Analysing the Molva and Di Pietro Private RFID Authentication Scheme

Mate Soos

INRIA Rhône-Alpes

Abstract. The private RFID authentication scheme by Molva and Di Pietro uses a non-cryptographic element for private identification. Through an exhaustive analysis of this element we have found multiple unintended features, flaws and attack vectors in the scheme. Based on this analysis we describe a set of design flaws and offer some remedies for them.

1 Introduction

RFIDs, also called tags, are small electronic devices that are currently used in the supply chain to monitor the movement of goods. They can be remotely read by an RFID-reader by use of radio waves, without line-of sight. RFIDs are mostly passive, that is, they are powered by the reader's energy field. This gives them a very long lifespan and also makes them very cheap. Cheapness and convenience are however problematic when it comes to privacy – nobody wants to be tracked cheaply and conveniently by anyone, especially by use of a device that is tiny and uses a mode of communication that cannot be sensed by human beings. Therefore, privacy is a huge concern for RFIDs. Once the privacy problem is solved, authentication is a natural next step: authenticating using a privacy-preserving tag would be extremely convenient: key-chains would be a thing of the past. These dreams are however difficult to realise due to the hardware constraints of the tags: since tags are tiny, cheap, and must consume very little power, they require innovative protocol designs to overcome these difficulties.

In this paper we examine a protocol by Refik Molva and Roberto Di Pietro [1] that tries to solve the problems mentioned above with an ambitious protocol. This protocol is part of a class of protocols we call non-cryptographic, that is, it relies on an innovative approach to replace a function that is normally provided by a standard cryptographic algorithm. In this class of protocols are for instance the HB$^+$[2], and LMAP [3]. These non-cryptographic protocols are notorious for being broken sooner or later and mended only to be broken again. This happened with HB$^+$ [4] which produced HB^{++} [5] which got broken [6] or with LMAP [7] which produced M2AP [8] which got broken [9].

Organisation

This paper is structured as follows. In Section 2 we present the protocol and in Section 3 we analyse one of its function's unintended behaviour. In Section 4

we analyse the private identification part of the protocol then in Section 5 we show a theoretical passive and an practical active attack against it. Finally, in Section 6 we present a list of design flaws and their remedies and in Section 7 we conclude our paper.

2 A short summary of the Molva - Di Pietro scheme

The scheme of Molva and Di Pietro [1] is a private RFID mutual authentication scheme. As such it solves three problems at the same time for RFID tags: it privately indicates the tag ID to the reader, it authenticates the tag to the reader and authenticates the reader to the tag. For clarity of explanation, we will make a clear distinction between these three parts.

There are n tags $\mathcal{T}_i$ in the system, each of which is configured with a unique key k_i which serves as the tag ID and the key at the same time. This key is used as a bitvector with $k_i[x]$ representing the x^{th} bit of k_i. Each reader also has a unique ID_j. At system initialisation, each reader $\mathcal{R}_j$ is configured with the reader-specific key of each tag, $k_{i,j} = h(k_i||ID_j||k_i)$, where $h()$ is a secure hash function available both on the tag and the reader.

Private identification

For private identification the protocol relies on the function DPM. The input to DPM is l bits where l is divisible by 3, and the output is one bit. DPM is defined as:

$$DPM(x) = \bigoplus_{i=0}^{l/3} M(x[3i], x[3i+1], x[3i+2]) \quad (1)$$

where M is the majority function: its input is 3 bits, and its output is one bit. M decides whether there are more `1`-s in its input is than `0`-s and returns the value `1` or `0` accordingly.

The private identification part of the protocol is as follows:

1. $\mathcal{R}_j$ sends ID_j to the tag
2. $\mathcal{T}_i$ computes $k_{i,j} = h(k_i||ID_j||k_i)$. It then generates q l-bit random nonces, r_p $(p = 1 \dots q)$. It then sends q α_p-s where $\alpha_p = r_p \oplus k_{i,j}$ and it sends a q-bit long vector V that is set up as $V[p] = DPM(r_p)$ to the reader
3. $\mathcal{R}_j$ computes $DPM(\alpha_p \oplus k_{i,j})$ for all keys $k_{i,j}$ it possesses and checks it against $V[p]$. This is called the *Lookup Process*. The key $k_{i,j}$ that fits on all pairs $(\alpha_p, V[p]), p = 1 \dots n$ is the tag suspected of sending the packets

At the end of the identification part, the reader suspects which tag it is talking to. The authors explain in detail how large q (the number of pairs sent) should be so that with very high probability only the correct key will fit on all pairs $(\alpha_p, V[p]), p = 1 \dots n$.

