Solving the Simultaneous Scanning Problem Anonymously: Clumping Proofs for RFID Tags

Pedro Peris-Lopez, Julio C. Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda
Carlos III University of Madrid
Email: {pperis, jcesar, jestevez, arturo}@inf.uc3m.es

Abstract

The vast majority of works on RFID security focus only on privacy and tracking (violation of location privacy). However, in this paper we are interested in a new mechanism that enables a pair of RFID tags to generate a proof of having been simultaneously scanned by a reading device. In 2004, Juels introduced this concept and presented a proof named "yooking proof". Saito and Sakurai (2005) showed how Juels's proof was vulnerable to replay attacks, and proposed a new scheme based on using timestamps. Nevertheless, Piramuthu later demonstrated that this new proof was also vulnerable to replay attacks. Although Piramthu's proposed scheme seems to be resistant to replay attacks, and attack using multi-proofs sessions can be attained. Furthermore, Piramuthu claims that privacy and location privacy is guaranteed in his scheme, which is not the case as tags transmit their static identifiers in clear. A new anonymous proof, named "clumping proof", is proposed here that solves the multi-proofs session attack and provides privacy while also protecting against tracking.

Index Terms— *RFID, yooking-proofs, grouping-proofs, security, privacy, tracking*

1 Introduction

RFID is a pervasive technology, perhaps the most pervasive technology in history. One of the main problems that ubiquitous computing has to solve before its wide development is privacy [17]. Products labeled with insecure tags could reveal sensitive information when queried by readers. Moreover, even if we assume that tag's contents are secure, trucking (violation of location privacy) protection is not guaranteed. Tags usually answer with the same identifier.

In addition to the previous threats, there are some other aspects that must be considered: eavesdropping, counterfeiting, physical attacks, active attacks, etc. To depth in all these matters we recommend the reading of [7, 10, 13] where surveys of the most important advances in RFID technology are presented.

In this paper, we are interested in a rather different security-oriented problem. In 2004, Juels introduced the problem of providing a proof for the simultaneous reading of two RFID tags [6]. In other words, a proof that a pair of RFID tags has been scanned simultaneously, but not necessarily by the same reading device. In the future we refer to this problem in general terms as the simultaneous scanning problem. Concretely, Juels denominated his proof as "yoking proof" (applying "yoke" wit its meaning "to join together").

2 Motivation and Related Work

The aim of Juels's proof is to allow two tags to generate a proof of having been scanned simultaneously, which is verifiable off-line by a trusted entity [6]. The proposal relies on the assumption that the protocol will always terminate within a given time interval. For example, in the UHF frequency band, a frequency hooping by the reader is required ($t = 400$ ms under FCC), which entails a termination of the tag reading process in this interval. This assumption only implies that each tag is read within a t interval, but it does not imply that the two tags have been simultaneously read. Consider the following example, tag A is read at time T_1, the reader waits Ts seconds, and finally tag B is read. Under this scenario the reader has a proof that these two tags have been consecutively read but no simultaneity is guaranteed, so the main proof objective is not met. This attack is similar to the replay attack against the "yooking proof" presented by Saito et al. [15]. Additionally, the verifier would not have any information about when the proofs were generated (initial and final time) as the verifier does not initiate the proofs, and the proofs are not immediately sent to verifiers either. From all the above, the participation of the verifier in the proof generation seems to be necessary.

As we mentioned above, Juels's proposal is susceptible to replay attacks. Although Saito a Sakurai proof continues

Third International Workshop on Security
Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2007)
0-7695-2863-5/07 $25.00

to be vulnerable to this kind of attacks, they protocol will be presented as a couple of interesting ideas can be extracted from it. Following, Piramuthu's proof will be presented. Despite the fact that the protocol seems to be resistant to replay attacks, we will show how a multi-proof session attack can be accomplished. Moreover, both protocols present privacy and tracking problems due to the fact that tag's static identifier is transmitted on the channel in clear.

