and security risks can end up in corporate espionage. The technology may also allow real-time tracking of customers in stores, or even after they leave stores. Furthermore, there can be cases like retailers and other companies are tracking consumers long after a consumer purchases an item. As shown in figure 1.1, in an RFID environment, the reader to server communication is assumed to be secure. On the other hand, the communication channel between a tag and a reader is considered to be insecure. Tags are usually kept by the owners/users/consumers. Whereas the server-side is the party where the application is mounted and run to verify the identities of different tags. As the communication between

a reader and a tag is through an insecure channel, this is clearly under threat of various attacks launched by an adversary or unwanted entity.

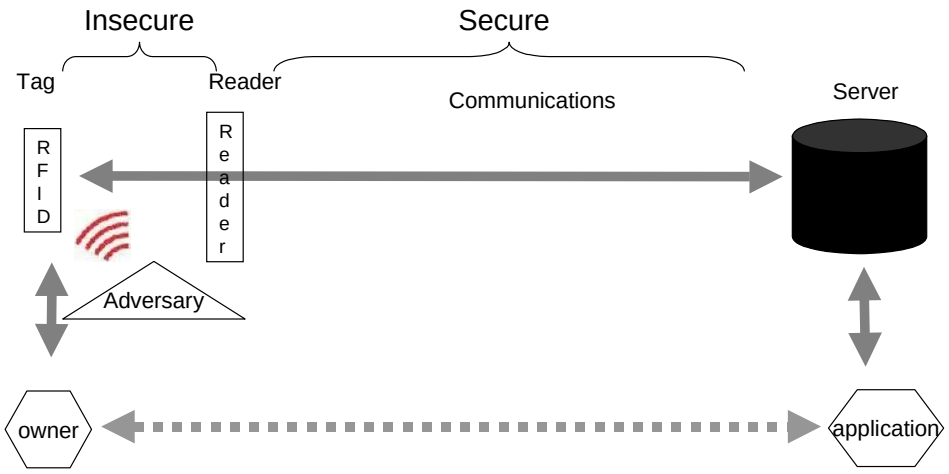

Figure 1.1: Security Problem in RFID System

If unique RFID devices pass beyond the point of sale and are carried out into the consumer's world, they pose a strong threat to privacy. The unique ID in a garment or a car could be read silently by any organization and associated with an individual, allowing subsequent re-identification by that organization (or indeed by any other organization to which the data was sold). The organization doing the surveillance need not be the manufacturer or retailer. As readers cost as little as USD 20, a hobbyist snoop or private investigator could even set one up near a doorway to record people who revisit an area.

If common items such as clothing, wallets and car tires become traceable, marketers will certainly start installing RFID readers in entrances to stores and car parks, to obtain more information about visitors. Once one has been identified with a bugged item he is carrying, that item can give him away anywhere. If one makes a purchase with a credit or loyalty card, the seller could link hie identity with the RFID number of any tagged articles he is carrying, and use it later or even sell that linking information to other organizations. Vast databases of records of people's movements would become available to telemarketers, government investigators and divorce lawyers. This scenario must be avoided, by ensuring that no RFID devices contaminate the consumer world.

Many works on security and privacy of RFID has been done is recent past[8, 10, 18, 20, 34, 3, 16]. Some works have been done targeting specific RFID standard EPC Class-1 Generation-2 [35, 19, 16, 11, 22].

1.2 Contribution

In this thesis, we have studied the recent privacy preserving RFID authentication protocols. We also studied the vulnerabilities and threats in the RFID. Although many works have been done recently on RFID security and privacy, a few works have focused on reducing the communication cost in the RFID environment [13, 21]. We have introduced a two-way message authentication protocol where both tag and reader authenticate each other. Compared to the previous RFID authentication protocols, our protocol is an improvement 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. In our two-way authentication protocol, a reader helps a tag to recover from the non-operative state. Furthermore, our scheme provides reader authentication to thwart tag cloning.

In the extended version of our scheme, the reader uses partial authentication to keep the rogue tags out of the aggregate function. A simple aggregate function enables a server only to find out the anomaly of the resultant XOR operations of the hash values. That means, if the pre-computed aggregate hash value does not match with the received aggregat hash value, 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. 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. However, the level of security is improved compared to the previous works as it provides resistance against DoS, Cloning, Replay, Tracking attacks. It also provides forward security and does not allow a tag to become non-operative unless explicitly done by the reader.

1.3 Outline of this Dissertation

In chapter 2, we give the preliminaries on RFID environment, define the notations and different attacks. In the next chapter, we briefly describe some of the recent works and compare their performance. Chapter 4 is a preliminary version of our proposed RFID authentication scheme. Next, chapter 5 describes our two-way RFID authentication protocols. In chapter 6, we compare our works with the previous works and we describe the features of our works. Finally, in chapter 7, we put the concluding remarks and possible future research directions. The chapter 8, includes the list of the publications made from the parts of our works.

Chapter 2

Preliminary

2.1 Radio Frequency IDentification

Radio Frequency IDentification (RFID) is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags or transponders. RFID's wireless identification capabilities promise to revolutionize our industrial, commercial, and medical experiences that is envisioned as the replacement for traditional barcodes. 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 radiowaves. Some tags can be read from several meters away and beyond the line of sight of the reader. Most RFID tags contain at least two parts. One is an integrated circuit for storing and processing information, modulating and demodulating a (RF) signal, and other specialized functions. The second is an antenna for receiving and transmitting the signal. A technology called chipless RFID allows for discrete identification of tags without an integrated circuit, thereby allowing tags to be printed directly onto assets at a lower cost than traditional tags.

2.1.1 RFID Environment

In an RFID environment, an RFID system consists of three components: tags, reader and server.

RFID tags come in three general varieties:- passive, active, or semi-passive (also known as battery-assisted). Passive tags require no internal power source, thus being pure passive devices (they are only active when a reader is nearby to power them), whereas semi-passive and active tags require a power source, usually a small battery.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 sends its corresponding information when interrogated by a valid reader. LF tags (125-135 kHz) can typically be read up to 30 cm away, HF tags (13.56 MHz) up to 1 m away, UHF tags (2.45 GHz) up to 7 m away, and active tags up to 100 m away or more.

Reader(s) communicate with a server and with the tags. They are responsible of per-

forming 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. 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.

2.1.2 Passive RFID Tags

Passive RFID tags have no internal power supply. The minute electrical current induced in the antenna by the incoming radio frequency signal provides just enough power for the CMOS integrated circuit in the tag to power up and transmit a response. Most passive tags signal by backscattering the carrier wave from the reader. This means that the antenna has to be designed both to collect power from the incoming signal and also to transmit the outbound backscatter signal. The response of a passive RFID tag is not necessarily just an ID number; the tag chip can contain non-volatile, possibly writable EEPROM for storing data.

Passive tags have practical read distances ranging from about 10 cm (4 in.) (ISO 14443) up to a few meters (Electronic Product Code (EPC) and ISO 18000-6), depending on the chosen radio frequency and antenna design/size. Due to their simplicity in design they are also suitable for manufacture with a printing process for the antennas. The lack of an on-board power supply means that the device can be quite small: commercially available products exist that can be embedded in a sticker, or under the skin in the case of low frequency RFID tags. A passive RFID may also have privacy enhancing technologies built-in including built-in firewall access controls, communication encryption and a silent mode ensuring that the consumer at point of sales can get exclusive control of the key to control the RFID. The RFID does not respond unless the consumer authorizes it, the consumer can validate presence of a specific RFID without leaking identifiers and therefore the consumer can make use of the RFID without being trackable or otherwise leak information that represents a threat to consumer privacy. Projections on the likely resources in several years of Class 1 RFID tags with cost in the vicinity of USD 0.05 include several hundred bits of memory and somewhere between around 1000-10,000 gates with around 200- 2000 gates being available for security features. Few gates are available for security functionality. Thus such RFID tags may be expected to perform some basic computational operations, but not conventional cryptographic ones. Even hardware-efficient symmetric-key encryption algorithms are well beyond the reach of RFID tags of this kind.

2.1.3 Applications of RFID

Low cost RFID tags promise to accelerate supply chain processes, from moving goods through loading docks to managing the huge data collected from these goods. The US Department of Defense are already conducting RFID trials at the pallet, case, and item level and have issued a mandate to its suppliers that each item sold to them must be marked with a passive radio frequency identification (RFID) tag. Procter and Gamble, and the U.S. Department of Defense are to deploy RFID as a tool for automated oversight

of their supply chains [4]. Automatic Payment Automatic payment is another popular application of RFID. Wal-Mart is stepping up pressure on suppliers to comply with its 3-year-old inventory-technology mandate based on RFID. Beginning from January 2009, it will charge suppliers a USD 2 fee for each pallet they ship to its Sam's Club distribution center in Texas that doesn't have an RFID tag.

Various industry sectors have conducted trials of RFID-enhanced cashless payment technology, from RFID-augmented credit cards and public transportation tickets, to RFID-like Near Field Communications (NFC) in consumer devices. Electronic toll collection using EZ-Pass is widespread which enables the automatic deduction of the toll from a prepaid account. The active EZ-Pass transponder, attached to a car windshield, sends account information to a toll plaza as the car drives underneath.

The United States Department of Homeland Security (DHS) and the International Civil Aviation Organization (ICAO) plan to use passive RFID to police access at airports. The ICAO wants to replace all (approx. 1 billion) passports with digital passports, which store encrypted biometric data on an RFID chip, by 2015. The DHS also wants to use passive RFID to record who is entering/leaving the U.S. across land routes. Suica, Felica cards are already hugely popular in Japan as prepaid automatic rail pass.

RFID-based animal tracking has been used now-a-days to monitor cows, pigs, cats, dogs. Being aware of the number and location of their stock at all times allows ranchers and other livestock producers to optimize the livestock's value. It also helps controlling outbreaks of animal diseases like bird flu or mad-cow disease. Start-up firm Somark Innovations is touting technology designed to help tag and trace livestock with RFID-enabled ink tattoos [7].

Wearable RFID wristbands, backpacks and clothing have tracked prisoners, school kids, and even the elderly. Applied Digital Corp. has also created an injectable RFID tag called the Verichip which is used for personal data storage, readable in venues as varied as nightclubs and hospitals.

RFID-tagging allows physical objects to be represented in cyberspace and entered into data bases. Candidates include clothes (to be queried by smart washing machines), packaged foods (to be queried by smart refrigerators), medicine bottles (to be queried by smart medicine cabinets), rental cars, airline baggage, library books, banknotes, drivers' licenses, employee badges, and even surgical patients (to avoid mix-ups).

2.1.4 The RFID Industry

The RFID industry is growing- in 2006, the global market for RFID was USD 2.77 billion [15]. By 2017, it is predicted to reach USD 26.88 billion [17]. However, despite the value of RFID itself, the application domains that use RFID equipment are worth even more. Pharmaceuticals are also big ticket RFID-tagged items. But most importantly, the retail economy (and its frequently tagged supply chain) transacts over USD 1,845 trillion annually. With these amounts of money at stake, it is important to provide reliable and secure operational environment for the success of RFID in any given application. Most informed observers of emerging technologies believe that the cost of RFID tags will drop

dramatically. To be widely used, tag costs will need to drop from 50 cents each to 5 cents each in 5-10 years [27].

2.1.5 Threats on RFID

RFID systems can be abused in the form of corporate security breaches, behavioral profiling, and universal surveillance. So it is likely that RFID will continue to inspire progress in security and privacy research in the future.

Personal or financial data can be resold on the black market through identity theft. According to the US Federal Trade Commission, approximately 10 million Americans have their personal information abused every year, costing consumers USD 5 billion and businesses USD 48 billion annually [6]. RFID applications are likely to be troubled with identity theft. There are several ways that criminals can perform RFID-enabled theft of personal/ financial information; Benjamin et. al. [34] showed that. Criminals can also use RFID malware [24] as a tool for identity theft. RFID tags could be used to install remote malware to extract personal/financial data from the back-end database. Vulnerable RFID readers could be used to perform RFID wardriving, where criminals wander the streets looking for exploitable RFID readers.

RFID tags could also be used for spamming purposes; for example, an EPC Gen2 tag could have a bogus Uniform Resource Identifier (URI) to be used as spam.

Estimates vary on exactly what space resources might be available at the low-end of the market. The notion of gate equivalents (GE) corresponds directly to the physical space that is occupied on silicon. This in turn depends on the manufacturing technology used. A reasonable consensus on the number of GE available for computation is summarized in Juels and Weis and Weis. There it is suggested that RFID tags at the lower-end of the market might have a total gate count of around 1000-10,000 GE with around 200- 2000 GE being available for security features. While we might be looking at around 2000 GE for security functionality in the cheaper tags today, enthusiasts of Moores Law (Moore, 1965) will observe that this suggests around 10,000 GE may be available for a similar price in 4-5 years [32].