Tag authentication

During tag authentication the reader would like to make sure that it is indeed talking to tag $\mathcal{T}_i$. This is important since a malicious tag could simply replay an instance of the private identification part of the protocol to the same reader and the reader would not be able to differentiate between the two tags.

The tag authentication part of the protocol is as follows:

1. $\mathcal{R}_j$ sends a nonce n_j to the tag
2. $\mathcal{T}_i$ computes and sends $\omega = h(k_{i,j}||n_j||r_1||k_{i,j})$ to the reader
3. $\mathcal{R}_j$ computes $r_1 = \alpha_1 \oplus k_{i,j}$ and checks ω against $h(k_{i,j}||n_j||r_1||k_{i,j})$. If they match, the tag is authenticated

Reader authentication

After tag authentication the reader authenticates itself to the tag:

1. $\mathcal{R}_j$ computes $r_1 = \alpha_1 \oplus k_{i,j}$ and sends $h(k_{i,j}||r_1||k_{i,j})$ to the tag.
2. $\mathcal{T}_i$ computes $h(k_{i,j}||r_1||k_{i,j})$ and checks it against the received hash. If they match, the reader is authenticated

3 The DPM function

The function DPM is such that if an even number of majority functions' outputs are inverted, the output is not inverted. This property of the DPM function will have two unfortunate consequences: key- and pair-equivalences, which we will detail in this section.

3.1 Key equivalences

Let us divide the key $k_{i,j}$ into blocks of 3 bits, which we will simply call key *blocks* from now on. If an even number of key blocks are inverted, the resulting key will be indistinguishable by the reader from the original key using only the $(\alpha_p, V[p])$ pairs. This is because $V[p] = DPM(\alpha_p \oplus k_{i,j}) = DPM(\alpha_p \oplus k_{i,j} \oplus \text{inversions})$ and so the result of the Lookup Process will not depend on whether the blocks were inverted or not. One such pair of key-equivalents is $k_{i,j} =$`[001 000 100]`$\approx$`[110 000 011]`.

Key equivalences mean that any key of size l belongs to a key-equivalence group of size $\sum_{i=0}^{\lfloor (l/3)/2 \rfloor} \binom{l/3}{2i} = 2^{l/3-1}$, i.e. in a keyspace of 2^l there are $2^{l-l/3+1}$ key-equivalence groups (or key-eqgroups for short). Keys in a key-eqgroup are equivalent if the reader only looks at the $(\alpha_p, V[p])$ pairs. Naturally, if the reader checks ω, each of these tags is distinguishable from one another. However, as key sizes of at least 80 bits must be used to thwart brute-forcing of keys, the number of possibilities that could be left after the pairs are processed is $2^{81/3-1} \approx 7$ million, which is impossible for the reader to check, since executing a hash function this many times is too time-consuming for a quick identification session.

Under normal conditions, the keyspace (2^l) is extremely sparsely populated – there may be less tags in the whole system than the size of one such key-eqgroup. However, care must be taken to have a very low ratio of $n : 2^{2l/3+1}$, otherwise the hash $h(k_i||ID_j||k_i)$ could produce many keys that are in the same key-eqgroup for a certain reader (i.e. for a certain ID_j). This can be a problem, as there might be a time-limit for the reader to find which tag it is talking to within the key-eqgroup. We can calculate the chance that for a random reader ID_j there will be at least one key-eqgroup with more than one tag inside as $1-(1-n/2^{2l/3+1})^{n-1}$. For example, for $n = 10^7$, $l = 81$, this probability is 0.003, which means that if there are 1000 readers deployed, then there is a $1-(1-0.003)^{1000} \approx 95\%$ chance that at least one reader will have at least one key-eqgroup with more than one tag inside.

Key-eqgroups also mean that the attacker's keyspace is limited to $2^{l-l/3+1}$ if the attacker is only interested in the key-eqgroup (=anonymity group) the tag belongs to. If the attacker is interested in the exact tag, it can try to do $2^{l/3-1}$ hash operations to find $k_{i,j}$ using n, r_1 and ω of a session. This is feasible even for $l = 99$ and h =SHA-1: the key-eqgroup would be $2^{l/3-1} = 2^{32} = 4$ billion large and a Xeon quad-core can do about 4 million SHA-1 operations per second, so in less than half an hour would find the key $k_{i,j}$. Let us note that the hash function on the tag would be more simple than SHA-1 due to hardware constraints, and would be much easier to brute-force.