Notation Used:

- T_i, V, R: an RFID tag, verifier, and reader
- PS: pseudonym derived from its static identifier (ID)
- r, r_A, r_B: random numbers (nonces)
- x_i: secret keys shared between T_i and V
- x_V: secret key of the Verifier
- $MAC_{x_i}[m]$: a cryptographic message authentication code, e.g. HMAC with secret key x_i applied to message m. MAC: $\{0,1\}^d \times \{0,1\}^* \rightarrow \{0,1\}^d$.
- TS: timestamp
- P_{AB}: proof that $A\&B$ were scanned simultaneously

2.1 "Yooking proof"using a timestamp

Saito and Sakurai proposed a "yooking proof" using a timestamp [15]. In this scheme, RFID tags compute a MAC of a timestamp under a secret key. These keys are shared between tags and verifier/s in advance. Following, the proposed proof is described:

- A reader gets a timestamp TS from a database and queries to T_A and T_B including TS in the query.
- T_A computes $m_A = MAC_{x_A}[TS]$ and submits it to the reader.
- The reader submits m_A to T_B.
- T_B computes $m_B = MAC_{x_B}[TS, m_A]$ and submits it to the reader.
- The reader submits $P_{AB} = (A, B, TS, m_B)$ to a verifier V. The verifier V checks it using the shared secret keys x_A and x_B.

As Piramuthu demonstrated [12], "grouping proofs" are vulnerable to replays attacks too. Imagine the following scenario: an attacker queries a tag T_A with a future timestamp TS obtaining its corresponding answer m_A. Next, some time later, when time TS becomes true, a proof of simultaneous reading of T_A and T_B can be obtained by the attacker, without the presence of tag T_A.

Additionally, an exhaustive search can be accomplished. Suppose that the timestamp TS has a 16 bits length (as defined for `time()` function in DJGPP), and a tag reading time is 400 ms. Under this scenario, an exhaustive search trying all possible timestamps can be carried out in around 7 hours. Once all possible values are obtained, an attacker may be able to impersonate tag T_X. In the next section, the selection of a suitable length value for this variable will be discussed. According to Saito and Sakurai, timestamp TS is generated by V and used to build the proof P_{AB} in order to check the time when the proof was generated. The problem of this solution is that time is transmitted as a plaintext, which allows attackers to construct future proofs. As a consequence of this in our proposed proof timestamps will still be used, but will be passed to the channel only after the application of a MAC, thus in seemingly random wraps.

2.2 On Existence Proofs for RFID Tags

In 2006, a new proof inspired in Juels's "yoking proofs" was proposed by Piramuthu [12]. In it, the reader is assumed to be authenticated by the verifier: first a random number is obtained from V, and at the end of the process the proof P_{AB} is returned to V. Next, an execution of Piramuthu's proof is outlined:

- A reader gets a nonce r from the verifier and queries T_A including r in the query.
- T_A generates a random number r_A using r as seed and submits $a = (A, r_A)$ to the reader.
- The reader sends a request ($request$, r_A, r) to T_B.
- T_B computes $m_B = MAC_{x_B}[r_A, r]$ and generates a random number r_B with the seed r. Next, tag T_B answers (B, m_B, r_B) to the reader.
- The reader forwards m_B to tag T_A.
- T_A computes $m_A = MAC_{x_A}[m_B, r_A]$ and sends it back to the reader.
- The reader submits $P_{AB} = (A, B, r_A, r_B, r, m_A, m_B)$ to a verifier V. The verifier V checks it using shared secret keys (x_A, x_B) and r.

3 Attacks against Piramuthu's proof

In Piramuthu's scheme, tag T_B computes a MAC of a nonce generated by the verifier (r) and a random number r_A generated by T_A. Piramuthu states that the variable r is used as a seed for generating r_A but its use as an authentication method is never mentioned. Due to this, the inclusion of r_A does not guarantee generation by T_A , as only seed r (which can be easily eavesdropped) is needed to obtain r_A, and no secret keys (x_A) are involved. For that reason, Piramuthu's

