# SECURITY ANALYSIS OF ULTRALIGHTWEIGHT RFID PROTOCOLS

by

Serdar AKTAŞ

B.S., Electronics Engineering, Gebze Institute of Technology, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2011

SECURITY ANALYSIS OF ULTRALIGHTWEIGHT RFID PROTOCOLS

APPROVED BY:

| | |
|---|---|
| Prof. Emin Anarım<br>(Thesis Supervisor) | .................. |
| Prof. Kemal Cılız | .................. |
| Assist. Prof. Gökay Saldamlı | .................. |

DATE OF APPROVAL: 17.08.2011

## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Prof. Emin Anarım for his invaluable guidance and help during the preparation of this dissertation. I would like to mention his patience, giving me inspiration and hope when I was stuck at dead-ends.

I am heartily thankful to Assist. Prof. Gökay Saldamlı and Dr. İmran Ergüler whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject.

I would like to thank all the authors in the references list for making all the data available.

I also thank Tübitak Bilim İnsanı Destekleme Daire Başkanlığı for supporting this thesis with BİDEB 2210 programme.

# ABSTRACT

## SECURITY ANALYSIS OF ULTRALIGHTWEIGHT RFID PROTOCOLS

RFID (Radio Frequency Identification) is a technology that uses radio waves as a medium to exchange data between a reader and an electronic tag for the purpose of object identification and tracking. RFID tags are considered as a replacement technology for barcodes and other means of traditional identification tools which traditionally find applications in manufacturing, supply chain management and inventory control. Security and privacy aspects of RFID are becoming more important as RFID technology continues to flourish as an inherent part of virtually every ubiquitous environment. As the nodes of RFID systems mostly suffer from low computational power and small memory size, strong cryptographic protocols are not appropriate for low-cost RFID tags. Therefore, designing security mechanisms for RFID systems turn out to be very challenging that many authentication protocols have been proposed recently by an increasing number of researchers. Nevertheless, it is shown that majority of these proposals do not provide security and privacy. The work done in this M.S. thesis is to analyze the privacy and security aspects of ultralightweight RFID protocols defined in the previous literature and outline the weaknesses in these protocols. Also in a new ultralightweight RFID protocol we manage to attain a total breakdown by compromising the secret key information by revealing each bit using the weakness of XOR operation. Other weaknesses, we report for this ultralightweight RFID protocol include desynchronization, replay and traceability flaws.

# ÖZET

## HAFİF AĞIRLIKLI RFID PROTOKOLLERİNİN GÜVENLİK ANALİZİ

RFID (Radyo Frekansı ile Tanımlama), nesnelerin tanımlanması ve izlenmesi amacıyla, radyo dalgaları kullanarak bir okuyucu ve bir elektronik etiket arasında veri alışverişini sağlayan bir teknolojidir. RFID etiketleri, barkod gibi üretim, tedarik zinciri yönetimi ve stok kontrolünde kendine uygulama bulan diğer geleneksel tanımlama araçları yerine teknolojik bir değişim olarak düşünülmektedir. RFID teknolojisi hemen hemen tüm çevrelerin hazır ve olağan bir parçası olarak gelişmesine devam ederken, RFID' nin güvenlik ve gizlilik yönleri daha da önemli bir hale gelmektedir. RFID sisteminin bileşenleri çoğunlukla düşük işlem gücü ve küçük hafıza boyutlarından sıkıntı çektiği için, güçlü şifreleme protokolleri RFID etiketleri için uygun değildirler. Buna bağlı olarak, artan sayıda araştırmacı tarafından birçok doğrulama protokolünün yakın zamanda önerilmiş olması RFID için güvenlik ve gizlilik mekanizmaları tasarımının çok zorlu olduğunu ortaya çıkarmaktadır. Ancak yinede, bu önerilerin çoğunun güvenliği ve gizliliği sağlayamadığı gösterilmiştir. Bu yüksek lisans tezinde önceki literatürde tanımlanan hafif ağırlıklı RFID protokollerinin güvenlik ve gizlilik analizleri yapılmıştır ve bu protokollerdeki zayıflıkların anahatları çıkarılmıştır. Ayrıca yeni bir hafif ağırlıklı RFID protokolünde XOR işleminin zayıflığından yararlanılarak her bitin tanımlanması ile gizli anahtar ele geçirilip, tüm sistemin bozulabilmesi sağlanmıştır. Bu hafif ağırlıklı RFID protokolü için elde edilen diğer zayıflıklar ise eşzamansız olma, izlenebilirlik ve tekrarlama kusurlarıdır.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS

| | |
|---|---|
| $\mathcal{DB}$ | Database |
| $\mathcal{HW}$ | Hamming Weight |
| $\mathcal{R}$ | Reader |
| $\mathcal{T}$ | Tag |

# LIST OF ACRONYMS/ABBREVIATIONS

| | |
|---|---|
| EMAP | An Efficient Mutual-Authentication Protocol for Low-cost RFID Tags |
| GA | Good Approximation |
| IC | Integrated Circuit |
| ID | Identification |
| IDS | Index Pseudonym |
| IDT | Dynamic Temporary Identification |
| LMAP | A Real Lightweight Mutual Authentication Protocol for Low-cost RFID Tags |
| LSB | Least Significant Bit |
| $M^2AP$ | A Minimalist Mutual Authentication Protocol for Low-cost RFID Tags |
| MSB | Most Significant Bit |
| RF | Radio Frequency |
| RFID | Radio Frequency Identification |
| P | Partial ID |
| SASI | A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity |
| SBAP | Spacing Based Authentication Protocols for Low-Cost RFID |
| SLMAP | A Secure Ultralightweight RFID Mutual Authentication Protocol |
| XOR | Exclusive OR |

# 1. INTRODUCTION

In this chapter we will give a brief information about the RFID system.

## 1.1. RFID System Overview

Nowadays, many traditional identification tools are being replaced with RFID tags in various applications including manufacturing, supply chain management and inventory control. This imminent trend is mostly because of the recent advances in the IC technology meeting the costs and size sensitivities of such applications. Comparison of RFID technology and other traditional identification tools are given in Table 1.1 [1]. RFID technology also has the following advantages:

- RFID tags do not require direct line of sight therefore they can be hidden within the item.
- Active RFID tags have greater operation ranges than other traditional identification tools.
- Reading data from RFID tags are less time consuming than barcodes.
- It is possible to modify the data within an RFID tag, whereas it is not possible for barcodes.
- Batch mode is available for RFID tags where large number of items can be read at once.
- Memory size of RFID tags can reach up to order of kB, whereas barcodes can read just a few digits.
- It is much harder for RFID tags to be replicated than barcodes.

These advantages give RFID technology an opportunity to find many application areas in everyday life such as:

- *Human Identification*: E-passport is one of the RFID applications that aims human identification.
- *Transportation payments*: Many countries use RFID gadgets to collect payments for

toll roads and public transportation.

- *Touch-free credit cards*: Some credit card companies have developed special credit cards that enable users to do the payment without handing over the credit card.
- *Product tracking*: Products with planted RFID tags enables tracking of the assets.
- *Logistics*: Yard management, shipping and distribution centers are some areas where RFID technology is used.
- *Animal Identification*: Especially for large ranches and rough terrain, RFID has become important in animal identification management.
- *Inventory Tracking*: RFID system is used for managing inventory in public and private foundations such as libraries, museums and theaters.
- *Access Management*: Many foundations use RFID gadgets to manage access of information and locations that needs to be protected from unauthorized usage.

A typical RFID system consists of tags, one or more readers, and a back-end server described with the following roles:

- *Tag*: A tag $\mathcal{T}$ is equipped with an antenna for receiving and transmitting radio-frequency (RF) signal. Tags are mostly classified as active and passive according to their source of energy; active tags have on-board batteries whereas passive tags use only the electromagnetic waves emitted from the reader. Since passive tags have no battery this allows them to be smaller than active tags and their lifespan is theoretically indefinite. But they generally lack strong cryptographic operations and their operation distance is limited whereas active tags have the advantage of high computational capabilities and long operation distance. Therefore, in active tags more complicated protocols can be implemented to reduce the possibility of any unathorized interactions.

Figure 1.1. Example of passive and active tag.

- *Reader*: A reader $\mathcal{R}$ has the privilege to read or write data to the tag and interrogates tags within its range. The readers are just simple tranceivers that have the ability to transmit and receive signal through their antenna. They may have one or more antennas to do their task. However, reader's role in RFID systems is quite limited. In general, it bridges the tag and server and just forwards messages to each other. The communication between the reader and the database is generally considered to be secure.

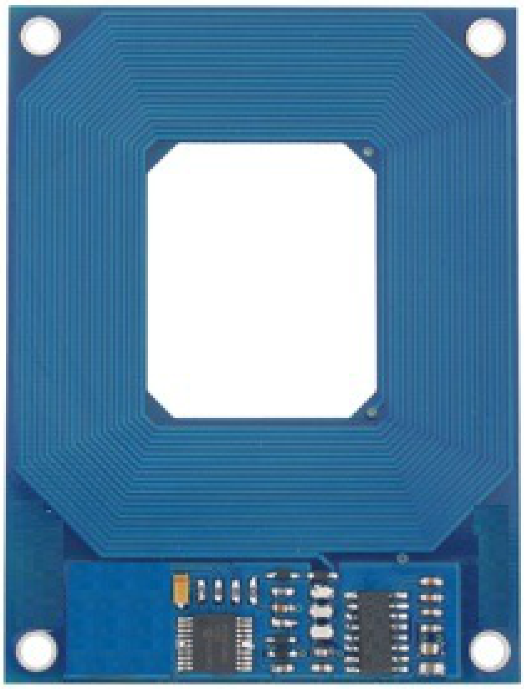

Figure 1.2. A simple RFID reader.

- *Back-end server/database*: A database is denoted by $\mathcal{DB}$ and contains all the relevant information about the RFID system. The server is responsible for the security and operability of the whole system and controls all the processes including authentication and identification.

Despite their advantages, RFID system has weaknesses that can be summarized as:

- When RFID tag is implemented under liquid or metal product, it is not possible for the reader to read the tag since metalic surfaces and liquids reflect and attenuate the radio waves. Therefore, for these kind of products, careful locations must be selected for the tags.

- Since RFID tags use radio waves, it is possible for an adversary to execute denial of service attack by generating noise at the same frequency that the RFID tags use.
- Many countries use different range of frequencies for same RFID applications. These differences make RFID technology hard to use for international markets.

These weaknesses are the simplest ones when compared to the threats for user privacy and counterfeiting. For instance, in a system without any security measure, a tag holder can easily be tracked by any adversary with simple low-cost devices. Although it is possible to raise a similar concern that the same tag holder could also be traced by tracking his/her mobile phone through a carrier, such a track is impossible once the phone is completely turned off. However, this countermeasure is not applicable for someone carrying a RFID gadget since in general a tag could not be turned off, and worse, it automatically responds to queries via radio signals. Therefore, in RFID systems, the attack scenarios and accompanying countermeasures are quite different than the typical wired or wireless systems.

Without any doubt, cryptographic challenge-response protocols address the authentication problems in distributed systems. However, utilizations of these protocols requires intensive computational power that most low-cost constraint device/tag does not have. Therefore, the interest for building security mechanisms for such limited devices has recently arouse in the security community. However, most of early studies are analyzed later that they do not fulfill the basic security and privacy measures [2]. In fact, active RFID security research exhibits how deep the authentication/identification problem in these systems and address the need for more efficient protocols.

## 1.2. Security Requirements for RFID Protocols

For a secure RFID protocol, various requirements are needed to provide privacy and security issues as reported in previous literature [3–6]. In this context, a secure protocol should satisfy the following security parameters to deal with several privacy issues in RFID systems:

Table 1.1. Comparing the primary auto-ID technologies.

| | **Barcodes** | **Contact memory** | **Passive RFID** | **Active RFID** |
|---|---|---|---|---|
| **Modification of data** | Unmodifiable | Modifiable | Modifiable | Modifiable |
| **Security of data** | Minimal security | Highly secure | Ranges from minimal to highly secure | Highly secure |
| **Amount of data** | Linear (8-30 characters) 2D (7200 numbers) | Up to 8 MB | Up to 64 kB | Up to 8 MB |
| **Costs** | Low | High | Medium | Very high |
| **Standards** | Stable and agreed | Proprietary; no standard | Evolving to an agreed standard | Proprietary; evolving open standards |
| **Lifespan** | Short unless laser-etched into metal | Long | Indefinite | 3-5 year battery life |
| **Reading distance** | Line of sight (3-5 feet) | Contact required | Distance up to 50 feet | Distance up to 100 meters and beyond |
| **Potential interference** | Optical barriers such as dirt | Contact blockage | Environments or fields that effect transmission of radio waves | Limited barriers since the broadcast signal from the tag is strong |

- *Location privacy / untraceability*: The transmitted messages should not leak any information that allows an adversary to trace the tag. Most obvious traceable tags are the ones transmitting fixed responses, hence, at each query, the tag should reply with a different message.
- *Forward security*: When the secret key within the tag is compromised by an adversary, the adversary must be unable to get any information about the tag owner's previous actions.
- *Key secrecy / anti-cloning*: The adversary should not be able to get the key or any other tag specific data without tampering. Without the key, no one could clone a tag.
- *Synchronization*: A very important measure for RFID protocols where any key update mechanisms are used. If the security protocol has a weakness that allows an adversary to set different keys for the database and tag, the tag would not be identified, hence, it would be useless for the present and forthcoming sessions.
- *Authentication*: The protocol has to ensure that the communicating parties are legitimate. Moreover, any information from the previous messages or any modification of the old messages should not give an advantage to an adversary.

Another important aspect in security analysis is the adversarial model. For analysis of the protocols, we assume that the attacker has full control over the communication line, i.e. he/she is able to intercept, modify and insert messages as well as eavesdrop the communication channel. Moreover, the attacker could instantiate new communication channels and directly interact with honest parties. Nevertheless, he/she is not able to compromise a target tag, i.e. cannot obtain the secrets of the tag by tampering. Hence, the attacker can be classified as a weak attacker as described in [7] and on duty he could perform the following types of attacks. We refer the reader to [7–11] for further discussions on the attacks.

- *Replay attack*: Adversary records the communication messages between the tag and the reader/server and afterwards he tries to realize a successful authentication between these entities by replaying the messages [8].
- *Tag impersonation attack*: In this type of attack, an adversary attempts to impersonate a tag to the reader/server. Thus, the adversary convinces the reader/server to believe the fake tags are legitimate [9].
- *Man-in-the-middle attacks*: In this type of attack the adversary intrudes into a com-

munication channel to intercept the exchanged data and inject false information. It involves eavesdropping on a connection, intruding into a connection, intercepting messages, and selectively modifying data [10].

- *Denial of service attack*: The adversary disturbs the communication channel between the tag and the reader by intercepting or blocking the communication messages to prevent the tag authentication. In some cases, the attacker tries to set different keys for the database and tag. So, the tag is desynchronized with the server and they would no longer be able to authenticate each other [9, 11].

In this study, we analyze the privacy and security aspects of ultralightweight RFID protocols in the previous literature and a recently proposed ultralightweight security method [12] named as the spacing based authentication protocol (SBAP). Although SBAP claims to provide both security and privacy in its design objectives, we outline very efficient attacks that SBAP fails to fulfill its claims. To be specific, we manage to attain a total breakdown by compromising the secret key information.

After briefly reviewing previous ultralightweight RFID protocols in the following chapter, a formal presentation of SBAP and its weaknesses are given in Chapter 3. Chapter 4 summarizes the weaknesses and main reasons for these vulnerabilities within these protocols. We close our study by some final notes in the last chapter.

# 2. REVIEW OF ULTRALIGHTWEIGHT RFID PROTOCOLS AND THEIR SECURITY ANALYSES

In this chapter, we make a survey over the previous ultralightweight RFID protocols.

## 2.1. LMAP Protocol

LMAP [13] is one of the first ultralightweight protocols. LMAP uses index pseudonyms (IDSs) with 96-bit length that shows the index of a table where all information about a tag is stored. Each tag is associated with a key, which is divided into four parts of 96 bits ($K = K1 \parallel K2 \parallel K3 \parallel K4$). The protocol can be defined in three steps:

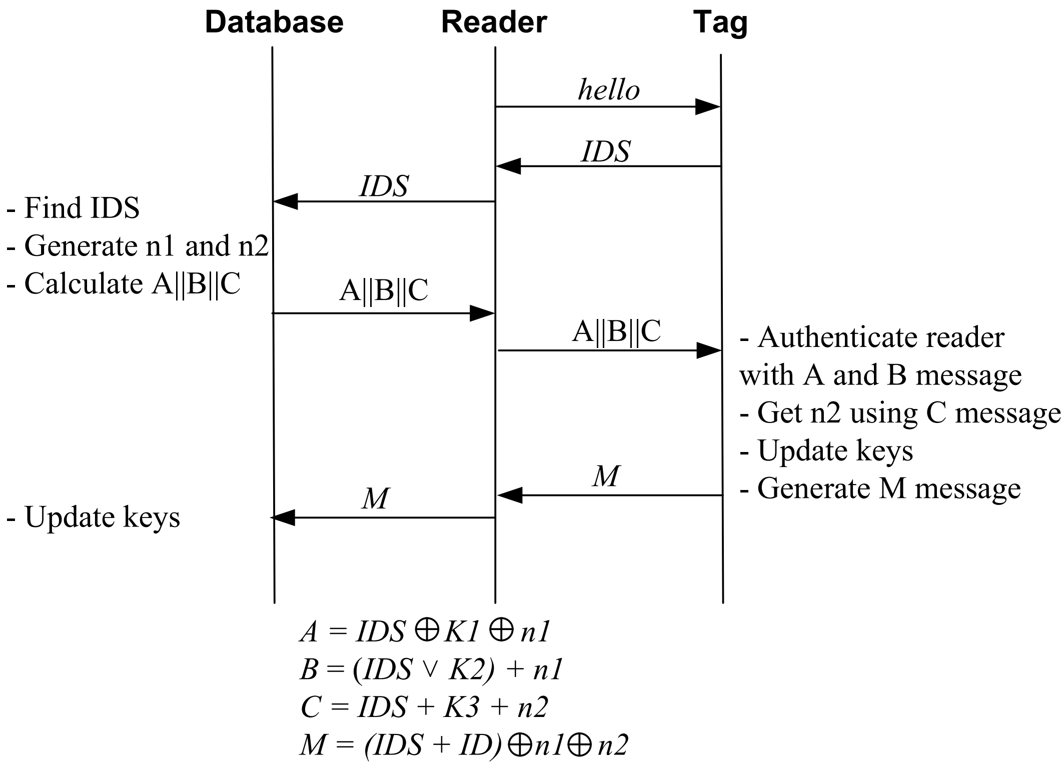


Figure 2.1. LMAP protocol flow.