3.2 Pair-equivalences

The design of the DPM function also implies $(\alpha_p, V[p])$ pair-equivalences: for multiple different pairs the same keys are found not to fit (i.e. pruned) during the Lookup Process.

If $V[p] = DPM(\alpha_p \oplus k_{i,j})$ then for any α'_p that has an even number of blocks inverted, $V[p] = DPM(\alpha'_p \oplus k_{i,j})$ will also hold. Therefore the Lookup Process will prune the same keys for these pairs. For example `[010 001]-0`≈`[101 110]-0`

If an odd number of blocks are inverted in α_p then an odd number of blocks in r must have been inverted, so $V'[p] = DPM(r') = \overline{V[p]}$. Therefore, the Lookup Process will prune the same keys for these pairs as well. For example `[100 000]-1`≈`[011 000]-0`.

The property of pair-equivalences implies that the Lookup Process is not running at maximum efficiency since the possibility that two equivalent r-s are produced by the tag during identification is higher than with a DPM function that does not have this property.

3.3 The effect of equivalences

It is important to note that during the Lookup Process both pair- and key-equivalences are acting in tandem, and have two different effects: the first slows down the pruning of the keyspace ($k_{i,j}$-s stored in the reader), and the second does not let the pruning go beyond a certain point.

4 Private identification

In this section we examine the identification part of the protocol from multiple angles. First, we examine the Lemmas it depends on, then investigate the number of pairs needed for it, and finally we make some practical observations regarding its bandwidth need.

4.1 Observations about Lemma 2

In the original paper, Lemma 3 states that for a randomly chosen r, the chance that $DPM(r) = 1$ is 0.5, and the chance that $DPM(r) = 0$ is also 0.5. This means that given a set of tags and their unique random keys, a randomly chosen $(\alpha_p, V[p])$ pair will on average fit on half of the keys. Using this lemma, the authors conclude in Lemma 2 that given q randomly chosen pairs, the probability that at least one key will survive out of n random keys is less than $n(1/2)^q$. For this to hold, the distribution of surviving keys should have been random after one pair. However, the distribution of the surviving keys is not random: for example, two identical pairs will prune the keyspace only once.

4.2 The true number of $(\alpha_p, V[p])$ pairs needed

We will calculate δq, the additional number of $(\alpha_p, V[p])$ pairs needed for the reader to identify a key-eqgroup with a probability of at least 99% given that there are $n'(\leq n)$ key-eqgroups among the tags in the system.

Since the distribution of key-eqgroups is not random after one or more $(\alpha_p, V[p])$ pairs, we will need to investigate whether we need to compensate for this with extra pairs. For the moment, let us use Lemma 2 to calculate an estimate of the number of pairs needed. The conclusion of Lemma 2 is that for an incorrect identification probability $\epsilon \leq 2^{-r}$, the number of pairs sent, q, must be at least $r + \log(n')$ [1]. As an example, for the parameters $n' = 10^6$ and $\epsilon \leq 0.01$, q must be at least 27. We will now investigate how large δq, the additional number of pairs needed should be to compensate for the fallacy of Lemma 2.

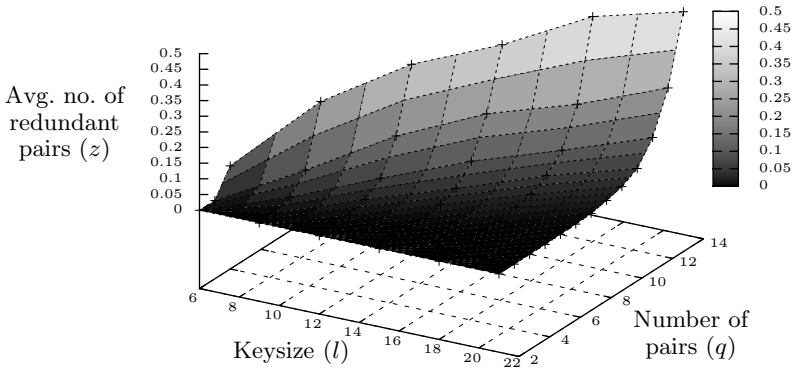
Redundant pairs We call redundant pairs a set of $(\alpha_p, V[p])$ pairs that are not equal in the sense of pair-equivalences, but as a set form a tautology: removing one or more pairs from the set will not reduce the information content of the set. One such set is for instance:

α_p	$V[p]$
011 100	0
011 111	1
010 100	0
010 111	1

[1] The authors meant log to be $\log_2$

In this set, given any 3 of the 4 pairs the 4th pair can be deduced. In other words, the information content of these 4 pairs is only 3 pairs. Theoretically determining the occurrence probabilities of redundant pairs is out of the scope of this paper. Instead, we ran some tests to observe what is the practical occurrence ratio of them for different keysizes and different number of non-equivalent pairs, which we present in Fig. 1.

Fig. 1. Average number of redundant pairs within q non-equivalent pairs for different keysizes. As the number of pairs increases, the occurrence rate of redundant pairs increases at an exponential rate



Using a logarithmic (ln) scale for the axis z, the equation $9.62l - 16.11q + 24z + 42.18 = 0$ describes the plane. Substituting $q = 27$ and $l = 81$ into this equation yields $z = -16.1$, i.e. the occurrence rate for these parameters is $e^{-16.1} \approx 10^{-7}$. This is so small that it does not need to be compensated with δq. However, for larger tag populations, the occurrence rate can be very high. For example, for 10^9 tags and consequently $q = 40$ the occurrence rate jumps to 10^{-4}, which cannot be ignored and must be compensated with a δq strictly larger than zero.

Pair-equivalents during identification The chance that among q random $(\alpha_p, V[p])$ pairs there will be at least one that is a pair-equivalent is as follows. There are $2^{2l/3}$ pair-eqgroups, so the chance that at least two pairs will be from the same pair-eqgroup among q pairs is

$$P_{repeat} \leq \binom{q}{2} 2^{-2l/3} \tag{2}$$

For $l = 81$, $n = 10^6$, $\epsilon \leq 0.01$ and $q = 27$, $P_{repeat} \leq \binom{27}{2} 2^{-54} \approx 2 \cdot 10^{-12}$. This probability is so small, that practically this will never occur and so it does not need to be compensated for with additional pairs.

4.3 The bandwidth needed in a common setup

Let us measure the total bandwidth cost of the protocol: for an even l-bit security goal, we need to set $|ID| = l$, $|n| = l$, $|\omega| = 2l$, and $|h(\cdot)| = 2l$ where $|x|$ means the bitlength of x. The protocol will use l bits to send ID_j, ql bits for the α_p-s, q bits for V, and $l + 2l + 2l$ bits for the two-way authentication. The total bandwidth cost is thus

$$B = l + ql + q + l + 2l + 2l = (6 + q)l + q \text{ bits} \tag{3}$$

In the case of $l = 81$ (which the authors seem to suggest through the use of a 160-bit hash function), $n' = 10^6$, $P_{find} \approx 0.99$, B is 2667 bits, which can be thought of as prohibitively large.

4.4 Implementation of the Lookup Process

Although the original paper mentions the processing overhead of the reader and concludes that it is $O(n \log n)$, we investigated the constant hiding behind the big O by implementing the Lookup Process on a Xeon E5345@2.33GHz computer. To speed up the calculations, we used all practical optimisations and programming techniques available to us, such as pure binary operations, loop-unrolling and memory bandwidth minimisation. The results are shown in Table 1.

Table 1. This table shows the average time and RAM required by the Lookup Process to find one tag. The Lookup Process was running on a Xeon E5345@2.33GHz with all optimisations other than assembly-level coding. P_{find} was set to 0.99 and keylength was 81 bits. As the number of tags in the system increases, the time it takes to identify the tag increases in an almost linear manner. Since memory usage mainly consists of storing tag keys, it increases linearly with the number of tags in the system

Number of tags	10^6	10^7	10^8
Time (s)	0.1	1.1	12
Memory (MB)	9.6	96	965

It is clear from Table 1 that even if the number of tags in the system is only 10^6, the reader would need to be very powerful – a hand-held reader rarely has the speed of a 2.33GHz Xeon processor. For larger tag populations, the RAM requirement would also become a problem. Therefore, it is more pertinent to speak about backend systems that process all incoming identifications and return the tag ID in batch mode. However, if backend systems must be used, then per-reader IDs are not a possibility and per-backend IDs must be used. Thus information confinement – one of the main goals of the paper – cannot be fully achieved.