2.2 Our Assumptions

- *Adversary*: 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.

- *Reader and Server*: All communication between server and reader is assumed to be over private and authentic channel. Both reader and server have ample storage and

computational capabilities.

- *Tags*: 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. The communication channel between a tag and a reader is assumed to be insecure.

- *Batch-mode*: 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.

2.3 Security Definitions

Definition 2.3.1 *`Tracking Attack:` If an entity's responses are linkable to each other, or distinguishable from those of other entities, then the entity's location could be tracked by an adversary.*

Definition 2.3.2 *`Replay Attack:` An adversary could intercept messages exchanged between a reader and a tag, and replay them.*

Definition 2.3.3 *`Denial-of-Service Attack:` An adversary could interrupt or impede messages sent between two valid entities. Such an attack could cause an entity to become non-operative.*

Definition 2.3.4 *`Forward Security:` If an adversary compromises an entity , then it might be able to derive previous keys to track the old transactions involving that entity.*

Definition 2.3.5 *`Timing Attack:` The attacker attempts to compromise a system by analyzing the time taken to execute cryptographic algorithms. The attack exploits the fact that every operation in a computer takes time to execute.*

Definition 2.3.6 *`Cloning Attack:` The data on a valid entity is scanned and copied by a malicious party and the copied data is embedded onto a fake entity. These cloned fake entities can be attached to counterfeit products.*

Definition 2.3.7 *`Swapping Attack:` An adversary exchanges the ciphertexts between two tags i and j which can have very serious consequences. It suffices to consider the possibility of an attacker exchanging ciphertexts associated with two medications or two spare aircraft parts. An adversary can also launch the attack by swapping the identities of two different tags.*

2.4 Notations

We describe the common notations used throughout the paper. Tags, Readers, Issuers, Servers, Certification Authority are denoted as T, R, P, S and C respectively. T_t are the timestamps from the tags; whereas T_r are from the readers.

Table 2.1: Notations Used in This Literature

H	A hash function, $h : \{0,1\}^l \rightarrow \{0,1\}^l$
S_j	The server of the j-th owner of T $(j \geq 1)$
l	The bit-length of a tag identifier t, or a random string r
s	A string of l bits assigned to T
t	Tag identifier of l bits, which equals $h(s)$
$\hat{x}$	The most recent value of x
r	A random string of l bits
$x \gg a$	Right circular shift operator, which rotates all bits of x to the right by a bits
R_T	Request for ownership of tag T
N_T	Random nonce sent by tag
N_R	Random nonce sent by reader
K_i	Secret key and ID of a tag

Chapter 3

Previous works

3.1 Untraceable RFID tags via Insubvertible Encryption [IE]

3.1.1 Introduction

Insubvertible Encryption [23] produces ciphertexts which can be randomized without any key material, provides properties such as semantic security and key privacy, includes certificates which are decrypted by authorized parties only, has randomized contents in RFID tags for security purposes especially eavesdropping and readers replace the contents of tags with randomized versions at each read operation.

Security Formalization: The certification authority C, the set of issuers P_i, the reader /randomizer R_i , the tags D_i, the adversary or malicious entity A can corrupt R_i only, a trusted party T through whom all parties interact, prime numbers p and q, Elliptic curve(MNT) E, Generator $g \in G_1$ and $\tilde{g} \in G_2$, group size is the value $k = \lfloor log_2 p \rfloor$ and elliptic curve key length k' - since the index of G_1 in $E(\mathbb{F}_q)$ is small, k and K' are close values ; authentication of message m : $MAC(K_i, m)$, Enc(K,M) denotes combined encoding of the message and its authentication tag under MAC, G_1 is subgroup of p in $E(\mathbb{F}_q)$ and G_1 is subgroup of $E(\mathbb{F}_{q'})$:- these are the specific notations used in this section.

3.1.2 Definition

Each legitimate issuer P_i of RFID tags should generate a public key and have it certified by the authority, which has public key C. Each issuer is allowed to track tags initiated (marked) by itself. On the other hand, no other party is able to mark RFID tags. That means only certified public key may store a mark in the tag that is traceable beyond the next interaction with an honest reader. Each honest reader is required to re-randomize the mark.

Definition 3.1.1 *XDH Assumption*

Let G_1 and G_2 admit a bilinear map. XDH assumption holds if G_1 is a DDH-hard group, i.e, it is not computationally feasible, given a set of values y, y_1, y_2, y_3 in G_1, to decide if there are integers a, b such that $y_1 = y^a, y_2 = y^b$ and $y_3 = y^{ab}$

Definition 3.1.2 ***Strong LRSW assumption*** *Let G_1 and G_2 admit a bilinear map with no morphisms between them. Let $S, T \in G_2, S = \tilde{g}^s, T = \tilde{g}^t$. Let $O_{S,T}(.)$ be an oracle that, on input a value $x \in \mathbb{F}_p$, outputs a quintuple $A = (a, a^t, a^{s+xst}, a^x, a^{xt})$ for a randomly chosen $a \in G_1$. Then for all probabilistic polynomial time adversaries A, $\nu(k)$ defined as follows is a negligible function:*

$Pr[(p, G_1, G_2, g, \tilde{g}) \leftarrow GenerateKey(1^k);$
$s \leftarrow \mathbb{F}_p; t \leftarrow \mathbb{F}_p; S = \tilde{g}^s; T = \tilde{g}^t;$
$(a_1, a_2, a_3, a_4, a_5) \leftarrow A^{O_{S,T}}(p, G_1, G_2, g, \tilde{g}, S, T) :$
$x \notin Q \wedge x \in \mathbb{F}_p \wedge x \neq 0 \wedge a_1 \in G_1 \wedge a_2 = {a_1}^t$
$\wedge a_3 = {a^{s+xst}}_1 \wedge a_4 = {a^x}_1 \wedge a_5 = {a^{xt}}_1] = \nu k,$

3.1.3 Description of the Scheme

• GenerateCAKey: The private-public key pairs of the authority C are as follows:
$C_{SK} = (s, t), C_{PK} = (S = \tilde{g}^s, T = \tilde{g}^t$,C)
C= $(a_1, a_2, a_3, a_4, a_5, c_1, c_2)$ is a dummy insubvertible encryption and consists of a certificate $(a_1, a_2, a_3, a_4, a_5)$.
• GenerateKey: Issuers of tags generate public keys in G_1. Each private-public key pair is of ElGamal type:
$(SK_i, PK_i) = (x_i, y_i)$
• KeyRegistration (RegisterPublicKey): This algorithm requires XDH assumption for security. The authority C after verifying the identity of P_i and its claim to public key y_i computes a random $w \in \mathbb{F}_p$ and set $a = g^w$. The certificate on the public key y_i is:
$(a_1, a_2, a_3, a_4, a_5) \leftarrow (a, a_t, a^{s+x_ist}, a^{x_i}, a^{x_i}t) \in {G^5}_1$
• Encryption (InitiateTag): The issuer generates a random value $r \in \mathbb{F}_p$ and uses to randomize the certificate:
$(a_1, a_2, a_3, a_4, a_5) \leftarrow ({a^r}_1, {a^r}_2, {a^r}_3, {a^r}_4, {a^r}_5)$
Then the issuer computes the authentication code for a message m under its master authentication key K into T. The issuer encodes both m and T as a single point in G_1. The issuer encrypts M using ElGamal with public key y_i and random value $k \in \mathbb{F}_p$:
$(c_1, c_2) \leftarrow (g^k, M{y^k}_i)$
The *insubvertible encryption* of m is then:
$(a_1, a_2, a_3, a_4, a_5, c_1, c_2) \leftarrow (a^r, a^{rt}, a^{r(s+x_ist)}, a^{rx_i}, a^{rx_it}, g^k, M{y^k}_i) \in {G^7}_1$
• Randomization (ReadAndRandomize): After obtaining $(a_1, a_2, a_3, a_4, a_5, c_1, c_2)$ from the tag, the certificate is verified. If found invalid, then the C from the public key of CA is retrieved and set $(a_1, a_2, a_3, a_4, a_5, c_1, c_2) \leftarrow$C. Then random values v and z in $\mathbb{F}_p$ are generated to compute new encryption:
$(a_1, a_2, a_3, a_4, a_5, c_1, c_2) \leftarrow ({a^v}_1, {a^v}_2, {a^v}_3, {a^v}_4, {a^v}_5, {a^z}_1c_1, {a^z}_2c_2)$
• Decryption(ReadAndDecrypt): To decrypt $(a_1, a_2, a_3, a_4, a_5, c_1, c_2)$, the issuer first verifies the randomized certificate, then checks that the certificate is for its own public key

by $a_{i\,1}^{x} = a_4$. If so it proceeds to decrypt $M = c_2 \div (c_1)^{x^i}$, and the decode the point M into the original message m.

Discussion of the scheme Only authorized users possess valid certificates and have the ability to create valid insubvertible encryption. Certificates have the property of being randomizable. If the certificate is invalid the randomizer will substitute the content of the tag with a randomized version of a dummy insubvertible encryption from the public key of the authority.

3.1.4 Analysis

Semantic Security This scheme inherits semantic security from ElGamal encryption scheme. The pair (c_1, c_2) is effectively an ElGamal encryption in G_1 in which DDH is hard.

Unforgeability of Certificates The unforgeability of certificates released by the certification authority requires the LRSW assumption. With the assumption, the public key registration scheme is resistant to forgery. The Strong LRSW assumption is defined over two groups, admit bilinear map and there is no morphisms between them.

Key Privacy It is computationally difficult to distinguish between two samples of marks, when the two samples are created using different issuer public keys. This property follows from the assumption that G_1 is DDH-hard.

3.1.5 Discussion

Cloning Attacks A dual-core single tag can be deployed to avoid cloning attack. The first part would store an immutable ID and can be only de-activated. The second part would store data encrypted by insubvertible encryption. The first part is deactivated and the other left untouched when the tag leaves the store. But this cannot fully prevent cloning attacks. Use of tamper-proof tags that engage in interactive protocols for proof-of-knowledge of secrets can be a good option to prevent cloning attacks.

Swapping Attacks Insubvertible encryption does not defend against swapping attack; in which an adversary exchanges two valid ciphertexts across RFID tags. However Juels et al. proposed the use of a proxy device REP (RFID Enhancer Proxy) to prevent swapping attacks. Tags can generate a certain amount of randomness. They consider a protocol in which a tag generates a new pseudonym $p_{t,i}$for a counter t maintained (internally) on the tag. If it receives an update command from the REP, along with a valid write key, the tag transmits $p_{t,i}$ to the reader and adopts $p_{t,i}$ as its new pseudonym. This approach would render swapping attacks infeasible, as the REP and thus an adversary would be unable to dictate tag pseudonyms. A tag may be unable to generate a full-length random pseudonym in each session. Even partial generation of a pseudonym by a tag, however,

can help alleviate the risk of swapping attacks. In particular, tag might emit a random nonce r of length $k' < k$ before accepting the writing of a new pseudonym. The tag then only accepts a new pseudonym if it matches this nonce, e.g., if the last bits of the pseudonym are equal to r. In other words, a tag can participate partially in the generation of its pseudonyms. An adversary attempting to swap pseudonyms, then, will be unable to do so unless it can locate a pair of tags simultaneously emitting the same nonces. But the problem remains with this proposal is that, a device has to be carried by the users when they have tags with them. Moreover, they do not provide the way to secure the device itself.

3.2 An Elliptic Curve and Zero Knowledge based Forward Secure RFID Protocol[EC-ZK]

3.2.1 Introduction

The proposed approach [26] is based on elliptic curve cryptography, which allows the use of fewer bits than conventional public key cryptography guaranteeing same security. This protocol requires reader authentication by means of a zero knowledge protocol.

Setup Phase: Reader has secret s the knowledge of which must be proven to tags, a finite field F_q and an elliptic curve $E(F_q)$, a generator Q of a cyclic subgroup of points of the elliptic curve where the ECDLP was unfeasible, the public key $P \in E(F_q)$ of valid readers, computed as $P = sQ$. Each tag needs a secret elliptic curve point $K \in E(F_q)$, from which it will compute its identifier. The database has the information related to the tags.

