Secret Shuffling: A Novel Approach to RFID Private Identification

Claude Castelluccia and Mate Soos

INRIA, 655 avenue de l'Europe, Montbonnot, France
{claude.castelluccia, mate.soos}@inrialpes.fr

Abstract. This paper considers the problem of private identification of very small and inexpensive tags. It describes a novel scheme that does not require any computation from the tag. The proposed scheme relies on an NP-complete problem and as such is proven to be difficult to breach. We show that our solution outperforms existing computation-free schemes such as the pseudonym-rotation scheme proposal by Juels et al.[1].

1 Introduction

An RFID (Radio-Frequency Identification) tag is an extremely small electronic device that can – within a short range – wirelessly communicate with a reader. There are various types of RFID tags, ranging from very powerful to very weak devices. This paper focuses on tags with very limited computation capabilities, such as EPC tags. These devices are powered by the reader's electromagnetic field, and so need no battery and subsequently no recharging. EPC RFID tags carry interesting possibilities for the end users: they could be used to return faulty items to shops without keeping receipts, or even help intelligent washing machines that know what kind of clothes are inside them. However, with these possibilities comes a price: the possibler loss of privacy. For example, anybody possessing a reader could read any passersby's tags, which can potentially reveal even the brand of his or her socks. Similarly, tracking of people would also become possible. These possibilities scare off potential adoption as was the case with the boycott of Benetton where the garment maker was forced to take off RFID tags from their clothes.

Contributions This paper considers the problem of private identification of very small and inexpensive tags that cannot perform any cryptographic operations. Our proposal is a probabilistic identification protocol (ProbIP) that does not require any computation from the tag. Our scheme resembles Juels' pseudonym-rotation scheme as presented in [1], but increases its security significantly. The presented scheme is an identification scheme. As such, it does not address authentication, and so can not be used to authenticate a tag. It simply serves to correctly identify a tag if no active attacker is present. Privacy of the tag is preserved to some extent even if an active attacker is present.

Organization This paper is structured as follows: Section 2 presents briefly the related work. Section 3 describes our identification protocol and Section 4 provides a security analysis of our protocol.

2 Related work

Existing solutions to the RFID private identification problem can be categorized as follows: hash-lock based systems, solutions based on special tags and ultra-lightweight crypto-primitives.

Hash-lock based systems have been studied deeply, interesting papers in this category include a tree-type approach from Molnar et al. [2], an optimization of key-trees by Buttyan et al. [3], a synchronization-type approach from Ohkubo et al. [4] and a mixed approach from Lu, Han et al. [5]. Although these schemes offer relatively good security, they all suffer from the same problem: the need of a secure one-way hash function on the tag.

Some solutions use special tags, that usually have a relatively good processing power, to supervise and control all communication between the regular RFID tags and the reader. The RFID blocker tag by Juels, Rivest and Szydlo in [6] is an example of such a solution. This avenue of research has the advantage of providing very strong privacy but requires that an intelligent device be present at all times when a tag is being queried.

Ultra-lightweight crypto-primitives are an interesting avenue in RFID security research. In this category are papers such as Vajda and Buttyan's paper [7] that has been studied by Li et al. in [8], and a tiny implementation of AES by Feldhofer et al. in [9]. Also in this category, is the paper that gave us the most inspiration, written by Juels and Weis [10], that introduced HB^+, a novel lightweight authentication protocol. We believe this avenue of research has the potential to provide the best solution to the proposed problems.

3 Probabilistic Identification Protocol (ProbIP)

In this section, we introduce our Probabilistic Identification Protocol (ProbIP). In ProbIP, each tag $\mathcal{T}_j$ is configured with a unique K-bit long random secret key, k_j. The key is used as a bit-vector, with $k_j[1]$ being the first bit, $k_j[2]$ being the second, etc. The reader, $\mathcal{R}$, stores all the keys that are assigned to each of the n tags.