- *Tag identification*: The reader sends a hello message to the tag and tag will answer with its current index pseudonym. By using the $IDS$, the legitimate reader will be able to access the tag secret key $K$.
- *Mutual authentication*: Upon receiving the $IDS$, the reader generates two random

numbers $n1$ and $n2$. By using these random numbers and $K1$, $K2$ and $K3$ subkeys, the reader generates $A = IDS \oplus K1 \oplus n1$, $B = (IDS \vee K2) + n1$ and $C = IDS + K3 + n2$ messages. When the tag receives these messages, it authenticates the reader with $A$ and $B$ message and gets the random number $n1$. In order to get the random number $n2$, $C$ message is used. After succesfully completing these steps, the tag generates the message $D = (IDS + ID) \oplus n1 \oplus n2$ and sends it to the reader.

- *Index-pseudonym and key update*: After succesfully completing the mutual authentication step, index-pseudonym and key of the tag must be updated using the functions below:

$$IDS_{new} = (IDS + (n2 \oplus K4)) \oplus ID \tag{2.1}$$
$$K1_{new} = K1 \oplus n2 \oplus (K3 + ID) \tag{2.2}$$
$$K2_{new} = K2 \oplus n2 \oplus (K4 + ID) \tag{2.3}$$
$$K3_{new} = (K3 \oplus n1) + (K1 \oplus ID) \tag{2.4}$$
$$K4_{new} = (K4 \oplus n1) + (K2 \oplus ID) \tag{2.5}$$

As stated in [14], LMAP suffers from desynchronization and full disclosure attacks. Desynchronization attack is applied by two methods:

- *Modifying C message*: The adversary intercepts the message $C$ and toggles any bit of it to get a new $C'$ message as $C' = C \oplus [I]_j$ $(0 \leq j \leq 95)$. When the tag receives the modified message, it can still authenticate the reader but it will get a wrong random number $n2'$. If this modification is applied, number of bit differences between $n2$ and newly created $n2'$ can be calculated by using hamming weight:

$$If \quad [C]_j = 1; [C']_j = 0; \ \rightarrow If \quad [n2]_j = 0, HW(n2 \oplus n2') \geq 2 \tag{2.6}$$
$$If \quad [C]_j = 1; [C']_j = 0; \ \rightarrow If \quad [n2]_j = 1, n2' = n2 \oplus [I]_j \tag{2.7}$$
$$If \quad [C]_j = 0; [C']_j = 1; \ \rightarrow If \quad [n2]_j = 0, n2' = n2 \oplus [I]_j \tag{2.8}$$
$$If \quad [C]_j = 0; [C']_j = 1; \ \rightarrow If \quad [n2]_j = 1, HW(n2 \oplus n2') \geq 2 \tag{2.9}$$

For the cases $n2' = n2 \oplus [I]_j$, the reader will accept the new $D'$ message created by the tag since $D' = D \oplus [I]_j = (IDS + ID) \oplus n1 \oplus n2' \oplus [I]_j = (IDS + ID) \oplus n1 \oplus n2$.

For the other cases, since the number of bit differences are more than one, this attack is not applicable. Suppose that $n2$ is randomly generated, there is probability of 50% success rate of desynchronization since the tag will update itself with $(n1, n2')$ and the reader will update itself with $(n1, n2)$.

- *Modifying A and B messages*: The adversary intercepts the $A$ and $B$ messages and modifies any bit of them to get new $A'$ and $B'$ messages as $A' = A \oplus [I]_j$ and $B' = B \oplus [I]_j$. Since $A = IDS \oplus K1 \oplus n1$, $n1$ is set as $n1' = n1 \oplus [I]_j$. In this case for $B$, the adversary obtains:

$$If \quad [B]_j = 1; [B']_j = 0; \rightarrow If \quad [n1]_j = 0, HW(n1 \oplus n1') \geq 2 \qquad (2.10)$$

$$If \quad [B]_j = 1; [B']_j = 0; \rightarrow If \quad [n1]_j = 1, n1' = n1 \oplus [I]_j \qquad (2.11)$$

$$If \quad [B]_j = 0; [B']_j = 1; \rightarrow If \quad [n1]_j = 0, n1' = n1 \oplus [I]_j \qquad (2.12)$$

$$If \quad [B]_j = 0; [B']_j = 1; \rightarrow If \quad [n1]_j = 1, HW(n1 \oplus n1') \geq 2 \qquad (2.13)$$

For the cases $n1' = n1 \oplus [I]_j$, the reader will accept the new $D'$ message created by the tag since $D' = D \oplus [I]_j = (IDS + ID) \oplus n1 \oplus n2' \oplus [I]_j = (IDS + ID) \oplus n1 \oplus n2$. For the other cases, since the number of bit differences are more than one, this attack is not applicable. Suppose that $n1$ is randomly generated, there is probability of 50% success rate of desynchronization since the tag will update itself with $(n1', n2)$ and the reader will update itself with $(n1, n2)$.

For full disclosure attack of LMAP protocol, the adversary disguises as a legitimate reader and gets the current $IDS$ of a tag. By using this valid $IDS$, the adversary queries the reader to get a valid $A \parallel B \parallel C$ message. Later, the adversary modifies j-th bit of $A$ and $B$ messages repeatedly and sends the modified messages to the tag. According to proper $D$ or an error message is received from the tag, the adversary concludes that j-th bit of $n1$ is equal to j-th bit of $B$ or not equal to j-th bit of $B$. In 96 trials, adversary can decide the whole bit values of $n1$. Then, from $A, B, IDS$ and $n1$, the adversary can now calculate $K1$ and $K2$.

After this point the unknown parameters are $n2$, $K3$, $K4$ and $ID$. Obviously, one can use the method above to obtain the value of $n2$, but to interact with the reader m times.

But many readers limits the number of interactions by a constant times. To avoid from reaching maximum reader interaction limit, adversary tries another method to find out the remaining parameters. Firstly, adversary disguises as a legitimate tag and sends the $IDS$ to the reader again. The reader will response with the message $A^{new} \parallel B^{new} \parallel C^{new}$. Since the adversary knows the $K1$, $K2$ and $IDS$, he can set the $n1^{new} = 0$. After modifying the reader's message, the adversary sends the modified $A^{new'} \parallel B^{new'} \parallel C^{new}$ message to the tag. Then the tag responses with $D^{new}$. From these interactions the adversary generates the following equations:

$$C = (IDS + K3) + n2 \tag{2.14}$$

$$D = (IDS + ID) \oplus n1 \oplus n2 \tag{2.15}$$

$$C^{new} = (IDS + K3) + n2^{new} \tag{2.16}$$

$$D^{new} = (IDS + ID) \oplus n2^{new} \tag{2.17}$$

The following equation is generated by combining the equations:

$$C^{new} - C = (IDS + ID) \oplus D^{new} - (IDS + ID) \oplus n1 \oplus D \tag{2.18}$$

which is equal to

$$x \oplus a = x \oplus b + c \ mod\, 2^{96} \tag{2.19}$$

where a = $D^{new}$, b = $n1 \oplus D$, c = $C^{new} - C$ mod $2^{96}$ and x = $(IDS + ID)$ mod $2^{96}$. To solve the $x$ in the Equation 2.19, one must note that $x$'s most significant bits do not effect the computation involving its less significant bits. So, adversary can divide the 96 bits into several parts and try to find all possible solutions for each part. Note that one given triple $(a, b, c)$ may not be enough to determine the value of $x$. In this scenario, adversary can interact with the reader several times to attain a few more instances of equations. By intersecting these equations, the value range of the $x$ can be significantly narrowed down. After $x$ is solved other unknown parameters $ID$, $K3$ and $K4$ can be derived since the $IDS$ is already known.

## 2.2. M²AP Protocol

M²AP is proposed in [15] and it is the modified version of LMAP protocol. M²AP uses the same parameters as LMAP: an index pseudonym (IDS) and a key which is divided into four parts of 96 bits ($K = K1 \parallel K2 \parallel K3 \parallel K4$). The protocol can be defined in three steps:

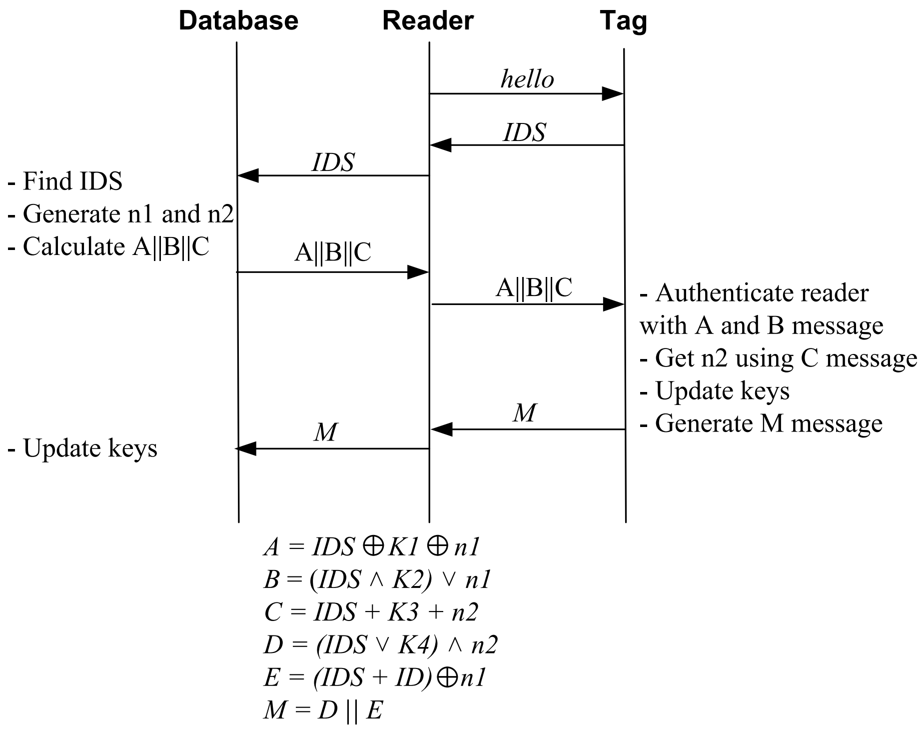


Figure 2.2. M²AP protocol flow.

- *Tag identification*: The reader sends a hello message to the tag and tag will answer with its current index pseudonym. By using the $IDS$, the legitimate reader will be able to access the tag secret key $K$.
- *Mutual authentication*: Upon receiving the $IDS$, the reader generates two random numbers $n1$ and $n2$. By using these random numbers and $K1$, $K2$ and $K3$ subkeys, the reader generates $A = IDS \oplus K1 \oplus n1$, $B = (IDS \wedge K2) \vee n1$ and $C = IDS + K3 + n2$ messages. When the tag receives these messages, it authenticates the reader with $A$ and $B$ message and gets the random number $n1$. In order to get the random number $n2$, $C$ message is used. After succesfully completing these steps, the tag generates two messages:

$$D = (IDS \vee K4) \wedge n2 \tag{2.20}$$

$$E = (IDS + ID) \oplus n1 \tag{2.21}$$

and sends them to the reader.

- *Index-pseudonym and key update*: After succesfully completing the mutual authentication step, index-pseudonym and key of the tag must be updated using the functions below:

$$IDS_{new} = (IDS + (n2 \oplus n1)) \oplus ID \tag{2.22}$$

$$K1_{new} = K1 \oplus n2 \oplus (K3 + ID) \tag{2.23}$$

$$K2_{new} = K2 \oplus n2 \oplus (K4 + ID) \tag{2.24}$$

$$K3_{new} = (K3 \oplus n1) + (K1 \oplus ID) \tag{2.25}$$

$$K4_{new} = (K4 \oplus n1) + (K2 \oplus ID) \tag{2.26}$$

Full disclosure attack can also be applied to $M^2$AP protocol as defined in [16]. Adversary has the advantage to reveal the tag $ID$, secrets keys $K1$, $K3$ and random numbers $n1$, $n2$ by simply eavesdropping two consecutive sessions. To recover $K2$ and $K4$, some other techniques are required. For the clarity of expression, we will denote the k-th bit of M in round n by $M_k^n$ and when $k = 96$ it represents the LSB of M. $M_k$ will represent the k-th bit of M in all sessions.

After eavesdropping two session, the adversary computes the LSB of n2 message:

$$[n2^n]_{96} = [E^n]_{96} \oplus [IDS^{n+1}]_{96} \tag{2.27}$$

Since $[n2^n]_{96}$ is known, next thing adversary computes is:

$$[K3^n]_{96} = [C^n]_{96} \oplus [IDS^n]_{96} \oplus [n2^n]_{96} \tag{2.28}$$

Using the updating formula of $K1^{n+1}$, the adversary can calculate the LSB of ID, since:

$$[ID]_{96} = ([K1^n]_{96} \oplus [ID]_{96}) \oplus ([K1^{n+1}]_{96} \oplus [ID]_{96}) \oplus [K3^n]_{96} \oplus [n2^n]_{96} \tag{2.29}$$

$$= ([A^n]_{96} \oplus [E^n]_{96}) \oplus ([A^{n+1}]_{96} \oplus [E^{n+1}]_{96}) \oplus [K3^n]_{96} \oplus [n2^n]_{96} \tag{2.30}$$

Adversary also computes the LSB of $K1^n$ and $n1^n$:

$$[n1^n]_{96} = [E^n]_{96} \oplus [IDS^n]_{96} \oplus [ID^n]_{96} \tag{2.31}$$

$$[K1^n]_{96} = [A^n]_{96} \oplus [IDS^n]_{96} \oplus [n1^n]_{96} \tag{2.32}$$

Adversary also computes $[K1^{n+1}]_{96}$ and $[K3^{n+1}]_{96}$ by using their update formulas. After that, he/she computes $[n1^{n+1}]_{96}$ from $[E^{n+1}]_{96}$. Finally, adversary calculates $[n2^{n+1}]_{96}$, $[IDS^{n+2}]_{96}$, $[K1^{n+2}]_{96}$, $[K3^{n+2}]_{96}$ respectively.

After computing the LSBs of the secret values, adversary tries to set up equations handling the addition modulo $2^{96}$ for the other 95 bits knowing the addends. After this step adversary obtains $K1$, $K3$, $n1$, $n2$ and $ID$.

The only unknown parameters in this step are $K2$ and $K4$. These parameters are captured in three different ways.

In the first method, adversary obtains $K2$ and $K4$ in 192 protocol runs. The updating formula for the round (n+2) for $K2$ and $K4$:

$$[K2^{n+2}]_{96} = [n1^n]_{96} \oplus [n2^n]_{96} \oplus [n2^{n+1}]_{96} \oplus [ID]_{96} \tag{2.33}$$

$$[K4^{n+2}]_{96} = [n1^n]_{96} \oplus [n2^n]_{96} \oplus [n1^{n+1}]_{96} \oplus [ID]_{96} \tag{2.34}$$

As the number of eavesdropped session increases, adversary computes the nonces at each session as it is described above and calculates the LSB of $K2$ and $K4$ at each session. Then, adversary computes the other bits by handling the addition modulo $2^{96}$ in the updating formulas of $K2$ and $K4$. Each bit is computed in two sessions, and after 192 protocol runs the whole $K2^{n+192}$ and $K4^{n+192}$ will be captured. In the other methods, adversary uses the fact that one fourth of $K2$ and $K4$ if the following implications are hold:

$$([IDS^i]_k = 1) \wedge ([n1^i]_k = 0) \quad \rightarrow \quad [B^i]_k = [K2^i]_k \tag{2.35}$$

$$([IDS^i]_k = 0) \wedge ([n2^i]_k = 1) \quad \rightarrow \quad [D^i]_k = [K4^i]_k \tag{2.36}$$

In the second method, adversary uses the above information to calculate the whole $K2$ and $K4$ and it is shown that adversary can calculate them in nearly 120 protocol runs.

In the third method, adversary splits 96-bit long $K2$ and $K4$ into 8-bit blocks and for each block tests all the $2^{96}$ possible $K2^n - K4^n$. After fixing a pair, adversary can generate the updated blocks by using the updating formula of $K2$ and $K4$. It is shown that after six eavesdropped sessions, there is only one possible $K2^{n+6} - K4^{n+6}$ with probability of 0.9. The expected value for having a unique $K2^n - K4^n$ is 4.5 and since there are 12 blocks, adversary can calculate $K2$ and $K4$ after 54 protocol runs.

### 2.3. EMAP Protocol

EMAP [17] is proposed by Peris-Lopez and it is the modified version of the legacy $M^2AP$ protocol. EMAP uses index pseudonyms (IDSs) with 96-bit length that shows the index of a table where all information about a tag is stored. Each tag is associated with a key, which is divided into four parts of 96 bits ($K = K1 \parallel K2 \parallel K3 \parallel K4$). The protocol can be defined in three steps:

- *Tag identification*: The reader sends a hello message to the tag and tag will answer with its current index pseudonym. By using the $IDS$, the legitimate reader will be able to access the tag secret key $K$.
- *Mutual authentication*: Upon receiving the $IDS$, the reader generates two random numbers $n1$ and $n2$. By using these random numbers and $K1$, $K2$ and $K3$ subkeys, the reader generates $A = IDS \oplus K1 \oplus n1$, $B = (IDS \vee K2) \oplus n1$ and $C = IDS \oplus K3 \oplus n2$ messages. When the tag receives these messages, it authenticates the reader with $A$ and $B$ message and gets the random number $n1$. In order to get the random number $n2$, $C$ message is used. After succesfully completing these steps, the tag generates the messages $D = (IDS \wedge K4) \oplus n2$, $E = (IDS \wedge n1 \vee n2) \oplus ID \oplus K1 \oplus K2 \oplus K3 \oplus K4$ and sends them to the reader.
- *Index-pseudonym and key update*: After succesfully completing the mutual authentication step, index-pseudonym and key of the tag must be updated using the functions

below:

$$IDS_{new} = IDS \oplus n2 \oplus K1 \quad (2.37)$$

$$K1_{new} = K1 \oplus n2 \oplus ([ID]_{[1:48]} \parallel F_p(K4) \parallel F_p(K3)) \quad (2.38)$$

$$K2_{new} = K2 \oplus n2 \oplus (F_p(K1) \parallel F_p(K4) \parallel [ID]_{[49:96]}) \quad (2.39)$$

$$K3_{new} = K3 \oplus n1 \oplus ([ID]_{[1:48]} \parallel F_p(K4) \parallel F_p(K2)) \quad (2.40)$$

$$K4_{new} = K4 \oplus n1 \oplus (F_p(K3) \parallel F_p(K1) \parallel [ID]_{[49:96]}) \quad (2.41)$$

where $F_p(X)$ is a parity function: the 96-bit number $X$ is divided into 24 4-bit blocks. A parity is generated for each block, with a total of 24 parity bits. $[ID]_{[j:k]}$ represents the bit sequence from j-th to the k-th positions of $ID$.