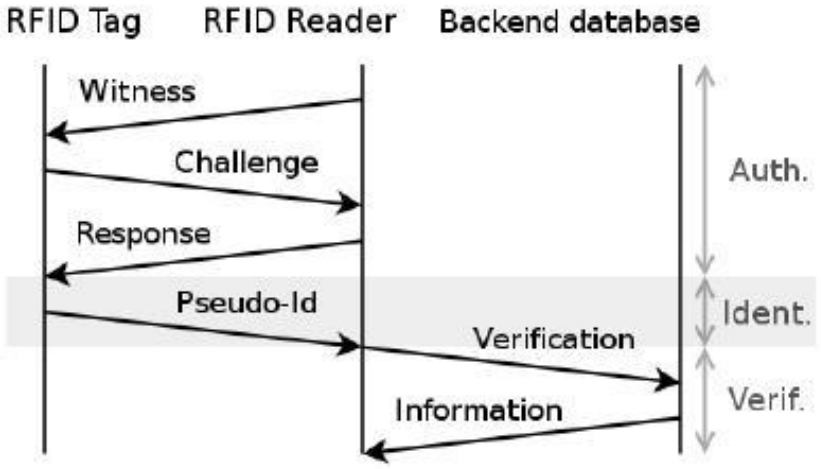

Figure 3.1: Basic Protocol

Reader Authentication Phase The figure below describes the reader authentication phase:

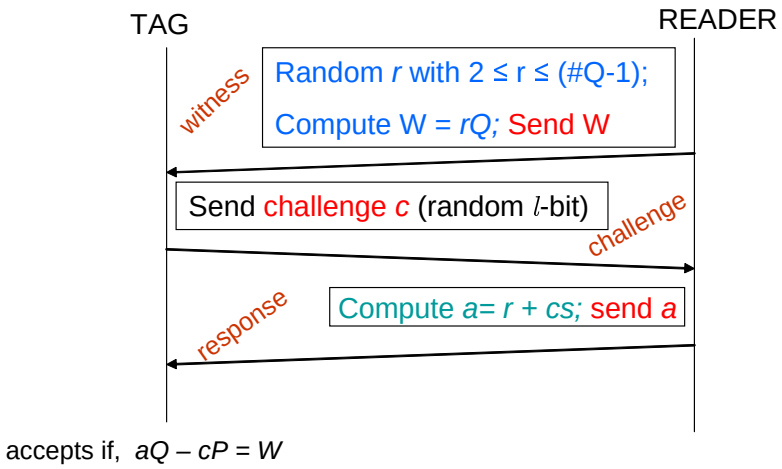

Figure 3.2: Reader Authentication

Tag Identification Phase The tag computes pseudo-id $id^i_j = LastBits(x(K^i_j)bxory(K^i_j))$, where $bxor$ is the bitwise xor operator, LastBits takes some of the last bits of its input, $(x(K^i_j)bxory(K^i_j))$ are the abscissa and the ordinate of the secret point. The tag computes its next secret point $K^i_{j+1} = x(K^i_j).Q$. The tag stores its new secret point and sends id^i_j

3.2.2 Advanced Protocol

W= rQ Reader sends $W = rQ$, current time stamp t, $a = r + cs$ to tag. The tag responds with messages id^i_j and u^i_j; where $u^i_j = v^i_j bxor FirstBits(y(K^i_j))$. v is a small value generated by possible attached sensor.

3.2.3 Discussion

Binary Finite field is defined over F_{2^m} (m= 137), the length of challenge l is 64 bits. Hash function input is last 128 bits of W's abscissa concatenated with 32 bits timestamp which makes it 160 bits. Output is l bits. To avoid birthday paradox, length of id is 48-bits (for 1 million tags). Reader verification and secret point generation takes < 10000 logical gates for a tag. Communications for the advanced protocol takes 297 bits for 1st message; 112 bits for the second message which makes a total of 409 bits < 520 bits. Database size for 1mio tags : 1000000 (48+137+1) $\approx$ 23MB.

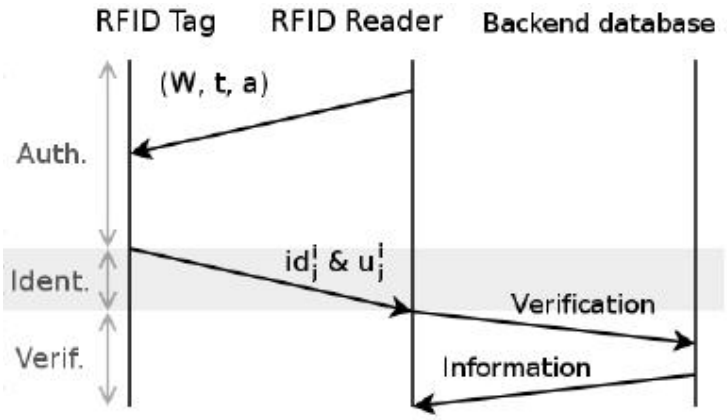

Figure 3.3: Advanced Protocol

Sniffing: In an sniffing attack the attacker eavesdrops the communications between a reader and a tag, trying to obtain useful information. The only public information is the needed for the protocol setup, the reader's public key, the information sent in the first message (the witness, a timestamp, and the response) and the information sent in the second message (the pseudo-id and, optionally, the encrypted sensor data). So, an sniffing attack is useless due to the use of a zero knowledge protocol in combination with the use of a pseudo-id which cannot be related to the tag's secret and the encryption of the sensor data.

Tracking of the tags: The tag's tracking attack consists on the tracking of the behavior of the owner of a tag. In this case, the only information to be considered is the pseudo-id (and the optional sensor data), since the rest of the data is sent by the reader. A pseudo-id sniffed at a certain moment, cannot be related with the information obtained before (or after), because a bitwise xor is considered a secure ciphering algorithm for small data (the same reasoning applies to the encrypted sensor data).

Replay attack: A replay attack consists on an attacker resending information that he had captured before, eavesdropping a previous session. If an attacker eavesdrops the reading operation of a tag, he will obtain a witness, a timestamp and a response. Since the timestamp must be greater than the last-heard time of any tag, he cannot reuse these values with the tag involved in the communication he has eavesdropped; but, if there were tags in the system whose last-heard time were still lower than the obtained by the attacker, these tags would be vulnerable to a replay attack. So, only the interactive version of the protocol avoids this attack. On the other hand, a pseudo-id cannot be reused, because the database will wait for the next pseudo-id of any tag, so if an attacker reuses a pseudo-id the reader will not recognize the value as valid.

Denial Of Service: Finally, a denial of service consists on a temporal (or permanent) incapacitation of the system or a part of it. Since a tag only changes its secret K_j^i when a reader has successfully authenticated, there is no danger of an attacker performing a denial of service attack of multiple read operations.

3.3 High Power Proxies for Enhancing RFID Privacy and Utility [REP]

3.3.1 Introduction

In this paper, a new design principle for a personal RFID-privacy device is proposed. Such a device is referred as a REP (RFID Enhancer Proxy). Briefly stated, a REP assumes the identities of tags and simulates them by proxy. By merit of its greater computing power, the REP can enforce more sophisticated privacy policies than those available in tags.

3.3.2 How REP works

The details on the four processes involved in REP management of tags are: Tag acquisition, Tag relabeling, Tag simulation, and Tag release.

Tag acquisition When the owner of a REP and RFID tag wishes the REP to simulate the tag, the REP must acquire all of the necessary tag information and place the tag in a state permitting the REP to act as its proxy. The main technical challenge occurs when the tag has associated secrets, like PINs for access control or killing, that must be transferred securely.

Tag relabeling The relabeling of tags by the REP is a means of protecting against privacy compromise through direct tag scanning. One way to accomplish this is to have the REP re-encrypt a public-key ciphertext carried by a tag, as proposed in previous work. The setting in which the REP serves as a proxy permits a simple approach involving the assignment of changing pseudonyms to tags.

Tags can generate a certain amount of randomness. A tag i generates a new pseudonym $p_{t,i}$ for a counter t maintained (internally) on the tag. If it receives an update command from the REP, along with a valid write key, the tag transmits $p_{t,i}$ to the reader and adopts $p_{t,i}$ as its new pseudonym. This approach renders *swapping attacks* infeasible, as the REP - and thus an adversary - would be unable to dictate tag pseudonyms. Tags are capable of generating only a limited number of random bits in the course of a given session. A tag may therefore be unable to generate a full-length random pseudonym in each session. A tag might emit a random *nonce* r of length $k' < k$, where k is the pseudonym size, before accepting the writing of a new pseudonym. The tag then only accepts a new pseudonym if it matches this nonce, e.g., if the last bits of the pseudonym are equal to r. An adversary attempting to swap pseudonyms, then, will be unable to do so unless it can locate a pair

of tags simultaneously emitting the same nonces. The probability of successful attack by an adversary is a function of the number of tags N managed by a REP, the number of time-slots s available to the adversary for its attack, and the bit-length k'. For example, a pallet carrying some 100 tags relabeled every minute, and seeking protection against attacks lasting up to one day (1440 minutes), and employing tags that generate 32-bit nonces. The probability that a given tag shares a random pseudonym with any of the 99 others may be crudely bounded above by $99/232$. Thus the probability of a successful swapping attack in this case is easily seen to be less than $(1-(1-99/2^32))\ 1440 < 0.000034$.

Tag simulation The REP simulates tags in interaction with readers. The REP may also simulate spurious tags to prevent leakage of information about the number of tags carried by its owner. As a REP is presumed to be a powerful device, it can enforce more-or-less any privacy policy desired by its owner. No simulation policies is specified in this paper. But these could include robust public-key based authentication schemes.

Tag release When the owner of a REP wishes it no longer to simulate a tag, the REP must release its control and re-imprint the tag with its original identity. On release of a tag, the randomly assigned portion of its identifier is retained, and the rest of its identifier is restored to its original state. For example, the identifier on an EPC tag has two segments, roughly speaking: (1) A (numerical) identifier segment that specifies the object the tag is attached to, e.g., says, This is a 100g tablet of Valhrona chocolate and (2) A unique numerical segment, effectively a serial number. During the period in which a tag is simulated by a REP, segment (1) can be effaced or overwritten by the REP, while segment (2) (or a portion thereof) is generated at random by the tag. When the tag is released, the randomness in (2) is retained, while (1) is restored. Effectively, then, a tag gets a new serial number at the time it is released.

3.3.3 Discussion

No Swapping Attacks One of the contributions is a simple technique for preventing an adversary from swapping the identities of two different tags, e.g., swapping the identifiers on two medications with differing dosages. The insubvertible encryption and universal re-encryption schemes are vulnerable to this type of attack. The technique proposed involves random input from the tag in the creation of pseudonyms, works even when tags do not have access-control features.

Denial of Service Attacks Even if an attacker cannot successfully initiate a swapping attack, corruption of tag data has a second effect: Denial of service. If an attacker is able to implant a pseudonym in a tag, the tag effectively becomes desynchronized with the REP: The REP no longer recognizes the tags pseudonym. If the REP were consequently to halt rotation of new pseudonyms into a tag, a breach of privacy could result, since tag identifier would remain static. A REP might alert a user to unexpected desynchronization events of this kind by emitting a warning tone.

Tag release problem A secondary effect of a corruption attack is that the REP cannot properly release tags: If it does not recognize their pseudonyms, it cannot manage them properly. Thus, one of two approaches might be needed for tag restoration: (1) If the REP possesses PINs for the tags in its control, it can try to match tags to PINs via exhaustive search or (2) Some kind of manual intervention on the part of the user might be necessary, e.g., the user might have to key in a printed product code from items that the REP has lost.

3.4 A Family of Dunces: Trivial RFID Identification and Authentication Protocols

3.4.1 Introduction

This paper describes a protocol family for inexpensive untraceable identification and authentication of RFID tags. Untraceable means that it is computationally infeasible to infer from interactions with a tag information about the identity of the tag or link multiple authentication sessions of the same tag. Proposed protocols are inexpensive, requiring as little as one light-weight cryptographic operation on the tag and storage for one key. The legitimate entities are: tags, readers and servers. All communication between a server and its readers is assumed to be over private and authentic channels. Furthermore, servers and readers maintain loosely synchronized clocks. We assume that an RFID tag has no clock and small amounts of ROM (e.g., to store a key) and non-volatile RAM (to store ephemeral state, such as a counter or time-stamp). With power supplied by a reader, whether contact or contactless, a tag is able to perform a modest amount of computation and commit any necessary state information of small constant length to non-volatile storage.

Two modes of tag identification are considered: real-time and batch. The $real-time$ mode is the one typically considered in the literature: it involves on-line contact between the reader and the server, in order to quickly identify (and, optionally, authenticate) the tag in question. If immediate feedback about a tag is needed the server must be contacted in real time. In *batch* mode, a reader scans numerous tags, collects replies and sometime later performs their identification (and optionally, authentication) in bulk.

3.4.2 YA-TRIP: Yet Another Trivial RFID Identification Protocol

When the tag keeps state of the last time-stamp it saw (assuming it was legitimate), then it can distinguish between future (valid) and past (invalid) time-stamps. A tag compares the time-stamp to the stored time-stamp value. If the former is strictly greater than the latter, the tag concludes that the new time-stamp is probably valid and computes a response derived from its permanent key and the new timestamp. A tag thus never accepts a time-stamp earlier than or equal to the one stored. However, to protect against

narrowing attacks, even if the timestamp supplied by the reader pre-dates the one stored, the tag needs to reply with a value indistinguishable from a normal reply (i.e., a keyed hash over a valid timestamp). In such cases, the tag replies with a random value which is meaningless and cannot be traced to the tag even by the actual server.