3.1 Protocol description

The protocol, between tag $\mathcal{T}_j$, and the reader $\mathcal{R}$, is as follows:

1. $\mathcal{R}$ initiates an identification by broadcasting a `HELLO` message.
2. Upon reception of a `HELLO` message, $\mathcal{T}_j$ replies with P *packets* and a `FINISHED` message, where P is a system parameter that will be defined in the following section. A *packet* is a list of $2L$ values, $a_1, b_1, a_2, b_2 \ldots, a_L, b_L$, where a_i is a

random index from the key $a_i \xleftarrow{r} [1, K]$ that is never repeated in the same packet, and b_i is a random bit $b_i \xleftarrow{r} \{0, 1\}$ that satisfy the following equation:

$$\sum_{i=1}^{L} k_j[a_i] \oplus b_i = L/2 \tag{1}$$

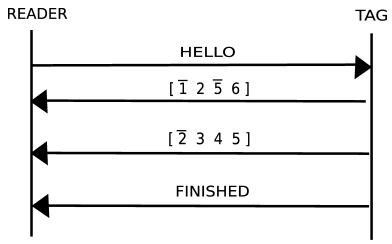
Since addition is commutative, as long as the pairs a_i, b_i for all i are not changed, the order of the pairs can change. We will note these pairs in the following fashion: $\bar{a}_i$ if $b_i = 1$ and a_i if $b_i = 0$.

3. Upon reception, $\mathcal{R}$ computes the result of eq. (1) for each packet for every tag's key in a relatively fast fashion. The key(s) that fits all the packets is suspected to have been used to send the packets.

3.2 An example

Let's consider, to illustrate our protocol, a system that uses the following artificially small system parameters, $L = 4$, $K = 6$ and $n = 4$. In this example, $\mathcal{T}_1$ is configured with the key $k_1 =$`011001`, $\mathcal{T}_2$ with the key $k_2 =$`100101`, $\mathcal{T}_3$ with the key $k_3 =$`011110` and finally $\mathcal{T}_4$ with $k_4 =$`001110`.

Let's assume that the tag the reader is trying to identify is $\mathcal{T}_2$. An example protocol run between $\mathcal{R}$ and $\mathcal{T}_2$ is the following:



In a step-by-step fashion, the following happens during this protocol run:

1. $\mathcal{R}$ broadcasts a `HELLO` message.
2. Tag $\mathcal{T}_2$ sends two packets and the `FINISHED` message. The first packet is defined by `[1̄ 2 5̄ 6]`, for which the eq. (1) wrt. k_2 is $(1 \oplus 1) + (0 \oplus 0) + (0 \oplus 1) + (1 \oplus 0) = 2 = L/2$. The second packet is defined by `[2̄ 3 4 5]` for which the eq. (1) wrt. k_2 is $(0 \oplus 1) + (0 \oplus 0) + (1 \oplus 0) + (0 \oplus 0) = 2 = L/2$.
3. Upon reception of the first packet, the reader computes for each of the 4 tags the eq. (1). $\mathcal{R}$ gets that for $\mathcal{T}_1$ it is 4, for $\mathcal{T}_2$ it is 2, for $\mathcal{T}_3$ it is 2 and for $\mathcal{T}_4$ it is 1. The reader, therefore, keeps only tags $\mathcal{T}_2$ and $\mathcal{T}_3$ as possible candidates.
4. Upon reception of the second packet, the reader computes for tags $\mathcal{T}_2$ and $\mathcal{T}_3$ the eq. (1). $\mathcal{R}$ gets that for $\mathcal{T}_2$ it is 2 and for $\mathcal{T}_3$ it is 3. At this point, tag $\mathcal{T}_2$ has been successfully identified by $\mathcal{R}$.

3.3 Minimum number of packets needed by the reader

Here, we compute the minimum amount of packets needed by $\mathcal{R}$ to correctly identify a tag. Since the protocol is probabilistic, there will always be a non-zero probability fp that the number of packets sent will not be enough. However, this probability can be arbitrary adjusted between $0 < fp < 1$.

