RFID Mutual Authentication Protocols based on Gene Mutation and Transfer

Raghav V. Sampangi, and Srinivas Sampalli

Abstract: **Radio Frequency Identification (RFID) is a technology that is very popular due to the simplicity in its technology and high adaptability in a variety of areas. The simplicity in the technology, however, comes with a caveat – RFID tags have severe resource restrictions, which make them vulnerable to a range of security attacks. Such vulnerability often results in the loss of privacy of the tag owner and other attacks on tags. Previous research in RFID security has mainly focused on authenticating entities such as readers / servers, which communicate with the tag. Any security mechanism is only as strong as the encryption keys used. Since RFID communication is wireless, critical messages such as key exchange messages are vulnerable to attacks. Therefore, we present a mutual authentication protocol that relies on independent generation and dynamic updates of encryption keys thereby removing the need for key exchange, which is based on the concept of gene mutation and transfer. We also present an enhanced version of this protocol, which improves the security offered by the first protocol. The novelty of the proposed protocols is in the independent generation, dynamic and continuous updates of encryption keys and the use of the concept of gene mutation / transfer to offer mutual authentication of the communicating entities. The proposed protocols are validated by simulation studies and security analysis.**

Index terms: **RFID security, RFID authentication, mutual authentication, genetic mutation, encryption key generation and management**

I. INTRODUCTION

Radio Frequency Identification (RFID) technology is an emerging wireless technology that finds application in nearly all domains. Be it uniquely identifying objects in the retail industry, or tracking an object or an entity through a manufacturing line, or managing patients in healthcare, or the recent concept of “Internet of Things”, RFID has opened the doors for a wide range of applications.

An RFID system typically comprises of electronic circuits known as RFID tags, devices to read data on these tags known as RFID readers, and enterprise servers that store data about the object the tag represents. RFID tags store an identifier (a number) to uniquely identify the objects to which they are attached. To read the data on these tags, RFID readers transmit radio-frequency electromagnetic signals, which energize the tags and allow them to respond with the identifier. The readers then forward this information to the enterprise server requesting for more information about the object the tag represents. The server retrieves and forwards relevant information to the reader after validating the reader, either through wired or wireless networks [1].

RFID Tags can be broadly categorized as passive tags, semi-passive tags, and active tags [1]. This categorization is based on the availability of an on-chip power supply or battery, which either facilitate or not facilitate the tags to initiate communication with readers. Passive tags do not have an on-chip battery, which necessitates the reader to initiate communication. Active tags, on the other hand, have an on-chip battery, which allows them to initiate communication with readers as well as respond to requests from them. Semi-passive tags have an on-chip battery, however, requiring energy from readers to broadcast their message.

Passive RFID tags are typically required to perform one basic function — respond to queries by any readers, and when required, perform data update tasks as instructed by the reader (inherently by the enterprise server). However, they do not have adequate resources for performing sophisticated authentication of the entity giving them the instructions. This makes passive RFID tags vulnerable to a range of attacks such as replay attack, tag killing, tag over-writing, etc. Furthermore, readers with high signal strength can also be used as “rogue” readers, can read information from any tag, even if separated by a large distance. This further increases the vulnerability of the tags, increasing doubts in their widespread acceptance.

Research on strengthening data privacy and security of RFID tags has therefore, assumed focus in recent years. Existing work has focused extensively on using pre-shared secret keys and performing simple bitwise operations such as XOR (exclusive-OR) to perform symmetric encryption and authentication. It has to be noted that the reason for RFID tags to use encryption is to authenticate entities, as significant information is stored securely in the server. However, one thing to note is that the tag, even with the capability to perform minimal computations, does not authenticate either the reader or the server, or does so in very a trivial manner. The reader queries themselves do not pose much threat to tags.

Manuscript received December 15, 2012; revised February 22, 2013.

This work has been funded by the Boeing Company.

Authors are with the Faculty of Computer Science, Dalhousie University, Canada. E-mails: raghav@cs.dal.ca, srini@cs.dal.ca.

