Lightweight Authentication Protocols for Low-Cost RFID Tags

István Vajda and Levente Buttyán
Laboratory of Cryptography and Systems Security (CrySyS)
Department of Telecommunications
Budapest University of Technology and Economics, Hungary
http://www.crysys.hu/

August 5, 2003

Abstract

Providing security in low-cost RFID tags is a challenging task because tags are highly resource constrained and cannot support strong cryptography. Special lightweight algorithms and protocols need to be designed that take into account the limitations of the tags. In this paper, we propose a set of extremely lightweight tag authentication protocols. We also provide an analysis of the proposed protocols.

1 Introduction

Radio Frequency Identification (RFID) systems are composed of RF tags and RF tag readers. Most tags consist of an antenna connected to a microchip. The use of silicon-based microchips enables a range of functionality to be integrated into the tags, including readable/writable storage and limited computing capability. Tag readers broadcast an RF signal to access information stored on the tags. This information can range from static identification numbers to user written data or data computed by the tag.

In the near future, low-cost RFID tags attached to consumer items as "smart-labels" may become an economical and efficient replacement for optical bar codes. Indeed, RFID tags offer several advantages over optical bar codes: data may be read automatically, without line of sight, through non-conducting material, at a rate of several hundred tags per second, and from a distance of several meters. Besides replacing optical bar codes, the above described characteristics of tags makes them useful in other applications as well, including access control to buildings, environmental sensing, livestock and automobile identification, inventory control, theft detection, etc. Because of their numerous applications and their low cost, RFID tags has a strong potential to become a ubiquitous computing technology.

However, several researchers have pointed out that the universal deployment of RFID systems may create new security and privacy problems (see e.g., [7, 10, 4]). The potential risks include corporate espionage, and violation of consumer (personal) privacy and location privacy. In traditional computing systems, many security and privacy problems can be solved by using cryptographic technics [8]. Unfortunately, RFID tags are highly resource constrained and cannot support strong cryptography. To be more precise, tags could be equipped with resources to support strong cryptographic primitives, tamper resistant packaging, and other security enhancing features, but only at a higher cost of 0.5–1 USD/tag. On the other hand, significant market penetration can be expected only if tags are priced below 0.1 USD or even below 0.05 USD [7]. In this price range, tags come with the following typical characteristics:

- storage capacity of a few hundreds bits;
- a few thousands gates available for logical functions;
- tags are passively powered, which excludes background calculations in idle time when the tag is not "powered" by the tag reader;

- limited distance and quality of radio transmission due to the low gain antenna and severe power constraints of the tags;
- no tamper resistance.

This means that supporting strong cryptographic primitives on low-cost tags is not a viable option today. Note that implementing even a standard cryptographic hash function, such as MD5 [5] or SHA-1 [1], is beyond the capabilities of today's tags. Hence, there is a strong need for new, lightweight cryptographic primitives that can be supported by low-cost RFID tags.

In this paper, we propose a set of extremely lightweight challenge-response type authentication protocols that can be used in low-cost RFID systems for authenticating the tags. Tag authentication is an important primitive that serves as a fundamental building block in more sophisticated security and privacy protecting mechanisms. As a motivating example, consider a theft detection system where RFID tags are attached to stocked items, and a central tag reader periodically polls and authenticates each tag. The system should be designed in such a way that a thief cannot steal an item and go undetected by installing a cloned replacement tag that can continue authenticating itself successfully to the reader. Our protocols are designed to prevent this cloning attack.

A main contribution of our work is that we also provide an analysis of the proposed protocols. The goal of the analysis is to obtain a lower bound on the resource requirement of the best guessed attack against a given protocol, where the resource requirement is measured in the computational complexity of the attacking algorithm as well as in the number of runs that the attacker needs to observe or interfere with. Identifying a lower bound on the "cost" of the attack may help system designers to choose the best trade-off between complexity of the protocol and resistance against attacks. This is important, as the resource constraints of the tags do not allow arbitrarily complex protocols.

Outline. In Section 2, we report on some related work. In Section 3, we introduce our system model. In Section 4, we present our proposed protocols together with their analysis. Finally, in Section 5, we conclude the paper.