The total number of packets possible for *all* keys is $\binom{2K}{L}$, as a_i comes from a set of size K and b_i comes from a set of size 2, whereas for a *given* key, the number of possible packets is only $\binom{K}{L/2}\binom{K-L/2}{L/2}$ since eq. (1) must hold and indices cannot be repeated in a packet. The ratio of these two numbers

$$R = \frac{\binom{K}{L/2}\binom{K-L/2}{L/2}}{\binom{2K}{L}} \tag{2}$$

is the probability that a random packet is valid for a random tag. As an example, for $K = 400, L = 10$, $R \approx 0.232$.

Given n tags, the false positive probability, fp, that p packets generated by a given tag match another tag's key can be calculated as $fp = n * R^p$. The number of packets sent from the tag to the reader should then be

$$P = \left\lceil \frac{\log(1/n * fp)}{\log(R)} \right\rceil \tag{3}$$

which is, for the example parameters of $L = 10$, $fp = 0.1$ and $n = 10^7$, $P = \lceil 12.62 \rceil = 13$. If these packets do not suffice (which has a low chance of happening), repeated identification attempts will be carried out by the reader until it finds the correct tag.

3.4 Parameters

The parameter K must be at least $\lceil \log_2(n) \rceil$ bits, but as the security of the system will rely on the condition that $n \ll 2^K$, the larger this parameter is, the more secure the system. Also, K should be at least an order of magnitude larger than L. The parameter L must be such that $L/2$ is a whole number. When deciding the parameters, the number of bits sent in one identification

$$B = P * L * (\lceil \log_2(K) \rceil + 1) \tag{4}$$

which is also the minimum amount of random bits that need to be generated during an identification, must be kept in mind. The parameters L, K and n all influence this number. As an example, for $K = 400, L = 10$ and $n = 10^7$, $B = 1300$ bits. It is important to note that sending this information is just a fraction of a second given a 52.969 kb/s label-to-interrogator link in Class 1 EPC tags [11].

3.5 Algorithm used by the reader

This section describes the algorithm used by the reader or a set of back-end servers, to identify the tag using the packets it sent. It is assumed that $\mathcal{R}$ knows the keys $k_1 \dots k_n$ of all the tags in the system. These keys are stored not in their natural order $k_j[1], k_j[2], \dots k_j[K]$ for all $\mathcal{T}_j \in \{\mathcal{T}_1, \dots, \mathcal{T}_n\}$, but in their column-like order $k_1[i], k_2[i], \dots k_n[i]$ for all $i \in [1, K]$. These K columns we will call $Col_1, \dots, Col_K$. Naturally, storing these columns needs exactly the same amount of memory as simply storing the keys, i.e. $n * K$ bits.

At each protocol instance, the following is executed by $\mathcal{R}$:

1. $\mathcal{R}$ fills with zeros a temporary n-long byte-vector *temp*. This will store the result of the eq. (1), for each tag. A byte is sufficient for any $L < 1.5 * 256$.
2. Upon reception of a packet, $\mathcal{R}$ performs the following algorithm for each of the packets' L pairs (a_i, b_i): For each tag $\mathcal{T}_j$ in the system, $temp[j]$ is incremented by one if $Col_{a_i}[j] \oplus b_i = 1$. Iteration through the *temp* and the Col_{a_i} can be parallel, so for a given index, on average $n + 8n$ bits of memory need to be read and $8n/2$ bits of memory written.
3. Once all pairs in the packet have been considered, all tags $\mathcal{T}_j$ for which $temp[j] = L/2$ could have sent the packet.
4. Steps 2-3 are repeated for all packets with different *temp*s, i.e. $temp_1$ for packet no. 1, $temp_2$ for packet no 2, etc.
5. The identified tag is the tag $\mathcal{T}_j$ for which $temp_i[j] = L/2$ for all $i \in [1, P]$.

The proposed algorithm identifies a tag by looking through, on average, $P * L * (n + 8n)$ bits of memory and, by writing $P * L * 8n/2$ bits of memory space, while doing $L * n$ comparisons and $L * n/2$ incrementation per packet, plus $P * n$ comparisons for evaluating all the packets' results in step 5, which gives $P * L * (n + n/2) + P * n$ processing steps in total. For instance, if $L = 10, K = 400, n = 10^7$ then the overall memory requirement is $400 * 10^7 + 13 * 8 * 10^7$ bits= 630MB and the overall processing requirement is $13 * 10 * (1.5 * 10^7) + 13 * 10^7 \approx 2.08e9$ processing steps. Parallelization of this algorithm is relatively simple, and can bring down both the memory and processing requirement of individual computers.

