RFID Privacy based on Public-Key Cryptography
(Abstract)

Serge Vaudenay

EPFL
CH-1015 Lausanne, Switzerland
http://lasecwww.epfl.ch

Abstract. RFID systems makes it possible for a server to identify known tags in wireless settings. As they become more and more pervasive, people privacy is more and more threatened. In this talk, we list a few models for privacy in RFID and compare them. We review a few protocols. We further show that strong privacy mandates the use of public-key cryptography. Finally, we present a new cryptosystem which is dedicated to tiny hardware and which can be used to design secure RFID systems achieving strong privacy.

Note: this paper contains new definitions and results that are announced in this talk. Details and proofs will appear in future papers.

Credits: the work on RFID was done together with Salvatore Bocchetti as a part of his Master Thesis [3]. We received many suggestions from Gildas Avoine. The work on the new cryptosystem was done together with Matthieu Finiasz [4] and was extended together with Jean-Philippe Aumasson, and Willi Meier. Part of it was done in the Master Thesis of Jean-Philippe Aumasson [2].

1 RFID Schemes

We consider an environment with several participants. Some are called *systems*, others are called *tags*. Every tag is associated to a system. We say that the tag *belongs* to the system. Every tag is given an identification string ID. The purpose of RFID protocols is to design a communication protocol between a system and a tag so that the system knows whether or not the tag belongs to the system and learns the tag identification string ID when the tag belongs to the system.

Tags have memory which contains a *state*. Systems have a database which contains pairs of data associated to the tags that they own. This pair consists of the ID and a key. Systems may also have cryptographic key materials.

An RFID scheme is defined by the following processes.

– An initialization algorithm for the system. This produces cryptographic key materials (if any).

– An algorithm to set up a tag. This algorithm takes an ID as input and produces a tag key K and an initial state. The latter is the initial state of the tag. The former is inserted together with ID in the database of the system that owns the tag. Note that from this definition tags do not necessarily know their own ID and key. This may (or not) be part of the initial state though.
– A 2-party communication protocol between a system and a tag. Protocols are usually initiated by the system and produce two types of outputs on the reader side: a public output and a private output. We distinguish two types of protocol: identification protocols and authentication protocols. As for public outputs, the two kinds of protocols do the same. The private output of an identification protocol should be the tag ID if it belongs to the system or $\perp$ if it does not. Both outputs of an authentication protocol should be the tag 1 if it belongs to the system or 0 if it does not.

A protocol is *complete* if the output of the protocol is correct with high probability. Depending on the application, we may want to have a stronger security notion, namely *soundness*, which says whether an adversary can make the protocol output some wrong information. A critical issue is *privacy*, which means that protocols do not leak any information which may be used by adversaries to trace tags.

2 Adversaries

In an *attack*, one system is first initialize and an adversary can play with it. In addition to this, he can create tags with chosen *ID* which belong to the system or not. That is, the tag initialization algorithm is run, the tag with specified initial state is created, and the database of the system is updated in the case where the tag belongs to the system. Here the adversary does not see the tag key or initial state. In addition to creating new tags, the adversary can play with the system and the tags. We distinguish two kinds of tags: tags that are *free* from tags that are *drawn*. Tags can move from a free status to a drawn one and vice versa. A drawn tag is a tag which is close to the adversary so that the adversary can trace it during the entire time it is a drawn tag. For this, drawn tags are identified by a temporary identity that we call a *virtual tag*.

More concretely, we assume that the adversary has access to the following oracles.

– $\mathsf{Init}(\mathsf{ID}, b)$ initializes new (free) tags of specified ID which belongs to System or not depending on bit b.
– $\mathsf{GetTag}(\mathsf{distribution}) \to (\mathsf{vtag}_1, b_1, \ldots, \mathsf{vtag}_n, b_n)$ draws one or several free tags at random with chosen probability distribution. This oracle returns “virtual tags” names and bits telling whether they belong to the system or not.
– $\mathsf{Free}(\mathsf{vtag})$ frees a drawn tag.
– $\mathsf{Launch} \to \pi$ launches a new protocol instance with reader.
– $\mathsf{SendReader}(m, \pi) \to m'$ resp. $\mathsf{SendTag}(m, \mathsf{vtag}) \to m'$ sends protocol message m to reader resp. a drawn tag and returns the answer m' (if any). By convention, we write $\mathsf{Execute}(\mathsf{vtag}) \to (\pi, \mathsf{transcript})$ as a macro oracle call instead of one $\mathsf{Launch} \to \pi$ followed by a succession of $\mathsf{SendReader}(m_i, \pi) \to m_{i+1}$ and $\mathsf{SendTag}(m_{i+1}, \mathsf{vtag}) \to m_{i+2}$ calls. The protocol transcript is the concatenation of all messages m_i.

- $\mathsf{Result}(\pi) \rightarrow x$ tells 1 if the output of the protocol instance π is a tag ID or 0 if the output is $\perp$.
- $\mathsf{Corrupt}(\mathsf{vtag}) \rightarrow S$ corrupts a drawn tag and gets its internal state S.

We define several classes of adversaries.

