Noisy Tags: A Pretty Good Key Exchange Protocol for RFID Tags

Claude Castelluccia[1] and Gildas Avoine[2]

[1] INRIA, France
Claude.Castelluccia@inria.fr
[2] EPFL, Switzerland
Gildas.Avoine@a3.epfl.ch

Abstract. We propose a protocol that can be used between an RFID tag and a reader to exchange a secret without performing any expensive computation. Similarly to the famous blocker tag suggested by Juels, Rivest, and Szydlo, our scheme makes use of special tags that we call *noisy tags*. Noisy tags are owned by the reader's manager and set out within the reader's field. They are regular RFID tags that generate noise on the public channel between the reader and the queried tag, such that an eavesdropper cannot differentiate the messages sent by the queried tag from the ones sent by the noisy tag. Consequently, she is unable to identify the secret bits that are sent to the reader. Afterwards, the secret shared by the reader and the tag can be used to launch a secure channel in order to protect communications against eavesdroppers. It can also be used to securely refresh a tag's identifier by, for example, xoring the new identifier with the exchanged secret key. Refreshing tags' identifiers improves privacy since it prevents tracking tags.

1 Motivations

An RFID (Radio-Frequency IDentification) tag is small circuit attached to a small antenna, capable of transmitting data to a distance of several meters to a reader device (reader) in response to a query. Most RFID tags are passive, meaning that they are battery-less, and obtain their power from the query signal. They are already attached to almost anything: clothing, foods, access cards and so on.

Unfortunately, the ubiquity of RFID tags poses many security threats: denial of service, tag impersonation, malicious traceability, and information leakage. We focus in this paper on this latter point that arises when tags send sensitive information, which could be eavesdropped by an adversary. In the framework of a library, for example, the information openly communicated by the tagged book could be its title or author, which may not please some readers. More worryingly, marked pharmaceutical products, as advocated by the US Food and Drug Administration, could reveal a person's pathology. For example, an employer or an insurer could find out which medicines a person is taking and thus work out his state of health. Large scale applications like the next generation of passports

J. Domingo-Ferrer, J. Posegga, and D. Schreckling (Eds.): CARDIS 2006, LNCS 3928, pp. 289–299, 2006.

are also subject to such an issue. The *e-passports* [8] will be equipped with an RFID chip that will store some kind of biometric information about the bearer (fingerprint and digital picture).

Avoiding eavesdropping can be done by establishing a secure channel between the tag and the reader. This requires the establishment of a session secret key, which is not always an easy task considering the very limited devices' capacities. This difficulty is reinforced by the fact that tags and reader do not share a master key in most of the applications. In the future, implementing a key establishment protocol may become a mandatory feature. For example Californian Bill 682 requires such a technical measure to be implemented in ID-cards deployed in California. Furthermore, as explained in Section 4.2 for the library application, a secret key can be used to improve privacy and prevent tracking by securely refreshing a tag identifier.

This paper describes a novel way of establishing a key through a public channel between an RFID tag and a reader. The paper is structured as follows: Section 1 motivates our work. Section 2 presents the related work. Section 3 describes our 3 proposed key-exchange protocols. Section 4 presents some possible applications of our schemes. Section 5 discusses the relevance of our proposals and their security.

2 Related Work

The problem of secure pairing of wireless devices has been tackled by several researchers. The proposed solutions can be classified into the following categories:

Public-Key Cryptography Based Solutions: These solutions rely on public-key based key exchange protocols such as Diffie-Hellman or RSA [5, 6, 7]. In Diffie-Hellman based schemes, devices exchange their Diffie-Hellman components and derive a key from them. In RSA-based schemes, one of the devices selects a secret key and encrypts it under the other device's public key. The main problem of these solutions is performance. They require that devices perform CPU-intensive operations such as exponentiation, which are prohibitive for CPU-constrained devices such as RFID tags.

PIN-Based Schemes: In Bluetooth, two wireless devices derive a shared key from a public random value, the addresses of each device and a secret PIN. The PIN is provided to each device by the user via an out-of-band channel, such as a keyboard. While this solution is computationally efficient, it requires that *both* devices be equipped with some kind of physical user interface. As a result, this solution cannot be used to pair devices lacking physical interfaces, such as RFID tags.