2 Related work

To the best of our knowledge, the first paper that calls attention to the security and privacy risks and challenges of widely deployed RFID systems is the paper of Sarma *et al.* [7]. In particular, the authors mention scarcity of tag resources as a primary challenge in providing security and privacy mechanisms in low-cost RFID systems. They suggests that new, lightweight cryptographic primitives and protocols should be developed that take into account the strong resource constraints of RFID tags. Our work is an effort in this vein. The authors of [7] also mention the problem of tag spoofing, which would enable a thief to replace an item with a cloned tag and fool a "smart shelf" that the valid item were still in stock. The tag authentication protocols we propose can be used to protect against this attack.

Several papers propose lightweight cryptographic primitives for resource constrained applications such as smart cards and sensor networks. In [2], Hoffstein *et al.* propose a lightweight public-key cryptosystem called NTRU. In [9], Stern and Stern propose a lightweight digital signature scheme. While both of those proposals lead to very efficient mechanisms compared to previously known public-key cryptosystems and digital signature schemes, they still require resources well beyond what is available on low-cost RFID tags. In [6], Perrig *et al.* propose TESLA, an efficient broadcast authentication mechanism for sensor networks, which is based on symmetric-key cryptography. However, TESLA uses hash chains and standard message authentication codes, none of which can be implemented in low-cost RFID tags. Moreover, TESLA relies on time synchronization between the base station and the sensors, which is also beyond what is feasible in low-cost RFID systems. In contrast to these proposals, our protocols use only elementary logical and arithmetical operations that can be implemented with a few gates.

In [10], Weis *et al.* propose various schemes for controlling access to RFID tags. In their proposal, each tag can be in two states: in a *locked* state, where it responds to all queries with only its meta-ID and offers no other functionality, and in an *unlocked* state, where it can perform privileged operations related to security and configuration. The goal of the proposed schemes is to ensure that the tag enters into the unlocked state

only if it receives an appropriate command from a legitimate tag reader. Hence, the proposed protocols mainly provide reader authentication. In contrast to this, in this paper, we are mainly concerned with tag authentication. In addition, the protocols proposed in [10] use standard cryptographic hash functions and one of the protocols also requires a secure pseudo-random number generator to be implemented on the tags, which seems to be infeasible with current technology.

In [3], Juels addresses the problem of privacy protection in low-cost RFID systems. He proposes a scheme where each tag stores a list of pseudonyms. Each time the tag is queried, it emits the next pseudonym from its list. In addition, the query-response rate of the tags is deliberately reduced, which guarantees that tags emit pseudonyms with a relatively low rate. Thus, an attacker can track a tag only if he has access to it for a long period of time. Due to their small storage capacity, tags can store only a short list of pseudonyms. Juels solves this problem by allowing the list to be refreshed by authorized tag readers. For this reason, mutual authentication is required between the tag and the reader (otherwise an attacker could update the list of pseudonyms in a tag). Juels proposes a lightweight mutual authentication protocol, which is based on the release of "keys" that are supposed to be secret and associated to the parties running the protocol. After mutual authentication, the keys are refreshed; for this purpose, new keys are generated by the tag reader, and transmitted in an encrypted form to the tag. Encryption is based on a one-time pad, where the pad is selected from a series of pads maintained by the tag. The series of pads is also updated (overlayed) with new padding material in each run of the authentication protocol. The new padding material is sent in clear to the tag, but the updating procedure ensures that a pad is used (becomes "live") only after a certain number of updates. This number m should be chosen in such a way that an attacker cannot observe m consecutive updates. The rationale is that observing a run of the protocol requires physical proximity, and one may assume that an attacker cannot stay in the vicinity of a tag for an arbitrary long time.

The protocol proposed by Juels does not require the tag to perform any cryptographic operations (apart from XOR), hence it is feasible to implement it in current low-cost RFID systems. However, the protocol uses 4 messages, and updating the keys and the pads has a cost (in terms of bits that has to be pushed from the tag reader to the tag). In addition, in some applications, the assumption that the number of consecutive runs of the protocol that an attacker can observe can be upper bounded does not hold. In particular, if the tags do not move (e.g., they are attached to items stored in a depot), then the attacker can relatively easily install itself at a nearby location and eavesdrop a huge number of consecutive runs of the authentication protocol. In this case, Juels' protocol would not be appropriate for authenticating the tags. In contrast, our protocols were developed with this application in mind; on the other hand, we do not make any attempt to prevent the tracking of tags.