[1] Tag $\leftarrow$ Reader : T_r
[2] Tag :
- [2.1] $\delta = T_r - T_t$
- [2.2] if ($\delta \leq 0$) or ($T_r > T_{max}$)
- [2.2.1] $H_{id} = PRNG_i^j$
- [2.3] else
- [2.3.1] $T_r = T_t$
- [2.3.2] $H_{id} = HMAC_{K_i}(T_t)$

[3] Tag $\rightarrow$ Reader : H_{id}

The YA-TRIP protocol, as presented above, has a potential drawback. It is susceptible to a trivial denial-of-service (DoS) attack: an adversary sends a wildly inaccurate (future) time-stamp to a tag and incapacitates it either fully (if the time-stamp is the maximal allowed) or temporarily.

3.4.3 YA-TRAP: Adding Tag Authentication

The initial reader-tag message to includes a random challenge R_r. Then, a MAC of both (reader and tag) challenges in the tag's reply message. Later, once the tag is identified by the server, it can be authenticated by verifying the MAC. The identification step is the same as in YA-TRIP. The resulting protocol (YA-TRAP) is shown in Figure 2.

[1] Tag $\leftarrow$ Reader : T_r, R_r
[2] Tag :
- [2.1] $\delta = T_r - T_t$
- [2.2] if ($\delta \leq 0$) or ($T_r > T_{max}$)
- [2.2.1] $H_{id} = PRNG_i^j$
- [2.3] else
- [2.3.1] $T_t = T_r$
- [2.3.2] $H_{id} = HMAC_{K_i}(T_t)$
- [2.4] $R_t = PRNG_i^{j+1}$
- [2.5] $H_{auth} = HMAC_{K_i}(R_t, R_r)$

[3] Tag $\rightarrow$ Reader : H_{id}, R_t, H_{auth}

YA-TRAP is susceptible to DoS attacks whereby a rogue reader can easily incapacitate a tag by feeding it a futuristic (or even maximum)T_r value.

3.4.4 YA-TRAP*: Adding DoS Resistance

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 the offset of ET_{r_i} as ν 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 ν 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_{r_i} is not only plausible but is at most INT time units (e.g., one day) into the future. If the tag accepts the T_{r_i}, then the tag stores this value as T_{t_i} for the next authentication session, T_{t_i} is considered to be the last valid timestamp value the tag had accepted. The server has a database that contains the information corresponding to each tag. These information are stored in a hash table. When the reader forwards the messages from the tags to the server, the server LOOKUP its database to search the data corresponding to the received messages. The following Algorithm (YA-TRAP*) describes 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*

A tag needs to compute 3 keyed hash operations including a PRNG and in addition, needs to compute ν 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 5 n. This needs a huge amount of resource for communication in case of batch-mode authentication. Even though 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 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 value within the 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}. 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.

3.4.5 Discussion

The protocols described in this paper are vulnerable to DoS attacks as described above. Moreover, YA-TRAP* is susceptible to *timing attacks*. The execution the last OR clause

in step 2.3 in Figure 3 is dependent upon the offset of T_r which can be freely selected by the adversary. Consequently, the adversary can mount a timing attack aimed at determining the epoch corresponding to the tag's last time-stamp of use (T_t). This is because the number of repeated hash operations in step 2.3 is based on $\nu = \lfloor T_r/INT \rfloor - \lfloor T_t/INT \rfloor$ One obvious countermeasure is to artificially pad the number of iterated hashes or introduce a random delay in tag's reply to the reader. However, such countermeasure is counterproductive as it increase protocol execution time which is undesirable, especially in batch mode.

3.5 SQUASH- A New MAC with Provable Security Properties for Highly Constrained Devices Such As RFID Tags

3.5.1 Introduction

The requirements for a RFID tag are: small footprint, low power consumption (to maximize operating range), reasonable speed and good security. The hash functions used in authentication procedure should be one-way to protect the shared secret, need not be collision resistant. The proposed SQUASH is similar to a MAC, but operates on a single block. This is based on Rabin variant $c = m^2(modn)$ which is believed to be a very good one-way function where m (or some of its bits) from c is provably as difficult as factoring n. But, it requires large registers, produces large outputs, uses a lot of power and is too slow.

3.5.2 Building SQUASH

SQUASH stands for SQUare-hASH. Using composite Mersenne numbers of the form $2^k - 1$ for n is a requirement for a generic squash. Message m is not stored, whereas in the Rabin scheme, the modulus n should have 1000+ bits, and the message m should be full size. Instead, it uses a short window t of bits in the Rabin ciphertext as output of H. t=32 to 64 bits of output, not 1000+, unlike the case of the original Rabin scheme, a future complete factorization of such a universal modulus n will have only limited impact on SQUASH As because there is no known way how to use such a factorization to actually extract S from short windows of bits in the middle of $m^2(modn)$.
The RFID tag computes the bits of the middle window in c from LSB to MSB. Each bit is sent out as soon as it is computed, and erased from memory. The lower end of the t-bit window is extended with u=8 additional bits to guard against the extra carry bits. The value of the extended window is computed assuming that the carry into its LSB was zero. At the end of the computation of the $t + u$ bits of the extended window, lower u is thrown away and the top t is provided as the output of H. The final squash algorithm follows the steps below:

1. Initialize: Set j, $k \leftarrow 128$ and c $\leftarrow 0$

2. Convolve: Compute $c \leftarrow c + \sum_{v=0}^{k-1} m_v * m_{j-v}(mod k)$
3. Set c_j=LSB(carry), carry=rightshift(carry).
4. Repeat steps 2 and 3 40 times and output the 32 bits

The mixing function M is based on the single NFSR from GRAIN-128 with no input and no output. The secret S and random challenge R are both 64 bits. The upper half of the register is filled with $S \oplus R$ and the lower part is filled with S while initialized. They are mixed by clocking 512 times and the circular convolution goes on the bits afterwards. The resultant 128 bit is the squared m.

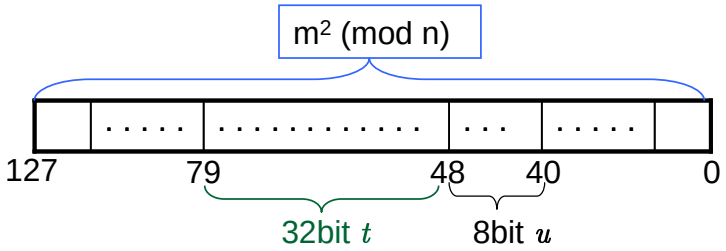

Figure 3.4: A squash output

3.5.3 Discussion

The probability that a reader will accept a random t from adversary is 2^{-t}. t=32 bits; so cheating probability is one in 4 billion. Unlike the case of the original Rabin scheme, a future complete factorization of such a universal modulus n will have only limited impact on SQUASH because of the use of window. Both the mixing function and the squash used the same 128-bit register. The other hardware needed are: a 8-bit carry register, few AND and XOR gates for feedback, short counters for v and j. The total number of gates is almost half of GRAIN-128 which is one of the smallest hardware cryptographic primitives (as only the NFSR part is used in SQUASH).

3.6 Data Synchronization in Privacy-Preserving RFID Authentication Schemes [Sync]

3.6.1 Introduction

This proposal [28] focuses on schemes where the tag shares a personal secret key with the reader and where this key is updated regularly. The studied schemes consequently necessitate synchronization mechanisms to be sure that the same version of the key is used at both sides. It shows first the capacity of an adversary to desynchronize a tag and

the reader; second the way the scheme is able to resynchronize a tag and the reader; third the efficiency of the search procedure in the worst case. These are necessary to compare schemes and to know what are the advantages and the drawbacks of each of them with respect to synchronization problem.

3.6.2 The Desynchronization Capacity

Desynchronization characterization means that for a given scheme the maximum number of desynchronizations between a tag and the reader an adversary can create. This is computed by the corresponding scale called the desynchronization value. To have a better characterization, the maximum number of desynchronization $D_{\mathcal{R}}$ an adversary can create when only focusing on the reader is expressed on one side, and on the other side this maximum number $D_{\mathcal{T}}$ when $\mathcal{A}$ only focuses on the tag. The desynchronization value consequently corresponds to the couple $(D_{\mathcal{R}}, D_{\mathcal{T}})$. Let $\mathcal{A}$ being the adversary chooses a tag T. Let $TK_T = K_T^{(i)}$ (resp. $RK_T = K_T^{(j)}$) the tag's version (resp. reader's version) of K_T. The maximum number of desynchronization obtained by the adversary $\mathcal{A}$, when $\mathcal{A}$ only focuses on the tag (resp. the reader), is $D_{\mathcal{T},\mathcal{A}} = j - i$ (resp. $D_{\mathcal{R},\mathcal{A}} = i - j$).

Definition 3.6.1 *`Desynchronization` For a given RFID authentication scheme, the desynchronization value of a scheme is the couple $(D_{\mathcal{R}}, D_{\mathcal{T}})$ with $D_{\mathcal{R}} = Sup_{\mathcal{A}}(D_{\mathcal{R},\mathcal{A}})$ and $D_{\mathcal{T}} = Sup_{\mathcal{A}}(D_{\mathcal{T},\mathcal{A}})$. The scheme is said $(D_{\mathcal{R}}, D_{\mathcal{T}})$ desynchronizable.*

3.6.3 The Desynchronization Capacity

The capacity a scheme has to resynchronize a tag and the reader by computing the corresponding scale, called the resynchronization value, defined as follows.

Definition 3.6.2 *`Resynchronization` For a given RFID authentication scheme, when considering an arbitrary valid tag T with $TK_T = K_T^{(i)}$ and a reader with $RK_T = K_T^{(j)}$, we denote by $R_{\mathcal{R}}$ (resp. $R_{\mathcal{T}}$) the maximum number of desynchronizations the scheme can tolerate to accept the tag $\mathcal{T}$ if only the tag (resp. the reader) has been updated, i.e. $RK_T = K_T^{(j)}$(resp. $TK_T = K_T^{(i)}$). The resynchronization value of the scheme is the couple$(R_{\mathcal{R}}, R_{\mathcal{T}})$ and the scheme is said $(R_{\mathcal{R}}, R_{\mathcal{T}})$resynchronizable.*

Definition 3.6.3 *`Synchronizable and Desynchronizable` For a given RFID authentication scheme, if $D_{\mathcal{R}} \leq R_{\mathcal{R}}$ and $D_{\mathcal{T}} \leq R_{\mathcal{T}}$, the scheme is said synchronizable. Else, the scheme is said desynchronizable.*

3.6.4 Protocol Description

The random values N_R and N_T are used to protect the scheme against replay attack. In the search procedure, for each $T \in D_{\mathcal{R}}$, the reader computes the value $H_1(RK_T \| N_R \| N_T)$ and compares it with r. In case of a match, the procedure outputs the corresponding identifier. Else, for each $T \in D_{\mathcal{R}}$, the reader computes the ephemeral identifier $ET = H_2(RK_T)$,

computes $H_1(ET||N_R||N_T)$ and compares it with r. In case of a match, the reader outputs the corresponding identifier and updates RK_T (so $RK_T = TK_T$). Otherwise, the tag is rejected.

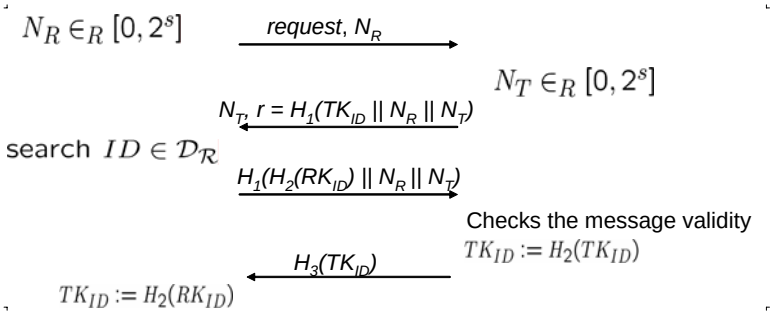

Figure 3.5: The C2 Scheme

3.6.5 Discussion

Privacy The scheme provides the anonymity of the tag under the assumption that the hash function is one-way. Moreover, the scheme is unlinkable since $\mathcal{A}$ has no control on the value N_T. Again, if the used hash functions are secure, the three first messages are useless for $\mathcal{A}$. One possibility for $\mathcal{A}$ is to use $H_3(TK_T)$ but this message is only sent when the key is updated: $\mathcal{A}$ cannot learn any information of it. Finally, the scheme provides the forward-privacy property using the same argument. So the scheme keeps privacy.

Theorem 1 *The scheme is $(1, 0)$-desynchronizable, $(1, 0)$-resynchronizable and the search procedure works in 3 operations per tag in the worst case. The scheme is consequently synchronizable.*

This suggests a new model for RFID authentication most adapted for schemes using a secret key infrastructure with an update key mechanism and has presented a protocol with good properties while being secure in the privacy model.