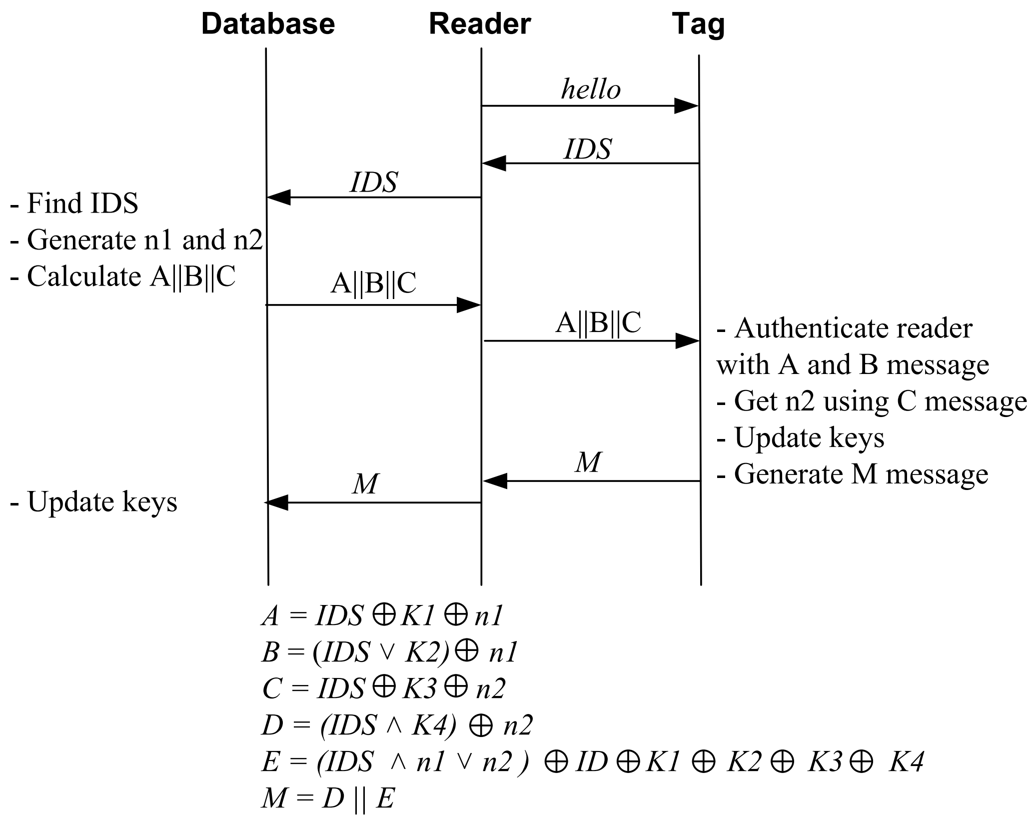


Figure 2.3. EMAP protocol flow.

EMAP suffers from desynchronization and full disclosure attacks [18]. Desynchronization attack is applied by two methods:

- *Modifying C message*: The adversary intercepts the message $C$ and toggles any bit of it to get a new $C'$ message as $C' = C \oplus [I]_j$ $(0 \leq j \leq 95)$. Then he/she sends the new message $A \parallel B \parallel C'$ to the tag. When the tag receives these messages, it can still authenticate the reader as $A$ and $B$ are not modified. However, the tag will get a wrong random number $n2'$ since $C$ message is modified. Tag will reply with $D = (IDS \wedge K4) \oplus n2'$ message calculated with this wrong random number. Adversary captures this message and modifies it as $D' = D \oplus [I]_j = (IDS \wedge K4) \oplus n2$ which is accepted by the reader. Also, $E$ message will be modified as $E' = E \oplus [I]_j = (IDS \wedge n1 \vee n2') \oplus [I]_j \oplus ID \oplus K1 \oplus K2 \oplus K3 \oplus K4$. The reader will accept the message if $result_1 = (IDS \wedge n1 \vee n2)$ equals to $result_2 = (IDS \wedge n1 \vee n2') \oplus [I]_j$. The truth table of $result_1$ and $result_2$ is given in Table 2.1.

Table 2.1. Truth table of $result_1$ and $result_2$.

| $[IDS]_j$ | $[n1]_j$ | $[n2]_j$ | $[n2']_j$ | $result_1$ | $result_2$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |

Table 2.1 shows that for $[IDS]_j = 0$ the success rate is 100% whereas for $[IDS]_j = 1$ the success rate drops to 50%. Given that $IDS$ is known, an adversary can choose to change any bit of $C$ to achieve 100% success rate.

- *Modifying A and B messages*: The adversary intercepts the $A$ and $B$ messages and modifies any bit of them to get new $A'$ and $B'$ messages as $A' = A \oplus [I]_j$ and $B' = B \oplus [I]_j$. Since $A = IDS \oplus K1 \oplus n1$ and $B = (IDS \vee K2) \oplus n1$, the tag will get $n1' = n1 \oplus [I]_j$ from both $A'$ and $B'$. When the tag receives these messages, it authenticates the reader and generates the reply messages $D$ and $E$. As it is shown on Table 2.1, for $[IDS]_j = 0$ the adversary only forwards them to the reader and the reader will accept them with 100%. For the case $[IDS]_j = 1$, the success rate drops down to 50%.

Full disclosure attack for EMAP protocol consists of four stages. In stage one, the adversary derives some bit values of random number $n2$ and derives the other half of $n2$ in stage two. In stage three, based on $n2$ he/she derives as much as possible the tag's secret information in a single protocol run and in the last stage all secret information including the $ID$ of the tag is derived.

- *Stage one*: The adversary impersonates a legitimate reader and gets the current $IDS$ of a tag. Using this valid $IDS$, the adversary impersonates the tag to get a valid $A \parallel B \parallel C$ from a legitimate reader. The attacker then intercepts the reply messages $D \parallel E$. Since $D = (IDS \wedge K4) \oplus n2$ and $IDS$ is known, some bit values of $n2$ can derived from bitwise expression. If we let $\phi$ be the set of bit positions in which the corresponding bit values in $IDS$ are zero and $\tau$ be the set of bit positions in which the corresponding bit values in $IDS$ are one, we can derive the bitwise expression of $D$ as:

$$D_j = [n2]_j \quad (\forall j \in \phi) \tag{2.42}$$

$$D_k = [K4]_k \oplus [n2]_k \quad (\forall k \in \tau) \tag{2.43}$$

  From Equation 2.42 half of $n2$ is derived. The other half of $n2$ can not be derived since $K4$ is unknown. Following the same approach, we can derive the half of $n1$, since $[B]_k = [n1]_k$, $(\forall k \in \tau)$.

- *Stage two*: Firstly, the adversary launches the desynchronization attack by modifying $A$ and $B$ messages and sending $A' \parallel B' \parallel C$ where $A'$ is set as $[A']_\tau = [A]_\tau \oplus [I]_\tau$, or equivalently toggling all the bit values at positions of $\tau$ on $A$. $B'$ is set as $[B']_\tau = [B]_\tau \oplus [I]_\tau$ and $n1'$ is set as $[n1']_\tau = [n1]_\tau \oplus [A]_\tau$. After receiving these values, the tag obtains $n1'$ and $n2$ and replies with $D$ and $E'$. Since $E'$ is calculated from $n1'$ and $n2$, it is different from the message in stage one. If we set $result_1 = (IDS \wedge n1 \vee n2)$ and $result_2 = (IDS \wedge n1' \vee n2)$, Table 2.2 is created.
  From Table 2.2, Equation 2.44 can be derived for $(\forall k \in \tau)$ and $n2$ is fully disclosed:

$$[n2]_k = \begin{cases} 0, & \text{if } [E]_k \neq [E']_k \\ 1, & \text{if } [E]_k = [E']_k \end{cases} \tag{2.44}$$

  Since $n2$ is disclosed, other unknown parameters $K3$ and $[K4]_\tau$ are derived by solving

Table 2.2. Truth table of k-th bit value of $result_1$ and $result_2$.

| $[IDS]_k$ | $[n1]_k$ | $[n1']_k$ | $[n2]_k$ | $result_1$ | $result_2$ |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |

the expressions of $C$ and Equation 2.43.

Secondly, the adversary launches the desynchronization attack by modifying $C$ message and obtains $[n1]_\tau$. By solving the expression of $A$, the adversary also derives $[K1]_\tau$.

- *Stage three*: In this stage, the known parameters are $[n1]_\tau$, $n2$, $[K1]_\tau$, $K3$, $[K4]_\tau$ whereas the unknown parameters are $[n1]_\phi$, $[K1]_\phi$, $K2$, $[K4]_\phi$. The adversary expresses the $E$ message in the following forms:

$$E_j = [n2]_j \oplus [ID]_j \oplus [K1]_j \oplus [K2]_j \oplus [K3]_j \oplus [K4]_j \quad (2.45)$$

$$E_k = ([n1]_k \vee [n2]_k) \oplus [ID]_k \oplus [K1]_k \oplus [K2]_k \oplus [K3]_k \oplus [K4]_k \quad (2.46)$$

where $(\forall k \in \tau)$ and $(\forall j \in \phi)$. By using $A$ and $B$, the equations are transformed into:

$$[ID]_\phi \oplus [K4]_\phi = [n2]_\phi \oplus [E]_\phi \oplus [A]_\phi \oplus [B]_\phi \oplus [K3]_\phi \quad (2.47)$$

$$[ID]_\tau \oplus [K2]_\tau = ([n1]_\tau \vee [n2]_\tau) \oplus [E]_\tau \oplus [K1]_\tau \oplus [K3]_\tau \oplus [K4]_\tau \quad (2.48)$$

Since $[K4]_\phi$ and $[K2]_\tau$ are still unknown, the adversary can not derive the $ID$. It is obvious that the adversary needs more protocols runs to capture the $ID$.

- *Stage four*: In this stage the adversary eavesdrops a valid $IDS^{n+1}$ knowing that the formula for updating $IDS$ is $IDS^{n+1} = IDS \oplus n2 \oplus K1$. Given that $n2$ is known, $K1$ is derived. By using $A$ and $B$ messages, $n1$ and $[K2]_\phi$ are obtained. At this point, the only unknown parameters are $[K4]_\phi$, $[K2]_\tau$ and ID. By using the updating algorithm of $K1$ and the known parameters, the adversary can derive the most significant 48 bits of $ID$ denoted as $L$.

$$[ID]_L = [K1^{n+1}]_L \oplus [K1]_L \oplus [n2]_L \quad (2.49)$$

By using the updating algorithm of $K2^{n+1}$, $K4^{n+1}$ and the known parameters, the following parameters are derived:

$$[ID]_{R\cap\phi} = [K2^{n+1}]_{R\cap\phi'} \oplus [K2]_{R\cap\phi} \oplus [n2]_{R\cap\phi} \quad (2.50)$$

$$[ID]_{R\cap\tau} = [K4^{n+1}]_{R\cap\tau'} \oplus [K4]_{R\cap\tau} \oplus [n1]_{R\cap\tau} \quad (2.51)$$

It is calculated that in one protocol run, approximately 24 bits can be derived using the Equation 2.50 and Equation 2.51. It is assumed that nearly in six protocol runs, the whole $ID$ can be captured.

## 2.4. SASI Protocol

In 2007, Hung-Yu Chien proposed an ultralightweight protocol with rotations named shortly as SASI [19]. In SASI protocol tag shares four varibles with the database as one static identification $ID$, a pseudonym $IDS$, and two keys named $K1$ and $K2$. The length of these variables are 96 bits. The protocol can be defined in three steps:

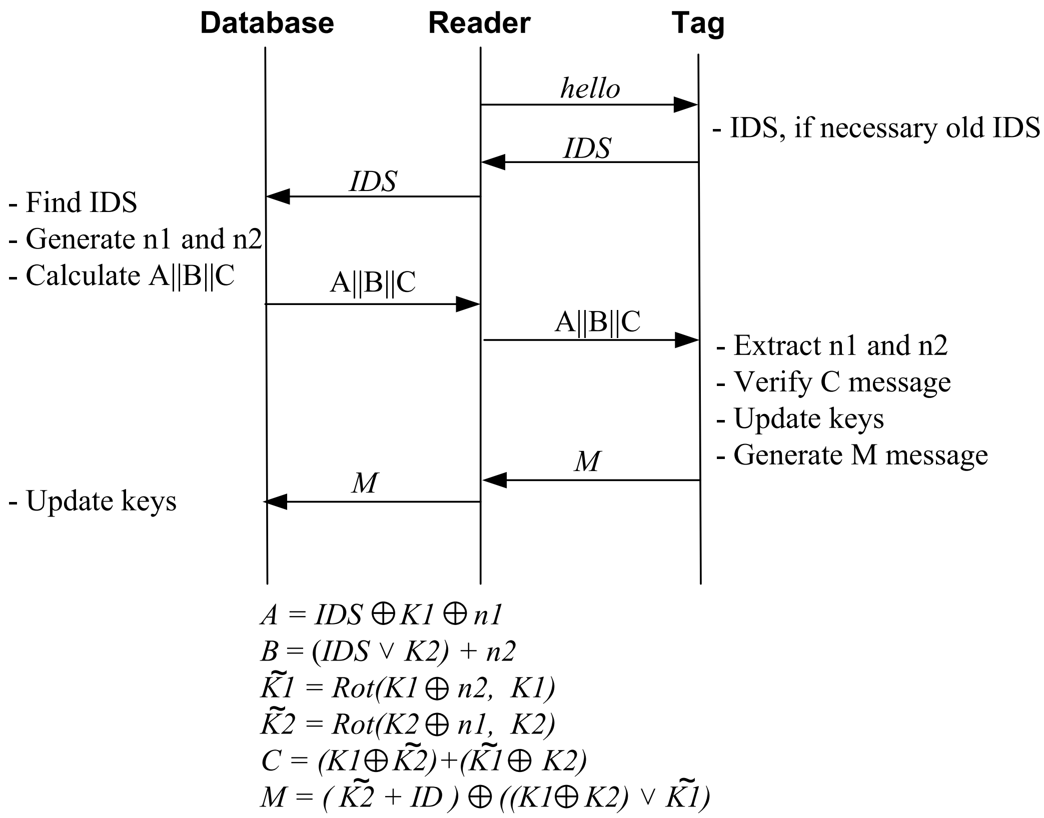


Figure 2.4. SASI protocol flow.

- *Tag identification*: The reader sends a hello message to the tag and tag will answer with its current index pseudonym. By using the $IDS$, the legitimate reader will be able to access the tag secret key $K$. If the tag is re-queried without completing a session successfully, the tag sends its old $IDS$ to the reader.
- *Mutual authentication*: Upon receiving the $IDS$, the reader generates two random numbers $n1$ and $n2$. By using these random numbers and $K1$, $K2$ subkeys, the reader generates $A = IDS \oplus K1 \oplus n1$ and $B = (IDS \vee K2) + n2$ messages. After that the reader generates two new subkeys as $\tilde{K1} = Rot(K1 \oplus n2, K1)$ and $\tilde{K2} = Rot(K2 \oplus n1, K2)$ where $Rot(x, y)$ means left rotation of $x$ with $y$ bits. Lastly, reader generates $C = (K1 \oplus \tilde{K2}) + (\tilde{K1} \oplus K2)$. From $A \parallel B \parallel C$, the tag extracts $n1$ from $A$, extracts $n2$ from $B$. With these random numbers it generates $\tilde{K1}$ and $\tilde{K2}$ to verify the value of $C$. If the messages are authentic, the tag generates $D = (\tilde{K2} + ID) \oplus ((K1 \oplus K2) \vee \tilde{K1})$ and sends it to the reader.
- *Index-pseudonym and key update*: After sending $D$ message to the reader, the tag updates its keys and pseudonym. The tag also stores its old keys and pseudonym to avoid desynchronization attacks. As soon as the reader verifies $D$ message, tag specific data is updated using the functions below:

$$IDS_{new} = (IDS + ID) \oplus (n2 \oplus \tilde{K1}) \tag{2.52}$$

$$K1_{new} = \tilde{K1} \tag{2.53}$$

$$K2_{new} = \tilde{K2} \tag{2.54}$$

The natural way of attacking this protocol is to consider what happens when modular rotations are not performed, that is, when amount of rotation is zero modulo 96 [20]. In this case, the update functions of $K1$, $K2$ and $IDS$ are defined as:

$$\tilde{K1} = Rot(K1 \oplus n2,\ K1\, mod\, 96) = Rot(K1 \oplus n2, 0) = K1 \oplus n2 \tag{2.55}$$

$$\tilde{K2} = Rot(K2 \oplus n1, K2) = K2 \oplus n1 \tag{2.56}$$

$$IDS_{new} = (IDS + ID) \oplus K1 \tag{2.57}$$

To recover the $ID$ of the tag, the adversary now has the function:

$$ID = IDS_{new} \oplus K1 - IDS \tag{2.58}$$

By taking the advantage of $K1 = K2 = 0\, mod\, 96$ and snooping two consecutive authentication sessions, few least significant bits of the secret $ID$ are recovered since with a probability of 33% it can be written as:

$$ID\ mod\ 96 \approx (IDS_{new} - IDS)\ mod\ 96 \tag{2.59}$$

which allows an adversary to trace the tag. Since the attack uses the case $K1 = K2 = 0\ mod\ 96$, it is important for an adversary to recognize it. Assume that $K1 = K2 = 0\ mod\ 96$ then:

$$\tilde{K1} = K1 \oplus n2 \tag{2.60}$$

$$\tilde{K2} = K2 \oplus n1 \tag{2.61}$$

So

$$C = (K1 \oplus \tilde{K2}) + (K2 \oplus \tilde{K1}) \tag{2.62}$$

$$= K1 \oplus K2 \oplus n1 + K2 \oplus K1 \oplus n2 \tag{2.63}$$

which implies that

$$C\, mod\, 96 \approx n1 + n2\ \ mod\, 96 \tag{2.64}$$

The value of $n1$ and $n2$ can be captured by using the messages $A$, $B$ and $IDS$, that is,

$$A = IDS \oplus K1 \oplus n1 \rightarrow n1 = A \oplus IDS \oplus K1 \tag{2.65}$$

$$B = (IDS \vee K2) + n2 \rightarrow n2 = B - (IDS \vee K2) \tag{2.66}$$

and since $K1 = K2 = 0\, mod\, 96$ these equations lead to

$$\begin{aligned} n1\ mod\ 96 &\approx (A \oplus IDS)\ mod\ 96 && (2.67) \\ n2\ mod\ 96 &\approx (B - IDS)\ mod\ 96 && (2.68) \\ C\ mod\ 96 &\approx (A \oplus IDS) + (B - IDS)\, mod\, 96 && (2.69) \end{aligned}$$

If Equation 2.69 holds, the $K1 = K2 = 0\, mod\, 96$ condition is satisfied. The adversary can eavesdrop authentication sessions to check if Equation 2.69 holds, and it is satisfied he/she can calculate the approximate $ID$ using Equation 2.59. As the adversary observes many consecutive sessions, success probability of the attack will increase.