- *Strong* adversaries can use the oracles as they want.
- *Forward* adversaries can only use Corrupt queries at the end of the attack. That is, a Corrupt query can only be followed by other Corrupt queries.
- *Weak* adversaries are not allowed to make Corrupt queries.
- *Narrow-strong* (resp. narrow-forward, narrow-weak) adversaries are strong (resp. forward, weak) adversaries who are not allowed to make Result queries.

3 Security of RFID Schemes

Let us consider an arbitrary adversary which can be written as follows.

```
1: $\mathsf{Init}(1,b_1)$, ..., $\mathsf{Init}(n,b_n)$
2: pick $i \in \{1,\dots,n\}$ at random
3: $(\mathsf{vtag},b) \leftarrow \mathsf{GetTag}(i)$
4: $\pi \leftarrow \mathsf{Execute}(\mathsf{vtag})$
```

The adversary creates n tags which belong or not to the system. Then, it draws one tag and runs a protocol. We say that this adversary fails iff the output of the protocol is what it is meant to be, namely i when $b_i = 1$ and $\perp$ otherwise. We say that the protocol is *complete* iff the probability of success of any of these adversaries is negligible.

Let us consider an arbitrary adversary which can be written as follows.

```
1: **for** $i = 1$ to $n$ **do**
2:    $\mathsf{Init}(i,1)$
3:    $\mathsf{vtag}_i \leftarrow \mathsf{GetTag}(i)$
4: **end for**
5: (training phase) do any oracle call except Init, GetTag, Free
6: $\pi \leftarrow \mathsf{Launch}$
7: (attack phase) do any oracle call except Init, GetTag, Free
```

We say that the adversary succeeds iff

- instance π is complete at the end of the attack phase,
- the output of π is $\mathsf{ID} \neq \perp$ (i.e. π identified a legitimate tag ID),
- tag ID did not complete a protocol run during the attack phase,
- tag ID was not corrupted.

We say that the protocol is *sound* iff the probability of success of any of these adversaries is negligible.

4 Privacy

To define privacy, we consider adversaries who output a list of virtual tags and a relation between their ID strings. The adversary wins if the ID strings of these tags satisfy the relation. Since some adversaries may win by giving trivial relations, we define the significance of an adversary by his ability to distinguish from a simulated run. More concretely, a *blinder* is an interface between the adversary and the oracles which let all queries pass except Lauch/SendReader/SendTag/Result queries whose output are simulated. The adversary "plugged" to a blinder is an adversary by itself who never queries the Lauch/SendReader/SendTag/Result oracles. The original adversary is significant if for any blinder the difference of the wining probability of the two adversaries is high.

An RFID system provides strong (resp. forward, weak, narrow-strong, narrow-forward, narrow-weak) privacy if there is no significant strong (resp. forward, weak, narrow-strong, narrow-forward, narrow-weak) adversary. The following implications are straightforward.

$$\begin{tikzcd}
\text{strong privacy} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{forward privacy} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{weak privacy} \arrow[d, Rightarrow] \\
\text{narrow-strong privacy} \arrow[r, Rightarrow] & \text{narrow-forward privacy} \arrow[r, Rightarrow] & \text{narrow-weak privacy}
\end{tikzcd}$$

5 Our Results

We can prove that

- our six privacy notions are pairwise different;
- strong privacy cannot be achieved;
- forward privacy can be achieved without public-key cryptography, in principle;
- an RFID system with narrow-strong privacy can be transformed into a secure key agreement protocol, which implies that this notion of privacy cannot be achieve without paying the price of public-key cryptography;
- a semantically secure public-key cryptosystem can be used to make an RFID system with *narrow*-strong privacy;
- a public-key cryptosystem secure against adaptive chosen ciphertext attacks can be used to make a secure RFID system with *forward* privacy and *narrow*-strong privacy.

The protocol in the last two results works as follows.

- The initialization algorithm for the system generates a public/private key pair for the cryptosystem.
- The setup algorithm for the tag generates a random key K for the tag and set the initial state to the vector including ID, K, and the public key of the system.
- The identification protocol works as depicted on Fig. 1. The system first picks a random a, sends it to the tag. Then, the tag encrypts ID, K, and a together by using the public key and sends the ciphertext to the system. Finally, the system decrypts using the private key, checks that a is correct and that ID and K are consistent with the database. The output of the protocol is ID.

$$\begin{tikzcd}
\text{Tag} & & \text{System} \\
& & \text{pick } a \arrow[ll, "a"'] \\
c = \mathsf{Enc}(\mathsf{ID}_{\mathsf{tag}} || K_{\mathsf{tag}} || a) \arrow[rr, "c"] & & \mathsf{Dec}(c), \text{ check}
\end{tikzcd}$$

Fig. 1. Identification Protocol Based on a Public-Key Cryptosystem.