However, since readers are a medium for the server to communicate important updates, such as security key updates, from the enterprise server, it becomes a necessity for the tag to validate (or, authenticate) the server. This is based on the premise that even though there can be rogue RFID readers, they cannot extract any valuable information from the tags themselves, since all important information is stored in the server. We present a mutual authentication protocol that focuses on authentication between the tag and the server, using mechanisms to dynamically update the encryption key independently at both the tag and the server.

The mechanism used to update the encryption key introduces an inherent authentication feature. This protocol is based on the concept of genetic mutation and gene transfer. Gene transfer is the process of propagation of characteristics or genes from one generation of an organism to the next. Mutations are changes introduced in this set of characteristics or genes. Our protocol is based employing this concept for generation of encryption keys, and to provide authentication as an implicit feature. We then present an enhancement to this protocol, which further increases the security offered. Prior research has examined the need for key exchange messages over a wireless channel to be authentic [2][3], and many of the RFID protocols use key management / encryption for authentication. Our protocols remove the need for key exchange messages by introducing independent key generation, and provide an inherent mutual authentication feature. The presented protocols are validated using key similarity analysis, complexity evaluation and security evaluation.

We hypothesize that:

H1: The encryption keys are updated with every instance (or, every communicated frame); and,

H2: The encryption keys generated for each frame will be least similar to each other (with similarity quantified by a number in the range 0–1).

The rest of this paper is organized as follows: we present the background and related work in the section II, followed by a description of the proposed protocols in section III. Following the description of the protocols, we discuss the methodology used for evaluating them and present the experimental results / analyses in sections IV and V, respectively. We then present a discussion of our work presented here, and discuss some benefits and challenges in section VI, following which we conclude the paper in section VII.

II. Background and Related Work

A. RFID Security

Security in passive RFID tag based systems is always a critical factor, since passive tags impose several resource and computational restrictions. This makes complex algorithms, which require a high degree of computation for achieving security. On the other end of the scale, simple algorithms may prove easier for an adversary to crack. Much of the current work focus on employing key updates being sent by the server, to synchronize with and update the keys in the tags. However, one scheme focuses on pseudonyms and focuses on updating the identity of the key with each query. We discuss these schemes in this section.

A protocol with the tags storing encrypted versions of their IDs, with their original IDs stored in the database on the server was proposed by Osaka et al. [4]. In this protocol, readers transmit a random number along with their queries, and the tags respond with a number comprising of the mathematically hashed value of the random number received XOR-ed with the encrypted ID. The reader forwards this combination along with the random number to the server, which authenticates the tag and releases information about the object the tag represents after validating the reader. The protocol can be configured to work either with or without a change in the symmetric key used for encryption by the tag. The scheme further supports ownership transfer, thereby supporting privacy protection. It is our understanding that the tag reply consisting of a constant entity (the encrypted ID) coupled with the server transmitting key updates to the tag would reduce the security offered by the scheme, as it opens up avenues to use cryptanalytic techniques to break the key sequence. If one key is retrieved, the succeeding conversations and thus, the system are vulnerable to attacks.

Osaka et al.'s work was slightly modified by Gui et al. [5], to support forward security and prevention of denial of service (DoS) attacks. Their protocol facilitates mutual authentication between the tag and the reader, with the reader / server considered as the same entity to illustrate secure communication channel between the reader and the server. They have introduced an additional XOR computation and hashing at the tag, to verify the reader, and have hence, updated the work proposed by Osaka et al. Their protocol generates number a, that is computed and sent by the tag (using the random number sent by the reader, similar to the Osaka scheme [4]), and numbers e and m, computed using the updated encrypted ID and a random number b. The reader verifies the tag using a, while the tag uses m to authenticate the reader, and e and m to update its encrypted ID. It has to be noted that with e, m and b being transmitted in the open, and the new key being a combination of the numbers so transmitted and the previously agreed key, the system could still be vulnerable to attacks if one of the keys are recovered by standard cryptanalytic techniques.