Note that an adversary does not know the configured set of keys, and would need to run this algorithm with $n = 2^K$ and so $\approx 1.63e113$ GB of memory would be needed for the same parameters ($L = 10, K = 400$). The processing need would increase to similar proportions.

4 Security analysis of ProbIP

In this section we will evaluate the security of our scheme using the "strong privacy" model proposed by Juels and Weis in [12]. In our scheme, *tags' keys are completely independent of each other* thus the corruption of one tag does not affect the security of the rest of the system. Therefore, it is useless for adversary

$\mathcal{A}$ to use the the SetKey procedure to change the key of tags. It is also useless for $\mathcal{A}$ to examine any other tags than the ones it will pick, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$. In our scheme, ReaderInit is a simple fixed `HELLO` message, so it need not be executed by $\mathcal{A}$ at all. Therefore, in view of our protocol, the privacy experiment of the model can be refined to what is present in Fig. 1.

Experiment $\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}}[K, n, x_A + x_B + x_C]$:

Setup:

(1) Generate keys $(k_1, \ldots, k_n)$ uniquely and randomly with GenKey
(2) Initialize $\mathcal{R}$ with keys $(k_1, \ldots, k_n)$
(3) Set each $\mathcal{T}_i$'s key k_i with a SetKey call

Phase 1 (Learning):

(4) Let $\mathcal{A}$ perform x_A TagInit calls with $\mathcal{T}_A$ and let it record the received packets into the set X_A
(5) Let $\mathcal{A}$ perform x_B TagInit calls with $\mathcal{T}_B$ and let it record the received packets into the set X_B

Phase 2 (Challenge):

(6) Let $\mathcal{T}_C \xleftarrow{r} \{\mathcal{T}_A, \mathcal{T}_B\}$
(7) Let $\mathcal{A}$ perform x_C TagInit calls with $\mathcal{T}_C$ and let it record the received packets into the set X_C
(8) Let $\mathcal{A}$ perform calculations on the recorded packets in order to make an educated guess whether $\mathcal{T}_C = \mathcal{T}_A$ or $\mathcal{T}_C = \mathcal{T}_B$.

Exp succeeds if $\mathcal{A}$ guessed $\mathcal{T}_C$ correctly

Fig. 1. The privacy experiment as proposed by Juels and Weis in [12], refined to the specifics of our protocol

In order to find out how the experiment can be optimized by the adversary and how he should choose the parameters x_A, x_B and x_C, we will first analyze what a packet is. Then we propose an algorithm to perform the attack, and finally, we will experimentally show what is the resistance of our proposed protocol for certain parameter combinations.

4.1 A closer look at the packets

Looking at the packets in a more mathematical way, they describe an $L/2$-in-L LSAT problem. In $L/2$-in-L LSAT, like in LSAT, the input instance is a collection of clauses, where each clause consists of exactly L literals, where each literal is either a variable or its negation. The $L/2$-in-L LSAT problem is

to determine whether there exists a truth assignment to the variables so that each clause has exactly $L/2$ true literals. This problem is NP-complete if $L > 2$ as indicated by Schaefer's dichotomy theorem [13]. Thus, if an attacker, given enough packets, wants to solve for k, if $L > 2$, he would have to solve an NP-complete problem.

Similarly, a packet can also be looked at as an L-long Linear Pseudo-Boolean Constraint (LPBC) as the original eq. (1) implied.