3 System model and assumptions

We consider a system that consists of one RFID tag reader and several RFID tags. We assume that each tag shares a secret with the reader, which is established in a secure manner before the beginning of the operation of the system. Message passing between the reader and the tags is based on single-hop wireless communication. Tags are passively powered, so they can operate only if the reader provides the necessary energy. Tags are highly resource constrained: they have limited computing power, limited storage capacity, and limited communication capabilities.

We assume that the reader regularly polls the tags, each time with a new challenge, and the tags must authenticate themselves by correctly responding to the challenge. There are known protocols to do this in a secure manner, but they use standard cryptographic primitives (MAC, digital signature, encryption), which are too costly for low-cost RFID tags. We assume that computing even a standard cryptographic hash function, such as MD5 or SHA-1, exceeds the capabilities of the tags, and thus, it is infeasible in our system. In the case of standard cryptography, the speed and simplicity of an algorithm are usually qualifying factors; in our case, however, low complexity of the primitives has a first place importance.

The attacker's aim is to produce a response to a challenge. If he can do this in a feasible way, then we say that the protocol is broken. Such a success of the attacker might be achieved with or without recovering the secret key shared by the reader and the tag. This means that it is not enough to prove that the secret key cannot be calculated in a feasible way by the attacker, because he might reach his goal without it. In the

analysis of the protocols presented below, we will consider also degrading if the attacker is able to reduce the key space or can calculate some of the bits of the key in a feasible way.

The data from which the attacker tries to prepare for a successful response could be obtained in a passive or in an active manner. In case of a passive attack, the attacker collects messages from one or more runs without interfering with the communication between the parties. In case of an active attack, the attacker impersonates the reader and/or the tag, and typically replays purposefully modified messages observed in previous runs of the protocol.

4 Protocols

In this section, we propose 5 lightweight tag authentication protocols that matches the stringent characteristics of the system described above. The presentation of each protocol is broken into three parts: first we describe the protocol and its rationale, then an analysis follows, and finally, based on the weaknesses explored by the analysis, some improvements and strengthening are suggested. Before starting the presentation of the protocols, we would like to illustrate some of the concepts on a simple example. Notations used in this example will be kept hereafter. Let us consider the following protocol:

$$\begin{aligned} R \rightarrow T &: x \oplus k = a \\ T \rightarrow R &: f(x) \oplus k = b \end{aligned}$$

where R and T stand for the reader and the tag, respectively; k is the secret key shared by R and T; x is an n bit random challenge; and f is an n-bits to n-bits mapping.

The mutual information $I(h,k)$ between the observable message pair $h = (a,b)$ and the key k is $I(h,k) = H(x \oplus f(x))$, where $H(x \oplus f(x))$ is the entropy of $x \oplus f(x)$. This can be proven as follows. By definition $I(h,k) = H(h) - H(h|k)$. Because of the random selection of x, $H(h|k) = n$. Furthermore,

$$\begin{aligned} H(h) &= H(a,b) = H(a, a \oplus b) = H(a \oplus b) + H(a|a \oplus b) = \\ &= H(x \oplus f(x)) + H(x \oplus k | x \oplus f(x)) = H(x \oplus f(x)) + n \end{aligned}$$

Thus, $I(h,k) = H(x \oplus f(x)) + n - n = H(x \oplus f(x))$.

For instance, if f is the identity mapping (i.e., $f(x) = x$), then no information about the key can be gained from the observation of the run. However, choosing f as the identity mapping would trivially be a bad choice, because the attacker could simply replay the challenging message as the response. The other extreme is when f maps to a constant (i.e., the response message is shifted by a know constant vector). In this case, the mutual information is at maximum. Furthermore, if $x \oplus f(x)$ is a uniform mapping into an m dimensional subspace of the n dimensional vectors, then $n - m$ bits of the key remain independently selected.

Selection of a linear binary mapping for f is dangerous. Let M and I be two $n \times n$ binary matrices, where M represents mapping f and I is the identity matrix. The attacker can set up the following system of linear equations:

$$(M \oplus I)k = Ma \oplus b. \tag{1}$$

The solution of this system for unknown k is not unique if the rank of matrix $M \oplus I$ is less than n. Note, however, that the attacker does not have to know the exact key to be able to efficiently produce a successful response message; it is enough to know an arbitrary solution of (1).