Yu et al. [6] proposed a protocol based on XTEA encryption, which addressed the issue associated with the unsecure radio frequency (RF) channel used for communication between RFID tags and readers. The nature of the RF channel makes communication vulnerable to attacks such as interception, data capture, data analysis, etc. Their protocol presents a way of encryption and exchange of messages using a non-volatile ID for the tag, a dynamically updated key set (128 bits). Their work is based on the assumption that the least significant 30 bits of a tag's ID can represent a tag uniquely. Server replies to each tag response with an acknowledgement, and instructs the tag to update its key set. For the purposes of authentication, least significant 30 bits of the tag ID are sent along with the message sent by the

server. It is to be noted that if the protocol reduces the number of bits that are significant for an authentication process (by employing the least significant 30 bits), it is also reducing the security and the uniqueness aspect, since it reduces the number of possible combinations of tag IDs.

The work proposed by Molnar et al. [7] employs the concept of pseudonyms and time limited delegation. The tag generates a different pseudonym, or a pseudo-random number, that enables the server to authenticate the tag. The server is referred to as a trusted center in the protocol and it delegates the responsibility of authenticating the tag to a reader by giving it a set of pseudonyms that it can use to verify the tag for a specified amount of time.

Their work has a central concept of "tree of secrets", where nodes of a binary secret tree has secret keys in each node. Each tag has a counter, which points to a leaf of the tree, which in turn represents a pseudonym. Therefore, a particular pseudonym can be used to represent one tag, depending on its present state and the pseudonym. This protocol necessitates the use of a trusted center, and assumes a protected channel of communication between the reader and the server, which might not always be the case.

A protocol for mutual authentication and privacy protection that conforms to EPC Class 1 Generation 2 stanard was proposed by Chen et al. [8]. This protocol necessitates a registration phase, where tags and readers have to register with the server, and a communication phase. The registration phase requires tags and readers to independently register with the server, which generates unique identifiers to represent their IDs. Further, each reader is assigned a set of tags, and it can only communicate with the assigned tags. This makes this protocol not employable in RFID systems with mobile readers, since such a system allows any reader to communicate with any tag.

During the communication phase of this protocol, tags use random numbers, XOR operations and CRC operations to authenticate the reader and vice versa. The protocol focuses on ensuring security and privacy by using a list of valid readers and tags, and by restricting the ability of readers to communicate with any tag in the system.

Vajda et al. [9] present multiple authentication protocols for RFID systems. One of their proposed protocols is based on a simple XOR operation that uses different encryption keys for securing the communication between tag and reader. The session keys are updated with each frame transmission. A block stream generator with a secret key is used to generate the session keys. The key used by the tag, however, remains a constant, which is also the seed used by the reader. The seed is either permuted or expanded by the block stream generator, and it uses recursive permutation of halves of the seed in case of the former. The resultant number is used as the updated key. The tag uses the seed it stores to verify the reader. In their analysis of lightweight authentication protocols, Defend et al. [10] critically analyze their work.

To summarize, we can say that most approaches require the server to perform key updates; all use random numbers, and one using time-limited delegation of responsibilities. One thing to note is that any approach that requires key updates to be performed by the server places makes the network vulnerable as there is a possibility, no matter how least likely, of an adversary cracking the updated keys and hijacking the sessions.

B. Biomimetics

Biomimetics is the use of concepts existing in biological sciences, such as the working mechanism of a neuron, the ascent of sap in plants and so on, in other fields such as engineering, robotics, electronics and so on to create new systems [11]. Traditionally, biomimetics has been used by several known people and organizations, such as Leonardo da Vinci (for his design of flying machines based on birds), Velcro (design derived from the hooked seeds of the burdock plant), anti-reflective surfaces (created using polythene sheets, based on insect eyes, wings and leaves of plants), and many more.

Although no framework exists to specifically use the concepts of biological sciences in other disciplines, one can choose to carefully understand the concept and design the system accordingly. The adoptability of any particular concept is however, subject to the prevailing conditions and requirements in the discipline where it is adopted.