4.2 Algorithm used by the attacker

In light of what the true nature of a packet is, the way for an attacker to attack our scheme in the given model is to execute an LPBC solver on $X_C \cup X_A$, and examine the output of the solver. If the result is unsatisfiable (UNSAT), then surely $\mathcal{T}_C \neq \mathcal{T}_A$ (consequently, $\mathcal{T}_C = \mathcal{T}_B$), since $\mathcal{T}_A$ only sends packets that have a solution, k_A. However, if the result is satisfiable (SAT), $\mathcal{T}_C$ can be either of the tags. But, if the attacker knows that he has gathered enough packets that had $\mathcal{T}_C \neq \mathcal{T}_A$ the result would surely have been UNSAT, then he knows that $\mathcal{T}_C = \mathcal{T}_A$. If he cannot gather this amount of packets in X_A and X_C, then he will almost always get the result SAT, which he cannot use. Therefore there is no need for him to gather packets from more than one of the tags in the learning phase, and it does not matter which one he picks. We decided to pick $\mathcal{T}_A$.

While keeping the above argument in mind, one could reason in the following way about NP-hard problems:

1. They are hard to solve exactly but can in certain cases be approximated easily. While this is true, it does not help the attacker in any way. If the attacker can find an approximate solution to the set of constraints defined by $X_A \cup X_B$, that does not tell anything about whether $\mathcal{T}_C = \mathcal{T}_A$ or $\mathcal{T}_C \neq \mathcal{T}_A$.
2. They are hard to solve in the worst case, but easy to solve most of their instances. This is true, and we will deal with this later, where we show that the problem generated by the tag is exactly what is suggested in [14] and which is known to be hard.

LPBC solver used We tried two of the most respected LPBC solvers, Minisat [15] and Toolbar [16], and a less known one, Galena [17]. Each one had its own distinct advantage. Galena was designed to solve exactly these kinds of problems, Toolbar had a large user-base and thus good support, while Minisat was the best performing in many SAT competitions. Minisat had multiple advantages over the competition: since it was a pure SAT solver, it could use the well-established DPLL algorithm as described by Davis, Putnam et al. in [18], and could also benefit from "learning" as first shown by Schulz et al. in [19], and improved by Marques Silva et al. in [20]. Galena adapted these techniques to solving LPBC problems, but it could not use the well-established literature of how to implement them in a way that is fast.

The fastest solver by far (and also, the most modifiable), was Minisat. It was the winner of multiple SAT-competitions in 2005 and came the first in the 2006 SAT-race, and so gave a good foundation to base our results on. One would expect that the speed slowdown of using the LPB constraints of the type in eq. (1) converted into regular CNF formulas is significant. However, as was shown by the version of Minisat called "Minisat+" this method is faster than directly tackling the original LPB-constraint problem: Minisat was much better than its rivals in the LPB-part of the SAT competition in 2006.

In order to use Minisat, we needed to convert each packet to a set of CNF clauses. We thus used a conversion that made from e.g the packet of `[1 2̄ 5 9]` the following CNF clauses:

$$(1 \vee \bar{2} \vee 5) \wedge (\bar{1} \vee 2 \vee \bar{5}) \wedge (\bar{2} \vee 5 \vee 9) \wedge (2 \vee \bar{5} \vee \bar{9}) \wedge \ldots$$

i.e. all $L/2 + 1$ combinations in their original and their negated form. These formulas simply mean that any $L/2 + 1$ literals that have the same value is forbidden. We used this conversion because the conversion algorithm included in Minisat+ converted the problem in such a way that it was slower than using the conversion given above.

Minimum number of packets needed by the attacker Let's assume that $\mathcal{T}_A \neq \mathcal{T}_C$. The attacker first needs to know how should he distribute the number of ReaderInit calls he has between $\mathcal{T}_A$ and $\mathcal{T}_C$ in order to have the highest possibility of finding out that $\mathcal{T}_A \neq \mathcal{T}_C$. We will now calculate this ratio and deduce equation for the minimum amount of packets needed.