SASI protocol also lacks the untraceability property [21]. For the traceability attack, the adversary uses the case that addition (+) equals to XOR($\oplus$) for the least significant bit. This leads to:

$$\begin{aligned} C_{LSB} &= K1_{LSB} \oplus \tilde{K}2_{LSB} \oplus \tilde{K}1_{LSB} \oplus K2_{LSB} && (2.70) \\ D_{LSB} &= \tilde{K}2_{LSB} \oplus ID_{LSB} \oplus ((K1_{LSB} \oplus K2_{LSB}) \vee \tilde{K}1_{LSB}) && (2.71) \end{aligned}$$

XOR and OR operation results the same with a probability of $p = 0.75$. Depending on the probability $p = 0.75$, Equation 2.71 can be rewritten as:

$$D_{LSB} = \tilde{K}2_{LSB} \oplus ID_{LSB} \oplus K1_{LSB} \oplus K2_{LSB} \oplus \tilde{K}1_{LSB} \quad (2.72)$$

Combining Equation 2.70 and Equation 2.72 gives:

$$C_{LSB} \oplus D_{LSB} = ID_{LSB} \quad (2.73)$$

Using the relations above, the adversary launches the untraceability attack:

- Adversary eavesdrops on a protocol session between reader and a tag $T_0$, to obtain $C$ and $D$.
- Adversary chooses two fresh tags $T_0$, $T_1$ with identifiers $ID_0$, $ID_1$, where $ID_0$ =

$0 \ mod \ 2$, $ID_1 = 1 \ mod \ 2$.

- Adversary is then given a candidate tag $T_*$ among $T_0$ and $T_1$. By using Equation 2.73, adversary guesses $T_*$ with a probability of 25% which is not negligible.

In [22], it is shown that SASI protocol suffers from desynchronization attack with two different methods. The first method uses the steps below:

- Adversary denotes the variables of a tag in the database as $IDS_1$, $K1_1$, $K2_1$.
- When a legitimate reader queries the tag, the adversary records the messages $A$, $B$, $C$ as $A'$, $B'$, $C'$ and interrupts the $D$ message. This causes the tag to update its variables as:

$$(IDS_{old}, K1_{old}, K2_{old}) = (IDS_1, K1_1, K2_1) \quad (2.74)$$
$$(IDS_{new}, K1_{new}, K2_{new}) = (IDS_2, K1_2, K2_2) \quad (2.75)$$

  whereas the reader will not update its variables.
- Next, the adversary allows the tag and the reader to run a protocol without interrupting them. Thus, the database will update its variables as $IDS_3$, $K1_3$, $K2_3$. In the tag, the values are now:

$$(IDS_{old}, K1_{old}, K2_{old}) = (IDS_1, K1_1, K2_1) \quad (2.76)$$
$$(IDS_{new}, K1_{new}, K2_{new}) = (IDS_3, K1_3, K2_3) \quad (2.77)$$

- In the final step, the adversary queries the tag as a valid reader, and the tag replies as $IDS_{new}$, which is $IDS_3$. The adversary pretends that he can not find the $IDS_{new}$ and queries the tag again. The tag will response with $IDS_{old}$, which is $IDS_1$, and the adversary now replies with the recorded $A'$, $B'$, $C'$ messages. Using these values the tag will update its values as:

$$(IDS_{old}, K1_{old}, K2_{old}) = (IDS_1, K1_1, K2_1) \quad (2.78)$$
$$(IDS_{new}, K1_{new}, K2_{new}) = (IDS_2, K1_2, K2_2) \quad (2.79)$$

which causes a desynchronization betweeen reader and the tag.

Table 2.3. The MSB of each variable.

| $K1 \oplus \tilde{K2}$ | $K2 \oplus \tilde{K1}$ | $carry$ | $C^R$ | $K1 \oplus \tilde{K2}^*$ | $C^T$ | $C_1^*$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

The second method uses man-in-the-middle attack to cause desynchronization.

- When a legitimate reader queries the tag, the adversary records the messages $A$, $B$, $C$ as $A_1$, $B_1$, $C_1$. This causes the tag to update its variables as:

$$(IDS_{old}, K1_{old}, K2_{old}) = (IDS_1, K1_1, K2_1) \tag{2.80}$$

$$(IDS_{new}, K1_{new}, K2_{new}) = (IDS_2, K1_2, K2_2) \tag{2.81}$$

and the database updates its variables as $(IDS_2,\ K1_2,\ K2_2)$.

- Next, the adversary queries the tag until it replies with $IDS_1$. The adversary tries to forge a tuple $(A_1',\ B_1',\ C_1')$ that is acceptable by the tag. The adversary makes $A_1' = A_1^*$, where $A_1^*$ is to flip k-th bit in $A_1$, $B_1' = B_1$ and $C_1' = C_1^*$ where $C_1^*$ is to flip the most significant bit of $C_1$, considering that flipping k-th bit in $A_1$ will flip the k-th bit in $n1$, therefore k-th bit of $K2_1 \oplus n1_1$ will flip and if the flipped bit is rotated to the MSB in $\tilde{K2}$, then C message will be changed in the MSB. The adversary replies the tag with $(A_1',\ B_1',\ C_1')$.
- When tag tries to verify message $C$, it is actually using $C_1^*$, $\tilde{K2}^*$ and $K1$, where $\tilde{K2}^*$ differs from $\tilde{K2}$ in the MSB. Table 2.3 shows that in all cases the value computed by the reader $C^R$ and the value computed by the tag $C^T$ are equal.

The adversary can obtain an authenticated tuple ($A'_1$, $B'_1$, $C'_1$) by at most 96 trials for all possible values of $k$. Once an authenticated tuple ($A'_1$, $B'_1$, $C'_1$) is accepted by the tag, the tag will update its variables as:

$$(IDS_{new}, K1_{new}, K2_{new}) \quad = \quad (IDS_2, K1_2, K2_2^*) \tag{2.82}$$

where $K2_2^*$ has the k-th bit flipped in $K2_2$. When the tag queried by the reader, the tag will reply with $IDS_2$. $IDS_2$ will be found in the database, but the tag will reject the reader since the $K2_{new}$ stored in the tag is no longer synchronized with the database.

## 2.5. Gossamer Protocol

Gossamer protocol [23] is proposed by Peris-Lopez and it is inspired by the SASI scheme. In Gossamer protocol, tag shares four varibles with the database as one static identification $ID$, a pseudonym $IDS$, and two keys named $K1$ and $K2$. The length of these variables are 96 bits. The protocol can be defined in three steps:

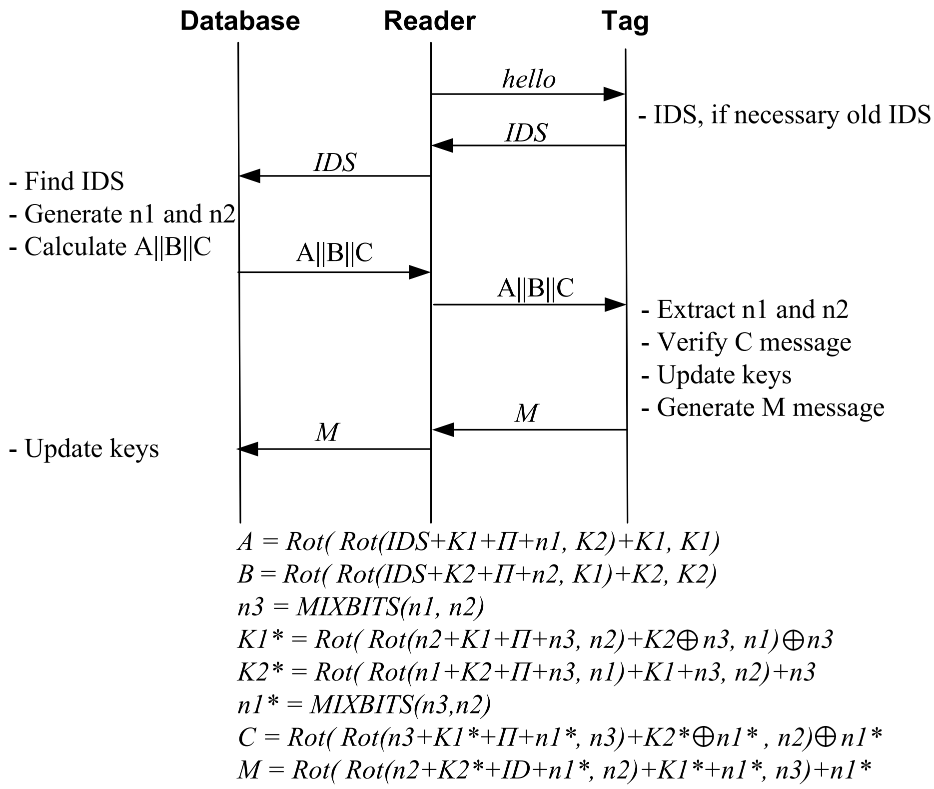


Figure 2.5. Gossamer protocol flow.

- *Tag identification*: The reader sends a hello message to the tag and tag will answer with its current index pseudonym. By using the $IDS$, the legitimate reader will be able to access the tag secret key $K$. If the tag is re-queried without completing a session successfully, the tag sends its old $IDS$ to the reader.
- *Mutual authentication*: Upon receiving the $IDS$, the reader generates two random numbers $n1$ and $n2$. By using these random numbers, $K1$ and $K2$ subkeys the reader generates $A$, $B$ and $C$ messages using the functions below:

$$A = Rot(Rot(IDS + K1 + \pi + n1, K2) + K1, K1) \quad (2.83)$$

$$B = Rot(Rot(IDS + K2 + \pi + n2, K1) + K2, K2) \quad (2.84)$$

$$n3 = MIXBITS(n1, n2) \quad (2.85)$$

$$K1^* = Rot(Rot(n2 + K1 + \pi + n3, n2) + K2 \oplus n3, n1) \oplus n3 \quad (2.86)$$

$$K2^* = Rot(Rot(n1 + K2 + \pi + n3, n1) + K1 + n3, n2) + n3 \quad (2.87)$$

$$n1^* = MIXBITS(n3, n2) \quad (2.88)$$

$$C = Rot(Rot(n3 + K1^* + \pi + n1^*, n3) + K2^* \oplus n1^*, n2) \oplus n1^* \quad (2.89)$$

where $\pi = 0x3243F6A8885A308D313198A2$. In this protocol $Rot(x, y)$ means a circular shift on the value of $x$, $(y \ mod \ 96)$ positions to the left. In order to obtain highly non-linear functions, MIXBITS function is created and it is presented in Figure 2.6:

```
Require X, Y
Z = Y;
for k = 0 to 32 do
  Z = (Z>>1) + Z + Z + Y ;
end for
return Z;
```

Figure 2.6. Algorithm of MIXBITS function.

When the tag receives these messages, it extracts the nonces $n1$ and $n2$. Then the tag computes a local version of submessage $C$. If it is verified, the tag sends $D = Rot(Rot(n2 + K2^* + ID + n1^*, n2) + K1^* + n1^*, n3) + n1^*$ to the reader.

- *Index-pseudonym and key update*: After sending $D$ message to the reader, the tag updates its keys and pseudonym. The tag also stores its old keys and pseudonym to avoid desynchronization attacks. As soon as the reader verifies $D$ message, tag specific data is updated using the functions below:

$$n2^* = MIXBITS(n1^*, n3) \quad (2.90)$$

$$IDS' = Rot(Rot(n1^* + K1^* + IDS + n2^*, n1^*) + K2^* \oplus n2^*, n3) \oplus n2^* \quad (2.91)$$

$$K1' = Rot(Rot(n3 + K2^* + \pi + n2^*, n3) + K1^* + n2^*, n1^*) + n2^* \quad (2.92)$$

$$K2' = Rot(Rot(IDS' + K2^* + \pi + K1', IDS') + K1^* + K1', n2^*) + K1' \quad (2.93)$$

where $IDS'$, $K1'$ and $K2'$ shows the updated values.

Later it is shown that Gossamer protocol suffers from desynchronization attack [24]. This attack is applied in three steps:

- The values stored in the database are named as $IDS_1$, $K1_1$, $K2_1$. When the tag is queried by the reader, the adversary stores the $A \parallel B \parallel C$ message, but the adversary does not allow $D$ message to reach the reader. Tag updates its keys without verifying whether $D$ message has reached the reader or not. The values in the tag are:

$$(IDS_{old}, K1_{old}, K2_{old}) = (IDS_1, K1_1, K2_1) \quad (2.94)$$

$$(IDS_{new}, K1_{new}, K2_{new}) = (IDS_2, K1_2, K2_2) \quad (2.95)$$

- The adversary allows the reader and the tag to run a successfull protocol. Since the $IDS_2$ is not recognized by the reader, reader asks for the older values. When the tag sends $IDS_1$, they complete the protocol. The values in the tag are now:

$$(IDS_{old}, K1_{old}, K2_{old}) = (IDS_1, K1_1, K2_1) \quad (2.96)$$

$$(IDS_{new}, K1_{new}, K2_{new}) = (IDS_3, K1_3, K2_3) \quad (2.97)$$

- The adversary sends hello message to the tag. Tag responds with $IDS_3$ and the adversary pretends that he/she can not identify $IDS_3$ and asks for $IDS_1$. Since the adversary has stored $A \parallel B \parallel C$ message in the first step, he/she sends it to the tag.

The tag updates its keys as:

$$(IDS_{old}, K1_{old}, K2_{old}) = (IDS_1, K1_1, K2_1) \tag{2.98}$$

$$(IDS_{new}, K1_{new}, K2_{new}) = (IDS_2, K1_2, K2_2) \tag{2.99}$$

whereas the keys in the database are $IDS_3$, $K1_3$, $K2_3$.

In the forthcoming sessions, the reader and the tag will not be able to communicate since their keys are different.

In [25], a possible weakness of the protocol is identified. If the nonces $n1$ and $n2$ satisfies the condition that $n1$, $n2$ $mod$ $96 = 0$, then adversary can rewrite the equations as:

$$C = K1^* + \pi + K2^* \tag{2.100}$$

$$D = K1^* + ID + K2^* \tag{2.101}$$

$$IDS_{new} = K1^* + IDS + K2^* \tag{2.102}$$

since $n1$, $n2$, $n3$, $n1^*$ $mod$ 96 are all zeros. Then, the adversary computes:

$$ID = D - C + \pi \tag{2.103}$$

$$ID = D - IDS_{new} + IDS \tag{2.104}$$

$$C - \pi = IDS_{new} - IDS \tag{2.105}$$

It is obvious that if Equation 2.105 is satisfied between two session, adversary can calculate ID from the exchanged messages using Equation 2.103 or Equation 2.104.

## 2.6. Lee Protocol

Lee protocol [26] uses three parameters; a dynamic temporary identification $IDT$, a secret key $K$, and a static identification $ID$. Their length are all 128 bits. The protocol mainly consists of three stages:

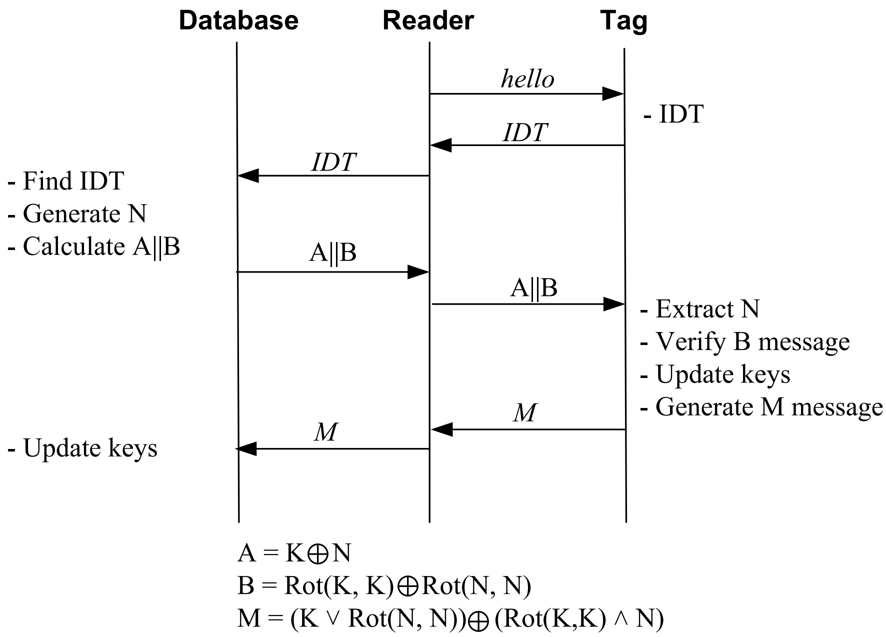


Figure 2.7. Lee protocol flow.

- *Tag identification*: The reader sends a inquire message to the tag and tag will answer with its dynamic temporary identification. By using the $IDT$, the legitimate reader will be able to access the tag secret key $K$.
- *Mutual authentication*: Upon receiving the $IDT$, the reader generates a random number $N$ and computes $A$ and $B$ messages as follows:

$$A = K \oplus N \quad (2.106)$$

$$B = Rot(K, K) \oplus Rot(N, N) \quad (2.107)$$

where $Rot(x, y)$ means left rotate of $x$ with the number of one in $y$. The reader sends $A$ and $B$ messages to the tag. Upon receiving the messages, tag obtains the random number $N$ from $A$, and computes a new $B'$ message to check if it is equal to the one received from the reader. If the message is authentic, the tag generates

$$C = (K \vee Rot(N, N)) \oplus (Rot(K, K) \wedge N) \quad (2.108)$$

and sends it to the reader.