Physical Contact or Imprinting: In [11], Stajano and Anderson propose a solution, called *imprinting*, based on physical contact. Two devices get paired by linking them together with an electrical contact that transfers the bits of a shared secret. No cryptography is involved, since the secret is transmitted in plaintext. Furthermore, the key validation phase is not necessary since there is no

ambiguity about the devices that are involved in the binding (i.e., man-in-the-middle attacks are impossible). While this solution is interesting, it requires each device to have some additional hardware to perform the electrical contact.

Physical Protection: The simplest solution is to shield the RFID and the reader by a Faraday cage – a container made of metal mesh or foil impenetrable by radio signals – and have the two devices exchange their secret in cleartext. While very simple, this solution is not practical for RFID tags that are embedded in larger objects, such as humans, cars, clothing, that can not easily be placed in containers.

Shake Them Up!: Another solution is proposed in [3]. This paper presents a new pairing protocol that allows two CPU-constrained wireless devices A and B to establish a shared secret at a very low cost over an anonymous channel. On an anonymous channel, an eavesdropper can actually read the content of the exchanged packets but cannot identify their source. [3] proposes to implement such a channel by bringing the devices close to each other and shaking them during the key exchange [1]. With the proposed protocol, A can send the secret bit 1 to B by broadcasting an (empty) packet with the source field set to A. Similarly, A can send the secret bit 0 to B by broadcasting an (empty) packet with the source field set to B. Only B can identify the real source of the packet (since it did not send it, the source is A), and can recover the secret bit (1 if the source is set to A or 0 otherwise). An eavesdropper cannot retrieve the secret bit since it cannot figure out whether the packet was actually sent by A or B. By randomly generating n such packets A and B can agree on an n-bit secret key.

This solution is interesting but requires to shake the tag and the reader, which is not always practical. Also, the security is based on the assumption that it is difficult to identify the packets sent by the two parties. This assumption does not hold in an RFID environment because distinguishing packets sent by a reader from packets sent by a tag is straightforward.

3 Noisy Tag Protocol (NTP)

3.1 Background

Our protocol is inspired by the tag blocker proposal [9] and a key-exchange scheme developed by some unknown researcher at Bell Telephone Labs during World War II.

The idea of using special device in RFID environment was already proposed by [9] and [12]. [12] proposes to use a special device to simulate RFID tags as a way of spoofing such systems into believing that stolen items are still present in a retail environment. [9] defines the concept of blocker tags that simulate the set of

[1] This is to guarantee that an eavesdropper cannot identify the packets sent by A from those sent by B using data from the RSSI (Received Signal Strength Indicator) registers available in commercial wireless cards.

all possible tag identifiers for privacy protection i.e., to prevent a malicious user to trace some tags. Our application is quite different: we use a special RFID tag – *the noisy tag* – to establish secret key on-the-fly between a reader and RFID tags. While the blocker tag is owned and borne by the consumer, the noisy tag is owned by the system.

The crux of the key-exchange scheme proposed by the Bell Telephone Labs is that a receiver can effectively drown out any signal by injecting noise onto a communication channel. An eavesdropper would only hear the noise, but the receiver could subtract the noise and recover the signal.

This idea can easily be extended to establish a key between two parties A and B over a public channel as follows: B starts by generating a sequence of random bits, noise $N(i)$, on the channel. Simultaneously A sends the secret key bits $k(i)$ over the channel. An eavesdropper will see the sequence of bits $s(i) = k(i)+N(i)$ and won't be able to recover the secret bits $k(i)$ while B can subtract the noise $N(i)$ and recover $k(i)$. A and B did then exchange a secret key.

We propose to apply this idea to allow an RFID tag to exchange a secret key with a reader without performing any expensive computation. For that, a noisy tag must be installed in the reader's field. A noisy tag is a regular RFID tag that shares a secret key K with the reader (this key can be pre-configured). It is used to generate the noise bits $N(i)$ as defined previously. The noise bits are generated from a pseudo-random function, the secret shared with the reader and some public nonce. As a result, they can be reconstructed (and subtracted) by the reader. However they look random to an adversary.

To illustrate this approach, we supply three examples of RFID key-exchange protocols based on noisy tags. In the rest of this paper, R denotes the reader, T the tag and NT the noisy tag. We also assume that T wants to exchange a n-bit long secret, s, with R.