x_A packets from $\mathcal{T}_A$ reduces the number of possible keys by a factor of R^{x_A}, the remaining set of possible keys we call S_A . x_C packets from $\mathcal{T}_C$ also reduces the number of possible keys by a factor of R^{x_C}, the remaining set of possible keys we call S_C . Intuitively, the union of these packets will leave the attacker with

$$PC = |S_A \cap S_C| = \max \begin{cases} \max(2^K * R^{x_A}, 1) * (2^K * R^{x_C}) \\ \max(2^K * R^{x_C}, 1) * (2^K * R^{x_A}) \end{cases} \quad (5)$$

number of possible key combinations. This equation tells two things. Firstly, the intersection is the smallest if $x_A = x_C$. Secondly, in case $\mathcal{T}_A \neq \mathcal{T}_C$, the closer PC is to 0, the higher the possibility of the attacker to find that the constraints $X_A \cup X_C$ is UNSAT. As an example, if $PC = 0.1$ then he has a 90% chance that he will find out that the resulting constraints are UNSAT (thus $\mathcal{T}_A \neq \mathcal{T}_C$) and if he finds that $X_A \cup X_C$ is SAT, then he can be 90% sure that indeed $\mathcal{T}_A = \mathcal{T}_C$.

Therefore, if the attacker wishes to attain a 90% chance ratio of finding out which tag is $\mathcal{T}_C$ then he needs

$$P_{att} = \left\lceil \frac{\log(1/2^K * 0.1)}{\log(R)} \right\rceil \quad (6)$$

packets equally distributed between x_A and x_B. As an example, if $K = 400$ and $L = 10$, then $P_{att} = \lceil 191.62 \rceil = 192$. If $n = 10^7$ and consequently, $P = 13$, this

is $\lceil P_{att}/P \rceil = 15$ identifications in total, i.e. 8 identifications for both $\mathcal{T}_A$ and $\mathcal{T}_C$. It is interesting to observe that the equations (6) and (3) completely match if $fp = 0.1$ and $n = 2^K$ i.e. the search space of the reader is not restricted to the configured set of keys.

The threshold phenomenon As it is hypothesized by Cheeseman et al. in [14] and further explained by B. Smith in [21], all NP-hard problems exhibit a so-called phase-transition, which states that given a randomly generated NP-hard CSP(Constraint Satisfaction Problem), there is always a point where it is the hardest to solve the generated problem, and this corresponds exactly to the point where there is a transition from SAT to UNSAT. From this point on, the difficulty of finding a solution decreases at an exponential rate, along with the possibility of having any solution at all.

This phenomenon plays a crucial part in the security evaluation of our protocol: as the possibility of UNSAT increases if $\mathcal{T}_A \neq \mathcal{T}_C$, the hardness of finding this out increases as well. The peak of the difficulty of finding out UNSAT is then at the minimum amount of packets the attacker needs (i.e near P_{att}) to find that the CPS defined by the packets is indeed UNSAT. If more packets are known by the attacker, this difficulty decreases at an exponential rate in relation to the extra number of packets gathered.

Precisely calculating the phase-transition and its corresponding graph is very hard for a given CSP: even for such a widely studied problem as k-SAT, since its inception in 1991 with the influential paper of Cheeseman et al. [14] it has taken more than 10 years to prove that the threshold for k-SAT is $2^k \log 2 - O(k)$ [22]. So, instead of mathematically calculating the threshold and its corresponding graph, we will experimentally show it using Minisat, and extrapolating the results, deduce the protocol's resistance to attacks.

An example threshold plot is present in Fig. 2. In this particular instance of the protocol the parameters were such that $P_{att} \approx 36$ and $\mathcal{T}_A = \mathcal{T}_C$ so that the Hamming distance of solution given by the satisfiability solver approached 0 as the number of packets given to it approached P_{att}. The phase-transition is clearly between no. of packets 30 and 36.

4.3 Results

In calculating the resistance to attacks, we used the parameter L =10. Knowing that it is hardest to solve at P_{att}, we gave our simulated attacker multiple number of P_{att} packets to evaluate the computation speed-up it can achieve if given more packets. All experiments were performed with a 3GHz Pentium-D machine with 2 GB of memory. The plot for 64, 192, 576 times P_{att} is in Fig. 3.

The scale on the time axis is logarithmic, since the time to break the system increases exponentially as the key-length is increased. This is a consequence of using an NP-hard problem to base the security of the protocol on. The breaking times for different values of P_{att}, n and K are shown in Table 1.