6 TCHo

We present here a simple cryptosystem that can be used for tiny hardwares. We use a security parameter s which defines some parameters w, d, ℓ, $d_{\min}$, $d_{\max}$, γ, k. To make it more precise, we can take

$$w = 45 \quad d = 25\,820 \quad \ell = 50\,000 \quad d_{\min} = 5\,800 \quad d_{\max} = 7\,000 \quad \gamma = 0.9810 \quad k = 128.$$

Asymptotically, we take

$$w = \Theta(s) \quad d = \Theta(s^3) = \ell \quad d_{\min} = \Theta(s^2) = d_{\max} \quad \gamma = 1 - \Theta\left(\frac{1}{s}\right) \quad k = \Theta(s).$$

Key generation. We pick a random polynomial K over $\mathsf{GF}(2)$ of degree d with constant factor 1 and weight w until it has a primitive factor P of degree d_P in $[d_{\min}, d_{\max}]$. The public key is P. It is of length at most $d_{\max} = O(s^2)$ The private key is K. It is of length at most $w \log_2 d = O(s \log s)$ The complexity is $O(s^6 \log s \log \log s)$.

Encryption. To encrypt a k-bit plaintext, we set an LFSR with characteristic polynomial P to a random string, we produce a bit stream of length ℓ. We XOR it to ℓ/k repetitions of the plaintext. We XOR it again to the output of a random source producing ℓ independent bits with bias γ. The result is the ciphertext: a bit-string of length ℓ. Encryption is depicted on Fig. 2. The complexity is $O(s^5)$. The plaintext is of length $k = \Theta(s)$. The ciphertext is of length $\ell = O(s^3)$.

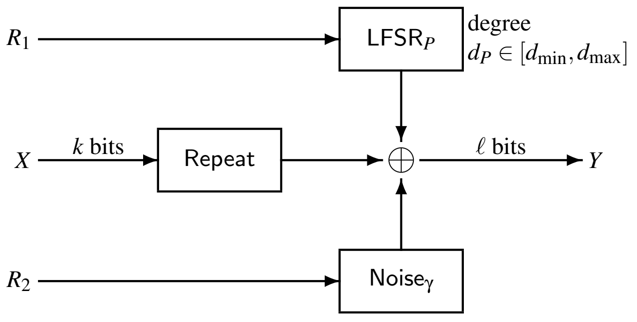

Fig. 2. TCHo Encryption.

Decryption. To decrypt, we make combination of the ciphertext bits by using K. We obtain $\ell - d$ bits. We do majority logic decoding and recover the plaintext. The complexity is $O(s^4)$.

Reliability, performances, and security. Heuristic arguments show that the probability that decryption does not produce the right plaintext is less than $9 \cdot 10^{-9}$ for our parameters. (Asymptotically, this probability is $\exp\left(-\Omega(s^2)\right)$, heuristically.) The cost to implement encryption in RFID tags relates to the cost of a random generator and an LFSR of d_P bits (that is at most 7 000 gates here). We can encrypt arbitrary messages using hybrid encryption, requiring an additional symmetric encryption scheme. Our scheme is semantically secure provided that

- it is hard to find a multiple of a polynomial of degree d_P with weight w and degree d when such polynomial exists;
- we cannot distinguish the output of the LFSR of length d_P XORed to biased bits of bias γ from a uniformly distributed string.

So far, the best algorithm to break semantic security in TCHo with our parameters vector has an advantage/complexity ratio of 2^{-65} with pessimistic estimates. Asymptotically, this best ratio is $\exp(-\Omega(s))/s^2$.

7 IND-CCA Construction

Given two random oracles F and H, the [1] construction based on tag-KEM/DEM transforms TCHo into an IND-CCA secure public-key cryptosystem. To encrypt a plaintext X with random bits σ, we compute $y = X + F(\sigma)$ (the *data encapsulation*) and the TCHo encryption of σ with random bits $(R_1, R_2) = H(\sigma, y)$ (the *key encapsulation* with tag y). The ciphertext is the TCHo ciphertext concatenated with y. Namely,

$$\mathsf{Enc}(X;\sigma) = \mathsf{TCHo.enc}(X; H(\sigma, y)) \,||\, y.$$

To decrypt, we first decrypt the TCHo ciphertext to recover σ. We subtract $F(\sigma)$ to y and get the plaintext X. Additionally, we check that the TCHo encryption used the right $H(\sigma, y)$ random bits.

References

1. M. Abe, R. Gennaro, K. Kurosawa, V. Shoup. Tag-KEM/DEM: a New Framework for Hybrid Encryption and a New Analysis of Kurosawa-Desmedt KEM. In *Advances in Cryptology EUROCRYPT'05*, Aarhus, Denmark, Lecture Notes in Computer Science 3494, pp. 128–146, Springer-Verlag, 2005.
2. J.-P. Aumasson. A Novel Asymmetric Scheme with Stream Cipher Construction. Master Thesis. 2006. http://lasecwww.epfl.ch/abstracts/abstract_tcho.shtml
3. S. Bocchetti. Security and Privacy in RFID Protocols. Master Thesis. 2006. http://lasecwww.epfl.ch/abstracts/abstract_RFID_bocchetti.shtml
4. M. Finiasz, S. Vaudenay. When Stream Cipher Analysis Meets Public-Key Cryptography. (Invited Talk.) To appear in the proceedings of SAC' 2006. Lecture Notes in Computer Science, Springer. 2006.