Keeping the main structure of the protocol, possible ways of strengthening would be the following:

- *non-linearity:* use of a nonlinear f could make it harder for the attacker to calculate the set of preimages (practically one way property);
- *mixed operations:* instead of the XOR operation (which is linear over binary vectors), modular integer addition or modular integer powering could be used; this would make it harder to combine the messages as well as to analyze them;

- *compression:* explicit dimension shrinking "provably" decreases the information about sensitive elements (key, the actual challenge) that the attacker has access to (in other words, authentication does not need to be based on invertible transformations).
- *keys:* different keys are used in the two directions.

However, strengthening must be done carefully and gradually: only light weight modifications are allowed and they must be followed by the reiteration of the analysis.

4.1 Protocol 1: XOR

Our first protocol has a structure similar to that of our example above, but it uses different keys in different directions:

$$\begin{aligned} R \to T &: x \oplus k_1 \\ T \to R &: x \oplus k_2 \end{aligned} \tag{2}$$

This would be provably secure if the keys k_1 and k_2 are selected uniformly at random in each run of the protocol. However, this clearly leads to a key establishment problem, which is further complicated by the special characteristics of our system (limited storage capacity of tags, impossibility of frequent manual key refreshing, etc.). In order to address these problems, a provably secure algorithmic key update scheme is needed, and it should also be based on a one-time pad. Note, however, that no more fresh random bits can be sent by one-time pad than that consumed for transmission!

One possibility for XOR rekeying is the following: in run i, R randomly selects a new key $k^{(i)}$ and transmits it XOR encrypted with the key $k^{(i-1)}$ used in the previous run. This leads to the following protocol:

$$\begin{aligned} R \to T &: a^{(i)} = x^{(i)} \oplus k^{(i)}, k^{(i)} \oplus k^{(i-1)} \\ T \to R &: b^{(i)} = x^{(i)} \oplus k^{(0)} \end{aligned} \tag{3}$$

where $i = 2, 3, \ldots$ is a counter, which is increased by one in every new run of the protocol, $x^{(i)}$ is the i-th challenge, and $k^{(0)}$ and $k^{(1)}$ are initially shared secret keys. This protocol uses only XOR operations, and from this point of view it would be ideal. However, it is breakable from two observed consecutive runs. Note that the series of keys $k^{(1)}, k^{(2)}, \ldots$ do not change randomly; the difference of them is a known value for the attacker.

Therefore, we diverge from the pure XOR protocols and we determine the consecutive session key used by R by a lightweight block stream generator with secret seeding value $k^{(0)}$, which outputs a new block in each new session. Consider the following protocol:

$$\begin{aligned} R \to T &: a^{(i)} = x^{(i)} \oplus k^{(i)} \\ T \to R &: b^{(i)} = x^{(i)} \oplus k^{(0)} \end{aligned} \tag{4}$$

where $k^{(i)} = \Pi(k^{(i-1)})$, and $\Pi : \{0,1\}^n \to \{0,1\}^n$ is a permutation, a special key stream generator that expands seed $k^{(0)}$. Permutation Π is defined as follows.

For simpler explanation, let us assume that the key length is 128 bits (i.e., $n = 128$). Let us cut each byte of $k^{(i-1)}$ into two half bytes. The left halves $k_{1,L}^{(i-1)}, k_{2,L}^{(i-1)}, \ldots k_{16,L}^{(i-1)}$ are concatenated into a vector denoted by $k_L^{(i-1)}$. Similarly, the right halves $k_{1,R}^{(i-1)}, k_{2,R}^{(i-1)}, \ldots k_{16,R}^{(i-1)}$ are collected into a vector denoted by $k_R^{(i-1)}$. Then, the calculation is continued in two main steps:

- *Step 1:* The elements of $k_R^{(i-1)}$ are permuted, where the permutation is controlled by $k_L^{(i-1)}$. The result is denoted by $k_R^{(i)}$.
- *Step 2:* The elements of $k_L^{(i-1)}$ are permuted, where the permutation is controlled by $k_R^{(i-1)}$. The result is denoted by $k_L^{(i)}$.

$\Pi(k^{(i-1)}) = k^{(i)}$ is obtained from rearranging (interleaving) the vectors of half bytes $k_L^{(i)}$ and $k_R^{(i)}$ into a vector $k^{(i)}$ of bytes.