3.2 Bit-Based Protocol, Version 1

This protocol assumes that collisions are allowed and therefore several tags can reply simultaneously to a reader. When several tags replies simultaneously, it is assumed that the amplitude (i.e. voltage) of the different bits get added. Assuming this property, a tag T can send a sequence of secret bits b to the reader R using the protocol described in this section. This protocol is composed of two phases:

Exchange Phase:

(1) R broadcasts a random nonce N.
(2) Both NT and T reply simultaneously with one bit (one bit per time slot) until the reader halts the protocol. The ith bit sent by NT is the ith bit of $\mathrm{h}(K, N)$, where $h(.)$ is a pseudo-random function such as a hash function. The ith bit sent by T is random.
(3) Since the reader can predict the sequence of bits sent by the noisy tag, it can easily filtered them out, and recover the bits sent by T.

If, for example, T sends the bit 1, implemented by a pulse of xmV, and NT the bit 0, implemented by a pulse of 0mV, R will receive the bit 1. Since it can compute that the bit sent by NT was 0, it can retrieve the bit sent by T, i.e. 1. Note however that this protocol does not work if both T and NT reply with the same bit. In fact, if both T and NT send simultaneously the bit 1, a pulse of $2.x$mV will be generated on the channel. In that case, an adversary knows that both T and NT sent the bit 1. Similarly if both T and NT send the bit 0, a pulse of $0mV$ will be generated on the channel and the adversary knows that both T and NT sent the bit 0. This is what we refer to as the "same-bit" problem. When T and NT send the same bit in a given time slot, this bit should not be used to generate the secret key. Hence, the reader halts the exchange phase when at least n time slots contain different bits. On average, the reader halts the protocol after $2n$ time slots. The reconciliation phase consists for the reader to inform the tag which bits should be used to generate the secret key.

Reconciliation Phase:

(4) R sends to T the relevant time slots' numbers. T uses this information to recover the secret bits that should be used to compute the shared secret.

The security relies on the fact that an adversary is not able to separate T's signal from NT's signal. This implies that T and NT are close enough otherwise an adversary may be able to determine which bit comes from T and which one comes from NT using specific material, e.g., directed antennas. This also implies that T and NT use the same standard, i.e. frequency and transmission power. Because one may think that an adversary would be able to separate a few bits, it may be preferable to generate a secret longer than the expected secret key, and then hash it.

3.3 Bit-Based Protocol, Version 2

In the previous protocol, T and NT must reply simultaneously and their bits get added. This requires that the tag T and the noisy tag NT be perfectly synchronized. This might not always be practical or even possible. In this section, we present a solution that removes this assumption. Like in the previous version, this protocol assumes that collisions are allowed.

Exchange Phase:

The exchange phase contains several rounds. Each round is composed of 2 consecutive time slots: $slot_0$ and $slot_1$. In a given round, T sends the bit 1 by setting $slot_1$ to 1. It sends the bit 0 by setting $slot_0$ to 1.

The protocol operates as follows:

(1) R broadcasts a random nonce N.
(2) NT computes a sequence of pseudo-random bits, c, from the nonce N, the secret K it shares with R and a pseudo-random function $h(.)$, i.e. $c = h(K, N)$.

(3) At round i, T picks a random bit b_i and sets the slot number b_i to 1. Similarly NT sets the slot number c_i to 1. When R receives the 2 slots, it can identify the slot set by the NT and retrieve the secret bit send by T.

Note however that this protocol suffers also from the "same-bit" problem: if T and NT select the same slot, the secret bit can be retrieved by an adversary. In fact, if both T and NT select the slot 1 (resp. 0), an eavesdropper can conclude that the secret bit sent by T was 1 (resp. 0). As a result, such rounds must be ignored and the reader halts the exchange phase when at least n rounds have both slots set to 1. On average, the reader halts the protocol after $2n$ time slots. The reconciliation phase consists for the reader to inform the tag which bits should be used to generate the secret key.

Reconciliation Phase:

(4) R sends to T the relevant round numbers. T uses this information to identify the secret bits to be used in the shared secret.

3.4 Code-Based Protocol

The two previous protocols suffer from the "same-bit" problem. As a result, on average, $2 \times n$ rounds are required to agree on a n-bit long key, s.

A solution to this problem consists of having the tag and noisy tag send codes (as in the CDMA protocol) instead of individual bits. If the code is large enough (we use n-bit long codes), the probability of code collision is very small and the number of rounds can be reduced to n. As we will see it later in this section, using codes instead of bits has several other important benefits.

As in the previous schemes, we assume that the reader shares a secret key, K, with the noisy tag. This key is used by the noisy tag together with a pseudo-random function to generate its codes.