- *Temporary identification and key update*: After sending $C$ message to the reader, the tag updates its key and temporary identification. The tag also stores its old key and temporary identification to avoid desynchronization attacks. The server also updates its key and stores the old key and temporary identification as well. The key and temporary identification is updated using the functions below:

$$IDT_{new} = K \oplus Rot(N, N) \tag{2.109}$$

$$K_{new} = Rot(K, K) \oplus N \tag{2.110}$$

Also, this protocol does not fullfill its security claims including synchronization and key secrecy [27]. For full-disclosure attack, the adversary eavesdrops two consecutive sessions to acquire the following equations:

- The adversary eavesdrops the first authentication session between an authentic tag and a genuine reader to acquire the following equations:

$$A = K \oplus N \tag{2.111}$$

$$B = Rot(K, K) \oplus Rot(N, N) \tag{2.112}$$

- In the following authentication session, the adversary also eavesdrops the following equations:

$$IDT_{n+1} = K \oplus Rot(N, N) \tag{2.113}$$

$$A_{n+1} = K_{n+1} \oplus N_{n+1} \tag{2.114}$$

$$B_{n+1} = Rot(K_{n+1}, K_{n+1}) \oplus Rot(N_{n+1}, N_{n+1}) \tag{2.115}$$

$$C_{n+1} = (K_{n+1} \vee Rot(N_{n+1}, N_{n+1})) \oplus (Rot(K_{n+1}, K_{n+1}) \wedge N_{n+1}) \tag{2.116}$$

The secret key in session n+1 is $K_{n+1} = Rot(K, K) \oplus N$. By using these equations, the adversary can now write:

$$A \oplus B \oplus IDT_{n+1} = K \oplus N \oplus Rot(K, K) \oplus Rot(N, N) \oplus K \oplus Rot(N, N) \tag{2.117}$$

$$= Rot(K, K) \oplus N \oplus (K \oplus K) \oplus (Rot(N, N) \oplus Rot(N, N)) \tag{2.118}$$

$$= Rot(K, K) \oplus N \tag{2.119}$$

$$= K_{n+1} \tag{2.120}$$

By using this equation, the adversary obtains the current key of the tag easily. Therefore, Lee protocol fails to hold key secrecy and since the secret key is revealed, the adversary can now reveal random number $N$ and clone the whole tag data. After capturing the secret key, the adversary can desynchronize the database and the tag as:

- During an authentication session, the adversary intercepts the $A$ and $B$ messages calculated with the random number $N$. Since the secret key and random number $N$ is known, he/she calculates new messages with a random value $N^*$.

$$A = K \oplus N^* \tag{2.121}$$

$$B = Rot(K, K) \oplus Rot(N^*, N^*) \tag{2.122}$$

- The tag updates its values with the random number $N^*$, and calculates $C$ message with $N^*$. The adversary intercepts this message and calculates the $C$ message with the random number $N$.

$$C = (K \vee Rot(N, N)) \oplus (Rot(K, K) \wedge N) \tag{2.123}$$

Since the tag and the server is updated with different random numbers, they are out of synchronization in the forthcoming sessions. In the second desynchronization method the adversary can use older messages and the non-resistance of the bitwise operations to create new valid messages:

- The adversary eavesdrops an authentication session and captures the messages $IDT$, $A$, $B$, $C$. After authentication the secret values are $IDT_{n+1}$, $K_{n+1}$.
- Adversary selects a $C_1$ value with the restriction that its hamming weight is 2.
- Adversary selects a $C_2$ value from the subset of $x \in \{0, 1, ..., 2^{128}\}$ that has hamming weight of 2.
- Adversary computes the values:

$$A_{n+1} = A \oplus C_1 = K \oplus N \oplus C_1 \tag{2.124}$$

$$B_{n+1} = B \oplus C_2 = Rot(K, K) \oplus Rot(N, N) \oplus C_2 \tag{2.125}$$

- If the tag accepts $A_{n+1}$ and $B_{n+1}$ and replies with $C_{n+1}$ to the adversary, the attack is successfull. Otherwise, starting from $C_2$ selection, adversary repeats the steps.
- If the whole step completely fails, the adversary repeats the steps from the beginning.

It is calculated that the average number of trials until desynchronization is 8128 which is feasible to implement.

## 2.7. SLMAP* Protocol

In 2007, Li and Wang [28] introduced an ultralightweight RFID protocol denoted as SLMAP that is mainly based on LMAP protocol. In [29], it is shown that this protocol has some security flaws and later an improved version, SLMAP* , is proposed by Li, Deng and Wang [30]. SLMAP* protocol uses five parameters; an index pseudonym $IDS$, three secret

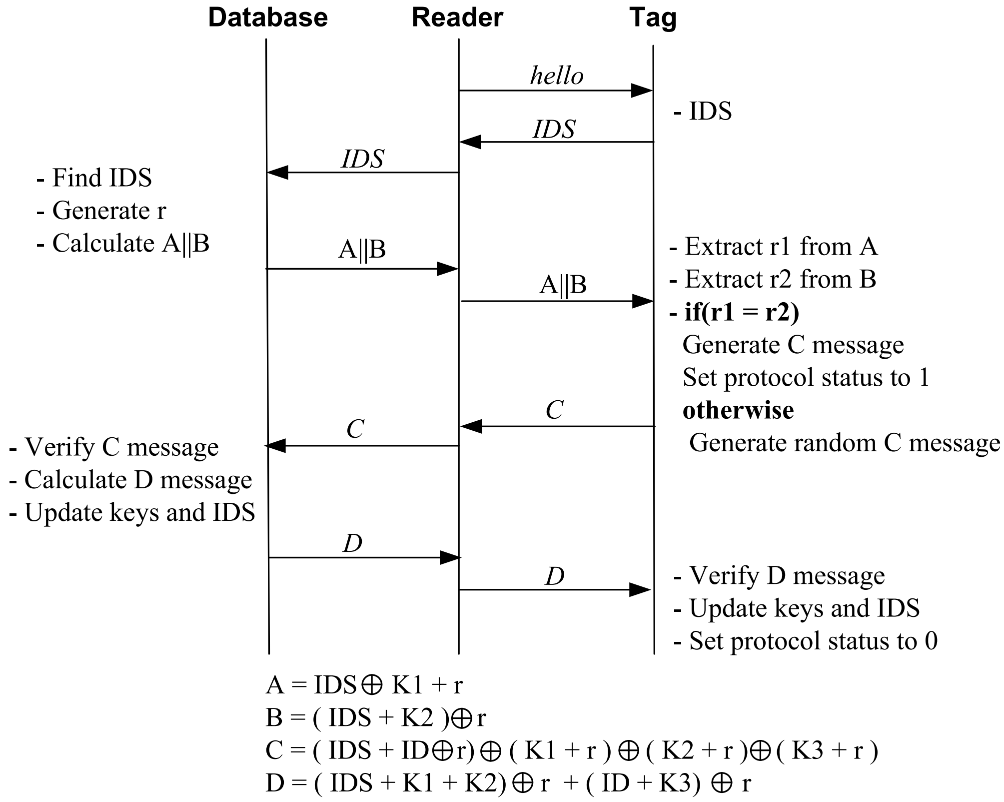


Figure 2.8. SLMAP* protocol flow.

keys $K1$, $K2$, $K3$, and a static identification $ID$. The protocol consists of three stages:

- *Tag identification*: The reader sends a hello message to the tag and tag will answer with its current index pseudonym. By using the $IDS$, the legitimate reader will be able to access the tag secret key $K$.

- *Mutual authentication*: Upon receiving the $IDS$, the reader generates a random number $r$. With this random number, reader generates $A = IDS \oplus K1 + r$ and $B = (IDS + K2) \oplus r$ messages and sends them to the tag. From $A \parallel B$, the tag computes $r1$ and $r2$ values:

$$r1 = A - (IDS \oplus K1) \tag{2.126}$$

$$r2 = B \oplus (IDS + K2) \tag{2.127}$$

  If $r1 = r2$, tag sets the protocol status bit to one. Later, the tag prepares an answer message $C = (IDS + ID \oplus r) \oplus (K1 + r) \oplus (K2 + r) \oplus (K3 + r)$ and sends it to the reader.

- *Index-pseudonym and key update*: When the message is received by the reader, it computes a local $C$ message. If it is equal with the one received from the tag, the reader computes:

$$D = IDS_{new} \tag{2.128}$$

$$= (IDS + K1 + K2) \oplus r + (ID + K3) \oplus r \tag{2.129}$$

  After updating $IDS, K1, K2$ and $K3$, the reader sends $D$ message to the tag. When the tag receives $D$ message, the tag also updates its secret values and sets the protocol status bit to zero.

The protocol status bit is set to zero if protocol is completed succesfully and set to one otherwise. If protocol status is one, the tag will expect $D$ message from the reader. If no valid $D$ message arrives for a maximum number of trials, for example $c$, tag resets its state and returns its status to the beginning of the session without changing the secrets.

In [31], authors have shown a way to desynchronize the tag and the server in two phases. The authors note that key update function is not defined in the protocol description, but they assume that the random number $r$ is used in update function since most similar protocols, including SLMAP, use the random nonces in their update functions.

Table 2.4. Truth table of the MSB values of messages.

| $X_{MSB}$ | $Y_{MSB}$ | $r_{MSB}/\hat{r}_{MSB}$ | $D_{MSB}/\hat{D}_{MSB}$ |
|---|---|---|---|
| 0 | 0 | 0/1 | 0/0 |
| 0 | 0 | 1/0 | 0/0 |
| 0 | 1 | 0/1 | 1/1 |
| 0 | 1 | 1/0 | 1/1 |
| 1 | 0 | 0/1 | 1/1 |
| 1 | 0 | 1/0 | 1/1 |
| 1 | 1 | 0/1 | 0/0 |
| 1 | 1 | 1/0 | 0/0 |

- In phase one, the adversary eavesdrops a session between the tag and the reader but stops $D$ message from reaching the tag. The reader will update its keys whereas the tag will not update its keys since $D$ message is blocked. At the end of the session, adversary obtains the tuple $A \parallel B \parallel D$.
- In phase two adversary queries the tag until it reaches maximum number of trials to force the tag to reset the status bit to zero without changing any secret value. After the tag resets its state, the adversary initiates a session between the tag and tries to change the random number $r$ as $\hat{r} = r \oplus I$ where $I = [100\ldots000]$. Then adversary modifies $A$ and $B$ messages as $\hat{A} = A + I$ and $\hat{B} = B \oplus I$. After these modifications, the tag obtains

$$r1 = \hat{A} - (IDS \oplus K1) \tag{2.130}$$
$$= A + I - (IDS \oplus K1) \tag{2.131}$$
$$= r + I \tag{2.132}$$
$$r2 = \hat{B} \oplus (IDS + K2) \tag{2.133}$$
$$= B \oplus I \oplus (IDS + K2) \tag{2.134}$$
$$= r \oplus I \tag{2.135}$$

Since $r1 = r2 = \hat{r}$, the tag accepts $A$ and $B$ messages and sends the $C$ message.

Adversary ignores this message and sends the old $D$ message to the tag. Tag computes $\hat{D} = (IDS + K1 + K2) \oplus \hat{r} + (IDS + K3) \oplus \hat{r}$. Tag updates its keys since it is equal to the $D$ message as proven below:

The two messages can be written as:

$$D = (X \oplus r) + (Y \oplus r) \quad (2.136)$$

$$\hat{D} = (X \oplus \hat{r}) + (Y \oplus \hat{r}) \quad (2.137)$$

Two messages only differ in their MSB and Table 2.4 shows that they are equal in all cases. After the session, tag updates its keys with $\hat{r}$ whereas the server updates its values with $r$. Therefore, tag and the reader will not be able to authenticate each other in the forthcoming sessions.

## 2.8. LMAP++ Protocol

LMAP++ is proposed by Li in [32], and it is a modified version of SLMAP protocol. LMAP++ uses four variables: a static identifier $ID$, a dynamic pseudonym $PID$, and two keys $K1$ and $K2$. All parameters are 96-bit. $PID$ shows the index of the tag specific data in the database. The protocol steps are as follows:

- *Tag identification*: The reader sends a hello message to the tag and tag will answer with its current index pseudonym. By using the $PID$, the legitimate reader will be able to access the tag specific data.
- *Mutual authentication*: Upon receiving the $PID$, the reader generates a random number $r$. With this random number, reader generates $A = PID \oplus K1 + r$ and $B = (PID + K2) \oplus r$ messages and sends them to the tag. From $A \parallel B$, the tag computes $r1$ and $r2$ values:

$$r1 = A - (PID \oplus K1) \quad (2.138)$$

$$r2 = B \oplus (PID + K2) \quad (2.139)$$

If $r1 = r2$, tag prepares an answer message $C = (PID + ID \oplus r) \oplus (K1 + K2 + r)$ and

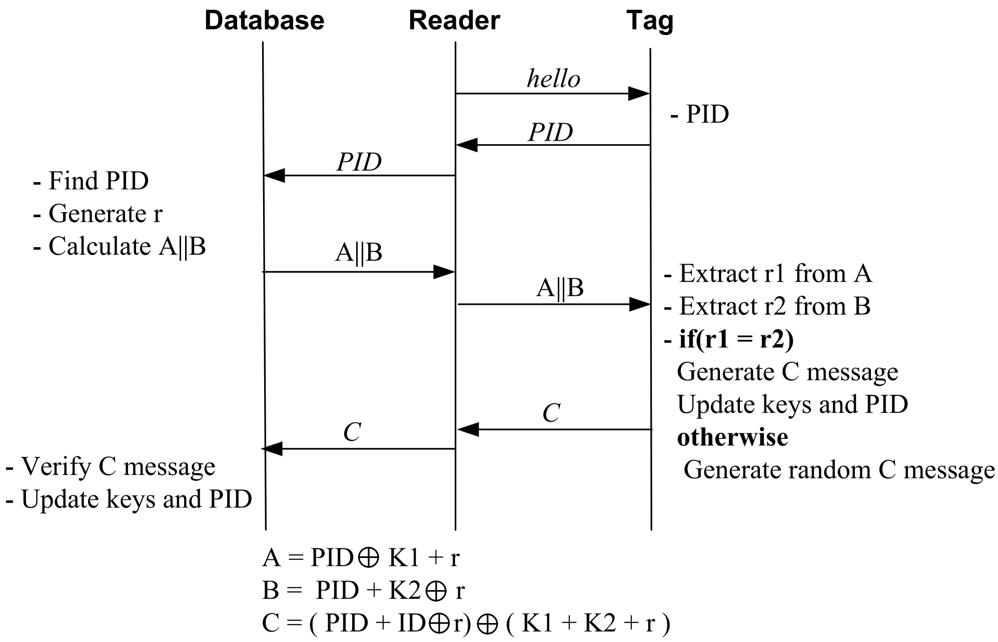


Figure 2.9. LMAP++ protocol flow.

sends it to the reader. If $r1 \neq r2$, a random $C$ message is created.

- *Index-pseudonym and key update*: After sending the $C$ message, tag updates its variables. When $C$ message is received by the reader, it computes another local $C$ message. If it is equal with the one received from the tag, the reader updates $PID$, $K1$, and $K2$:

$$PID_{new} = (PID + K1) \oplus r + (ID + K2) \oplus r \tag{2.140}$$

$$K1_{new} = K1 \oplus r + (PID_{new} + K2 + ID) \tag{2.141}$$

$$K2_{new} = K2 \oplus r + (PID_{new} + K1 + ID) \tag{2.142}$$

In [33], authors presented a way to desynchronize the tag and the server. To mount the attack, authors assumes that the LSBs of $PID$, $K1$, $K2$ and $ID$ are zero. Based on this assumption, the adversary eavesdrops a legitimate session and modifies $A$ and $B$ messages as $A' = A \oplus I$ and $B' = B \oplus I$ where $I = [000 \ldots 0001]$. Based on the fact that, modular addition for LSBs can be replaced by XOR, the tag authenticates the reader but calculates a wrong random number $r' = r \oplus I$.

After authenticating the reader, tag generates $C$ message with $r'$:

$$C = (PID + ID \oplus r') \oplus (K1 + K2 + r') \tag{2.143}$$

Since LSBs of $PID$, $K1$, $K2$ and $ID$ are zero, replacing $r$ by $r'$ will have no effect on the computation of $C$ message, and the reader will authenticate the tag. At the end of the protocol, the tag will update its variables with $r'$, whereas the reader will update its keys with $r$. Therefore, the synchronization between the reader and the tag exists no more.

The success probability depends on the assumption that LSBs of $PID$, $K1$, $K2$ and $ID$ are zero, and it has a success probability of 0.0625.

Traceability attack against LMAP++ also uses the fact that modular addition for LSBs can be replaced by XOR operation. For LSBs the message equations can be rewritten as:

$$A_{LSB} = PID_{LSB} \oplus K1_{LSB} \oplus r_{LSB} \tag{2.144}$$

$$B_{LSB} = PID_{LSB} \oplus K2_{LSB} \oplus r_{LSB} \tag{2.145}$$

$$C_{LSB} = PID_{LSB} \oplus ID_{LSB} \oplus r_{LSB} \oplus K1_{LSB} \oplus K2_{LSB} \oplus r_{LSB} \tag{2.146}$$

Using these equations the adversary can detect the LSB of its $ID$ by calculating:

$$ID_{LSB} = A_{LSB} \oplus B_{LSB} \oplus C_{LSB} \oplus PID_{LSB} \oplus PID_{LSB} \oplus PID_{LSB} \tag{2.147}$$

Keeping this equation in mind, if an adversary takes two tags with $ID^0_{LSB} = 1$ and $ID^1_{LSB} = 1$, he can distinguish these tags with a probability of one.

## 2.9. David-Prasad Protocol

In MobiSec'09, David and Prasad proposed a new ultralightweight mutual authentication protocol [34] for low-cost RFID tags. This protocol uses five parameters: an old pseudonym $P_{ID}$, a potential pseudonym $P_{ID2}$, two secret keys $K1$, $K2$ and a static identifier $ID$ with all 96-bit length. The protocol flow can be summarized as:

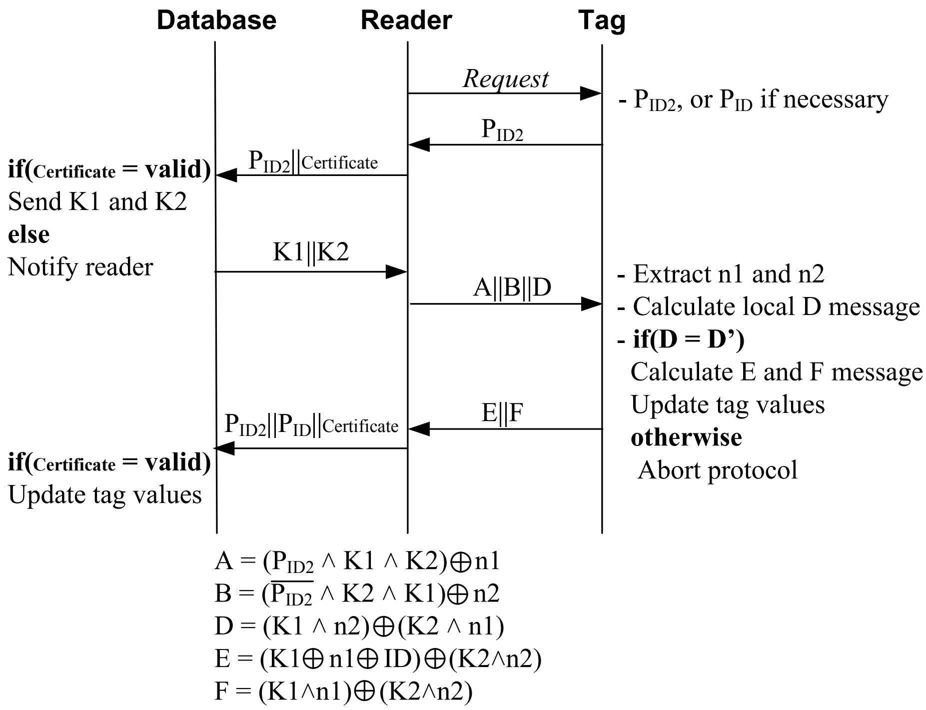


Figure 2.10. David-Prasad protocol flow.