Within Step 1, the following sub-steps are made (similar sub-steps are made within Step 2): In the first sub-step, the first and $k_{1,L}^{(i-1)}$-th elements of $k_R^{(i-1)}$ are swapped. In the second sub-step, the second and the $k_{2,L}^{(i-1)}$-th elements of the vector resulted after the first sub-step are swapped. And so on until in the 16th sub-step, the 16th and $k_{16,L}^{(i-1)}$-th elements of the vector resulted after the 15th sub-step are swapped. For a small instance $(k_L^{(i-1)}, k_R^{(i-1)}) = ((3,2,4,1),(2,4,1,3))$ is mapped into $(k_L^{(i)}, k_R^{(i)}) = ((4,1,3,2),(2,4,3,1))$.

Passive attack. By observing the messages of the i-th and $(i+1)$-st runs of the protocol, the attacker can calculate the following:

$$k^{(i)} \oplus k^{(0)} = a^{(i)} \oplus b^{(i)} \quad (5)$$

$$k^{(i+1)} \oplus k^{(0)} = a^{(i+1)} \oplus b^{(i+1)} \quad (6)$$

Adding together equations (5) and (6), the attacker gets

$$k^{(i)} \oplus k^{(i+1)} = a^{(i)} \oplus a^{(i+1)} \oplus b^{(i)} \oplus b^{(i+1)} \quad (7)$$

where the right side is known to the attacker. This means that the attacker is able to see the difference of consecutive session keys. The aim of the attacker is to find the seed $k^{(0)}$ in order to be able to impersonate T.

Assume that the attacker has a guess on session key $k^{(i)}$, $i \geq 1$. Then, according to (7), he also has a guess on the new session key $k^{(i+1)}$. He can check this guess by also calculating $k^{(i+1)}$ from $k^{(i)}$ using Π. However, guessing the session key is a brute force attack, which is easily foiled by selecting the dimensions sufficiently large. It is an open question whether knowing the vector difference of two consecutive outputs of the above defined block stream generator can be used for feasible attacks against its seeding block $k^{(0)}$.

4.2 Protocol 2: SUBSET

Consider the following protocol, where an XOR encrypted n-bit challenge is sent to the tag, which sends an m-bit portion of the challenge back as the reply:

$$\begin{aligned} R \to T &: x \oplus k \\ T \to R &: f(x) = (x_{L,x_{R,[0..7]}}, x_{L,x_{R,[8..15]}}, \ldots, x_{L,x_{R,[8m..8m+7]}}) \end{aligned} \quad (8)$$

The challenge is divided into two parts: $x = (x_L, x_R)$. The j-th byte of x_R, denoted by $x_{R,[8j..8j+7]}$, addresses a bit of x_L, denoted by $x_{L,x_{R,[8j..8j+7]}}$, which is considered as the j-th bit of the output vector.

For concreteness and simpler presentation, let us assume the following parameters: $n = 384 (= 256 + 128)$, $|x_L| = 256$, $|x_R| = 128$ (bits), and $m = 16$. The probability of successful impersonation of the tag using a random response message is $2^{-m} = 2^{-16}$ for the above instance, which is considered unacceptably high in "standard" cryptography. This might not be the case for some RFID applications. Consider, for instance, our theft detection example, where the needed strength of security depends on the value of the protected goods. Taking the bit selector bytes overlapped makes it possible to increase m without increasing the length of the challenge (and the key).

One might think that, in practice, it would be easy to thwart random attacks in general, because in case of wrong responses, the reader could send an alarm signal toward the physical security subsystem. Unfortunately, it seems to be a plausible assumption that low capability devices like RFID tags are not able to run efficient forward error control mechanisms. This fact together with the poor quality of the radio channel might lead to packet error probabilities preventing the application of reliable alarms.

Passive/active attack. The attacker listens to one run of the protocol and stores messages. He guesses the first byte of k_R, therefore he also has a guess on the first byte of x_R (byte guess) and a guess on one bit of k_L (bit guess). By listening to different runs, the attacker builds a list of guesses on the bits of k_L. He continues listening to new runs until inconsistent bit guesses appear on his list (i.e., he gets both a guess of

0 and 1 for $k_{L,j}$ for some j). When the attacker gets contradicting values on the same key bit, then he can be sure that the actual guess on the first byte of k_R is wrong.

The probability distribution of the number of s-fold collisions when drawing with replacement from within a single set is known (the case of 2-fold collisions is known as the birthday paradox). For instance, the probability of two byte guesses leading to a guess of the same bit of k_L (2-fold collision) is around 0.504 if at least 19 runs are observed. Because the bit guesses are random (except when the byte guess is right), the probability that the two bit guesses obtained from incorrect byte guesses are different is 1/2. Therefore, according to the birthday paradox, the attacker detects an incorrect byte guess with probability $0.252 (= 0.504/2)$ (from 19 observed runs). Refining the calculation by taking into account both 2- and 3-fold collisions we get, for instance, that by observing 50 runs, the detection probability of a false byte guess improves to 0.83.