5 Retrieving $k_{i,j}$

In this section we present two attacks that find $k_{i,j}$: a simple passive brute-force attack and a fast and efficient man in the middle attack. Since the key of the tag is always masked with the ID_j of the reader through $k_{i,j} = h(k_i||ID_j||k_i)$, the attacker will only be able to break the privacy of the tag when the tag is communicating with the same reader. Since both tag-to-reader and reader-to-tag authentication only requires the knowledge of $k_{i,j}$, retrieving it allows the attacker to both authenticate himself to the original reader as a legitimate tag, and to authenticate himself to the original tag as a legitimate reader.

5.1 Brute-force passive attack

The authors strangely forget to mention, indeed they might have overlooked, the simple brute-force passive attack against the identification part of the protocol, an attack vector that all schemes must exhibit that are not information theoretically secure. Naturally, the identification part of the protocol cannot be information theoretically secure since it sends a secure message (the ID of the tag) possibly infinite number of times while sharing just a few bits of secret information with the reader.

The attack simply executes the Lookup Process with all possible $2^{2l/3+1}$ key-eqgroups to find the key-eqgroup of $k_{i,j}$. Given different non-redundant $(\alpha_p, V[p])$ pairs, the number of possible keys is reduced by factor of 2 by each pair. Using the same formula as in Section 4 and setting $r = 0$, $n' = 2^{2l/3+1}$ we find that we need $2l/3 + 1$ non-equivalent non-redundant pairs to mount the attack.

We have implemented the attack and found that it performs as detailed in Table 2 on a Xeon E5345@2.33GHz CPU. The attack simply tries all key-eqgroups on the minimum amount of $(\alpha_p, V[p])$ pairs needed. Due to the time required to execute the algorithm for large keysizes, it can only be used to brute-force a key that the attacker has some information about. Using some supporting information, the brute-force attack can be used to fill out the gaps in (i.e. compute the unknown parts of) the key.

Table 2. This table shows the performance of our brute-force attack implementation on a Xeon E5345@2.33GHz CPU. As the key size increases, the time required to break the privacy of the tag increases at an exponential rate

Keysize (bits)	27	30	33	36	39	42	45
Time	0.38s	2.9s	27.2s	209s	1462s	13003s	76738s

Once the key-eqgroup of $k_{i,j}$ is found, the attacker can simply try to execute the hash function $h()$ implemented in the tag to try each of the $2^{l/3-1}$ combinations left against an ω response to find the exact $k_{i,j}$. As discussed in Section 3.1 this should not be difficult even for $l = 99$ and $h =$SHA-1.

5.2 Man-in-the-middle attack

The man-in-the-middle (or MiM for short) attack gains information about $k_{i,j}$ based on the success or failure of the authentication. Since success or failure can be represented in one bit, at each authentication attempt the attacker will learn exactly one bit of information. The attack exploits that with the exception of α_1, none of the α_p-s are authenticated: if an attacker modifies α_2 into α'_2 and the Lookup Process still finds the key $k_{i,j}$, then $DPM(k_{i,j} \oplus \alpha'_p) = V[p]$, so he either managed not to invert the output of any of the majority functions (M-s) in the DPM, or he managed to invert of an even number of them. However, if $DPM(k_{i,j} \oplus \alpha'_p) = \overline{V[p]}$, the authentication will fail since the reader will fail to find the key $k_{i,j}$ during the Lookup Process, in which case he can be sure that he must have inverted the output of an impair number of majority functions in the DPM.

The attacker will take the simplest approach to modifying α_2: he will try to invert one majority function's output. As a simple example, let us consider the block $\alpha_2[x \dots x+2] = \texttt{000}$. In this case $M(k_{i,j}[x \dots x+2] \oplus \alpha_2[x \dots x+2]) = M(k_{i,j}[x \dots x+2])$, so this block's M will depend solely on the key bits $k_{i,j}[x \dots x+2]$. Let us now invert the x+2nd bit of α_2: M will not change if and only if $k_{i,j}[x] = k_{i,j}[x+1]$, since then no matter what $k_{i,j}[x+2]$ is, $M = k_{i,j}[x] = k_{i,j}[x+1]$. However, M will change if $k_{i,j}[x] \neq k_{i,j}[x+1]$ since then the majority is decided by $k_{i,j}[x+2] \oplus \alpha_2[x+2]$, which the attacker just inverted. Therefore, by inverting the x+2nd bit of α_2, and observing the authentication, the attacker can conclude whether $k_{i,j}[x] = k_{i,j}[x+1]$ or not.