III. PROPOSED PROTOCOLS

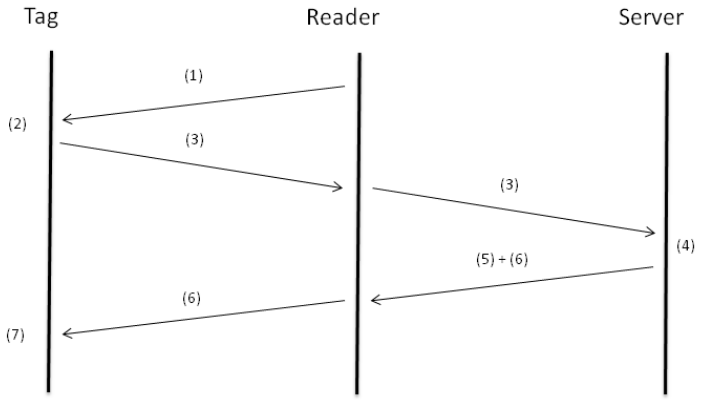
In this section, we describe the security protocol based on the gene mutation and transfer proposed in [12] [1] (Protocol A), and an enhanced version of the same (Protocol B).

A. Overview of Gene Mutation and Transfer

Deoxyribonucleic Acid (DNA) is the basic molecular structure in all living organisms, which contains genetic instructions that help in the organisms carrying out the various functions such as development and other activities. DNA is also responsible for propagating the genetic instructions from the parent generation to the progeny, thereby playing a significant role in the continuity of each species and in preserving characteristic elements (typically hereditary characteristics) of each species. The process by which genetic information is passed from one generation to the next is referred to as gene transmission [13][14].

There may be instances when genes alter, perhaps due to factors in the environment external or internal to an organism. In such cases, there is a very high probability that this alteration is passed on to the next generation, and for subsequent generations of the organism. Such alterations in genetic pattern are referred to as mutations. Let us consider the example of a pea plant. If there are seven peas in one pea pod, and one of them has a small genetic abnormality that has resulted in a dark brown patch on its surface, then, when this pea germinates and grows into a plant, the peas that grow from this plant will have a very high probability of having the brown patch. This depends on the characteristic gene being

[1] A preliminary version of this work has been published in the proceedings of the 2012 IEEE Symposium of Computational Intelligence for Security and Defence Applications (CISDA) [12].



(1) : Reader Query
(2) : $em = E_{ki}$ (ENC_ID)
$k_{i+1} = k_X$ or k_Y
$asv_{i+1} = h$ (numX ⊗ow $pattern_{asv}$)
(3) : *parent* || *generation* || *em* || asv_{i+1}
(4) : Regenerate *em'* using k_i and *ENC_ID*
if *em'* = *em*, tag is not invalid, then:
(4.1) Regenerate asv'_{i+1}
if $asv_{i+1} = asv'_{i+1}$, tag is authentic, then:
(4.2) Retrieve information about tag from database
(4.3) Update key, k_{i+1}
(4.4) Encrypt acknowledgement: E_{ki+1} *(ACK)*
(5) : Retrieved information about the tag
(6) : *Encrypted acknowledgement,* E_{ki+1} *(ACK)*
(7) : Tag: Regenerate encrypted acknowledgement, E'_{ki+1} *(ACK)*
if *E (ACK)* = *E' (ACK)* then:
(7.1) Server is valid

Figure 1. Overview of the proposed protocols.

either dominant (resulting in the patch being visible) or recessive (resulting in the patch not being visible).

Thus, in general, characteristics of a generation of any organism are transferred from one generation to another by means of gene transmission and the transferred characteristics may include abnormalities (or, mutations) as well.

B. Mutual Authentication Protocol based on Gene Transfer and Genetic Mutation (Protocol A)

This protocol mimics the concept of generations and genetic mutation that was described in the previous section. We consider its application on light-weight to heavyweight RFID tags, which have the capacity to perform minimally complex computations, such as the mathematical one-way hash function, and have sufficient storage capability.