Note that the attacker can accelerate the guessing process by guessing in parallel all possible values of the first byte of k_R. He builds a matrix with 256 columns, where the columns correspond to different possible values of the key byte, and different rows correspond to different observed runs. Within a row, we have a guess for different bits of k_L. Exactly one of these key bits is guessed correctly, all others are guessed randomly. Therefore the attacker can filter out all incorrect byte guesses in parallel by finding inconsistent guesses in the corresponding rows.

Once having a reliable byte guess on the first byte of k_R, the attacker can start to explore further bits of k_L. In case of an active attack, the attacker – knowing the 1st byte of x_R – shifts the value of this byte by XOR adding the corresponding shift values to the corresponding – observed – challenge message. In this way, he can obtain all the bits of k_L from the answers by close to 256 active attacks (note that at least one bit of k_L can be obtained during the byte guess step). If an active attack is costly, then the attacker has to wait until the first byte of the randomly chosen x_R scans all (or a large portion of the) bit positions of k_L, which may lead to a number of observed runs well above 256.

The exploration of the remaining bytes of k_R needs less effort, because the attacker already knows k_L.

Strengthening. One of the main weaknesses of the above protocol is that the attacker is able to scan the bit positions of k_L through the manipulation of a single byte. Therefore, instead of only one byte, all bytes of k_R should be involved in the selection of each bit of the output. This could be done in the following way: the bits of $f(x)$ are linearly combined by bits of a separate portion of the key. In this way, the attacker is forced to guess multiple bytes of k_R.

4.3 Protocol 3: SQUARING

Consider the following protocol:

$$\begin{aligned} R \to T &: x \\ T \to R &: k_L \oplus ((k_R + x)^2 \bmod 2^n) \end{aligned} \tag{9}$$

where k_L and k_R are two halves of a $2n$ bit secret key $k = (k_L, k_R)$, furthermore "+" denotes integer addition.

Active attack. The attacker challenges T two times by sending $x = 0$ and $x' = 1$ as challenges. Then, he calculates the XOR sum of the responses:

$$z = k_R^2 \oplus (k_R + 1)^2 \bmod 2^n \tag{10}$$

$$= k_R^2 \oplus (k_R^2 + 2k_R + 1) \bmod 2^n \tag{11}$$

Note that in expression (11), two different operations are mixed.

Let b_i denote the i-th bit of b, where the LSB is b_0. Using this notation, we get that $z_0 = 1$, and $z_1 = k_{R,0}$. This means that the attacker can obtain the LSB of the key. If this key bit is zero, he can obtain also $z_2 = k_{R,1}$, and so on until he arrives at the first non-zero key bit (starting from the LSB). He cannot proceed further from this point, because he should also know the appropriate bit of k_R^2 that affects the actual carry bit.

A further helpful observation is the following: when adding two bits, there is a carry bit only in case of addition $1 \oplus 1$, which happens only for 1/4 of the combinations of bit pairs. Therefore, when we have

to stop at the first non-zero bit of the key, we know that - in average - the chance that the appropriate bit of k_R^2 will not cause a carry bit is 3/4, so we proceed further in exploring more key bits under the assumption that there is no carry bit.

The above attack can be extended to the case when $x = a$ and $x' = a + 2^j$ for arbitrary a. As an example, let us consider $a = 0$, $j = 3$, and $k_R = 12_{|10} = 110\mathbf{0}_{|2}$. Then, $12^2 = 144_{|10} = 100\mathbf{1}0000_{|2}$, and $(12 + 2^3)^2 = 400_{|10} = 1100\mathbf{1}0000_{|2}$. The XOR of the fourth bits from the right (shown in bold) gives the LSB bit of 12.

Note that if we find a few bits from k_R, we also know some bits of k_R^2, and consequently, we can learn also a few bits of k_L.

Strengthening. The use of XOR addition instead of the integer addition before squaring could be a modification to be considered. This is because $z \oplus 1 = z + 1$ if $z_0 = 0$, and $z \oplus 1 = z - 1$ if $z_0 = 1$ (where "$-$" denotes integers subtraction), and hence, the "difference" depends on the bit which the attacker would like to find.