The results clearly show that there is a trade-off between the number of packets collected and the hardness of breaking the system privacy (i.e. winning

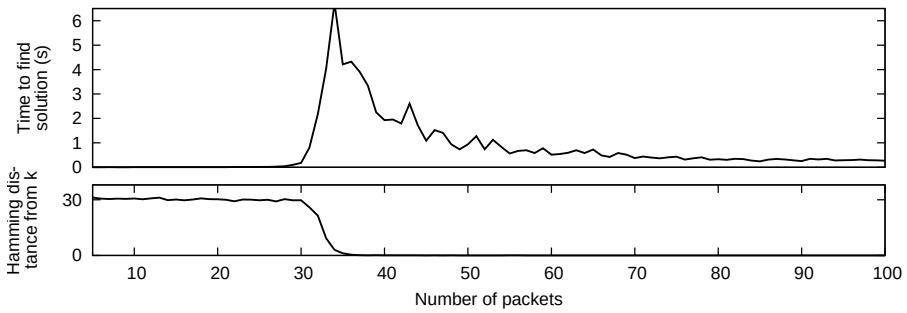


Fig. 2. The threshold phenomenon illustrated. Top part of the plot shows the time to find a solution versus information given, the bottom part shows the Hamming distance of the solution found from the key of the tag

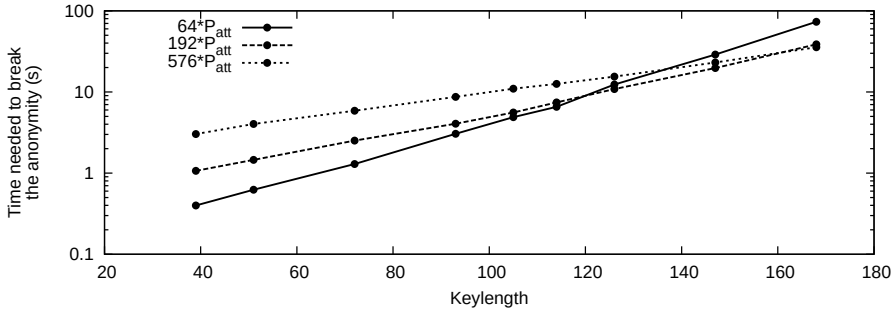


Fig. 3. Time to break the anonymity versus the keylength for different $overinfo$ rates (i.e. the number of times P_{att} packets was given to the attacker)

the privacy experiment). The more packets an attacker can collect, the easier it is for him to break the system.

4.4 Analysis

The results clearly show that there is a trade-off between the number of packets collected and the hardness of breaking the system privacy (i.e. winning the privacy experiment). The more packets an attacker can collect, the easier it is for him to break the system. This interesting property is a direct result of the threshold phenomenon.

An interesting property of our scheme is that n is not directly present anywhere in the results. This is because the only help a greater n gives to the attacker is that the number of packets per identification, P, will be greater, and so P_{att} number of packets will be attainable by less TagInit queries.

packets/K	100	200	400	1000
$1*P_{att}$	1.47e2 s	3.17e11 s	1.46e28 s	1.46e78 s
$3*P_{att}$	3.33e1 s	7.41e5 s	3.67e14 s	4.49e40 s
$9*P_{att}$	6.31e0 s	4.54e3 s	2.35e9 s	3.27e26 s
$27*P_{att}$	4.27e0 s	6.37e2 s	1.42e7 s	1.57e20 s
$64*P_{att}$	4.02e0 s	4.87e2 s	7.15e6 s	2.27e19 s
$192*P_{att}$	5.34e0 s	7.31e1 s	1.37e4 s	9.01e10 s
$576*P_{att}$	1.00e1 s	7.28e1 s	3.86e3 s	5.74e8 s

Table 1. This table shows the trade-off in time and calculations that are possible in terms of the $overinfo$ rate (i.e. the number of times P_{att} packets was given to the attacker). Larger keysizes make the attacker's job more difficult - he will either need to do much more calculations or gather much more packets than with smaller keysizes.

Our scheme, unlike the usual cryptographic schemes, is simple to analyze. The detailed examination of the problem that leads to the results shows that any algorithm that can break the privacy of the proposed protocol significantly faster than the presented one is a technological breakthrough.