3.7 RFID Tag Ownership Transfer [OT]

3.7.1 Introduction

In some RFID applications it is necessary to allow for transfer of tag ownership. This paper [29] has identified three requirements for secure and privacy-preserving tag ownership transfer: new owner privacy, old owner privacy, and authorization recovery. It proposes novel RFID authentication protocols for tag ownership transfer that meet such requirements. The scheme consists of three protocols: P1, an ownership transfer protocol, P2,

a secret update protocol, and P3, an authorization recovery protocol. We believe that this latter protocol is the first proposed practical authentication scheme for authorization recovery.

3.7.2 Protocol Description

P1: Ownership Transfer Protocol In this protocol (referred to as P1), the new owner of a tag first requests the current owner for ownership of the tag. If the request is valid, the current owner then transfers all the information related to the tag to the new owner via a secure channel. In order to use this protocol, the database of server S_j must contain the following entries for a tag T that it manages: the current secrets (K, s), the most recent secrets $(\hat{K}, \hat{s}$, and any other necessary information for T. Tag T stores a secret K as its identifier. In order to take ownership of tag T, server S_{j+1} communicates with both T and S_j, as follows:

1. S_{j+1} generates a random string r_1 of l bits, and sends it to T;

2. In response, T generates a random string r_2 of l bits, computes $M_1 = t \oplus r_2$ and $M_2 = f_K(r_1 \oplus r_2)$, and then sends M_1 and M_2 to S_{j+1};

3. S_{j+1} forwards r_1, M_1 and M_2 to S_j , with a request for ownership of T;

4. On receipt of this message, S_j performs the following steps:
   i) If the received request is valid, then S_j searches through its database for a pair (K, s) for which the value of K satisfies $M2 = f_K(r_1 \oplus M_1 \oplus K)$.
   ii) If such a pair (K, s) is found, S_j sets $r_2 = M_1 \oplus K$ and computes $M_3 = s \oplus (r2 \gg l/2)$. Otherwise, the session stops.
   iii) S_j updates the copies of the most recent secrets as $\hat{s} \leftarrow s$ and $\hat{K} \leftarrow K$, and the copies of the current secrets as $s \leftarrow (s \ll l/4) \oplus (K \gg l/4) \oplus r_1 \oplus r_2$ and $K \leftarrow h(s)$.
   iv) S_j sends M_3 to S_{j+1}, and transfers the updated secrets (K, s) and the other necessary information for T (info) to S_{j+1} via a secure channel.

5. When S_{j+1} receives (K, s), Info, and M_3 from S_j, it stores (K, s) and Info in its database, and forwards M_3 to T;

6. T then performs the following steps:
   i) T computes $s = M_3 \oplus (r_2 \gg l/2)$, and checks that $h(s) = K$.
   ii) If the verification succeeds, T has authenticated S_{j+1} as an authorized server, and updates its secret as $K \leftarrow h((s \ll l/4) \oplus (K \gg l/4) \oplus r_1 \oplus r_2)$. Otherwise, the session stops.

P2: Secret Update Protocol After carrying out P1, server S_{j+1} has ownership of tag T. However, if S_j is malicious, it could still identify or trace T using the information it passed to S_{j+1}. Thus S_{j+1} needs to execute another protocol (P2) to establish new secrets with T, in order to protect the new owners privacy. Protocol P2 is assumed to

be performed at a distance from any readers connected to S_j , in order to prevent S_j eavesdropping on the messages. Prior to running P2, it is assumed that server S_{j+1} has a pair of secrets (K, s) for tag T, obtained as a result of executing P1; T has identifier K. P2 involves the following steps.

1. S_{j+1} generates random strings r_1 and s' of l bits, and computes $K' = h(s')$. S_{j+1} then computes $M_1 = f_K(r_1) \oplus K'$ and $M_2 = s \oplus (K' \gg l/2)$, and sends r_1, M_1 and M_2 to T;

2. When T receives r_1, M_1 and M_2 from S_{j+1}, it performs the following steps:
   i) T computes $K' = M_1 \oplus f_K(r_1)$ and $s = M_2 \oplus (K' \gg l/2)$.
   ii) If $h(s) = K$, T has authenticated S_{j+1} as an authorized server. Otherwise, the session stops.
   iii) T updates its secret as $K \leftarrow K'$, and generates a random string r_2 of l bits.
   iv) T computes $M_3 = f_K(r_1 \oplus r_2)$ using the new secret K, and sends r_2 and M_3 to S_{j+1}.

3. S_{j+1} checks that M_3 is equal to $f_{K'}(r_1 \oplus r_2)$. If the validation succeeds, S_{j+1} now knows that T has had the new secret K', and updates secrets s and K for T to s' and K', respectively. Otherwise, S_{j+1} goes to step 1, and starts a new session.

If P2 completes successfully, S_{j+1} and T share new secrets known only to them, and S_j is no longer able to identify or trace T. Both S_j and S_{j+1} can also keep the pair (K, s) provided by S_j for use in the event that S_j needs to identify T again.

P3: Authorization Recovery Protocol P3 enables server S_{j+1} to make T change its secret back to the value it had when S_{j+1} took ownership of T from S_j. Prior to running P3, it is assumed that S_{j+1} stores the following information for tag T: the current secrets (K, s), the most recent secrets $(\hat{K}, \hat{s})$, and the old secrets also known to S_j , $(\bar{K}, \bar{s})$. P3 is the same as P2 except that K' is set equal to $\bar{K}$, i.e. the old value of K. After successful execution of P3, T stores $\bar{K}$ as its identifier. As a result, S_j can identify T again. If S_{j+1}executes P1 and P2 again, it can recover authorization on T from S_j.

3.7.3 Discussion

Tag Information Leakage Ts responses are a function of its secret K, and thus only the server that knows the secret is able to identify T and access the tag information.

Tracking T's responses are anonymous, since they are a function of random strings r_1 and r_2, and are independent of one another. The messages exchanged between the server and the tag are computed using r_1 and r_2, secrets K and s, and a keyed hash function f.

Tag Impersonation To impersonate a tag, an attacker must be able to compute a valid response M_3. However, it is difficult to compute such a message without knowledge of a secret K which is used as a key for a keyed hash function f.

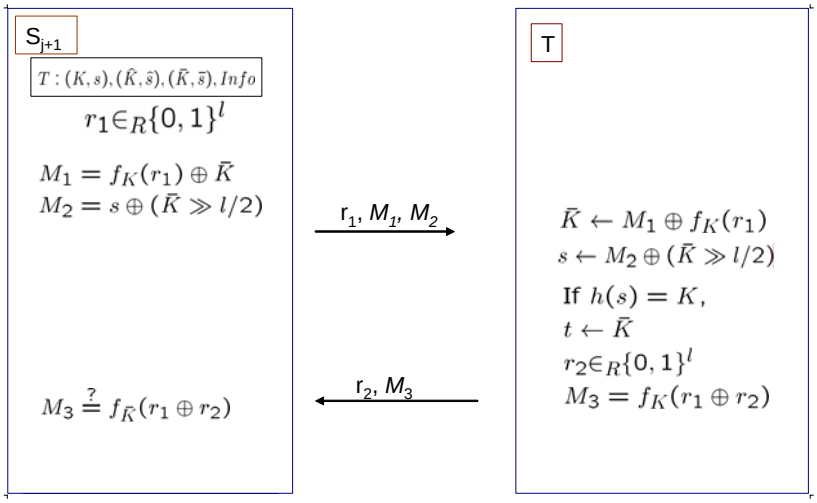

Figure 3.6: Authorization Recovery

Denial-of-Service If an adversary prevents message M_3 reaching S_{j+1} in P2 (or P3). Then T will update its identifier, but S_{j+1} will not. However, S_{j+1} knows the updated value of K, and can use the value to recover synchronization with T.

Forward Security T's secret K is updated using a one-way hash function h, and hence the previous secret identifiers of T are not computable.

The protocols also resist eavesdropping, replay attacks, and man-in-the-middle attacks.

The schemes satisfy the identified privacy and security requirements. P2 and P3 have desirable performance characteristics; a tag needs l bits of non-volatile memory to store its secret K, performs just three hash function computations. P1, P2, and P3 also provide all the identified requirements for tag ownership transfer.

3.8 RFIDDOT: RFID Delegation and Ownership Transfer made simple [DOT]

3.8.1 Introduction

This work [30] focuses on secure delegation and ownership transfer of RFID tags. A protocol suite has been showed that allows a user to acquire, access, transfer and release tags without undermining the privacy of current or future owners. This protocol is demonstrated to be secure against attacks such as violation of privacy, tag cloning, tag/reader spoofing, desynchronization. Furthermore, RFIDDOT ensures forward and backward privacy which is a prerequisite in secure RFID ownership transfer.

3.8.2 Protocol Description

This protocol assumes that user carries mobile reader embedded in cell/PDA [31] which is also justified by the establishment of the Near Field Communications (NFC) forum [33] that aims in integrating mobile phones with existing passive RFID products. Each tag T contains secret keys K_{-1} and K_i shared with the original owner's server and the current owner respectively. During the Key update operation, tag-to-reader responses (back channel) are protected from eavesdropping. This protocol is applicable to the cases where a user has acquired a set of tagged products and wants to use them in a private environment (i.e. home). While buying, user connects to a check-out server through a secure channel to receive tag information.

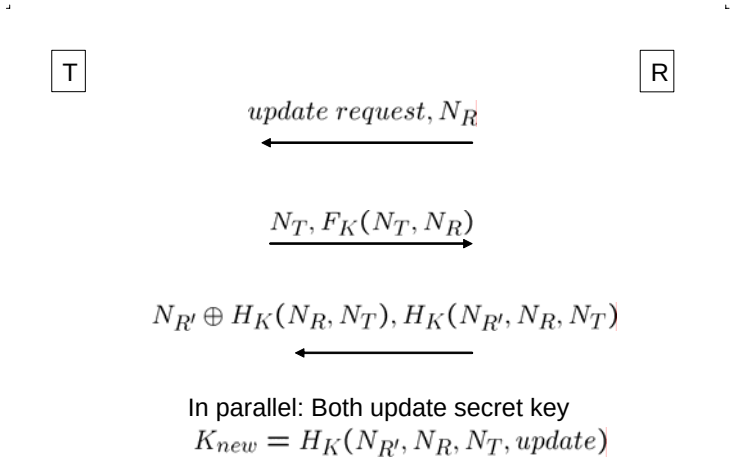

Figure 3.7: Ownership transfer

Ownership Transfer In the ideal situation, the user should take complete control of the tag and that no one should be able to trace the tag any more. The mobile or home

reader of the user signifies the message exchange to be an update request so that the tag in the end of the protocol updates its secret key. Towards that purpose, the reader first computes random number N'_R and transmits the third message shown in the figure. Upon receiving this message, the tag first computes $H_K(N_R, N_T)$ and then extracts the random number N'_R. The second part, $H_K(N'_R, N_R, N_T)$ is used to verify that N'_R was retrieved successfully by the tag, since if an adversary modifies the first part, the tag would end up with a wrong N'_R and eventually with a wrong key. If all tests pass successfully, both the reader and the tag update the current key to a new key K_{new}. This new key is used in subsequent transactions between the tag and the reader. And since K_{new} is generated by an unpredictable N'_R and fresh random data like N_T and N_R, the back-end database cannot any more track or identify the tag.

3.8.3 Discussion

In case of more than one owners, a timeline is considered, where $Owner_i$ denotes the i-th owner of the tag and K_i the key that is handed over from $Owner_i$ to $Owner_{i+1}$. $Owner_0$ denotes the back-end database and K_0 the key that is handed over to the first owner. K_{-1} is the original key of the tag, one that is never revealed and only used by the back-end system itself.

Privacy The protocol enforces privacy since no fixed identifiers are emitted (continuous use of unpredictable numbers) and no information ever leaks because of the one-wayness of H. $Owner_{i-1}$ also cannot recover N'_R since she has no access to the update messages that take place in the privacy of $Owner_i$'s home. Thus, while $Owner_{i-1}$ knows K_{i-1}, she does not have access to the message containing N'_R.

Cloning The approach used here is to let the tag generate the new key using the one-way function H. The one-wayness of H guarantees that given a target key K and a nonce N_T supplied by the tag, a malicious owner cannot come up with N_R and N'_R that generate this given key. Thus tag cloning is practically impossible.