The code-based protocol is composed of n rounds. Round i is described as follows:

- **step1:** The reader broadcasts a random nonce, N_i.
- **step2.1:** The noisy tag, NT, replies with a noisy code, $ncode_i$, which is generated from a pseudo-random function, the nonce N_i and the secret key, K. For example, $ncode_i = h(K||N_i)$. This code looks random to an eavesdropper, but can easily be recomputed by the reader.
- **step2.2:** The tag replies with a *random* code, $code_i$. Note that the order of step2.1 and step2.2 must be random at each round otherwise an eavesdropper could easily identify the code coming from the noisy tag from the code coming from the tag. This could be implemented as in the CSMA (Carrier Sense Multiple Access) protocol: upon reception of N_i, the tag and the noisy tag set a timer with a random value $\in [0; t]$, where t is the duration of a round. The tag whose timer expires first, sends its reply first.
- **step3:** Upon reception of $ncode_i$ and $code_i$, the reader filters out $ncode_i$ (by computing $h(K||N_i)$ and retrieve the code sent by the tag $code_i$. For an

eavesdropper, both $code_i$ and $ncode_i$ look random and she, therefore, cannot retrieve the code sent by the tag.

At the end of the n rounds, the reader and the tag share n codes. They can then generate a secret key, s, as follows: $s = code_1 \oplus code_2 \oplus \cdots code_n$.

This protocol has several benefits compared to the previous scheme:

1. It prevents the "same bit" problem, since the probability of the tag and the noisy tag selecting the same code is very small, and therefore reduces the number of rounds to n.
2. The tag and the noisy tag can potentially send several codes per rounds (unlike the previous solution which requires the tag and noisy tag to send one bit per slot). This makes the adversary's analysis more difficult.
3. The noisy tag functionality can be *distributed* over several tags, i.e. several noisy tags can be used. In this case, the reader shares a different secret key, K_i, with each of the noisy tag. When the reader broadcasts a random nonce N, all the noisy tags (or a random subset of the noisy tags- this is only possible because the reader can identify the participating noisy tags from the ncodes) compute their corresponding ncodes, $ncode_i = h(K_i, N)$, and send them back to the reader. As in the basic scheme, the tag replies with a random code. The reader can then filter out the codes sent by the noisy tags and recover the one sent by the legitimate tag. Using several noisy tags increases the noisy codes diversity (power, frequency,...). It is therefore more difficult for the adversary to identify the *codes* from the *ncodes* using, for example, power or energy measures. Also, as described in the following section, using several noisy tags, can reduce the number of necessary rounds for the same level of security.

Since the adversary does not know the secret key, K, shared by the noisy tag and the reader, she cannot differentiate the codes sent by the tag from the codes sent by the noisy tag. As a result, at each round, the probability of selecting the correct code is $\frac{1}{2}$. After n rounds, the probability for the adversary of selecting the n correct codes, and therefore computing the secret key k, is $\frac{1}{2^n}$. Therefore 80 rounds are required in order to obtain a level of security of 2^{80}.

If several noisy tags are used, the probability of selecting the correct code, at each round, becomes $\frac{1}{Q+1}$, where Q is the number of noisy tags replying per round. After n rounds, the probability for the adversary of computing the secret key k, is then $\frac{1}{2^{n.log_2(Q+1)}}$. Therefore if $Q = 15$, only 20 rounds are needed to obtain a level of security of 2^{80}.

4 Applications

4.1 E-Passports

NTP can be used in many applications to establish a secret channel between a reader and an RFID tag. The key is established opportunistically, i.e., it does

not authenticate the end-points of the secure channel. This authentication has to be provided by another mean.

For example, NTP can be used to establish a key between an e-passport and a reader. The next-generation passport, called *e-passport*, will contain an RFID chip, capable of storing and transmitting over the air biometric data together with standard information such as the name, date of birth, nationality of the bearer. This technology creates many security and privacy problems [8]. If no encryption and access control mechanisms are provided, it becomes trivial for anyone to skim e-passports and retrieve their information. In order to mitigate this problem, it is expected that the covers of the e-passports will contain RF blocking material. As a result, once closed it becomes impossible to skim an e-passport. A user can then authorize the reading of his e-passport (for example at a custom) by physically opening it. This simple solution improves the security considerably but does not prevent an eavesdropper from snooping on a legitimate reading. Encryption, and therefore a key-exchange protocol, is required to solve this problem. One proposed solution takes advantage of the fact that passports carry optically readable information. The idea is then to have the reader scan the e-passport and use the scanned information to generate an encryption key. This solution has at least two limitations. Firstly, it requires optical contact, which somehow alleviates the benefits of using RFID. Secondly, since the optically readable information is constant, the same key will be used by all readers. Consequently, it can leak.