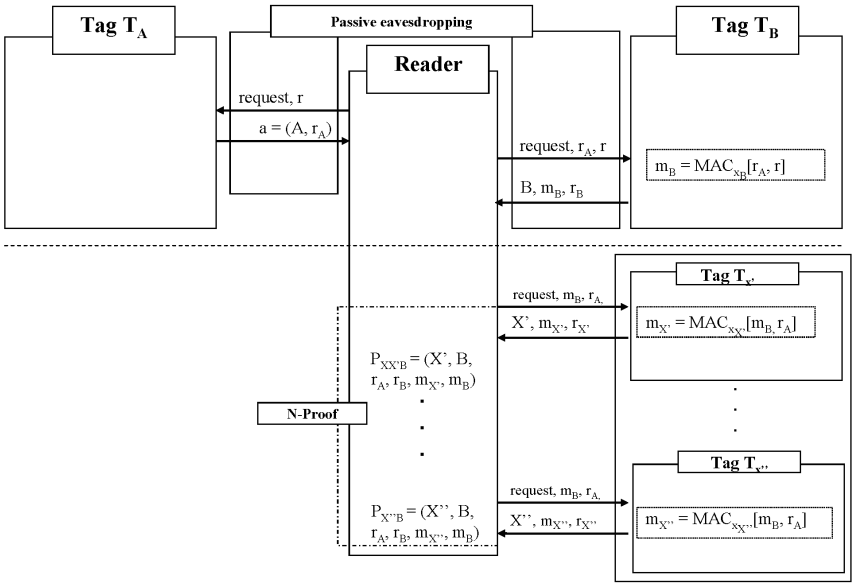

Fig. 1. Multi-proof (N) session attack

protocol is vulnerable to a multi-proof (N) session attack as we will show.

In any other case, when use of r_A is oriented towards providing authentication of the participant tags, Piramuthu's protocol resembles a lot that of Juels's [6]. The only difference is that tags use counters ($(r_A = f_{x_A}[c_A])$ instead of the seed r sent by the verifier $(r_A = f_{x_A}[r])$. Function f could be derived from a keyed hash function, a pseudo-random number generator, or from any other cryptographic primitive. As Juels mentions, the vast majority of tags can not execute standard cryptographic primitives, so only light-weight functions are possible in low-cost environments.

Finally, note that in the case that r_X is used to authenticate tags, the multi-proof session attack does not work. A multi-proof (N) session attack for "on existence proofs" is described below, which is also illustrated in Fig. 1.

- A reader gets a random number r from the verifier and queries T_A, including r in the query.
- T_A generates a random number r_A using r as a seed, and submits $a = (A, r_A)$ to the reader.
- The reader sends a request $(request, r_A, r)$ to T_B.
- T_B computes $m_B = MAC_{x_B}[r_A, r]$ and generates a random number r_B with the seed r. Next, tag T_B sends (B, m_B, r_B) to the reader.

  Next, the following process is repeated N times:

  – The reader sends a request $(request, m_B, r_A)$ to T_X.

  – T_X computes $m_X = MAC_{x_X}[m_B, r_A]$ and generates a random number r_X with seed r. Next, tag T_X sends (X, m_X, r_X) to the reader.

  – The reader discards r_X and submits $P_{XB} = (r_A, r_B, r, m_X, m_B)$ to a verifier V. The verifier V checks it using the shared secret keys (x_X, x_B) and the nonce r.

By means of this attack, an attacker is able to generate a proof of simultaneous reading for any tag T_X and tag T_B being present (only a passive listening of the air channel is required). Additionally, there are some other drawbacks that suggest a modification of the proof:

- Tags are very constrained devices with limited memory and processing capacity. Piramuthu's proofs relies on the fact that tags support on-board a pseudo-random number generator and a message authentication code. This two requirements lie well beyond the capabilities of a great number of RFID tags, specially the low-cost ones. Therefore, the computing requirements of our proposal have been designed to fit in this constrained environment.
- Another interesting point, is the security analysis presented in Piramuthu's scheme. The author claims neither privacy nor location privacy is jeopardized. This is clearly not true because the static identifiers are included in tag answers. In order to avoid these two attacks, the identifiers should not be transmitted as a plain text and should change each time a reader interacts with a tag. This is also a feature of our proposal.

4 Proposed protocol

This section is divided in three parts, in order to clarify the exposition of the proposed proof. First, the main concepts on which the proof is based are introduced. Additionally, we briefly explain how a new high-entropy function has been obtained. Then the proposed proof, named "clumping proof", is presented. The name of clumping proofs is based on the analogy with a tree ϒ : the trunk is the reader and the group of branches (clump) are the tags that are to be read simultaneously. In the last section, we present a security analysis of our proof.