- The reader sends a $C_{request}$ message to the server. If server authenticates the reader, the server sends certificate $C$ that is valid for one day.
- If the reader has a valid certificate, the reader sends a request message to the tag, and tag replies with its pseudonym $P_{ID2}$.
- The reader sends the tuple $P_{ID2} \parallel C$ to the server to get the tag specific data. If the certificate is valid and $P_{ID2}$ matches the one in the database, the server sends secret keys $K1$ and $K2$ to the reader. If $P_{ID2}$ is not found, server informs the reader and the reader sends another request to get the old pseudonym $P_{ID}$ of the tag.
- After getting the tag specific data, reader generates two random numbers $n1$, $n2$ and computes:

$$A = (P_{ID2} \wedge K1 \wedge K2) \oplus n1 \quad (2.148)$$

$$B = (\overline{P_{ID2}} \wedge K2 \wedge K1) \oplus n2 \quad (2.149)$$

$$D = (K1 \wedge n2) \oplus (K2 \wedge n1) \quad (2.150)$$

where $\overline{X}$ represents the bitwise NOT of $X$. Later, the reader sends these messages to the tag.

- From $A$ and $B$ messages, the tag gets the random numbers $n1$ and $n2$. Then, it computes a local version of $D$ message to check if the reader is authentic. If the reader is not authentic, protocol is aborted. After authenticating the reader, tag computes:

$$E = (K1 \oplus n1 \oplus ID) \oplus (K2 \wedge n2) \quad (2.151)$$

$$F = (K1 \wedge n1) \oplus (K2 \wedge n2) \quad (2.152)$$

Finally, the tag updates its values:

$$P_{ID} = P_{ID2} \quad (2.153)$$

$$P_{ID2} = P_{ID2} \oplus n1 \oplus n2 \quad (2.154)$$

- When the reader gets $E$ and $F$ messages, the reader computes a local version of $F$ message and checks if it is equal to the received one. If they are equal, the reader can obtain static identifier of the tag as:

$$ID = E \oplus (K2 \wedge n2) \oplus K1 \oplus n1 \quad (2.155)$$

  Then, the reader updates the tag specific data and sends the updated pair $\{P_{ID}, P_{ID2}\}$ and its certificate C to the server. If the certificate is valid, server updates the tag specific data.

It is later analyzed that this protocol suffers from traceability and full-disclosure attacks [35]. For traceability attack adversary eavesdrops a session between a legitimate reader and a tag. During the session, adversary captures and stores $P_{ID2}$ and $A$, $B$, $C$, $D$, $E$, $F$. By computing XOR between E and F, adversary obtains:

$$E \oplus F = (K1 \oplus n1) \oplus (K1 \wedge n1) \oplus ID \quad (2.156)$$

Table 2.5 shows that for a probability of 0.75, XOR operation is the complement of AND operation for any bit position, so XOR of $K1 \oplus n1$ and $K1 \wedge n1$ is equal to one. This gives

Table 2.5. Truth table of XOR and AND operation.

| $K1$ | $n1$ | $K1 \oplus n1$ | $K1 \wedge n1$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

adversary an opportunity to compute complement of ID for each bit:

$$E \oplus F = \overline{ID} \tag{2.157}$$

Keeping this equation in mind, adversary takes two tags $T_0$ and $T_1$. Using the method above, adversary computes the approximate value of these tags as $ID^{T0}$ and $ID^{T1}$. Then, he takes a candidate tag $T^*$, and computes approximate value of its $ID^*$. Finally, by looking at the bits of candidate tag, the adversary decides the tag as $T_0$ or $T_1$ with a probability of 0.125 for each bit.

Full-disclosure attack can be made by two different methods. In the first method, adversary eavesdrops two consecutive sessions to obtain two pseudonyms $P_{ID2}(i-1)$, $P_{ID2}(i)$ and messages $\{A_{i-1}, B_{i-1}, C_{i-1}, D_{i-1}, E_{i-1}, F_{i-1}\}$, $\{A_i, B_i, C_i, D_i, E_i, F_i\}$. By using these values, adversary computes:

$$Y = P_{ID2}(i-1) \oplus P_{ID2}(i) = n1 \oplus n2 \tag{2.158}$$

$$Z = A_i \oplus B_i = (K1 \wedge K2) \oplus n1 \oplus n2 \tag{2.159}$$

Thus, XOR between $Y$ and $Z$ gives:

$$Y \oplus Z = K1 \wedge K2 \tag{2.160}$$

So, where $Y \oplus Z$ is one, this implies that both $K1$ and $K2$ are equal to one for that position. Eventually, after two consecutive sessions on average $k/4$ bits of both keys will be revealed.

The second method is named as Passive Tango Cryptanalysis by the authors. This attack consists of two phases: selection of good approximations and combination of these good approximations.

- Diffusion properties of triangular functions are known as very poor. Thus, adversary uses $\{A,\ B,\ C,\ D,\ E,\ F\}$ messages to create good approximations (GA) for secret values of the tag. The adversary selects a set of approximations for which hamming distance between an approximation and the secret value deviates from the expected value, 48. After that, adversary lists the approximations in Table 2.6 as the best for each secret value.

Table 2.6. Best approximations for secret values.

| Target | Good Approximation |
|---|---|
| K1 | $GA-K1=\{D,F,(A\oplus D),\overline{(A\oplus F)},\overline{(B\oplus D)},$ $(B\oplus F),(A\oplus B\oplus D),(A\oplus B\oplus F)\}$ |
| K2 | $GA-K2=\{D,F,\overline{(A\oplus D)},(A\oplus F),(B\oplus D),$ $\overline{(B\oplus F)},(A\oplus B\oplus D),(A\oplus B\oplus F)\}$ |
| ID | $GA-ID=\{\overline{(E\oplus F)},(A\oplus B\oplus E),(A\oplus D\oplus E),$ $(A\oplus E\oplus F),(B\oplus D\oplus E),(D\oplus E\oplus F),$ $\overline{(A\oplus B\oplus D\oplus E)},(A\oplus D\oplus E\oplus F),\overline{(B\oplus D\oplus E\oplus F)}$ |

- In this phase, adversary tries to combine different approximations obtained in different sessions to construct a global approximation which is highly correlated with the secret value. This is done by eavesdropping authentication sessions between legitimate parties. For each session, adversary computes and stores the approximations as rows of three $(N_S N_A)$x$L$ matrices, namely $G_{K1}$, $G_{K2}$ and $G_{ID}$, (one for each $K1$, $K2$ and $ID$) where $N_A$ is the number of approximations for each secret value, $N_S$ is the number of eavesdropped sessions and L is the bitwise length of the secret value. The matrice is built as shown in Figure 2.11 where $GA_i^j$ represents the i-th good approximation in the j-th session. Then, adversary builds three 1x$L$ matrices in which each column shows the total number of ones in each column of the corresponding $G$ matrice. These matrices, which we can call as approximation matrices, represent the approximate value of the corresponding secret value. Finally, adversary replaces each column of the ap-

$$\begin{bmatrix} GA^{1}_{1} \\ GA^{1}_{2} \\ \vdots \\ GA^{1}_{NA} \\ GA^{2}_{1} \\ GA^{2}_{2} \\ \vdots \\ GA^{2}_{NA} \\ \vdots \\ GA^{NS}_{1} \\ GA^{NS}_{2} \\ \vdots \\ GA^{NS}_{NA} \end{bmatrix}$$

Figure 2.11. Approximation matrice structure.

proximation matrices with zero if the value in the column is below a given threshold $\gamma$, or one in any other case. The value of the $\gamma$ is calculated by:

$$\gamma = 0.5 N_A N_S \qquad (2.161)$$

These final matrices show the value of the corresponding secret value. As the number of eavesdropped sessions increase, more bits are revealed accurately. It is shown that after five or 10 sessions, more than 90 bits of each secret value is revealed with this attack.

# 3. A RECENT ULTRALIGHTWEIGHT PROTOCOL

A new type of attack for a recent ultralightweight protocol is defined in this chapter.

## 3.1. Spacing Based Authentication Protocol (SBAP)

SBAP is an ultra lightweight protocol that uses an XOR operation and a spacing algorithm to generate a new secret ID for each session [12]. SBAP proposes two authentication methods for the same protocol where the second method is an enhanced version of the first one that reduces server time complexity by saving the key that will be used in the forthcoming sessions.

Assume that each tag keeps its own secret $S$ and a tag $ID$, and the reader stores a list of secret $ID$s. For tag authentication, SBAP uses a partial ID which is denoted by $P$ and generated as follows:

$$P = P_{odd} \oplus P_{even} \tag{3.1}$$

where $P_{odd} := f(S, u, odd)$ and $P_{even} := f(S, u, even)$ for an ultralightweight extracting function $f(S, u, b)$ having three inputs: bit string $S$, random spacing factor $u$ and a Boolean variable $b$.

Generation of the $f$ function is quite simple: let $L$ be the length of the bit stream $S$, $u$ be a positive integer dividing $L$. Thus, we may write $S = s_0s_1 \ldots s_{L-1}$, and partition $S$ to get smaller bit streams $q_i = (s_{iu}s_{iu+1} \ldots s_{iu+u-1})$ for $i = 0, 1, \ldots, L/u - 1$. Once $S$ is partitioned into $q_i$ values, $P_{odd} := f(S, u, 1)$ and $P_{even} := f(S, u, 0)$ are simply calculated by concatenating the odd and even indexed $q_i$ digits respectively.

For instance, in Figure 3.1, we illustrate how the spacing process works for a secret $S$ having $L = 16$ bits long, and the spacing factor $u = 2$.

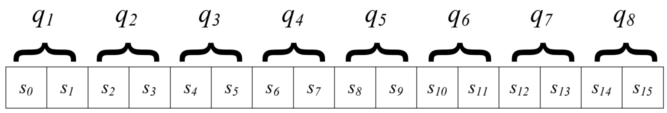


Figure 3.1. An example for the spacing algorithm.

Note that after the partitioning, one computes $P_{odd}$ and $P_{even}$ values as follows:

$$P_{odd} = q_1 \parallel q_3 \parallel q_5 \parallel q_7 \tag{3.2}$$

$$P_{even} = q_0 \parallel q_2 \parallel q_4 \parallel q_6 \tag{3.3}$$

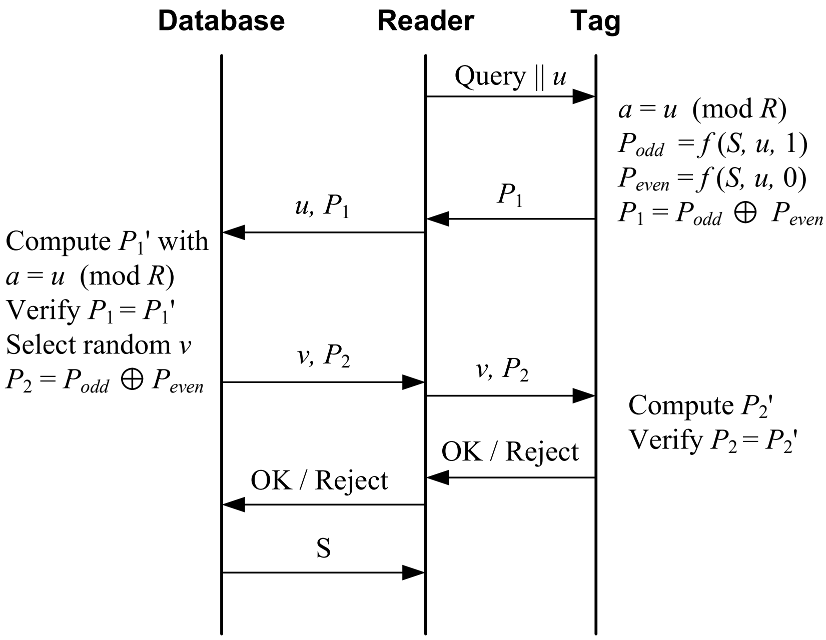


Figure 3.2. Protocol flow for regular method in SBAP.

Our next step is to describe SBAP with its regular and enhanced methods. The enhanced SBAP is proposed to reduce the searching/authentication time in the server if there is a fair amount of tags in the system. The main difference between two protocols is that in the enhanced method, both tag and server generate and save a $P_n$ value for the next section's use. In other words, the tag always responses with $P_n$ and the server initially performs a quick search for $P_n$ in its saved $P_n$ database. If a match exist, the server authenticates the tag otherwise it goes through the regular process. Such an approach surely reduces the server load as enhanced SBAP performs less spacing operations.

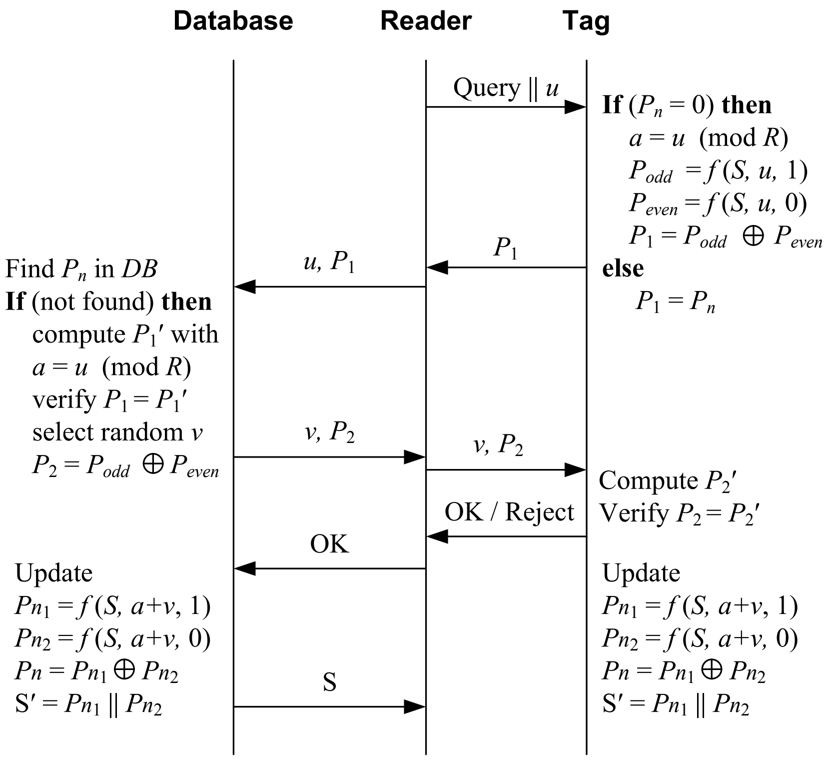


Figure 3.3. Protocol flow for enhanced method in SBAP.

With these remarks in mind, the details of both regular and enhanced SBAP are described in the following paragraphs and further illustrated in Figure 3.2 and 3.3 respectively.

(i) *Step one*: Partial ID generation

- The reader generates a random nonce, and sends a request along with this nonce to the tag.
- The first method reduces the range of this random nonce and generates a new partial ID using the calculations above, and responses with it. But the second method only computes a new partial ID if it does not have any $P_n$ computed in the previous successful session. After a successful session, the second method always responses with the last computed $P_n$ value.

(ii) *Step two*: Authentication

- Upon receiving partial ID $P_1$, the reader forwards it to the database.
- The server looks for a match for $P_1$ in the $P_n$ database, if no such $P_1$ exists, it

calculates $P_1$ with the random nonce $u$ and secret key $S$

- After verification database generates a random nonce $v$, and calculates a message $P_2$ with this nonce using the same spacing algorithm. The reader then forwards $(P_2,\ v)$ pair to the tag.

(iii) *Step three*: Verification & Key Update

- Upon receiving verification message $P_2$, the tag verifies its correctness.
- After checking $P_2$ value, in both methods, the tag simply sends OK/Reject response to the reader, and the reader forwards it to the database.
- However, the enhanced SBAP performs an additional key update session. After verification, it updates the secret key by using the following two values:

$$P_{n_1} = f(S, a + v, odd) \tag{3.4}$$

$$P_{n_2} = f(S, a + v, even) \tag{3.5}$$

The $P_n$ value for the coming session is calculated as $P_n = P_{n_1} \oplus P_{n_2}$ and the secret key $S$ is updated as $S = P_{n_1} \parallel P_{n_2}$.

- On the other hand, once the database receives the last OK message it performs the same update to calculate the fresh $P_n$ and $S$ values.

## 3.2. Attacks

Although SBAP claims to provide both security and privacy in its design objectives, we outline very strong attacks that SBAP failed to fulfill its claims. We manage to perform a total breakdown by compromizing the secret key information. Other weaknesses, we report for SBAP includes the strong attacks such as traceability, replay and desynchronization attacks.

### 3.2.1. Full Disclosure

We claim that SBAP does not hold the key secrecy. In order to prove this this claim, we exhibit an attack for the enhanced method that captures the secret key of the tag without an exhaustive search.

Assume that $P_n$ is not assigned (means it is equivalent to zero). In this phase any request to the tag would be replied back with a fresh $P_1$. Notice that the adversary could get the length information of the secret key $S$ by simply sending the nonce $N_r = 1$ to the tag since the tag would response such a message with $P_1$ having a length equals to the half of the secret key length.

**Remark 3.1.** *The authors in [12] did not explicitly discuss the relation of the secret length $L$ with the spacing factor $u$. However, according to our analysis the protocol would be extremely weak if*

- *spacing factor $u$ does not divide $L$*
- *$u$ divides $L$ but $L/u$ is odd*

*Notice that, in both cases the secret $S$ needs a padding in order to generate $P_1$ value. Since XOR of the padding and the secret key bits are open the adversary would extract the secret key bits from $P_1$.*

Next proposition shows that even if the assumptions in Remark 3.1 are satisfied, the key space for $S$ can be shrunk.