NTP can be used to establish a temporary and fresh key between an e-passport and its reader as follows: the user opens its e-passport in front of the legitimate reader. The NTP is then executed between the reader and the e-passport to exchange a key. We assume that the reader has deployed one or several noisy tags. The e-passport can then send its encryption data to the reader.

The use of NTP is not limited to e-passports. It can be used in any applications where the link between a tag and reader need to be secret.

4.2 Libraries

In the e-passport application, the threat was the leakage of sensitive information on the backward channel, i.e., the channel from tags to readers. The problem is even worse when considering the forward channel, i.e., the channel from readers to tags, because the data sent can be eavesdropped at a much longer distance, e.g., one hundred meters.

In the famous paper [10], Molnar and Wagner suggested a protocol that mitigates the privacy problem in libraries. Their protocol roughly consists in refreshing the book's random identifier each time it is borrowed. Although the adversary can still track the book borrowed by Mister X, she cannot determine that this book is the same than the one previously borrowed by Mister Y.

More precisely, in [10], on each check-out the reader reads the data D contained in the tag (e.g., title, author, etc.), picks a random identifier N, stores the pair (N, D) in the system's database, erases D from the tag, and finally

writes N in the tag. On check-in, the reader obtains the identifier N from the tag, looks for (N, D) in the system's database, erases N from the tag, and writes D instead.

However, eavesdropping the forward channel smashes the purpose of the protocol. Consequently, reducing the risk of malicious traceability by avoiding the adversary to eavesdrop the forward channel is important. This can be done by using a secure channel, which requires a key agreement protocol. Note that preventing passive attacks does not required authentication. Ensuring both authentication and privacy using only symmetric cryptography is actually a hard problem in practice because this involves in large scale applications a heavy key management, as explained in [1, 10].

When dealing with very low-cost tags, using a hash function or a symmetric cipher is still unrealistic today, even if a few lightweight implementations of symmetric cryptographic functions have been proposed [4, 13]. However, NTP is suited to such tags because NTP can be used to refresh the identifier of the tagged book without involving symmetric cryptographic functions on the tag's side (a symmetric cryptographic function must be implemented in the noisy tags but not in the books' tags). Indeed, since the identifier of the tag is random, reader and tag can agree on a common identifier instead of a secret key used to secure the channel. Thus, no symmetric cipher is required because we no longer use a secure channel, and moreover the privacy amplification phase, which requires a hash function, is not mandatory. Note that this is possible because the identifier is random, but NTP cannot be used to exchange a chosen information without establishing a secure channel, since reader and tag does not know, *a priori*, which bits will be withdrawn during the information reconciliation phase.

5 Discussion and Security Analysis

5.1 NTP Purpose

The primal purpose of this work is to provide a key agreement protocol between a reader and a tag that is resistant in presence of *passive* adversaries. NTP focuses only on passive adversaries because it does not ensure authentication. Clearly, dealing with passive adversaries instead of active ones is sometimes irrelevant. However, NTP is relevant in many environments, as explained below.

Active attacks require the adversary being able to stay close to the tag or reader in order to carry out her attack. Certain environments do not allow an adversary to be close enough to the tag or reader, e.g., in private areas (house, building, etc.). Furthermore, it is much easier to perform a passive attack, in particular on the forward channel, which can be eavesdropped from long distance.

Very low-cost tags are not able to use symmetric cryptography. That is the case for example with the tags used in libraries as described by Molnar and Wagner in [10]. In their protocol (and in most of the protocols used in libraries today – probably all of them), no security features are implemented, neither on the forward channel, nor on the backward channel. In that case, NTP is an interesting security measure, because it can be implemented cheaply.

In some cases, the authentication that could protect the tag against active adversaries, could be provided through another channel. For example, with the electronic passports, the officer swipes the data page through an optical reader and thus obtains information (name, date of birth, etc.), which can be used to authenticate the radio frequency channel. As explained above, the great interest of the noisy tag is to generate a fresh random session key, while the ICAO (International Civil Aviation Organization) recommends to generate a key directly from static data available on the passport.