Reasoning this way, all possible α_2 blocks will lead to a conclusion: see Table 3 for all the conclusions that can be drawn given any α_2 block and an inversion at either the 2nd or the 3rd bit of the block. From the attack's point of view a block in α_2 behaves the same as the inversion of the same block (i.e. $\texttt{000} \approx \texttt{111}$), so only one of the two is listed.

Using Table 3 the attacker only needs two authentication sessions per key block to narrow down the possible key-combinations to $2^{l/3}$. At this point, he will have two possibilities for each key block. It is sufficient for the attacker to simply try the recorded set of $(\alpha_p, V[p])$ pairs on one of the $2^{l/3}$ combinations. If at least one pair does not match, then he simply needs to invert the first block to recover the key-eqgroup of the tag. The whole process thus takes 2 authentication sessions per key block, and less than one millisecond of processing. For $l = 81$ this means the privacy can be broken to a key-equivalence level in a mere 54 protocol sessions.

The MiM attack can be used in tandem with the brute-force attack. If the attacker is willing to invest some time into breaking the scheme, he can use the MiM attack to learn some information about the first x bits of the key, and then use the brute-force attack to learn the rest of the $l - x$ bits. As long as $l - x$ is less than 39, this should take very little time for the attacker, and would let him use less active rounds: in the case of $l = 81$, instead of the normal 54 active rounds needed, he would only need 30 active rounds and a couple of minutes to find the key-eqgroup of $k_{i,j}$.

Table 3. This table shows the conclusions that can be drawn by actively modifying an ongoing protocol session: the attacker needs to invert one bit of α_2 at a block's 2nd or 3rd position and observe the outcome of the protocol. If the tag does not get accepted as authentic, then he can deduce the information that is present in the row marked with $\times$, if the tag does get accepted he can deduce the information that is present in the row marked with ✓

Inverted bit	Auth	Original $\alpha_2[x \dots x+2]$ block `000`	`001`
$\alpha_2[x+2]$	✓	$k_{i,j}[x] = k_{i,j}[x+1]$	$k_{i,j}[x] = k_{i,j}[x+1]$
$\alpha_2[x+2]$	$\times$	$k_{i,j}[x] \neq k_{i,j}[x+1]$	$k_{i,j}[x] \neq k_{i,j}[x+1]$
$\alpha_2[x+1]$	✓	$k_{i,j}[x] = k_{i,j}[x+2]$	$k_{i,j}[x] \neq k_{i,j}[x+2]$
$\alpha_2[x+1]$	$\times$	$k_{i,j}[x] \neq k_{i,j}[x+2]$	$k_{i,j}[x] = k_{i,j}[x+2]$
		Original $\alpha_2[x \dots x+2]$ `010`	`100`
$\alpha_2[x+2]$	✓	$k_{i,j}[x] \neq k_{i,j}[x+1]$	$k_{i,j}[x] \neq k_{i,j}[x+1]$
$\alpha_2[x+2]$	$\times$	$k_{i,j}[x] = k_{i,j}[x+1]$	$k_{i,j}[x] = k_{i,j}[x+1]$
$\alpha_2[x+1]$	✓	$k_{i,j}[x] = k_{i,j}[x+2]$	$k_{i,j}[x] \neq k_{i,j}[x+2]$
$\alpha_2[x+1]$	$\times$	$k_{i,j}[x] \neq k_{i,j}[x+2]$	$k_{i,j}[x] = k_{i,j}[x+2]$

To find the exact $k_{i,j}$ of the tag, the attacker would need to perform the same actions as in the last stage of the brute-force passive attack, i.e. execute the hash function $h()$ for each of the $2^{l/3-1}$ remaining combinations and compare it against an ω response. As discussed in Section 3.1 this should not be difficult even for $l = 99$ and h =SHA-1 – it would take less than an hour on a desktop PC.

6 Design flaws and their remedies

It is very difficult to design security protocols for RFIDs since the resources available on this platform are extremely limited. Some security properties must always be sacrificed in order to fit the security functions on the tag. However, we believe that the scheme being analysed is not only imperfect due to the limitations of the platform, but it also exhibits limitations that are due to design flaws. In this section we will list a set of design flaws exhibited by the scheme and then propose some modifications to overcome the problems detailed.