4.1 Considerations

The proposed proof is inspired in "grouping proofs" and "existence proofs". More concretely, the protocol grounds in the following principles:

- Timestamps are used for allowing the verifier to validate when the proof starts and ends.
- Each tag has an "internal state", in particular a counter initialized to a random value.
- At the beginning, there is a mutual authentication phase between the reader and the verifier. Only after this phase is successfully accomplished the proof starts.
- All the inputs to a given tag (except the first one) are derived from computations that can only be carried out by fellow tags participating in the proof. This guarantees freshness and causality.
- Due to the very limited tag resources, we assume that they do not have on-board pseudo-random number generators. This needs not be true, as the authors have shown that PRNGs can be implemented even in low-cost tags [11], but is a reasonable assumption when they should also implement a MAC.

In previous works [6, 12, 15], tags always include static identifiers in their answers. However, static identifiers should be disguised (anonymized) before transmission in order to avoid privacy and tracking problems. A common solution to this problem will be to compute a hash function of the ID, but standard cryptographic primitives are not at hand in this low-resource environments. A light-weight function, instead, has been employed in our proposal. A new function, named Nun, has been obtained by means of Genetic Programming [8]. Particularly, the experimentation has been performed with the `lil-gp` library [1]. We tried to find a highly nonlinear, but efficient, function. This property has been measured through the avalanche effect. In fact, an even more demanding property has been used: the Strict Avalanche Criterion [5].

Table 1. Function Nun

$x_1 = Nun[m, n]$

```
SP = m
for(i=0;i<32;i++){
 SP = (SP>>1)+(SP<<1) + n  }
x1 = SP
```

In order to evaluate how nonlinear and unpredictable Nun is, a sort of linear cryptanalysis has been applied [9, 16]. We have assumed for this analysis that variables have a length of 32-bits. After a random initialization of m (the first Nun input), and for each mask pair, 2^{25} 32-bits outputs haven been generated. Note, that the parameter n is an incremental counter in our proposal. From the calculations described above, we obtain that the bias of Nun is bounded by $\frac{1}{2^{14.56}}$, which is only slightly worst than that expected of a random function.

4.2 Anonymous clumping proofs

The proposed proof is described below. After a successful mutual authentication phase between the reader and the verifier, the reader asks the verifier for a new timestamp. However, the verifier does not send the timestamp in plain text, but encrypted by means of a keyed hash function ($t = g_{X_V}[TS]$). Next, the reader starts the protocol, as illustrated in Fig. 2.

- The reader divides t in two halves (t_{MSB}, t_{LSB}) and sends a query to T_A including t_{MSB}.
- Tag T_A computes its new pseudonym ($a_1 = Nun[ID_A, counter_A)]$ and builds an evidence ($a_2 = MAC_{x_A}[t_{MSB} \oplus a_1]$ that t_{MSB} was seen before. Then, T_A sends back a message ($Y = (a_1, a_2, counter_A)$) to the reader. Finally, T_A increments its counter ($counter_A = counter_A + 1$).
- Reader sends a request ($request$, t_{LSB}, a_2) to T_B.
- Tag T_B calculates its new pseudonym ($b_1 = Nun(ID_B, counter_B)$) and computes the message authentication code ($b_2 = MAC_{x_B}[a_2, b_1 \oplus t_{LSB}]$). Then, T_B sends back a message ($Z = (b_1, b_2, counter_B)$) to the reader. After that, the counter is incremented ($counter_B = counter_B + 1$).
- Reader forwards b_2 to T_A.
- T_A computes a message authentication code $m_{AB} = MAC_{x_A}[a_2, b_2]$ and sends back it to the reader.