4.4 Protocol 4: RSA

Assume an RSA encryption function E with block length n, public exponent e and secret exponent d. Consider the following protocol:

$$\begin{array}{lcl} R \rightarrow T & : & x \\ T \rightarrow R & : & E(x \wedge k) \end{array} \tag{12}$$

where x is a mask vector, in which there are $0 < m \leq n$ bits set to 1 at randomly selected positions; $x \wedge k$ denotes the bitwise AND of x and k, which masks the vector k by keeping the values of the bits of k at positions where the corresponding bits in x are 1 and setting all other bits of k to 0.

The number of operations when an RSA encryption function is evaluated depends on the binary weight of the exponent (according to the repeated square and multiply algorithm) as well as on the binary weight of the plain text. The second step of the protocol ensures that the binary weight of the plain text is at most m. Furthermore low weight public exponent is applied (e.g. $2^{16} + 1$).

Passive attack. The attacker can also try to determine m bits of k by listening on the channel and breaking the encryption $E(x \wedge k)$ by exhaustive search for the bits of k on coordinates determined by the mask x. This means at most 2^m trials. The workload of the attacker can be increased by increasing m, but this also increases the workload of T. Moreover, the more runs of the protocol are attacked, the more information can be obtained about k.

Active attack. The attacker changes two bit positions of an observed challenge x: a 1 bit of x is turned to 0 bit and a 0 bit of x is changed to 1 bit. In other words, a difference vector with Hamming weight 2 is XOR added to the challenge. The observation is that the response message will not change if key k had zeros at both of these two bit positions and it typically changes for the other three pairs (i.e., 01, 10, 11). Obviously, the probability of a zero pair is 1/4. In this way, the attacker has a probabilistic chance for scanning bit pairs of the key.

Strengthening. Key k is cyclically shifted with shift S before the masking operation is done, where the random shift is an appropriate mapping of an additional secret value k' and of the actual mask x: $S = g(k', x)$.

4.5 Protocol 5: KNAPSACK

The base station maps a random challenge by a trapdoor (practically) one way function and sends the result of this mapping together with the XOR encrypted one time trapdoor value to the tag:

$$\begin{array}{lcl} R \rightarrow T & : & d \oplus k, \kappa(x, d) \\ T \rightarrow R & : & x \oplus k' \end{array} \tag{13}$$

where k is an m-bit secret key and k' is an n-bit secret key, x is an n-bit challenge, and d is an m-bit trapdoor. Furthermore, κ is a punctured multiplicative knapsack: A public set of s-bit prime numbers consisting of n primes is stored both by R and T. R selects randomly $n/2$ elements from this set and multiplies together the selected primes. The n-bit challenge x contains 1 at those bit positions which correspond to the primes selected (an order is assumed among the primes) and 0 at the remaining positions. R chooses t integers randomly from the range of $1, 2, \ldots, s \cdot n/2$, and marks bits of binary representation of the product at bit positions corresponding to the selected integers. The marked bits are deleted and the binary string is shrunk in size accordingly. The resulted punctured string is the output of mapping κ. Trapdoor d consist of the integers used in puncturing, by appending these integers in order. It follows that the output of κ has length $s \cdot n/2 - t$ bits, furthermore the trapdoor is $m = t \cdot \log(s \cdot n/2)$ bits long.

For illustration consider the following instance: $s = 8$, $n = 24$. The set of primes are the primes from 127 to 255: 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251.

The tag knows the punctured position according to the trapdoor and by exhaustive search finds the punctured bits. Consequently t must be a small integer. The tag checks the correctness of the guessed values of the punctured bits by dividing the guessed product by a few primes. Let u denote the product of $n/2$ primes selected by R. When T guesses the punctured bits an integer $v = u + w$ is produced, where integer w equals 0 only for the right guess of punctured bits. The probability that a randomly selected integer w (hence v) is a multiple of a given prime from the set of primes is $2^{-(s-1)}$. Similarly, the probability that a wrong guess passes the test with j primes is $2^{-j \cdot (s-1)}$.

Note that this protocol – in contrast to the previous ones – provides some level of reader authentication, as $\kappa(x, d)$ can be viewed as a "message" and $d \oplus k$ as A keyed "checksum" of the message.