**Proposition 3.2.** *Let $S$ be a secret having a length $L$; the spacing factor $u = 2$ divide $L$ and $L/2$ be even, then the bit search space for $S$ can be shrunk to $L/4$.*

*Proof.* Since $L/u$ is even for $u = 1$ and $u = 2$, the adversary may send the nonces one and two without completing a session successfully. If $P_1$ and $P_1'$ are the respective responses of the tag and $S = s_0 s_1 \ldots s_{L-1}$ represents the secret bit stream, $P_1$ and $P_1'$ can be given as follows:

$$P_1 = (s_0 \oplus s_1)(s_2 \oplus s_3) \ldots (s_{L-2} \oplus s_{L-1}) \tag{3.6}$$

$$P_1' = (s_0 \oplus s_2)(s_1 \oplus s_2) \ldots (s_{L-3} \oplus s_{L-1}) \tag{3.7}$$

From these values, the following linear equations could be written for $i = 0, 1, \ldots, L/4 - 1$:

$$P_1[2i] = P_{odd}[2i] \oplus P_{even}[2i] = s_{4i} \oplus s_{4i+1} \tag{3.8}$$

$$P'_1[2i] = P'_{odd}[2i] \oplus P'_{even}[2i] = s_{4i} \oplus s_{4i+2} \tag{3.9}$$

$$P_1[2i+1] = P_{odd}[2i+1] \oplus P_{even}[2i+1] = s_{4i+1} \oplus s_{4i+3} \tag{3.10}$$

but these leads to

$$s_{4i+1} = P_1[2i] \oplus s_{4i} \tag{3.11}$$

$$s_{4i+2} = P'_1[2i] \oplus s_{4i} \tag{3.12}$$

$$s_{4i+3} = P_1[2i+1] \oplus s_{4i+1} = P_1[2i+1] \oplus P_1[2i] \oplus s_{4i} \tag{3.13}$$

which means that the bit search space for $S$ can be shrunk to $L/4$ since $s_{4i+1}, s_{4i+2}$ and $s_{4i+3}$ can be written as a sum of $s_{4i}$ for $i = 0, 1, \ldots, L/4 - 1$. □

In fact the choice of $L$ shrinks the bit search space of $S$ even further, in particular, if $L$ is a power of two, $S$ can be compromised with a single bit search. We give this result as a corollary for the following theorem.

**Theorem 3.3.** *Let $L = k2^m$ for some positive integer $m$ and odd $k$, then the bit search space for $S$ can be shrunk to $L/2^m$.*

*Proof.* Since $L/u$ is even for $2^i$ for $i = 0, 1, \ldots, m-1$, assume that the adversary sends the nonces $1, 2^1, 2^2, \ldots, 2^{m-1}$ without completing a session successfully, and gets the following respective responses from the tag:

$$P_1^0 = (s_0 \oplus s_1)(s_2 \oplus s_3) \ldots (s_{L-2} \oplus s_{L-1}) \tag{3.14}$$

$$P_1^1 = (s_0 \oplus s_2)(s_1 \oplus s_3) \ldots (s_{L-3} \oplus s_{L-1}) \tag{3.15}$$

$$P_1^2 = (s_0 \oplus s_4)(s_1 \oplus s_5) \ldots (s_{L-5} \oplus s_{L-1}) \tag{3.16}$$

$$\vdots \qquad\qquad \vdots \tag{3.17}$$

$$P_1^m = (s_0 \oplus s_{2^m})(s_2 \oplus s_{2^m+1}) \ldots (s_{L-2^{m-1}-1} \oplus s_{L-1}) \tag{3.18}$$

Notice that for $i = 0, 1, 2, \ldots, 2^m - 1$, any $s_i$ could be written as a combination of the bits of $P_1^0$, $P_1^1, \ldots, P_1^m$ and $s_0$. Since $L = k2^m$, a similar analysis could be done for each adjacent disjoint subset of $S$ having $2^m$ bits. In other words, any $s_i$ in $S$ could be written as a combination of the bits of $P_1^0$, $P_1^1, \ldots, P_1^m$ and $s_0, s_{2^m}, s_{2\cdot 2^m}, \ldots, s_{(k-1)2^m}$. Since there are $k$ such base elements and $P_1^i$ for $i = 0, 1, 2, \ldots, m$ are known, it suffices to search the bit space $k = L/2^m$ instead of searching $L$. □

**Corollary 3.4.** *Let $L = 2^m$ for some positive integers $m$ then the bit search space for $S$ in SBAP shrinks to a single bit search.*

*Proof.* The length of $S$ is $L = 2^m$ implies $k = 1$, hence, the bit search space is single bit search by Theorem 3.3. In other words, $S$ can be written as a linear combination of $s_0$. □

We give the following toy example in order to present the power of the described attack.

**Example 3.5.** *Assume that secret ID $S$ is a 8 bit key where $S = s_0 s_1 \ldots s_7$. Let the adversary send three nonces one, two and four without completing a session succesfully, then the tag responses with $P_1^0, P_1^1$ and $P_1^2$ messages calculated as follows:*

$$P_1^0 = (s_0 \oplus s_1)(s_2 \oplus s_3)(s_4 \oplus s_5)(s_6 \oplus s_7) \tag{3.19}$$
$$P_1^1 = (s_0 \oplus s_2)(s_1 \oplus s_3)(s_4 \oplus s_6)(s_5 \oplus s_7) \tag{3.20}$$
$$P_1^2 = (s_0 \oplus s_4)(s_1 \oplus s_5)(s_2 \oplus s_6)(s_3 \oplus s_7) \tag{3.21}$$

*Lets say that $P_1^0 = 1010, P_1^1 = 0110$ and $P_1^2 = 0011$ are recorded responses by the adversary. Using these values, the following linear equations can be written using the bits of the secret key $S$.*

$$s_0 = s_0 \tag{3.22}$$
$$s_1 = P_1^0[0] \oplus s_0 = s_0 \tag{3.23}$$
$$s_2 = P_1^1[0] \oplus s_0 = s_0 \tag{3.24}$$
$$s_3 = P_1^0[1] \oplus s_2 = P_1^0[1] \oplus P_1^1[0] \oplus s_0 = \bar{s}_0 \tag{3.25}$$

$$s_4 = P_1^2[0] \oplus s_0 = \bar{s}_0 \tag{3.26}$$

$$s_5 = P_1^2[1] \oplus s_1 = P_1^2[1] \oplus P_1^0[0] \oplus s_0 = \bar{s}_0 \tag{3.27}$$

$$s_6 = P_1^1[2] \oplus s_4 = P_1^1[2] \oplus P_1^2[0] \oplus s_0 = \bar{s}_0 \tag{3.28}$$

$$s_7 = P_1^0[3] \oplus s_6 = P_1^0[3] \oplus P_1^1[2] \oplus P_1^2[0] \oplus s_0 = s_0 \tag{3.29}$$

Note that using a spacing parameter which is close to the range $R$ is subject to even simpler key recovery attack. In fact, this is the case where a padding is necessary to either $P_{odd}$ or $P_{even}$ to generate a legitimate $P_1$. Although the authors did not mention the padding scheme explicitly, any padding which does not involve random bits would face SBAP to simpler attacks.

### 3.2.2. Location Privacy and Untraceability

Observe that the first method of SBAP protocol does not use any update mechanisms. If the adversary always queries the tag with the same nonce, the output will always be the same therefore making the tag traceable. This method is also applicable to the second method of SBAP when $P_n$ is equal to zero.

When $P_n$ is not equal to zero, the second method sends the same $P_n$ value until next successful session. If the adversary queries the tag without completing a session, the tag will always respond with the same value which makes the tag an easy target for tracking.

### 3.2.3. Desynchronization

Desynchronization attack is only applicable to the second method of SBAP protocol since the first method does not update the keys. At the end of the second method if the tag updates its key, it sends OK response to the reader. Upon receiving this reply, the database also updates its key. If the adversary intercepts this message from reaching to the reader, the tag will update its key whereas the key in the database will not be updated. This will render the tag useless for further interactions.

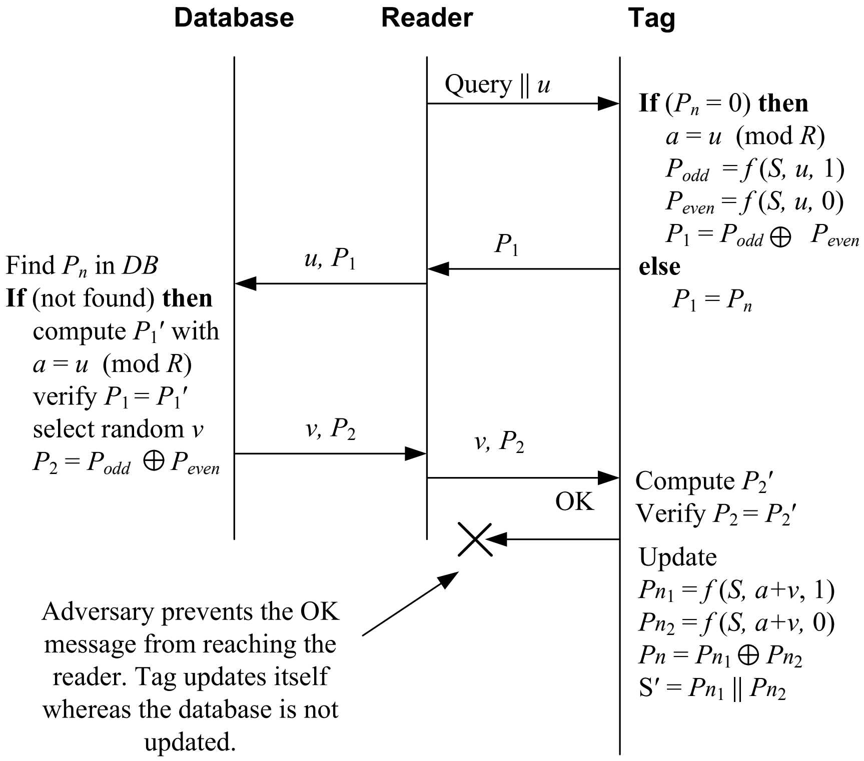


Figure 3.4. Desynchronization attack scenario for SBAP.

### 3.2.4. Replay Attack

In the enhanced method of SBAP protocol when $P_n$ is not equal to zero, the tag does not use the nonce generated by the server. Adversary can obtain access, by simply forwarding one of the tag's $P_n$ message to the reader, and send OK when he receives $P_2$ message from the reader.

This attack will also make the legitimate tag to be desynchronized with the server. It is obvious that the protocol lacks the mutual authentication property.

# 4. RESULTS

In the RFID protocol analysis, following parameters should be considered:

- Number of exchanged messages
- Number of protocol steps
- Number of gates
- Storage size (EEPROM or FLASH memory)
- Functions used in the protocol

According to the functions used in the protocol, RFID protocols can be classified into four categories [19]:

- *Full-fledged*: These are the strongest protocols [36,37] since they can use cryptographic one-way functions and even public key algorithms in their implementation. But, as a result they require larger memory size than other types of protocols.
- *Simple*: Random number generators and one-way hash functions are available for this kind of protocols [38–40].
- *Lightweight*: They include random number generators like simple protocols but instead of one-way hash functions they use simpler functions like CRC [41–44].
- *Ultralightweight*: This kind of protocols are implemented on RFID tags with small computational size and therefore they can only support bitwise operations to encrypt the exchanged messages.

In this context, protocols that are analyzed in this thesis are ultralightweight protocols, and therefore they have limited number of gates less than 1K. Because of limited number gates, extensive cryptographic functions can not be implemented on ultralightweight RFID tags. As seen on Table 4.1, all of the existing cryptographic functions are over 1K.

Parameters of the analyzed protocols are given in Table 4.2 where $L_1 = 96\ bits$, $L_2 = 128\ bits$ and $L_3$ is not defined in the protocol description.

Table 4.1. Number of gates for cryptographic functions.

| *Function* | *Number of gates* |
|---|---|
| Amphion [45] | 25K |
| Fast SHA-256 Helion [46] | 23K |
| Fast SHA-1 Helion [46] | 20K |
| MD5 Helion [46] | 16K |
| Feldhofer [6] | 3595 |
| JungFL [47] | 3089 |
| Universal Hash Yksel [48] | 1.7 K |

Table 4.2. Comparison of protocol parameters.

| *Protocol* | *Number of exc. messages* | *Number of auth. steps* | *Used functions* | *Storage size* |
|---|---|---|---|---|
| LMAP | 4 | 4 | $\oplus, +, \vee$ | $6L_1$ bits |
| $M^2AP$ | 5 | 4 | $\oplus, +, \vee, \wedge$ | $6L_1$ bits |
| EMAP | 5 | 4 | $\oplus, \vee, \wedge$ | $6L_1$ bits |
| SASI | 4 | 4 | $\oplus, \vee, \wedge, +, Rot$ | $7L_1$ bits |
| Gossamer | 4 | 4 | $\oplus, +, Rot, Mixbits$ | $7L_1$ bits |
| Lee | 4 | 4 | $\oplus, Rot, \vee, \wedge$ | $5L_2$ bits |
| SLMAP* | 4 | 5 | $\oplus, +, -$ | $(5L_1+1)$ bits |
| LMAP++ | 3 | 4 | $\oplus, +$ | $(4L_1+1)$ bits |
| David-Prasad | 5 | 4 | $\oplus, NOT, \wedge$ | $5L_1$ bits |
| SBAP | 2 | 4 | $\oplus, Spacing$ | $2L_3$ bits |

SBAP is the simplest protocol among other protocols in terms of used functions and storage size. All of the proposed protocols use simple operations like XOR to hide sensitive information but none of them are fully secure, and worse all protocols except LMAP++ and SLMAP* are proved to be insecure in terms of key-secrecy which makes them vulnerable to other kind of attacks.

Ultralightweight functions used in the protocols suffer mainly from the following weaknesses:

- *XOR*: All of the protocols use XOR operation in their implementation, but it lacks the avalanche effect to hide all secret data and gives an advantage to define bits by simple operations. The most obvious secure way to hide data with XOR operation is to use a random value. For example, if $A$ is the data to hide then it must be XOR-ed with a random value $B$ that is unknown to the adversary.

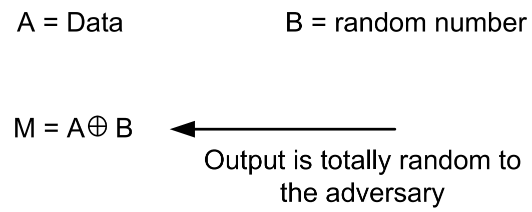


Figure 4.1. Most secure way to use XOR operation.

But if the same data $A$ or random number $B$ is used within other messages, the adversary can try to calculate approximate functions as in the passive tango analysis of David-Prasad protocol or use any weakness in the other messages to reveal information. That is why, same data must be handled carefully in other messages.
Although this is the most secure way to hide data, many protocols use the random values in the update sessions to update their hidden values. For example, if a protocol sends:

$$M = (A \oplus B) \tag{4.1}$$

to tag or reader. Adversary captures this message and modifies any bit of the message as:

$$M' = M \oplus [I]_j \;\; = \;\; A \oplus B \oplus [I]_j \;\; (0 \leq j \leq 95) \tag{4.2}$$

When the legitimate party receives the modified message, it extracts a wrong random number $B \oplus [I]_j$. If this wrong random number is used in update session, legitimate party may be useless for forthcoming sessions. That's why consistency of the random number must be assured with other messages if the random number is used in the update session or in the computation of any other crucial step.

- *OR & AND operation*: The truth table of simple bitwise operations are given in Table 4.3.

Table 4.3. Truth table of AND, XOR and OR operation.

| $A$ | $B$ | $A \wedge B$ | $A \oplus B$ | $A \vee B$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |

The following statements can be derived from Table 4.3:

(i) The output of AND operation is equal to the complement of XOR operation for the 0.75 of the time.

(ii) The output of OR operation is equal to XOR operation for the 0.75 of the time.

(iii) When $A$ and $B$ are equal to zero, the output of all three operations are zero.

(iv) The output of AND operation is equal to the OR operation for the 0.5 of the time.

(v) The output of AND operation is equal to the complement of OR operation for the 0.5 of the time.

These relations give adversary an opportunity to replace operators with a probability and if the probability is not negligible then the security of the protocol can be breached. For example, as depicted in Figure 4.2, if the adversary eavesdrops two messages in a session as:

$$M_1 = A \wedge B \tag{4.3}$$

$$M_2 = (A \oplus B) \oplus C \tag{4.4}$$

where A and B represents any data and C is the data to hide. Adversary writes $M_1$ as the complement of XOR operation, $M_1 = \overline{A \oplus B}$ with a probability of 0.75. Then adversary computes:

$$M_1 \oplus M_2 = \overline{(A \oplus B)} \oplus (A \oplus B) \oplus C \tag{4.5}$$

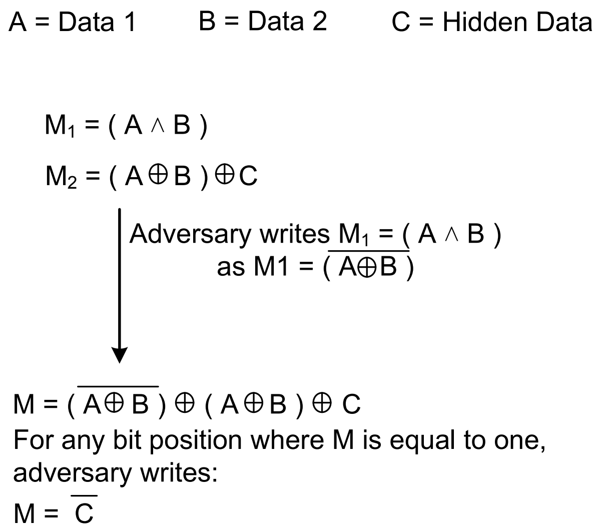


Figure 4.2. Attack by replacing AND operation with XOR operation.

If the j-th bit of $M_1 \oplus M_2$ is equal to one, j-th bit of hidden data C is the complement of $M_1 \oplus M_2$:

$$M_1 \oplus M_2 = \overline{C} \tag{4.6}$$

This method is used in the David-Prasad protocol to breach the security of the system. Therefore, precautions must be taken to avoid for any other bitwise operation replacement attacks.

- *Addition*: Addition operation is actually XOR operation with carry bits. Assume that;

$$A = 10100110 \tag{4.7}$$

$$B = 01110101 \tag{4.8}$$

Then

$$A \oplus B = 11010011 \tag{4.9}$$

$$A + B = 100011011 \tag{4.10}$$

One must notice the overflow bit in the addition operation and this bit is omitted. In the addition operation, carry bit array is captured as:

$$Carry = 11100100 \tag{4.11}$$

Later, adversary pads the carry bit array with a zero since the carry bit array is calculated for the second bit where first bit represents the LSB. Finally, the adversary computes $A \oplus B$ and the carry bit array as:

$$Carry \oplus (A \oplus B) = 11100100 \oplus 11010011 \ = 100011011 \tag{4.12}$$

which is equal to $A + B$. This example is depicted in Figure 4.3.