Listed below are the assumptions of this protocol:

- RFID tags are initially loaded with the data by the owner. The owner of the tag is the organization where the tag will be deployed.
- The data contained on the tag is an encrypted identifier (ID) uniquely identifying the object it is associated with. For example, if 1234 is the ID associated with an object, the data stored on the tag will be ENCRYPTED (1234), encrypted using any standard encryption scheme by the enterprise server. We denote this as ENC_ID.
- We denote the ENC_ID subjected to one round of simple encryption as the encrypted message, *em*.
- The tag and the server share a pre-loaded 128-bit initial encryption key (IK).
- The tag and the server store states of the three (3) previously used keys in their memory to retrieve the previously saved synchronization state in case of dropped messages. These are referred to as the key states.
- Key states include the seeds of the random number generator, integer numbers (*parent*, *generation*) and the keys. These are required by the communicating entities to restore state in case of dropped messages, as will be explained in the sections below.
- The tag and the server store a pre-loaded 128-bit authentication-synchronization vector (ASV), which is used to authenticate the tag and synchronize with the server for the very first message. The ASV is unique for all tags deployed in the said environment, and is updated with every key update. The ASV helps in identifying loss of synchronization or any attempt of data modification.
- The authentication of readers with servers is beyond the purview of this work. The readers are assumed to use any standard authentication mechanism for this purpose.

Protocol A works as follows. The reader queries the tag, to which the tag responds with an encrypted version of ENC_ID. The ENC_ID is encrypted using the initial key (IK) to generate *em*. Following the encryption, the tag updates its encryption key to the new key (NK), and generates the ASV during the key update. The tag transmits a message that consists of two integers, *parent* and *generation*, which indicate the current state of the tag, the *em*, and the mathematically hashed ASV.

On receiving this message, the reader forwards it to the server. The server then authenticates the tag by first verifying the state of the tag using the parent and generation, then the encrypted message and finally the hashed ASV, and validates the tag. After the tag is validated (and implicitly, the reader is validated), the server retrieves and releases information about the object the tag represents to the reader, along with an encrypted acknowledgement (E_{ki+1}*(ACK)*) message. The key used for encrypting the acknowledgement is the updated key, which is generated at the server during the tag authentication and key update process. On reception of E_{ki+1}*(ACK)*, the tag authenticates the server, and performs key updates (if needed for synchronization) and is ready for the next query. This operation is summarized in Figure 1.

Updates to the encryption key and the ASV is achieved in our protocol using bitwise operations, which are used to mimic the action of genetic mutation and gene transmission across

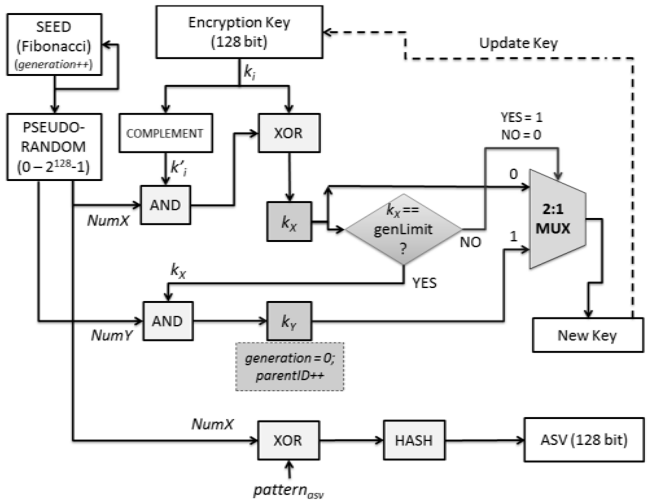


Figure 2 (a). Illustration of the working of the protocol (Protocol A)