6.1 Design flaws

We have found the following list of design flaws during our analysis of the protocol:

– Identification and authentication boundaries should have been clearly defined. Had identification and authentication been designed and analysed independently, many of the shortcomings described could have been averted

– Identification and authentication keys should have been generated differently. Had this been the case, the attacks presented would only have recovered the identification key and so would have been restricted to breaking the privacy. A simple difference between the generation of the two $k_{i,j}$-s would have been enough
– Given that the identification was not cryptographically secured, the integrity of the data exchanged during identification should have been authenticated during authentication. It is clear that the identification was not cryptographically secured since it only used a xor and a majority function to do its work
– The choice of the DPM function is not clearly motivated and its design is not analysed in a separate paragraph. Such a crucial function of a scheme should have been fully analysed

6.2 Remedies for the problems found

We believe that so-called "lightweight cryptography" as in home-brewn cryptography-like functions is not the way forward. A function used in RFID protocols should either be a well-analysed cryptographic function like AES, DES, the upcoming Grain etc. or it should not attempt to be a cryptographic function at all, instead it should be a standard hard problem whose attack vector is well-known and its hardness can be tested accordingly. In the case of crypto-functions, the attacks on the chosen well-known crypto-functions apply. In the case of standard hard problems, the standard well-known attack vectors apply. In both cases a hard limit can be calculated regarding the speed of breaking. Naturally, we expect the solution based on standard hard problems to be less secure, but let us remember that security is a relative term and in case of RFIDs it might mean that an RFID tag's certain property cannot be broken within its lifetime, which may be as short as one day.

Given our reasoning above, there are two possible ways of mending the protocol. One is to use a standard cryptographic primitive such as hash-chains for private identification as in [10], or to use a standard encryption function and key-trees as in [11]. The other way of mending the protocol is to use a standard hard problem such as HB$^+$'s Learning Parity with Noise [2].

7 Conclusions

During our analysis of the Molva-Di Pietro scheme we have uncovered multiple flaws, among them tag identification ambiguity, possibility of active attack and unanticipated processing slowness. We have fully analysed the scheme from multiple viewpoints and have found a multitude of obscure features such as pair-equivalences. We presented a list of detailed design flaws that we have found during our analysis and finally, we have given a set of improvement ideas.

References

1. Pietro, R.D., Molva, R.: Information confinement, privacy, and security in RFID systems. In: Proceedings of the 12th European Symposium On Research In Computer Security. (September 2007) 187–202
2. Juels, A., Weis, S.: Authenticating pervasive devices with human protocols. In Shoup, V., ed.: CRYPTO'05. Volume 3126 of LNCS., IACR (August 2005) 293–308
3. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J., Ribagorda, A.: LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags. Proceedings of RFIDSec'06 (July 2006)
4. Gilbert, H., Robshaw, M., Sibert, H.: An active attack against HB+ - a provably secure lightweight authentication protocol. In: IEE Electronic Letters 41, 21. (2005) 1169–1170
5. Bringer, J., Chabanne, H., Emmanuelle, D.: HB^{++}: a lightweight authentication protocol secure against some attacks. In: IEEE SecPerU 2006, IEEE (June 2006)
6. Gilbert, H., Robshaw, M.J., Seurin, Y.: Good variants of HB+ are hard to find. In: Financial Cryptography, Springer (January 2008)
7. Bárasz, M., Boros, B., Ligeti, P., Lója, K., Nagy, D.: Breaking LMAP. In: RFIDSec'07. (July 2007) 69–78
8. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J., Ribagorda, A.: M2AP: A minimalist mutual-authentication protocol for low-cost RFID tags. In: UIC06. Volume 4159. (September 2006) 912–923
9. Bárász, M., Boros, B., Ligeti, P., Lója, K., Nagy, D.A.: Passive attack against the M2AP mutual authentication protocol for RFID tags. In: RFID 2007 – The First International EURASIP Workshop on RFID Technology. (September 2007)
10. Ohkubo, M., Suzuki, K., Kinoshita, S.: Efficient hash-chain based RFID privacy protection scheme. In: Ubicomp 2004, Workshop Privacy: Current Status and Future Directions. (September 2004)
11. Buttyán, L., Holczer, T., Vajda, I.: Optimal key-trees for tree-based private authentication. In: PET 2006. (June 2007) 332–350