Forward and Backward Security Before $Owner_{i-1}$ handovers her secret to $Owner_i$, she updates the key one more time. Hence the key that is given to $Owner_i$ is not really connected to the key that $Owner_{i-1}$ used to access the tag and therefore traceability is not possible. Similarly, in order to protect the back-end database($Owner_0$), the key K_0 that is handed over to the first owner does not have to be the original key (say K_{-1}) that is used by the retail store but it can be the result of an update operation on K_{-1}. Backward security suggests that knowledge of a tag's state at time t cannot help in identifying the tag's transactions at time $t' > t$. This is true under the assumption that an adversary compromising the current tag key of $Owner_i$ cannot eavesdrop on all future interactions of the tag. Thus, if the same owner ever updates the tag key again or hands over the tag to a new owner who securely updates the key, then future interactions are secured.

This proposal allows users to securely own, transfer or release tags. It also ensures tag-reader authentication and protection against such attacks as invasion of privacy, tracking, tag or reader impersonation, tag cloning. The protocol supports both forward and backward security and captures the requirements that are necessary to make ownership transfer secure under a tight assumption. Using RFIDDOT, ownership transfer is achieved by a simple update operation that ensures the privacy of past and future owners.

3.9 A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags [MSW]

MSW protocols by Molnar, et al. [10] use hierarchical tree based keying to allow for gradual and efficient tag identification/ authentication. Tree-based keying involves a tree of degree k and depth $t = log_k(n)$ where n is the total number of tags. Each tag holds $t = log_k(n)$ distinct keys and, upon being challenged by a reader, responds with t MACs each based on a different key. But MSW protocols have a security flaw 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. This vulnerability of MSW protocols has been explored by Avoine, et al. [12].

3.10 Comparison

We summarize the features of all the previous works in this section. The table 3.1 focuses on the security features provided by various protocols. The table 3.2 shows the comparison of the protocols from the view point of computational, searching and communication cost.

All the works described in this chapter are compared in this section. The YA-TRIP protocol, as presented above, has a potential drawback. It is susceptible to a trivial denial-of-service (DoS) attack: an adversary sends a wildly inaccurate (future) time-stamp to a tag and incapacitates it either fully (if the time-stamp is the maximal allowed) or temporarily. YA-TRAP is susceptible to DoS attacks whereby a rogue reader can easily incapacitate a tag by feeding it a futuristic (or even maximum)T_r value. Moreover, YA-TRAP* is susceptible to *timing attacks*. The execution the last OR clause in step 2.3 in Figure 3 is dependent upon the offset of T_r which can be freely selected by the adversary. Consequently, the adversary can mount a timing attack aimed at determining the epoch corresponding to the tag's last time-stamp of use (T_t). Although the scheme is not initially forward secure, it can be made forward secure by adding an extra step in tag's operation by introducing another hash function.

MSW protocols have a security flaw 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.

REP prevents an adversary from swapping the identities of two different tags. corruption of tag data has a second effect: Denial of service. The technique proposed involves random input from the tag in the creation of pseudonyms, works even when tags do not

have access-control features. If an attacker is able to implant a pseudonym in a tag, the tag effectively becomes desynchronized with the REP: The REP no longer recognizes the tags pseudonym. An effect of a corruption attack is that the REP cannot properly release tags: If it does not recognize their pseudonyms, it cannot manage them properly.

IE cannot avoid cloning attack as it stores the ID and encrypted data in the same memory area; also it does not assume tamper-proof tags that engage in interactive protocols for proof-of-knowledge of secrets. Insubvertible encryption does not defend against swapping attack; in which an adversary exchanges two valid ciphertexts across RFID tags. In EC-ZK, if an attacker eavesdrops the reading operation of a tag, he will obtain a witness, a timestamp and a response. Since the timestamp must be greater than the last-heard time of any tag, he cannot reuse these values with the tag involved in the communication he has eavesdropped; but, if there were tags in the system whose last-heard time were still lower than the obtained by the attacker, these tags would be vulnerable to a replay attack. So, only the interactive version of the protocol avoids this attack. Reader verification and secret point generation takes < 10000 logical gates for a tag. Communications for the advanced protocol takes 297 bits for 1st message; 112 bits for the second message which makes a total of 409 bits < 520 bits. A pseudo-id sniffed at a certain moment, cannot be related with the information obtained before (or after), because a bitwise xor is considered a secure ciphering algorithm for small data (the same reasoning applies to the encrypted sensor data).

The scheme in Sync provides the anonymity of the tag under the assumption that the hash function is one-way. Moreover, the scheme is unlikable since adversary has no control on the value of random nonce sent by the tag. Again, if the used hash functions are secure, the messages are useless for adversary. The scheme also provides the forward-privacy property using the same argument.

RFIDDOT allows users to securely own, transfer or release tags. It also ensures tag-reader authentication and protection against such attacks as invasion of privacy, tracking, tag or reader impersonation, tag cloning. The protocol Supports both forward and backward security and captures the requirements that are necessary to make ownership transfer secure under a tight assumption. Using RFIDDOT, ownership transfer is achieved by a simple update operation that ensures the privacy of past and future owners.

The OT protocols resist eavesdropping, replay attacks, and man-in-the-middle attacks. The schemes satisfy the identified privacy and security requirements. P2 and P3 have desirable performance characteristics; a tag needs l bits of non-volatile memory to store its secret K, performs just three hash function computations. P1, P2, and P3 also provide all the identified requirements for tag ownership transfer.

Table 3.1: Performance Comparison (Security)

Features	IE	REP	YA-TRIP	YA-TRAP	YA-TRAP*
Cloning Attack	P	NP	NP	NP	NP
Swapping Attack	P	NP	NP	NP	NP
Timing Attack	NP	NP	NP	NP	P
Tag Tracking	NP	NP	NP	NP	NP
DoS Attack	NP	NP	P	P	P
Reader Authentication	No	No	No	No	No
Forward Secure	yes	yes	yes	yes	yes
Other Tag Compromise	NP	NP	NP	NP	NP

Features	MSW	EC-ZK	SQUASH	Sync	OT	DOT
Cloning Attack	NP	NP	NP	-	NP	NP
Swapping Attack	NP	-	NP	-	NP	NP
Timing Attack	NP	-	-	-	-	-
Tag Tracking	NP	NP	NP	-	NP	NP
DoS Attack	NP	NP	P	NP	NP	NP
Reader Authentication	no	yes	no	yes	yes	yes
Forward Secure	no	yes	no	yes	no	yes
Other Tag Compromise	P	NP	NP	NP	NP	no

- NP = Not Possible
- P = Possible
- - = Not considered by the protocol

Table 3.2: Performance Comparison (Cost)

Features	IE	REP	YA-TRIP	YA-TRAP	YA-TRAP*
Server Cost(searching)	O(n)	O(n)	O(n)	O(n)	O(n)
No. of Tag Computation	no	1 Hash	1 Hash	3 Hash	5 Hash
Tag Storage Cost	1key	1key	1key	1key	1key
No. of Msg. flows	2	2	2	2	2
Tag-Reader Comm.	5b	-	b	3b	3b
Reader-Server Comm.	5nb	-	nb	3nb	(3n+2)b
NAND Gate	< 1000	< 1000	< 1000	< 1000	< 1000

Features	MSW	EC-ZK	SQUASH	Sync	OT	DOT
Server Cost	$O(n\log_k n)$	-	k^2	O(n)	O(n)	O(n)
No. of Tag Computation	O (log n)	EC point	k^2	4 hash	4 hash	4 hash
No. of Msg. flows	2	4	2	4	2	3
Tag Storage Cost	O (log n) keys	2 keys	1 key	1 key	1 key	1 key
Tag-Reader Comm.	3b	2b	b	2b	2b	2b
Reader-Server Comm.	3nb	2nb	nb	2nb	2nb	2nb
Gate count	< 1000	< 10000	< 1000	< 1000	< 1000	< 1000

- n = total number of tags.
- b = bit length of Messages (assuming all are equal in bit size).
- - = Not considered by the protocol

$\diamondsuit$ REP requires an extra device with each tag to be attached with.

In the tables above, we assume that the hash functions take the same computational cost like the PRNG. From the tables we can see that, not all the protocols can satisfy the security requirements. Some of them also require high communication, computation cost. EC-ZK takes about 10000 gates, which is well beyond the reach of today's passive tags. Moreover, reader authentication is also not supported by some of them. But reader authentication is required to thwart illegal tracking or cloning of the tags. It is also important to keep the number of messages passed as low as possible. More than 2-pass protocols require more communication overhead, and a tag requires to 'remember' all the intermediate states to complete the protocol. All this translates into needing more resources on the tag. In REP, the protocol assumes an external device to be attached with the tag. This is not practical for applications where batch-mode authentication is done. Compared to all of the previous works, YA-TRAP* is the most efficient one. YA-TRAP* satisfies most of the security requirements and requires minimum communication cost.

Chapter 4

Preliminary Version of A Low Cost and Secure RFID Authentication Scheme

4.1 Introduction

In this chapter, we improve the efficiency of a trivial RFID authentication protocol named YA-TRAP* in a cost effective way specially for batch-mode. The protocol YA-TRAP* is vulnerable to DoS attacks. 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 two-way 2-pass authentication protocol which helps a tag recover from the dead state. The tag to reader and reader to server communication costs are reduced while maintaining the same level of security. Moreover, we reduce a tag's computation requirement while communicating with the reader.

4.2 Motivation

As shown in the previous chapter, YA-TRAP* is the most efficient one compared to all of the previous works. YA-TRAP* satisfies most of the security requirements and requires minimum communication cost. This is why we pick YA-TRAP* to compare with our work mainly.

The protocol YA-TRAP* is vulnerable to DoS attacks. DoS resistance in YA-TRAP* is limited by the magnitude of the system-wide INT parameter. Once generated by the server and distributed to the genuine readers, and the readers forward to the tags, 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 within the 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}.

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 batch-mode authentication. Considering all the messages have same bit length b, the tag to reader message is $3b$ bits long and the reader to server message is $5b$ bits long. That means, for n number of tags, the reader to server message becomes $5nb$ bits long - which amounts a huge burden on communication.

Moreover, even though 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.

4.3 Our Scheme

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 [9] 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.

We will describe our scheme here. Instead of using HMAC, we use SQUASH- a new hash proposed by Shamir [9]. This requires the least number of gate counts so far among the existing hash functions. Also unlike MAC operations, it operates in single block. These features are very much suitable for highly resource constrained devices like RFID. We don't need a tag to generate random number R_t as it does not add to the security requirements. Rather it requires an extra computation by a tag. A batch-mode communication needs the involved entities make their computation as quick as possible. So, for the sake of computational burden on a tag, we avoid generating random challenge. A tag computes hash of its last updated T_t, hash of R_r and its key k_i and makes H_{auth_i} by XOR-ing them. The tag then sends this H_{auth_i} and T_{r_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 [21]. Moreover, the reader concatenates all the T_{r_i} into one message T_r. The reader forwards H and T_r to server. Upon receiving them, the server looks up its pre-computed hash values against each T_{r_i} 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 (Proposed Algorithm)

[1]$Tag \leftarrow$ *Reader:* $T^j_{r_i}, R^j_{r_i}, ET_{r_i}$

[2] Tag_i*:*

[2.1] $\delta = T^j_{r_i} - T^j_{t_i}$

[2.2] $\nu = \lfloor T^j_{r_i}/INT \rfloor - \lfloor T^j_{t_i}/INT \rfloor$

[2.3] *If* $(\delta \leq 0)$ *or* $T^j_{r_i} > T_{max_i}$ *or* $H^\nu(ET_{t_i}) \neq (ET_{r_i})$, *then* $H_{id_i} = PRNG^j_i$, $H_{id'_i} = PRNG^{j+1}_i$, $H_{id'_i} = PRNG^{j+2}_i$

[2.4] *else* $T^j_{t_i} = T^j_{r_i}, ET_{t_i} = ET_{r_i}, H_{id_i} = HASH(T^j_{t_i}), R_{t_i} = PRNG^{j+1}_i, H_{id_{i'}} = HASH(R_{t_i}, R_{r_i})$

[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] $if \bigoplus_{i=1}^{n}(HASH(T_{t_i}, \oplus HASH(R_{t_i}, R_{r_i})) \neq H$,

[6.1.1] *MSG=TAG-AUTH-ERROR*

[6.2] *else MSG=TAG-VALID*

[7] *Server* $\rightarrow$ *Reader: MSG*

4.4 Achievement of Our Scheme

The server keeps the list of T_r, corresponding tag's secret k_i. So, it becomes possible for the server to identify if there is any particular rogue tag. We can use SQUASH proposed by Shamir as the HASH functions, therefore needing a small number of gates in the Tags. We reduce the number of required PRNG and or hash computations in tags, while maintaining the same level of security. Keeping the same level of security is possible because of the use of one-way hash function. For a highly resource constrained device like RFID tags, to guess the ki from $H_{id_{i_0}}$ takes a huge amount of computation by an adversary which is infeasible. 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 n number of tags, the communications needs only 2 messages instead of 5n messages in case of the original scheme. Only using aggregate function does not guarantee to find

out single rogue tag. To make sure that a server can find out a single rogue tag, we use T_r- a concatenated value of all the T_{R_i}. Upon receiving this value, the server can extract the individual T_{r_i} values and search the hash table entry to find and match with the corresponding hash values those were precomputed.