Passive attack. Without knowing the trapdoor (i.e., the punctured positions in the product), the attacker can follow two brute force strategies. First, he can exhaustively try all the possible subsets of the primes by trying to match it to the punctured product (n choose $n/2$). The other way is an exhaustive search for the punctured bits (positions and values, 2^t times $s \cdot n/2$ choose t) and checking by integer divisions. Because t must be selected small, this second search would be favored by the attacker. Therefore, we guess that the security depends on the feasibility of performing around $2^t \cdot (s \cdot n/2)^t$ divisions (dividing an $s \cdot n/2$ bit integer with an s-bit integer). For the above instance and for $t = 4$, when the output of κ is $96 - 4 = 92$ bits and the attacker can break the protocol (find the keys) on the cost of around 10^9 divisions, This number is small, and cannot be increased without increasing the load of the tag (see also the strengthening below).

Active attack. The attacker produces the second part of the challenge $\kappa(x, d)$. However, without knowing key k, he can send the first part only at random. The tag starts running its algorithm, which will fail to produce the challenge value x, because the two parts of the challenging message is inconsistent. This way, the tag is able to detect the attack. Therefore, such an attack falls in the category of DoS attacks.

Strengthening. Parameter t can be increased if the t-bit long string z of the punctured bits is appended to the trapdoor and sent also to the tag. Let the new, lengthened trapdoor be $d' = (d, z \circ g(d))$ where function g compresses d into t-bits. Key k is lengthened by t bits. Selection of mapping g and operation $\circ$ affect the strength of the protocol.

5 Conclusion and future work

In this paper, we proposed a set of extremely lightweight authentication protocols for low-cost RFID tags, and we also provided an analysis of the proposed protocols.

In general, the ultimate goal of the protocol designer is to design a protocol that has provable security under a given attacker model. Security proofs may be based on arguments from information theory (e.g., in case of one time pad), or on a reduction of the problem of breaking the protocol to a mathematical problem that is believed to be hard. Unfortunately, provable security of this kind comes with some cost: the one time pad has known key management problems, while protocols that are based on hard mathematical problems require large amount of resources even from the legal participants.

Another, more empirical design approach is to check if the protocol is resistant to the strongest known

attacks against similar kind of protocols. However, the designer may have inaccurate knowledge of the capabilities of the attacker. In order to alleviate this problem, the protocol is usually oversized in dimension (e.g., number of rounds in block ciphers) and complexity. This, however, is not desirable in low-cost RFID tags where tag resources are extremely scarce.

In this paper, we followed another approach. We built protocols from primitives that can surely be supported on low-cost RFID tags, and analyzed them, in order to see how resistant they are against various attacks. Clearly, our protocols can be broken by a powerful attacker; our goal was not to prevent this. Rather, we wanted to propose simple protocols that are amenable to analysis, and to give lower bounds on the complexity of attacking them. This allows the designer to adjust the security parameters of the system appropriately, and to find the best trade-off between security and performance. For instance, knowing a lower bound on the complexity of compromising the secret shared by the tag reader and a given tag would be useful in choosing the frequency of re-keying.

The work presented in this paper is an ongoing work. Important related problems, such as the issue of re-keying, will be addressed in future reports. We also intend to work towards a general framework in which the kind of protocols we propose here can be analyzed more systematically.

References

[1] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). Internet RFC 3174, September 2001.

[2] J. Hoffstein, J. Pipher, and J. Silverman. NTRU: A ring based public key cryptosystem. In ANTS III (LNCS no. 1423), pp. 267–288, 1998.

[3] A. Juels. Privacy and authentication in low-cost RFID tags. In submission. Available at http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/

[4] A. Juels and R. Pappu. Squealing Euros: Privacy protection in RFID-enabled banknotes. In *Proceedings of the 7th Financial Cryptography Conference*, 2003.

[5] R. Rivest. The MD5 message-digest algorithm. Internet RFC 1321, April 1992.

[6] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *CryptoBytes*, 5(Summer), 2002.

[7] S. Sarma, S. Weis, and D. Engels. Radio-frequency identifcation: Security risks and challenges. *CryptoBytes*, 6(1), 2003.

[8] B. Schneier. *Applied Cryptography*. John Wiley and Sons, 1996.

[9] J. Stern and J. Stern. Cryptanalysis of the OTM signature scheme from FC'02. In *Proceedings of the 7th Financial Cryptography Conference*, 2003.

[10] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *Proceedings of the 1st International Conference on Security in Pervasive Computing*, 2003.