5.2 NTP Security

Assuming that (1) the bits sent by the tag are uniformly distributed; (2) the bits sent by the noisy tag are uniformly distributed as well; (3) the adversary is not able to determine (with a probability better than 1/2) which signal comes from the tag and which one comes from the noisy tag; then NTP is perfectly secure, meaning that the adversary learns nothing about the shared secret key.

Assuming that tags are able to generate random bits is a common assumption in RFID. For example, [2] shows that privacy cannot be ensured if tags do not possess a cryptographically secure pseudo-random number generator. Indeed, such a generator is mandatory in the communication layer to avoid an adversary tracking tags because of the collision-avoidance protocol.

The assumption on the noisy tag's side is stronger. The generated bits should be indistinguishable from random bits, but the reader must be capable of generating itself the same series. This can be achieved using a pseudo-random function. In practice, a hash function can be used. Note that synchronization is not required between the reader and the noisy tag because bits are generated from the secret key (shared by the reader and the noisy tag) and a nonce broadcast by the reader.

The third assumptions relies on the difficulty for an eavesdropper to differentiate the information sent by the noisy tags from the information sent by the legitimate tag. Note that the popular tag blocker scheme relies on a rather similar assumption. As admitted in [9], it is conceivable that a well-equipped attacker might actually be able to use the signals' characteristics (fingerprints) to identify the source of each message and filter out the tag blockers or noisy tags. However, such an attack is hard to be put into practice and requires very specialized material. Moreover, if we assume that an attacker is able to recognize tags' fingerprints then protecting privacy, in particular avoiding malicious traceability of the tags, is unsolvable. Last but not least, the adversary should not be able to distinguish the legitimate tag's signal from the noisy tag's signal according to the geographical position. This implies that the legitimate tag should be close to the noisy tag. Possibly, several noisy tags can be used simultaneously to render more difficult the adversary's job. Furthermore, shaking the tag during the key exchange protocol, as suggested in [3], might randomize the power of its transmitted bits and might also be another way to increases security.

References

1. Gildas Avoine, Etienne Dysli, and Philippe Oechslin. Reducing time complexity in RFID systems. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, Lecture Notes in Computer Science, Kingston, Canada, August 2005. Springer-Verlag.
2. Gildas Avoine and Philippe Oechslin. RFID traceability: A multilayer problem. In Andrew Patrick and Moti Yung, editors, *Financial Cryptography – FC'05*, volume 3570 of *Lecture Notes in Computer Science*, pages 125–140, Roseau, The Commonwealth Of Dominica, February–March 2005. IFCA, Springer-Verlag.
3. Claude Castelluccia and Pars Mutaf. Shake Them Up! In *ACM/Usenix Mobisys*, June 2005.
4. Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370, Boston, Massachusetts, USA, August 2004. IACR, Springer-Verlag.
5. Christian Gehrmann and Kaisa Nyberg. Enhancements to bluetooth baseband security. In *Nordsec'01*, Copenhagen, Denmark, November 2001.
6. Jaap-Henk Hoepman. Ephemeral pairing in anonymous networks. Available at: http://www.cs.kun.nl/∼jhh/publications/anon-pairing.pdf.
7. Jaap-Henk Hoepman. The ephemeral pairing problem. In *Financial Cryptography – FC'04*, LNCS, pages 212–226, Key West, Florida, February 2004. IFCA, Springer-Verlag.
8. Ari Juels, David Molnar, and David Wagner. Security and privacy issues in e-passports. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2005*, Athens, Greece, September 2005. IEEE.
9. Ari Juels, Ronald Rivest, and Michael Szydlo. The blocker tag: Selective blocking of RFID tags for consumer privacy. In Vijay Atluri, editor, *Conference on Computer and Communications Security – CCS'03*, pages 103–111, Washington, DC, USA, October 2003. ACM, ACM Press.
10. David Molnar and David Wagner. Privacy and security in library RFID: Issues, practices, and architectures. In Birgit Pfitzmann and Peng Liu, editors, *Conference on Computer and Communications Security – CCS'04*, pages 210–219, Washington, DC, USA, October 2004. ACM, ACM Press.
11. Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *International Workshop on Security Protocols*, pages 172–194, 1999.
12. Stephen Weis. Security and privacy in radio-frequency identification devices. Master thesis, Massachusetts Institute of Technology (MIT), Massachusetts, USA, May 2003.
13. Kaan Yüksel. Universal hashing for ultra-low-power cryptographic hardware applications. Master thesis, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, April 2004.