4.5 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} 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 two way challenge-response method where both the tag and reader authenticate each other and then the reader replaces T_{max_i} sending the new value $T_{max_{i_{new}}}$ to the tag. The server keeps the table of valid tags and the last issued timestamp T_r 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 saved by the tag and T_{max_i} . Sending the value 1 indicates the readers 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 2n 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 distinguishing the $H(k_i, T_{max_i})$ from a random value. This indistinguishability feature is required to protect against narrowing attack which leads into tracing the tag by an adversary. The use of PRNGs to obfuscate the tag identity was first introduced in [18]. 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. The reader sends the value T by XOR-ing the T_{max_i} and $T_{max_{i_{new}}}$. This two-way authentication is done by using hash functions. And its security is dependent on the one-wayness 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 (Recovering a non-operative tag)

[1] $Tag \leftarrow Reader$: *1,* $H(T_t, T_{max_i})$
[2] Tag_i:
  [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* $T_{max_{i_{new}}} > T_{max_i}$

[4.2] $T = T_{max_i} \oplus T_{max_{inew}}$
[4.3] *Set* $T_{max_i} = T_{max_{inew}}$ [5] $Tag \leftarrow Reader$
[5.1] *Extract* $T_{max_{inew}}$ *from* T
[5.2] *Set* $T_{max_i} = T_{max_{inew}}$

4.6 Performance of the Scheme

We present the performance of our scheme here. Table 4.1 shows the performance of security features like Forward Security, DoS resistance, Tag Tracing and whether tag becomes non-operative. Our proposed scheme achieves DoS resistance capability by helping a tag to get operative from non-operative state. Table 4.2 is performance of cost. It includes tags computation, Tag to Reader communication and Reader to Server communication costs. We improve the efficiency of an RFID authentication protocol 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 does not allow a tag to be non-operative.

Table 4.1: Performance (Security)

Forward Secure	DoS resistant	Tag Tracing	Tag Non-Operative
yes	yes	no	no

Table 4.2: Performance (Cost)

Tag computation	Tag to Reader communication	Reader to Server communication
2 HASH 1 PRNG	2b	(n+1)b

4.7 Discussion

The proposed protocols in the chapter are not secure and cost effective enough. The algorithm 2 is resistant to the attacks. But is subject to tracing attack while a tag tries to recover from a non-operative state in the algorithm 3. This is because, the recovery from the non-operative state requires the reader to send '1' as an indicator to update the tag. Moreover, the aggregate function only reduces the communication cost while there is no way available to the server to authenticate tags individually. The next chapter addresses these issues and presents fresh protocols which have low cost and satisfy the security requirements.

Chapter 5

A Low Cost and Secure RFID Authentication Scheme

5.1 Introduction

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 low cost and secure RFID authentication scheme solves the problem by allowing a tag to refresh its prestored threshold value. Moreover, our scheme provides reader authentication to thwart tag cloning. 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 rogue tags out of the aggregate function. The level of security is also improved compared to the previous works.

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.

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. 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 [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. Cloned fake RFID tags and malicious RFID readers pose a major threat to the RFID-based supply chain management system. 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. These

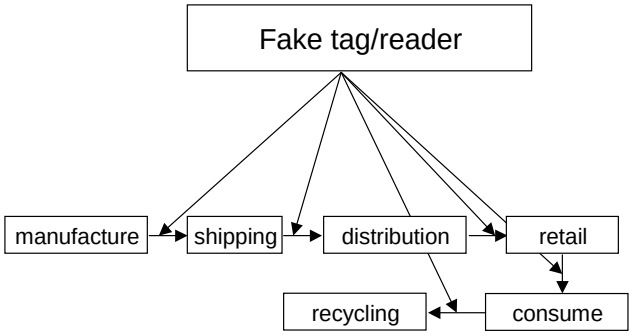

Figure 5.1: Attack model in RFID enabled Supply Chain

cloned fake tags can be attached to counterfeit products, which can be introduced into a genuine supply chain, or illegally sold at black and grey markets. Starting from shipping the tags from manufacturers to recycling by after being consumed, fake tags or readers might intrude in any of the steps shown in the figure 5.1. 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 chapter, 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* [8], 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 provides reader authentication to thwart tag cloning.

In the extended version of our scheme, the reader uses partial authentication to keep the rogue tags out of the aggregate function. A simple aggregate function enables a server only to find out the anomaly of the resultant XOR operations of the hash values. That means, if the pre-computed aggregate hash value does not match with the received aggregate hash value, 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. 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. However, the level of security is improved compared to the previous works as it provides resistance against DoS, Cloning, Replay, Tracking attacks. It also provides forward security and does not allow a tag to become non-operative unless explicitly done by the reader.

5.2 Operating Environment

In this section, we discuss our assumptions and the operating environment.

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 [9] 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.

5.3 Motivation

Many works have been done on secure RFID authentication [8, 4, 3, 12, 9, 10] in recent past. MSW protocols by Molnar, et al. [10] use hierarchical tree based keying to allow for gradual and efficient tag identification/authentication. But MSW protocols have a security flaw 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 [13]. This reduces the computation complexity and communication complexity, which are two prime factors related to less energy consumptions of an RFID system. But they do not show the use of the aggregate function in a fully authentication protocol. Moreover, it is not clear from their protocol how to find out individual rogue tags by the verifier.

Lee et.al. show ECRAC- a Provably Secure RFID authentication protocol based on ECDLP [20]. This is a two-way authentication protocol which authenticates only the tag enabling the vulnerability to cloning attack. More importantly, this protocol requires around 15000 gate counts (NAND gate equivalent) which is well beyond the capability of today's low cost passive RFID tags.

YA-TRAP* by Tsudik [8] is vulnerable to DoS attacks. DoS resistance in YA-TRAP* is limited by the magnitude of a system-wide parameter. Once generated by the server and distributed to the genuine readers, and the readers forward to the tags, the epoch token used is no more secret. Therefore, the adversary can still incapacitate tags for at most the duration of system-wide parameter if it queries each victim tag with the current epoch token and the maximum possible value within the current epoch. Moreover, an adversary can permanently incapacitate a tag by feeding arbitrary future timestamps and making it exceed the stored threshold value.

5.4 Our Low Cost and Secure 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 B, we extend the protocol into a more secure one which provides partial authentication of the tags. But in the extended version, a tag's computation increases by one hash function and the tag to reader communication cost increases 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., [14] for discussion). We briefly recall the underlying idea. If two parties share a secret onetime pad p, namely a random bit string, 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 intercepts 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 [13] 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 because 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 distinguishing the $Hash(T_{r_i}^{j-1}, T_{max_i})$ from a random value. This indistinguishability feature is required to protect against narrowing attack [8] which leads into tracing the tag by an adversary. The use of PRNGs to obfuscate the tag identity was first introduced in [18].

As discussed earlier, a valid tag can become non-operative after reaching the threshold T_{max_i} even 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}.

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 find out a valid tag when it becomes non-operative. 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 (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 $T_{max_{inew}}$. Generating $T_{max_{inew}}$ works as onetime padding as it is freshly computed every time 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 $T_{max_{inew}}$ is strictly greater than T_{max_i}. Even though 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 checks 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 [21]. 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 $MSG = TAG\text{-}VALID$ back to reader to end the whole process.

The Algorithm 4 below is the secure and low cost authentication protocol:

Algorithm 4 (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_i}, *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*

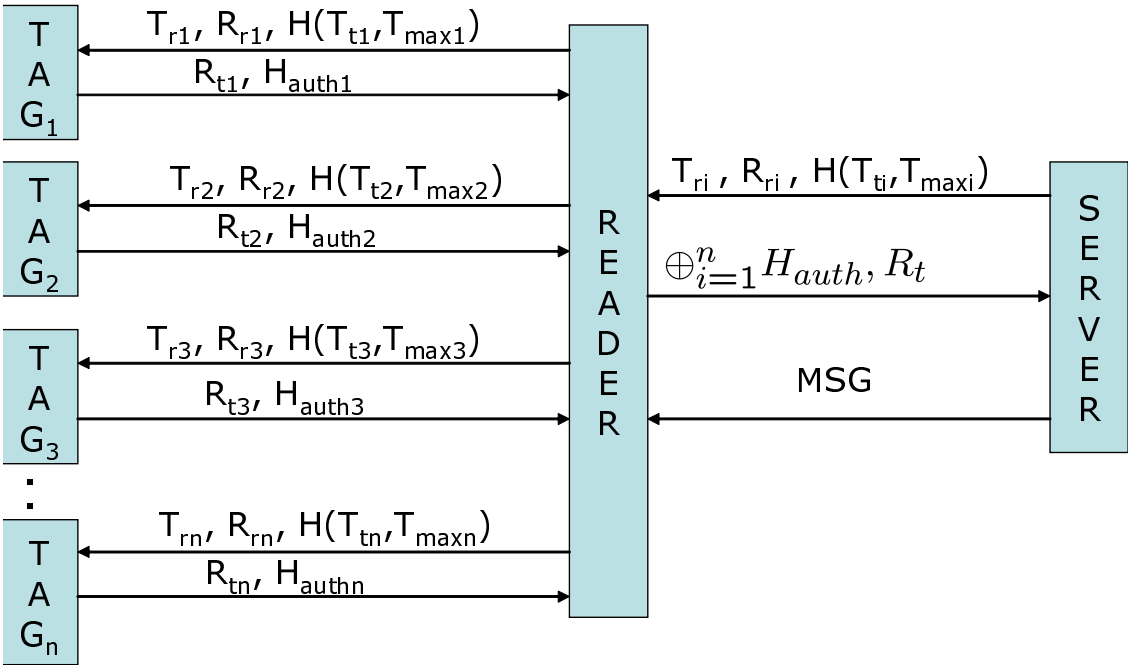

Figure 5.2: Our Scheme: Without Partial Authentication

5.4.1 Achievement of Our Scheme

The two-way authentication is done by using hash functions. And its security is dependent on the one-wayness of the underlying hash function. 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 anomaly in the aggregated value due to a rogue tag.

The cost of Reader to Server communication is drastically reduced by introducing aggregate function. For n number of tags, the communications needs only $(n + 1)b$ bits instead of $(3n + 2)b$ bits compared to YA-TRAP*.

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}. The reader replaces T_{max_i} sending the new value $T_{max_{i_{new}}}$ to the tag. 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.

Reader authentication is also provided in our scheme. This is particularly important to prevent tag cloning. In our scheme, a tag never generates genuine replies unless it verifies the reader first.

5.4.2 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 authentication 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 computations 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 corresponding 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 notify that, the extended version of the protocol increases communication cost for both the tag to reader and server to reader interaction. However, sacrificing a part of communication cost strengthens the security. We consider this trade-off 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 rogue 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 5 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 (Police checking drivers' licenses with mobile readers is such a condition, where the bulk of the licenses are finally authenticated later by the server; or, inventory control system where readers are deployed in a remote warehouse and have no means of contacting a back-end server, for examples). 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.

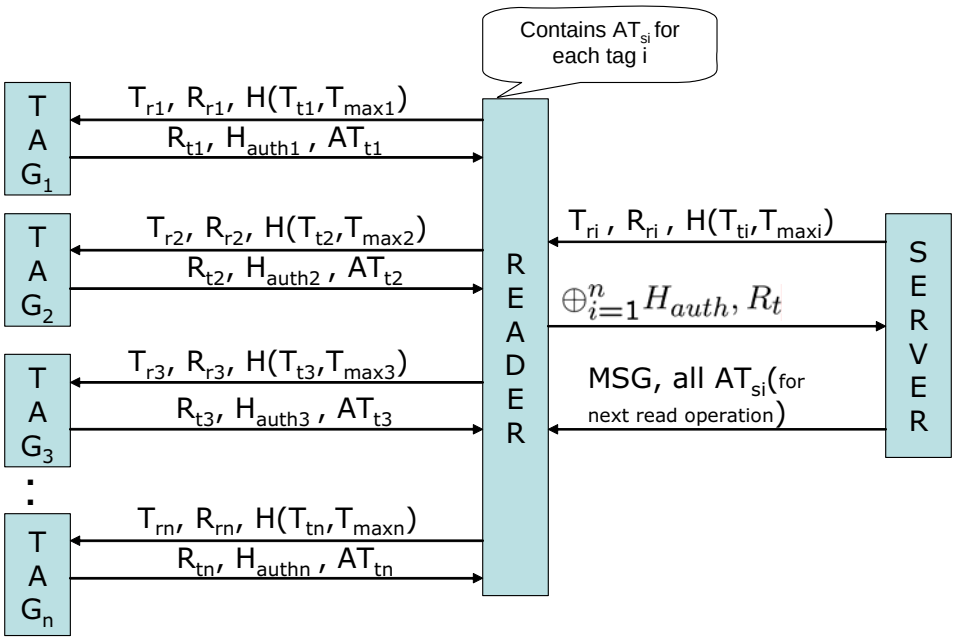

Figure 5.3: Our Scheme: With Partial Authentication

5.5 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 because 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 challenges 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 different sessions with a same tag are equal. This prevents the replay attack.