Comparison with other schemes The presented scheme is targeted for very simple RFID tags with no computational capabilities. Therefore, it is not meant to compete with schemes that use cryptographic functions such as a secure hash. As such, our scheme should be compared to schemes such as the pseudonym-rotation scheme, proposed in [1], where each tag is configured with a short list of random identifiers called pseudonyms. In this scheme, each time a tag is queried, it emits the next pseudonym in the list, cycling to the beginning when the list is exhausted. With this scheme, if $n = 10^7$ a tag that contains 15 pseudonyms which corresponds to a memory size of $15 * \lceil \log_2(10^7 * 15) \rceil = 420$ bits, loses its privacy after 16 requests by the attacker. Note that this is exactly the same as our protocol with parameters $K = 420, L = 10, \lceil P_{att}/P \rceil = 16$, with the exception that in our protocol, the attacker needs not only to *record* the sent information, but also to execute a difficult *computation* to win the privacy experiment. In other words, our protocol gives the same amount of *information* during one protocol run as the pseudonym-rotation scheme, but in a coded way. As such, for an attacker to break our scheme in a reasonable amount of time, he needs to collect much more information than would be necessary from a purely information theoretic point of view.

References

1. Juels, A.: Minimalist cryptography for low-cost RFID tags. In: SCN 2004
2. Molnar, D., Soppera, A., Wagner, D.: A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In: SAC 2005
3. Buttyán, L., Holczer, T., Vajda, I.: Optimal key-trees for tree-based private authentication. In: PET 2006

4. Ohkubo, M., Suzuki, K., Kinoshita, S.: Efficient hash-chain based RFID privacy protection scheme. In: Ubicomp 2004, Workshop Privacy: Current Status and Future Directions
5. Lu, L., Liu, Y., Hu, L., Han, J., Ni, L.: A dynamic key-updating private authentication protocol for RFID systems. In: PerCom 2007
6. Juels, A., Rivest, R., Szydlo, M.: The blocker tag: Selective blocking of RFID tags for consumer privacy. In: ACM CCS 2003
7. Vajda, I., Buttyán, L.: Lightweight authentication protocols for low-cost RFID tags. In: Ubicomp 2003
8. Li, T., Wang, G.: Security analysis of two ultra-lightweight RFID authentication protocols. In: IFIP SEC 2007
9. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: Aes implementation on a grain of sand. In: Information Security, IEEE (2005) 13–20
10. Juels, A., Weis, S.: Authenticating pervasive devices with human protocols. In: CRYPTO'05
11. EPCglobal: 13.56 mhz ism band class 1 radio frequency identification tag interface specification (2003). Technical report, Auto-ID cetner, MIT
12. Juels, A., Weis, S.: Defining strong privacy for RFID. Cryptology ePrint Archive, Report 2006/137 (2006)
13. Schaefer, T.J.: The complexity of satisfiability problems. In: STOC '78
14. Cheeseman, P., Kanefsky, B., Taylor, W.M.: Where the really hard problems are. In: IJCAI-91
15. Een, N., Soorensson, N.: An extensible sat-solver [ver 1.2]. Theory and Applications of Satisfiability Testing **2919/2004**
16. Bouveret, S., Heras, F., de Givry, S., Larrosa, J., Sanchez, M., Schiex, T.: Toolbar: a state-of-the-art platform for wcsp (2004)
17. Chai, D., Kuehlmann, A.: A fast pseudo-boolean constraint solver. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2003. 305–317
18. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**(3) (1960) 201–215
19. Schulz, M.H., Auth, E.: Improved deterministic test pattern generation with applications to redundancy identification. IEEE Transactions on computer-aided design **8**(7) (July 1989) 811–816
20. Silva, J.P.M., Sakallah, K.A.: Graspa new search algorithm for satisfiability. In: ICCAD '96
21. Smith, B.: The phase transition in constraint satisfaction problems: A CLoser look at the mushy region. In: Proceedings ECAI'94. (1994)
22. Achlioptas, D., Peres, Y.: The threshold for random k-sat is 2^ k(ln 2 - o(k)). In: STOC '03