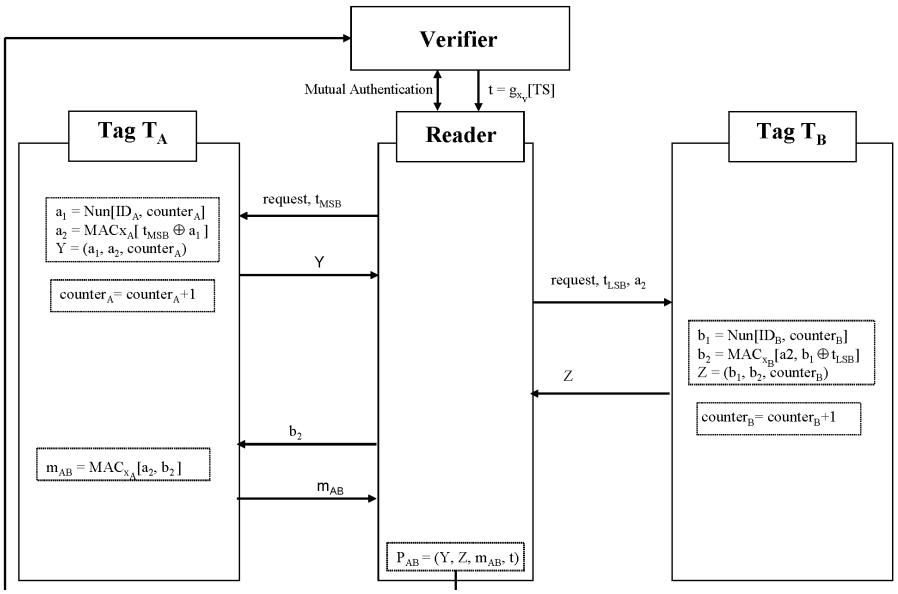

Fig. 2. Anonymous Clumping Proof

- Reader submits proof $P_{AB} = (Y, Z, m_{AB}, t)$ to the verifier V. The verifier first checks the validity of the anonymous timestamp t, then verifies the clumping proof using the shared secret keys (x_A, x_B).

4.3 Security Analysis

One of the most significative differences between "clumping proofs" an other simultaneous scanning proofs is that we have tried to address all the main security risks commonly found in RFID technologies.

Particularly, we have avoided the transmission of the static identifier in plain text over the channel, as the air channel can be easily eavesdropped, revealing confidential information and allowing a tracking attack. As such, we provide security against these two threats.

Another problem commonly found in previous proposals are replays attacks. In order to avoid this, all the inputs to a given tag (except the first one) are derived from computations that can only be carried out by fellow tags participating in the proof. This guarantees freshness and causality.

Additionally there are two other mechanisms to strengthen the proof's security. The protocol presents an asymmetry: reader sends T_{LSB} to the first tag and T_{MSB} to the second tag. Finally, tags include a counter ("internal sate") in their answers, which provide a sequence order in the proofs.

One of the problems connected with the usage of timestamps is the exhaustive search problem as discussed in Section II. If the length of the timestamp is fixed to 16 bits the attack can be very easy. In our case, the output of a keyed hash function is transmitted. Nevertheless, the length of the input (timestamp) will be the parameter that will fix the feasibility of this attack. Imagine the currently worst scenario (the quickest one) where it is using RFID tags according to Gen-2 specification, which allows high reading speed of around 200 tags/sec [4]. Even under these conditions, if the timestamp is fixed to a length greater than 32 bits, an attacker will have to interrogate the tag during more than a year to search all possible values. So 32 bits seem, at present speeds, as a secure choice for the timestamp lengths, but no less that this should be used.

Finally, we can define the security of our protocol in terms of a game, as done in [6]. It its assumed that tags are initialized (secret keys and counter), and the adversary ($\mathscr{A}$) may interact with tags $\{T_i\}$ during a long period of time. $\mathscr{A}$ will win the game if a submitted proof P_{AB} is accepted by the verifier. The success probability will be denoted by δ. Applying the random-oracle model to the underlaying cryptographic primitive, we can assert: *Given a random-oracle on MAC, the success probability δ of $\mathscr{A}$ for clumping proofs is bounded to* 2^{-d}.

Proof: Let $P_{AB} = (Y, Z, m_{AB}, t)$ where $(Y = (a_1, a_2, counter_A))$ and $(Z = (b_1, b_2, counter_B))$. *Case A*: Suppose that $\mathscr{A}$ did not sent initial message $(request, T_{MSB})$ to ask tag T_A for a proof. Under the random oracle assumption on MAC, the adversary could not guess $(a_2 = MAC_{x_A}[t_{MSB} \oplus a_1]$ with probability at most 2^{-d}. Analogously, $(b_2 = MAC_{x_B}[a_2, b_1 \oplus t_{LSB}])$ and $m_{AB} = MAC_{x_A}[a_2, b_2]$ could be correctly found with probability at most 2^{-d}. Similar arguments can be applied to Case 2: $\mathscr{A}$ sent $(request, T_{MSB})$ to tag T_A, but

($request$, t_{LSB}, a_2) did not send to tag T_B and Case 3: $\mathscr{A}$ did not send b_2 to tag T_A.