• *Tag Non-operative*: Even though a tag is valid, in YA-TRAP*, 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. 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, thus having full control over a tag.

5.6 Performance

As discussed earlier, our scheme achieves all the security requirements; i.e., forward security, DoS resistance, replay attack resistance. Moreover, it provides reader authentication, it does not allow a tag to become non-operative, and it also resists cloning attack and track tracing. Our scheme is also efficient from the view point of computational cost, Tag to Reader communication and Reader to Server communication costs. Considering the extended version, our scheme requires 4 hash functions and one PRNG by a tag. Also the tag to reader communication contains $3b$ bits. The bit length of reader to server communication is low in our scheme. For n number of tags, our scheme requires $(n+1)b$ bits for the reader to server communication. This clearly reduces the cost by a significant amount which is very important for batch-mode communication.

Table 5.1: Performance (Security)

Forward Secure	DoS resistance	Replay Attack resistant	Tag Non-Operative	Reader Authentication
yes	yes	yes	no	yes

Table 5.2: Performance (Cost)

Tag computation	Tag to Reader communication(bits)	Reader to Server communication(bits)
4 HASH 1 PRNG	3 b	$(n+1)b$

$\diamondsuit$ n = total number of tags.

$\diamondsuit$ b = bit length of HASH, R_{t_i} , R_{r_i} , T_{r_i} (assuming all are equal in bit size).

Chapter 6

Performance Comparison

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 identify a rogue 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 clarity 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 rogue 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*.

Tables show the comparison of security features like Reader Authentication, Forward Security, DoS resistance, Replay Attack resistance and whether tag becomes non-operative. Even though forward security is provided both by YA-TRAP* and our scheme, our scheme

achieves better performance considering DoS attack resistance, replay attack resistance. 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. 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. Furthermore, it enables the reader to synchronize a tag even if it becomes desynchronized due to an attack by the adversary.

Table 6.2 includes tag's computation, Tag to Reader communication and Reader to Server communication 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 one PRNG. Also the tag to reader communication contains $3b$ bits for both scheme. 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.

Moreover, in YA-TRAP*, a valid tag can become incapacitated after exceeding the T_{max_i} value. In our protocol, 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).

Table 6.1: Performance Comparison (Security)

Features	IE	REP	YA-TRIP	YA-TRAP	YA-TRAP*
Cloning Attack	P	NP	NP	NP	NP
Swapping Attack	P	NP	NP	NP	NP
Timing Attack	NP	NP	NP	NP	P
Tag Tracking	NP	NP	NP	NP	NP
DoS Attack	NP	NP	P	P	P
Reader Authentication	No	No	No	No	No
Forward Secure	yes	yes	yes	yes	yes
Other Tag Compromise	NP	NP	NP	NP	NP

Features	MSW	EC-ZK	SQUASH	Sync
Cloning Attack	NP	NP	NP	-
Swapping Attack	NP	-	NP	-
Timing Attack	NP	-	-	-
Tag Tracking	NP	NP	NP	-
DoS Attack	NP	NP	P	NP
Reader Authentication	no	yes	no	yes
Forward Secure	no	yes	no	yes
Other Tag Compromise	P	NP	NP	NP

Features	OT	DOT	Our Scheme-1	Our Scheme-2
Cloning Attack	NP	NP	NP	NP
Swapping Attack	NP	NP	NP	NP
Timing Attack	-	-	NP	NP
Tag Tracking	NP	NP	P	NP
DoS Attack	NP	NP	NP	NP
Reader Authentication	yes	yes	yes	yes
Forward Secure	no	yes	yes	yes
Other Tag Compromise	NP	NP	NP	NP

- NP = Not Possible
- P = Possible
- - = Not considered by the protocol

Table 6.2: Performance Comparison (Cost)

Features	IE	REP	YA-TRIP	YA-TRAP	YA-TRAP*
Server Cost(searching)	O(n)	O(n)	O(n)	O(n)	O(n)
No. of Tag Computation	no	1 Hash	1 Hash	3 Hash	5 Hash
Tag Storage Cost	1key	1key	1key	1key	1key
No. of Msg. flows	2	2	2	2	2
Tag-Reader Comm.	5b	-	b	3b	3b
Reader-Server Comm.	5nb	-	nb	3nb	(3n+2)b
NAND Gate	< 1000	< 1000	< 1000	< 1000	< 1000

Features	MSW	EC-ZK	SQUASH	Sync
Server Cost	O($n\log_k$ n)	-	k^2	O(n)
No. of Tag Computation	O (log n)	EC point	k^2	4 hash
No. of Msg. flows	2	4	2	4
Tag Storage Cost	O (log n) keys	2 keys	1 key	1 key
Tag-Reader Comm.	3b	2b	b	2b
Reader-Server Comm.	3nb	2nb	nb	2nb
Gate count	< 1000	< 10000	< 1000	< 1000

Features	OT	DOT	Our Scheme-1	Our Scheme-2
Server Cost	O(n)	O(n)	O(n)	O(n)
No. of Tag Computation	4 hash	4 hash	4 hash	5 hash
No. of Msg. flows	2	3	2	2
Tag Storage Cost	1 key	1 key	2 keys	2 keys
Tag-Reader Comm.	2b	2b	2b	2b
Reader-Server Comm.	2nb	2nb	(n+1)b	(n+1)b
Gate count	< 1000	< 1000	< 1000	< 1000

- n = total number of tags.
- b = bit length of Messages (assuming all are equal in bit size).
- - = Not considered by the protocol

$\diamondsuit$ REP requires an extra device with each tag to be attached with.

Chapter 7

Conclusion and Future works

7.1 Conclusion

In this literature, we have studied some recent secure RFID authentication protocols and have compared the various features provided by them. Then we have found out the vulnerabilities in YA-TRAP, YA-TRIP and YA-TRAP* protocols. We have proposed a two-way message authentication protocol where both tag and reader authenticate each other. Compared to the previous RFID authentication protocols, our protocol is an improvement 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. In our two-way authentication protocol, a reader helps a tag to recover from the non-operative state. Furthermore, our schemes provide reader authentication to thwart tag cloning.

In a extended version, the reader uses partial authentication to keep the rogue tags out of the aggregate function. A simple aggregate function enables a server only to find out the anomaly of the resultant XOR operations of the hash values. That means, if the pre-computed aggregate hash value does not match with the received aggregate hash value, 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. 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. However, the level of security is improved compared to the previous works as it provides resistance against DoS, Cloning, Replay, Tracking attacks. It also provides forward security and does not allow a tag to become non-operative unless explicitly done by the reader.

7.2 Future works

Searching cost by the server for individual tags has to be reduced. As we have seen, many privacy-preserving authentication protocols are proposed till now. But no research

still treats key management of the tags. RFID released within an RFID infrastructure should/would migrate to another RFID infrastructural domain; Gillette and Kobe-beef RFID empowered goods (once bought) should merge to another domain, as for example. Robust, Delay Tolerant computations are also yet to get focused by the researchers. Moreover, still unaddressed is how to handle revocation of rogue readers.

Acknowledgment

I would like to express my deep sense of gratitude to my advisor Professor Atsuko Miyaji and Associate Professor Masakazu Soshi from the Hiroshima-City University for their invaluable guidance, constructive advice and constant encouragement during this work. Professor Miyaji's deep intuition for theory and her eye for the detail, while never losing track of the big picture were really inspiring. I have been very fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. Associate Professor Soshi's enthusiasm and unlimited zeal have been major driving forces to his research. A very special thank to my whole laboratory which has brought unmatched influence to me.

Bibliography

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

[2] A. Juels, R. Pappu: Squealing euros: Privacy protection in RFID enabled banknotes. *Financial Cryptography-FC 03*, Vol. 2742 of Lecture Notes in Computer Science, pp. 103-121. IFCA, Springer-Verlag (2003).

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

[6] T. Zeller J. : Black Market in Stolen Credit Card Data Thrives on Internet. *http://www.nytimes.com/2005/06/21/technology/21data.html*

[7] M. L. Songini: RFID-enabled tattoos eyed for livestock tracking. *http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9009038*

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

[9] A. 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).

[10] 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).

[11] D. N. Duc, J. Park, H. Lee and K. Kim: Enhancing security of EPCglobal Gen-2 RFID tag against traceability and cloning. *Symposium on Cryptography and Information Security- SCIS*, IEICE(2006).

[12] 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).

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

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

[15] R. Das and Dr. P. Harrop: RFID Forecasts, Players, Opportunities 2006-2016, *http://www.idtechex.com/research/*

[16] P. P. Lopez, J. C. H. Castro, J. M. E. Tapiador and A. Ribagorda: Advances in Ultralightweight Cryptography for Low-cost RFID Tags: Gossamer Protocol. *Workshop on Information Security Applications- WISA*, Springer-Verlag(2008)

[17] R. Das and Dr. P. Harrop: RFID Forecasts, Players, Opportunities 2007-2017, *http://www.idtechex.com/research/*

[18] 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).

[19] N. W. Lo and K. Yeh: An Ecient Mutual Authentication Scheme for EPCglobal Class-1 Generation-2 RFID Systems. *Emerging Directions in Embedded and Ubiquitous Computing*, pp: 43-56, Sringer-Verlag(2007)

[20] Y.K. Lee, L. Batina, I. Verbauwhede: EC-RAC (ECDLP Based Randomized Access Control): Provably Secure RFID authentication protocol *IEEE International Conference on RFID*,pp. 97-104, IEEE(2008).

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

[22] M. Burmestera and B. de Medeiros: The Security of EPCGen2 Anonymous compliant RFID Protocols. *Applied Cryptography and Network Security- ACNS*, pp: 490-506, Springer-Verlag(2008).

[23] G. Ateniese, J. Camenisch, B.de Medeiros: Untraceable RFID Tags via Insubvertible Encryption. *CCS*, CSS(2005).

[24] M. Rieback, B. Crispo, A. Tanenbaum: Is your cat infected with a computer virus? *IEEE Pervasive Computing and Communications*, pp. 169-179, IEEE(2006).

[25] A. Juels, P. Syverson, and D. Bailey: High-Power Proxies for Enhancing RFID Privacy and Utility. *Privacy Enhancing Technologies (PET) Workshop*, pp. 210-226. Springer-Verlag(2005).

[26] S. Martinez, M. Vallas, C. Roig, F. Gine, J.M. Miret: An Elliptic Curve and Zero Knowledge based Forward Secure RFID Protocol. *RFIDSec* 2007.

[27] P. Singer: A new approach to low cost RFID tags. *http://www.semiconductor.net/article/CA499653.html* Semiconductor International, February 1, 2005.

[28] S. CANARD, I. COISEL: Data Synchronization in Privacy-Preserving RFID Authentication Schemes. *RFIDSec* 2008.

[29] B. Song: RFID Tag Ownership Transfer. *RFIDSec* 2008.

[30] T. Dimitriou: RFIDDOT: RFID Delegation and Ownership Transfer made simple. *4th International Conference on Security and Privacy in Communication Networks-SecureComm*, ACM(2008).

[31] Nokia unveils RFID phone reader. *http://www.rfidjournal.com/article/view/834*. RFID Journal,17 March (2004).

[32] M. J. B. Robshaw: An overview of RFID tags and new cryptographic developments. *information security technical report II*, pp: 82-88. Elsevier2006

[33] *http://www.nfc-forum.org*

[34] T. S. H. Benjamin, D. V. Bailey, K. Fu, A. Juels, T. OHare: Vulnerabilities in first-generation RFID-enabled credit cards. *Technical report, http://prisms.cs.umass.edu/ kevinfu/papers/RFID-CC-manuscript.pdf*, 2006

[35] J. Park, J. Na and M. Kim: A Practical Approach for Enhancing Security of EPCglobal RFID Gen2 Tag. *IEEE Future Generation Communication and Networking-FGCN*, pp. 436-441, IEEE(2007)

Chapter 8

List of Papers

- "An RFID Authentication Protocol Suitable for Batch-mode Authentication"– Computer Security Symposium (CSS) 2008, October 8-10, Okinawa, Japan. (part of chapter 4)

- "A Secure RFID Authentication protocol with Low Communication Cost" – The 3rd International Workshop on Intelligent, Mobile and Internet Services in Ubiquitous Computing (IMIS 2009), IEEE (to appear). (part of chapter 5)

- "A Low Cost and Secure RFID Authentication Scheme"– IPSJ Journal (submitted) (part of chapter 5)