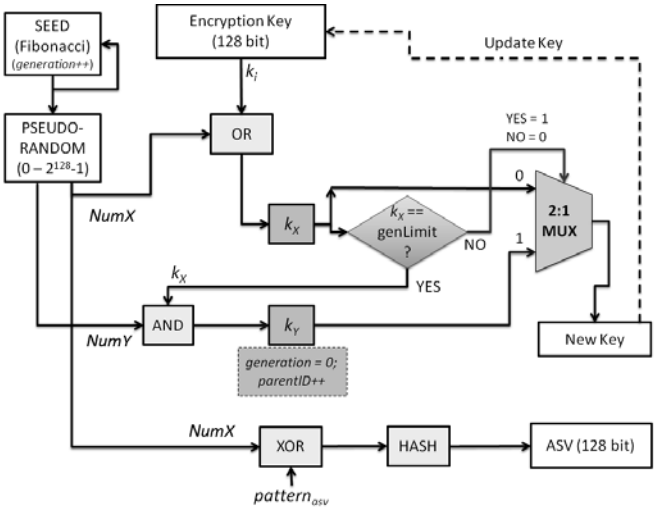


Figure 2 (b). Optimized version of Protocol A

generations of keys. Figure 2(a) illustrates the details of Protocol A. The operation to generate K_x in Figure 2(a) can be optimized and can be achieved with the help of one OR operation instead of a combination of XOR operations. This is illustrated in Figure 2(b). We have included Figure 2(a) to illustrate the concept of preserving the 1 bits of the parent key and applying mutations on the other bits. Note that the 2:1 MUX (the multiplexer) is an illustration for the logical operation to choose an appropriate key based on the condition being satisfied. The following paragraphs describe the scheme in detail.

B.1 Key Generation and Management

The novelty of our protocol is in using the concept of gene transfer and genetic mutation to generate the keys. In RFID systems, the simplicity of tags allows for adversaries to clone tags or to track tags using the responses. Some of the protocols described earlier [4][5][6] try to overcome this flaw by employing key refreshes and updates, i.e. the server transmits keys in either acknowledgement messages or explicit key refresh messages. On the other hand, the pseudonym protocol generates a new identifier (ID) using a generator that is synchronized with the server [7]. One aspect common to all these protocols is the transmission of keys, either from the server to the tag or from server to the reader.

In our protocol, we eliminate key exchange messages and key update messages. Using simple logical operations and pseudorandom numbers, we ensure that the tags and servers are able to:

- Generate keys independently
- Synchronize the keys
- Ensure mutual (or, bilateral) authentication

The tag uses the encryption keys so generated to further encrypt the already encrypted ID. It has to be noted that in RFID systems, encryption is used as a way of authenticating the entities, since RFID tags by themselves do not store any critical information. The tag and server are pre-loaded with an initial key (IK). When the tag is queried for the first time by a reader, the tag is energized and it encrypts ENC_ID to generate *em*. Following this, it initiates the process of generating the new key (NK).

New key generation process is as follows. The system first generates a mask of the IK, to preserve the pattern of its 1 bits by computing its one's complement. This is akin to the preserving the genetic characteristics of an organism. The tag has a synchronized pseudorandom number generator (PRNG) with the server, having a Fibonacci number as seed. A 128 bit pseudorandom sequence is generated by this PRNG, referred to as *NumX*. The protocol performs a bitwise AND operation on *NumX* with the mask to give a mutation pattern, which is a pattern of 1s and 0s (*NumX* ') to fill the 0 bits of the IK, without modifying the 1 bits. Finally, *NumX* ' is XOR-ed with the IK to generate K_X. This is analogous to applying mutation to an existing pattern. Please note that in an optimized version of the protocol illustrated in Figure 2(b), this set of operations to generate K_X can be achieved using one logical OR operation between the IK and *NumX*. The Fibonacci number is updated to obtain a new seed for the PRNG, in order to generate a new pattern for the subsequent communication.

By using a mask, we preserve the characteristics of the IK, which is similar to preserving the characteristics of each generation during gene transfer. By introducing a new pattern to occupy the 0-bit positions of the IK, we are introducing "mutations", and the K_X represents the "mutated" pattern. The variable *generation* keeps track of the modifications on the particular key.

The *genLimit* pattern is used to decide whether K_X will be used as the updated key. If K_X is different from *genLimit*, K_X will be used as the key for the next encryption cycle. In such a case, each key acts as the parent to the next key, and its bit pattern is preserved in the new key. When K_X becomes the same as the *genLimit*, the protocol decides in favor of abandoning K_X and generating a new - unrelated - key. In this case, the next usable 128 bit sequence from the PRNG is generated, and is AND-ed with K_X to generate K_Y. This is when a new parent key is generated, which is recorded in the key state being updated by incrementing the *parent* and