5 Discussions and Conclusions

In [2], Bolotnyy et al. introduce a new formulation problem, called anonymous yooking, which requires that tags preserve its privacy during the execution of the proof. The proposed scheme is based on the execution on tag of both a PRNG and a keyed hash function, as in others well-known proposals [3, 14]. Particularly, the tag transmits the output of a keyed hash function instead of transmitting its static identifier. However, the computational cost of this proposal will be very high on the verifier's side, as it has to accomplish an exhaustive search ($0(N^2)$). In our case, we use a pseudonym that is updated every time the tag is queried, and which is difficult to predict (from past values) as described in Section IV. Moreover, the result of this function is transmitted together with a counter, thus easing the search process in the verifier (which will take $0(N)$ steps). Since no private information is transmitted during the proof, and all messages are contained in random wraps (thus only seemingly random values are sent over the channel), privacy and privacy location is guaranteed.

The counter, together with the timestamp (t), provides the grounds for having a temporal order in every exchanged message in the proof. Specifically, suppose that P_{AB} and PAB' are generated using the same anonymous timestamp. As proofs have been generated with ($counter_A$, $counter_A$) and ($counter'_A$, $counter'_B$) respectively, where $counter_A > counter'_A$ and $counter_B > counter'_B$, it can be demonstrated that P'_{AB} was generated later than P_{AB}.

To summarize, in this paper we present a new proof of simultaneous tag scanning, named "clumping proofs". All literature we know in this area has been revised in order to identify its principles and problems. Moreover, a new attack to Piramuthu's protocol is presented. In the proposal we have put an special emphasis in solving all the security concerns while using only minimal processing capability on tags.

References

[1] The `lil-gp` genetic programming system. http://garage.cps.msu.edu/software/lil-gp/lilgpindex.html.

[2] L. Bolotnyy and G. Robins. Generalized "yooking proofs" for a group of RFID tags. In *Poster of MOBIQUITOUS06*, 2006.

[3] T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Proc. of SECURECOMM'05*, 2005.

[4] Class-1 Generation-2 UHF air interface protocol standard version 1.0.9: "Gen 2". http://www.epcglobalinc.org/standards/.

[5] R. Forré. The strict avalanche criterion: Spectral properties of boolean functions and an extended definition. In *Proc. of CRYPTO'88*, LNCS, pages 450–468, 1990.

[6] A. Juels. "Yoking-proofs" for RFID tags. In *Proc. of PerSec'04*, pages 138–143. IEEE Computer Society, March 2004.

[7] A. Juels. RFID security and privacy: A research survey. Manuscript, 2005.

[8] J.R. Koza. Evolving a computer program to generate random number using the genetic programming paradigm. In *Proc. of the 4th Int. Conference on GA*, pages 37–44, 1991.

[9] M. Matsui. Linear cryptanalysis method for DES cipher. In *Proc. of EUROCRYPT'93*, volume 1994, pages 386–397.

[10] P. Peris-Lopez, J. C. Hernandez-Castro, J.M. Estevez-Tapiador, and A. Ribagorda. RFID systems: A survey on security threats and proposed solutions. In *Proc. of PWC06*, volume 4217 of *LNCS*, pages 159–170, 2006.

[11] P. Peris-Lopez, J. C. Hernandez-Castro, J.M. Estevez-Tapiador, and A. Ribagorda. LAMED – A PRNG for EPC class-1 generation-2 RFID specification. *Journal of Information Science and Engineering (submitted)*, 2007.

[12] S. Piramuthu. On existence proofs for multiple RFID tags. In *SecPerU'06*. IEEE Computer Society Press, 2006.

[13] S. Piramuthu. Protocols for RFID tag/reader authentication. *Decision Support Systems*, page doi:10.1016/j.dss.2007.01.003, 2007.

[14] K. Rhee, J. Kwak, S. Kim, and D. Won. Challenge-response based RFID authentication protocol for distributed database environment. In *Proc. of SPC'05*, volume 3450 of *LNCS*, pages 70–84, 2005.

[15] J. Saito and Sakurai Kouichi. Grouping proof for RFID tags. In *Conference on Advanced Information Networking and Applications – AINA*, volume 2, pages 621–624, Taiwan, March 2005. IEEE.

[16] F.-X. Standaert, G. Piret, and J.-J. Quisquater. Cryptanalysis of block ciphers: A survey. Technical report.

[17] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, September 1991.