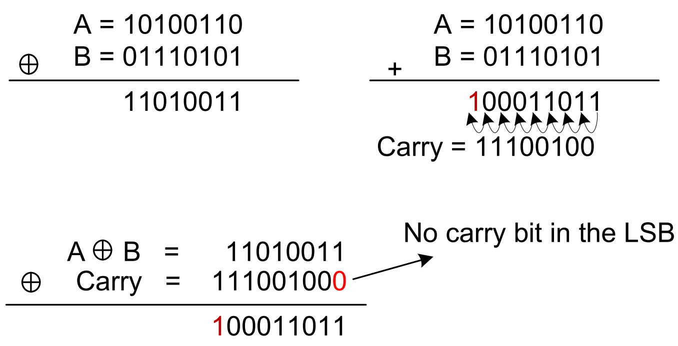


Figure 4.3. Relation between addition and XOR operation.

When XOR operation is used with addition, for the LSB, addition can be replaced with XOR operation. This case can be used as an analysis tool by the adversary to perform various attacks. Traceability attack of SASI and LMAP++ protocol uses this weakness.

Furthermore, after revealing the LSB, adversary can handle the carry bit to reveal other bits as it is done in the $M^2AP$ protocol. Therefore, addition operation must be handled carefully within the protocols.

- *Rotation functions*: Rotation function is firstly used in the SASI protocol. When rotation functions are used in the protocols, it is much harder to do the cryptanalysis. Nevertheless, when the output of rotation functions are predictable (same as input, equal to one or zero, ...), the algorithms are simplified to bitwise operations which is much easier to analyze. For example, if a protocol hides the sensitive information by calculating the message:

$$M = Rot(A, B) \oplus C + D \tag{4.13}$$

where $Rot(x, y)$ is a rotation function. If there is a case that $Rot(A, B) = 0$, the message can be simplified as:

$$M = C + D \tag{4.14}$$

which is much more easier to analyze. The attacks against the key secrecy property of SASI and Gossamer protocols use this weakness. Note that $MIXBITS$ function of the Gossamer protocol is also a rotation function and it suffers from the case that its output is equal to zero if $n1$, $n2$ $mod$ $96 = 0$ condition is satisfied.

- *Spacing function*: Spacing function is brand new in the ultralightweight protocol designs. It is actually some kind of remapping function that changes the position of bits in data. But it suffers from the case that if the spacing parameter is too big, then the data must be padded with a row of ones or zeros. This weakness reveals the bits of hidden data and the key secrecy fails to hold.

In the analyses of previous literature, the most outstanding attack is the Passive Tango defined for the cryptanalysis of David-Prasad protocol. In this attack, the adversary tries to create approximate functions to reveal the secret data and by eavesdropping the consecutive sessions he/she computes the secret information.

Although it is not mentioned in the analyses, all of the protocols are $k - th$ traceable that until next succesfull authentication they always reply with the same PID or IDS.

# 5. CONCLUSION

In this thesis work, the security analyses of previous ultralightweight RFID protocols are summarized and a new ultralightweight RFID protocol is examined in terms of security and privacy.

The severity of attacks for previous protocols show that these protocols are insecure. Two main reasons for these weaknesses are:

- Simple bitwise operations are not sufficient to provide secure RFID authentication against powerfull adversarial models since the resulting messages are strongly biased. Designers must handle these operators carefully within the public messages.
- Rotation functions are necessary for RFID protocol designs. Nevertheless, it is also important to select right type of rotation function and rotation functions, alone, are not sufficient to provide security in ultralightweight protocols.

A new type of attack is defined for SBAP protocol to reveal secret ID of the tag. Full-disclosure attack for SBAP protocol is based on the property of XOR operation. Although the protocol uses a spacing algorithm, it still gives adversary an opportunity to define the bits of secret key in terms of other bits. As the adversary eavesdrops consecutive sessions, the number of unknown bits decreases dramatically. Also if the spacing parameter is chosen close to the range, because of the padding the secret key is revealed much more easier. Location privacy is not achieved since the tag always responses with the same reply to the same nonce. Also replay attack is applicable to the enhanced SBAP, since the tag does not use the nonce generated by the reader.

Analyses show that there is still a need for a secure ultralightweight protocol. Some candidate protocols are proposed recently by various researchers [49, 50]. But it is later analyzed that [49] also suffers from full-disclosure and desynchronization attacks in [51]. To the best of our knowledge [50] has not received any attack yet.

Nevertheless, the security of candidate ultralightweight protocols must be proved with elaborated crypt-analysis to avoid vulnerabilities that are defined in this work and other literature.

# REFERENCES


1. Sweeney, P. J., *RFID for Dummies*, Wiley Publishing Inc., New York, NY, USA, 2005.

2. Zhang, Y. and P. Kitsos, *Security in RFID and Sensor Networks*, Auerbach Publications, New York, NY, USA, 2009.

3. Juels, A., "RFID Security and Privacy: A Research Survey", *IEEE Journal on Selected Areas in Communications*, Vol. 24, No. 2, pp. 381–394, February 2006.

4. Sarma, S., S. Weis and D. Engels, "Radio-Frequency Identification: Security Risks and Challenges", *Cryptobytes, RSA Laboratories*, Vol. 6, No. 1, pp. 2–9, Spring 2003.

5. Avoine, G., *Cryptography in Radio Frequency Identification and Fair Exchange Protocols*, Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, December 2005.

6. Feldhofer, M., S. Dominikus and J. Wolkerstorfer, "Strong Authentication for RFID Systems using the AES Algorithm", M. Joye and J.-J. Quisquater (Editors), *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, Vol. 3156 of *Lecture Notes in Computer Science*, pp. 357–370, IACR, Springer, Boston, Massachusetts, USA, August 2004.

7. Song, B. and C. J. Mitchell, "RFID Authentication Protocol for Low-cost Tags", V. D. Gligor, J.-P. Hubaux and R. Poovendran (Editors), *Proceedings of the 1st ACM Conference on Wireless Network Security – WiSec'08*, pp. 140–147, ACM, ACM Press, Alexandria, Virginia, USA, March–April 2008.

8. Dimitriou, T., "A Lightweight RFID Protocol to protect against Traceability and Cloning attacks", *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2005*, pp. 59–66, IEEE, IEEE Computer Society, Athens, Greece, September 2005.

9. Weis, S., S. Sarma, R. Rivest and D. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems", D. Hutter, G. Müller, W. Stephan and

M. Ullmann (Editors), *International Conference on Security in Pervasive Computing – SPC 2003*, Vol. 2802 of *Lecture Notes in Computer Science*, pp. 454–469, Springer, Boppard, Germany, March 2003.

10. Juels, A., "Minimalist Cryptography for Low-Cost RFID Tags", C. Blundo and S. Cimato (Editors), *International Conference on Security in Communication Networks – SCN 2004*, Vol. 3352 of *Lecture Notes in Computer Science*, pp. 149–164, Springer, Amalfi, Italy, September 2004.

11. Song, B. and C. J. Mitchell, "Scalable RFID Authentication Protocol", *3rd International Conference on Network and System Security – NSS 2009*, pp. 216–224, IEEE, IEEE Computer Society, Gold Coast, Australia, October 2009.

12. Jin, Z., Z. Cheng and K. Yoo, "Spacing Based Authentication Protocol for Low-Cost RFID", *Future Generation Communication and Networking",*, Vol. 1, pp. 160–163, Spring 2008.

13. Peris-Lopez, P., J. C. Hernandez-Castro, J. M. Estevez-Tapiador and A. Ribagorda, "LMAP: A Real Lightweight Mutual Authentication Protocol for Low-cost RFID tags", *Workshop on RFID Security – RFIDSec'06*, Ecrypt, Graz, Austria, July 2006.

14. Li, T. and G. Wang, "Security Analysis of Two Ultra-Lightweight RFID Authentication Protocols", H. Venter, M. Eloff, L. Labuschagne, J. Eloff and R. Von Solms (Editors), *IFIP TC-11 22nd International Information Security Conference – SEC 2007*, Vol. 232 of *IFIP*, pp. 109–120, IFIP, Springer, Sandton, Gauteng, South Africa, May 2007.

15. Peris-Lopez, P., J. C. Hernandez-Castro, J. M. Estevez-Tapiador and A. Ribagorda, "M2AP: A Minimalist Mutual-Authentication Protocol for Low-cost RFID Tags", J. Ma, H. Jin, L. T. Yang and J. J. P. Tsai (Editors), *International Conference on Ubiquitous Intelligence and Computing – UIC'06*, Vol. 4159 of *Lecture Notes in Computer Science*, pp. 912–923, Springer, Wuhan and Three Gorges, China, September 2006.

16. Bárász, M., B. Boros, P. Ligeti, K. Lója and D. Nagy, "Passive Attack Against the M2AP Mutual Authentication Protocol for RFID Tags", *First International EURASIP*

*Workshop on RFID Technology*, Vienna, Austria, September 2007.

17. Peris-Lopez, P., J. C. Hernandez-Castro, J. M. Estevez-Tapiador and A. Ribagorda, "EMAP: An Efficient Mutual Authentication Protocol for Low-Cost RFID Tags", *OTM Federated Conferences and Workshop: IS Workshop – IS'06*, Vol. 4277 of *Lecture Notes in Computer Science*, pp. 352–361, Springer, Montpellier, France, November 2006.

18. Li, T. and R. H. Deng, "Vulnerability Analysis of EMAP - An Efficient RFID Mutual Authentication Protocol", *Second International Conference on Availability, Reliability and Security – AReS 2007*, Vienna, Austria, April 2007.

19. Chien, H.-Y., "SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity", *IEEE Transactions on Dependable and Secure Computing*, Vol. 4, No. 4, pp. 337–340, December 2007.

20. Hernandez-Castro, J. C., J. M. Estevez-Tapiador, P. Peris-Lopez and J.-J. Quisquater, "Cryptanalysis of the SASI Ultralightweight RFID Authentication Protocol with Modular Rotations", *International Workshop on Coding and Cryptography – WCC'09*, pp. 286–296, Ullensvang, Norway, May 2009.

21. Phan, R. C.-W., "Cryptanalysis of a New Ultralightweight RFID Authentication Protocol - SASI", *IEEE Transactions on Dependable and Secure Computing*, Vol. 99, No. 1, p. 5555, 2008.

22. Sun, H., W. Ting and K. Wang, "On the Security of Chien's Ultralightweight RFID Authentication Protocol", *IEEE Transactions Dependable and Secure Computing*, Vol. 8, No. 2, pp. 315–317, April 2011.

23. Peris-Lopez, P., J. C. Hernandez-Castro, J. M. Estevez-Tapiador and A. Ribagorda, "Advances in Ultralightweight Cryptography for Low-cost RFID Tags: Gossamer Protocol", K.-I. Chung, K. Sohn and M. Yung (Editors), *Workshop on Information Security Applications – WISA'08*, Vol. 5379 of *Lecture Notes in Computer Science*, pp. 56–68, Springer, Jeju Island, Korea, September 2008.

24. Zeeshan Bilal, A. M. and F. Kausar, "Security Analysis of Ultra-lightweight Crypto-

graphic Protocol for Low-cost RFID Tags: Gossamer Protocol", *International Conference on Network-Based Information Systems – NBIS'09*, pp. 260–267, IEEE, IEEE Computer Society, Indianapolis, Indiana, USA, August 2009.

25. Ahmed, E., E. Shaban and M. Hashem, "Lightweight Mutual Authentication Protocol for Low Cost RFID Tags", *International Journal of Network Security and Its Applications*, Vol. 2, No. 2, pp. 27–37, April 2010.

26. Lee, Y.-C., Y.-C. Hsieh, P.-S. You and T.-C. Chen, "A New Ultralightweight RFID Protocol with Mutual Authentication", *WASE International Conference on Information Engineering – ICIE '09*, pp. 58–61, IEEE, IEEE Computer Society, Taiyuan, Shanxi, August 2009.

27. Peris-Lopez, P., J. C. Hernandez-Castro, J. M. Estevez-Tapiador and J. C. A. van der Lubbe, "Security Flaws in a Recent Ultralightweight RFID Protocol", *Workshop on RFID Security – RFIDSec Asia'10*, Vol. 4 of *Cryptology and Information Security*, pp. 83–93, IOS Press, Singapore, Republic of Singapore, February 2010.

28. Li, T. and G. Wang, "SLMAP-A Secure Ultralightweight RFID Mutual Authentication Protocol", *Proceedings of Chinacrypt 2007*, pp. 1–5, Cheng Du, China, October 2007.

29. Hernandez-Castro, J. C., J. E. Tapiador, P. Peris-Lopez, J. A. Clark and E.-G. Talbi, "Metaheuristic Traceability Attack against SLMAP, an RFID Lightweight Authentication Protocol", *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium – IPDPS 2009*, IEEE, IEEE Computer Society, Rome, Italy, May 2009.

30. Li, T., R. Deng and G. Wang, "The Security and Improvement of an Ultralightweight RFID Authentication Protocol", *Journal of Security and Communication Networks 2008*, Vol. 1, No. 2, pp. 135–146, 2008.

31. Erguler, I., C. Unsal, E. Anarim and G. Saldamli, "Security Analysis of an Ultralightweight RFID Authentication Protocol-$SLMAP^{*}$", *Security and Communication Networks 2011*, pp. 1–5, 2011.

32. Li, T., "Employing Lightweight Primitives on Low-cost RFID tags for Authentication", *Vehicular Technology Conference 2008*, pp. 1–5, 2008.

33. Bagheri, N., M. Safkhani, M. Naderi and S. K. Sanadhya, "Security Analysis of $LMAP^{++}$, an RFID Authentication Protocol", Cryptology ePrint Archive, Report 2011/193, 2011.

34. David, M. and N. Prasad, "Providing Strong Security and High Privacy in Low-Cost RFID Networks", *Proceedings of Security and Privacy in Mobile Information and Communication Systems*, pp. 172–179, Springer, Heidelberg, Germany, 2009.

35. Hernandez-Castro, J. C., P. Peris-Lopez, R. C. Phan and J. M. Estevez-Tapiador, "Cryptanalysis of the David-Prasad RFID Ultralightweight Authentication Protocol", S. O. Yalcin (Editor), *Workshop on RFID Security – RFIDSec'10*, Vol. 6370 of *Lecture Notes in Computer Science*, pp. 22–34, Springer, Istanbul, Turkey, June 2010.

36. Kinoshita, S., M. Ohkubo, F. Hoshino, G. Morohashi, O. Shionoiri and A. Kanai, "Privacy Enhanced Active RFID Tag", *International Workshop on Exploiting Context Histories in Smart Environments – ECHISE'05*, Munich, Germany, May 2005.

37. Kumar, S. and C. Paar, "Are Standards Compliant Elliptic Curve Cryptosystems Feasible on RFID?", *Workshop on RFID Security – RFIDSec'06*, Ecrypt, Graz, Austria, July 2006.

38. Ohkubo, M., K. Suzuki and S. Kinoshita, "Cryptographic Approach to "Privacy-Friendly" Tags", *RFID Privacy Workshop*, MIT, Massachusetts, USA, November 2003.

39. Rhee, K., J. Kwak, S. Kim and D. Won, "Challenge-Response based RFID Authentication Protocol for Distributed Database Environment", D. Hutter and M. Ullmann (Editors), *International Conference on Security in Pervasive Computing – SPC 2005*, Vol. 3450 of *Lecture Notes in Computer Science*, pp. 70–84, Springer, Boppard, Germany, April 2005.

40. Fouladgar, S. and H. Afifi, "A Simple Privacy Protecting Scheme Enabling Delegation and Ownership Transfer for RFID Tags", *Journal of Communications*, Vol. 2, No. 6, pp.

6–13, 2007.

41. Bringer, J., H. Chabanne and D. Emmanuelle, "HB$^{++}$: a Lightweight Authentication Protocol Secure against Some Attacks", *IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU 2006*, IEEE, IEEE Computer Society, Lyon, France, June 2006.

42. Piramuthu, S., "HB and Related Lightweight Authentication Protocols for Secure RFID Tag/Reader Authentication", *Collaborative Electronic Commerce Technology and Research – CollECTeR 2006*, Basel, Switzerland, June 2006.

43. Munilla, J. and A. Peinado, "HB-MP: A Further Step In The HB-family of Lightweight Authentication Protocols", *Computer Networks*, Vol. 51, pp. 2262–2267, June 2007, http://dl.acm.org/citation.cfm?id=1241112.1241356.

44. Sadighian, A. and R. Jalili, "FLMAP: A Fast Lightweight Mutual Authentication Protocol for RFID Systems", *The 16th IEEE International Conference on Networks (ICON 2008)*, pp. 1–6, New Delhi, India, December 2008.

45. Amphion, *CS5265/75 AES Simplex encryption/decryption*, 2005, http://www.amphion.com.

46. Datasheet, *High Performance MD5. Fast SHA-1. Fast SHA-256. Hash core for ASIC*, 2005, http://www.heliontech.com/auth.htm.

47. Jung, M., H. Fiedler and R.Lerch, "8-bit microcontroller system with area efficient AES coprocessor for transponder applications", *Workshop on RFID and Lightweight Crypto*, pp. 32–43, European Network of Excellence in Cryptology ECRYPT, Fraunhofer, Graz, Austria, 2005.

48. Yuksel, K., J. Kaps and B. Sunar, "Universal hash functions for emerging ultralow-power networks", *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation*, pp. 32–43, The Society for Modeling and Simulation International, SCS, San Diego, California, USA, January 2004.

49. Rama, N. and R. Suganya, "SSL-MAP: A More Secure Gossamer Based Mutual Authentication Protocol for Passive RFID Tags", *International Journal on Computer Science and Engineering*, Vol. 2, pp. 363–367, 2010.

50. Kianersi, M., M. Gardeshi and M. Arjmand, "SULMA: A Secure Ultra Light-Weight Mutual Authentication Protocol for Low Cost RFID Tags", *International Journal of Ubicomp*, Vol. 2, No. 2, pp. 17–24, April 2011.

51. Kianersi, M., M. Gardeshi and H. Yousefi, "Security Analysis of Ultra-lightweight Protocol for Low-Cost RFID Tags: SSL-MAP", A. Ozcan, J. Zizka and D. Nagamalai (Editors), *Recent Trends in Wireless and Mobile Networks*, Vol. 162 of *Communications in Computer and Information Science*, pp. 236–245, Springer Berlin Heidelberg, June 2011.