

Scalability Issues in RFID Systems

Gildas Avoine

EPFL, Lausanne, Switzerland



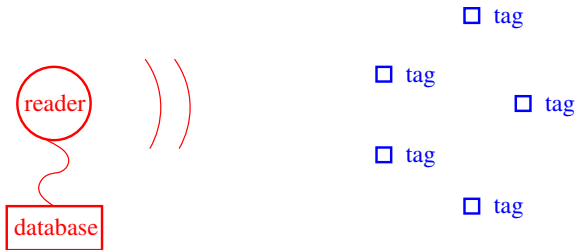
Yet Another Introduction to RFID

Existing Challenge-Response Protocols

Reducing Computation Complexity

Yet Another Introduction to RFID

Identify objects remotely by embedding in these objects tiny devices (**tags**) capable of transmitting data.

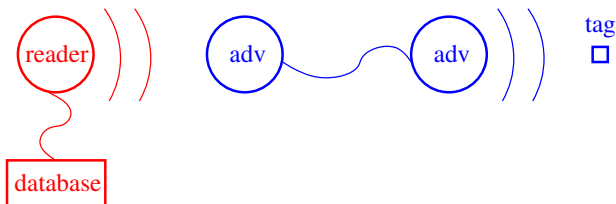


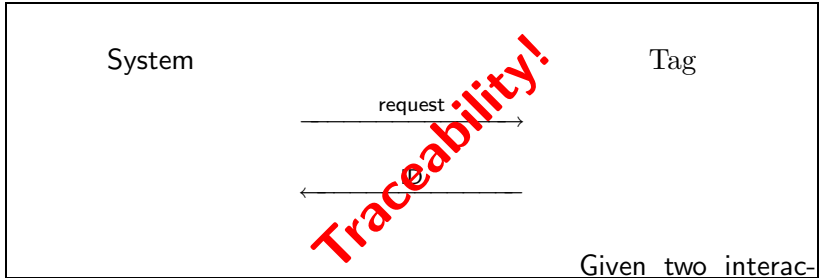
- RFID is **not new** but its **growth explodes** for a few years.
- The target of the **Auto-ID Center** (now EPC Global and Auto-ID Labs) is the improvement of the supply chains.

- Management of stocks (Wal-Mart, US DoD, etc.)
- Libraries (Santa Clara Library, KU Leuven, etc.)
- Anti-counterfeiting (luxury articles, etc.)
- Pets identification
- Sensor networks (Michelin's tyres, etc.)
- Automobile ignition keys (Texas Instruments, etc.)
- Access control (Famous Baja Beach Club, etc.)
- Localization of people (Amusement parks, etc.)
- Electronic documents (Passports, etc.)
- Transport Ticketing (Metro in Paris, etc.)

Identification: we want to know who we are speaking with (no proof required)

Authentication: we want to be sure that we are speaking with the correct party (proof required) (impersonation based on replay attacks, tag cloning, relay attack)





Easier to trace with RFID than other technologies e.g. video, credit cards, GSM, Bluetooth.

- Tags cannot be **switched-off**
- Tags answer **without the agreement** of their bearers
- Tags can be almost **invisible**
- Easy to **analyze the logs** of the readers
- Increasing of the **communication range**

Even if you do not think that **privacy is important**, some people think so and they are rather influential (CASPIAN, FoeBud, etc.)

Example: Boycotts against Gillette, Tesco, Metro, etc.

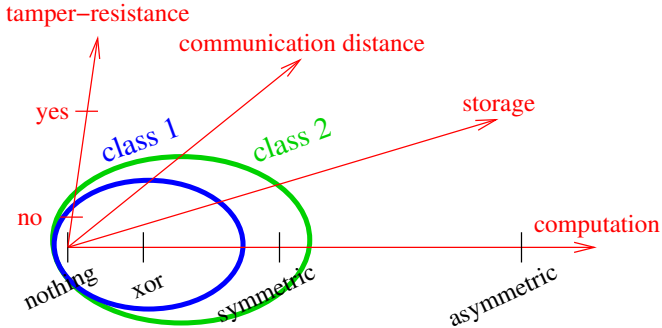
Palliative techniques exist like **kill-keys**, **blocker tags** and **Faraday cages**.

During the XVI century, soldiers already protected their privacy using Faraday cages...

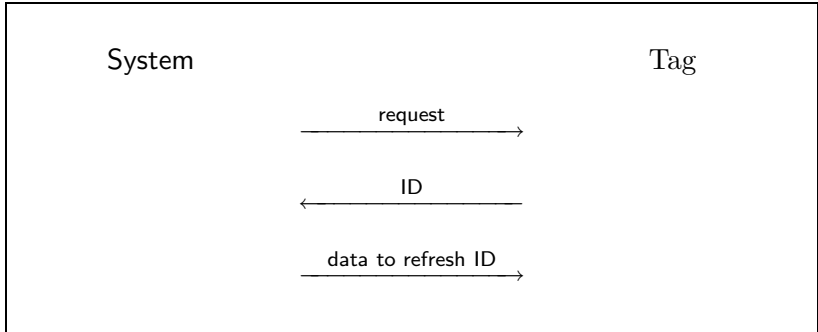


Landesmuseum Joanneum,
Landeszeughaus, Graz.

How designing an RFID protocol such that only an authorized party is able to **identify** (or **authenticate**) a tag while an adversary is neither able to **identify** it nor to **trace** it?

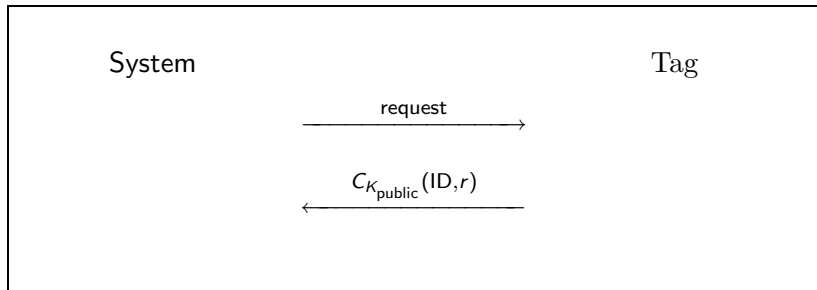


The boom which RFID technology is enjoying today relies essentially on the willingness to develop **small** and **cheap** RFID tags.

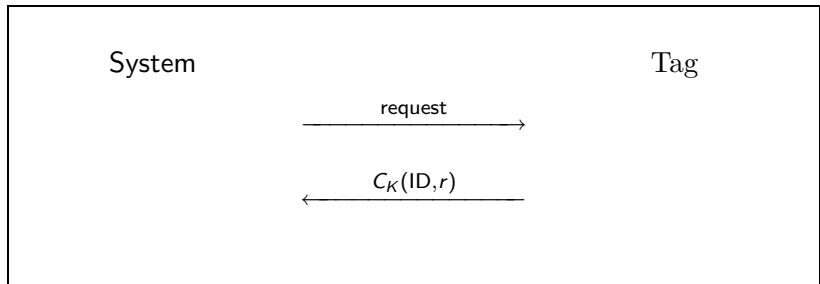


The basic idea is that each time the tag is queried, the information it sends is renewed **with the help of the reader.**

If we were able to use asymmetric cryptography on the tag's side



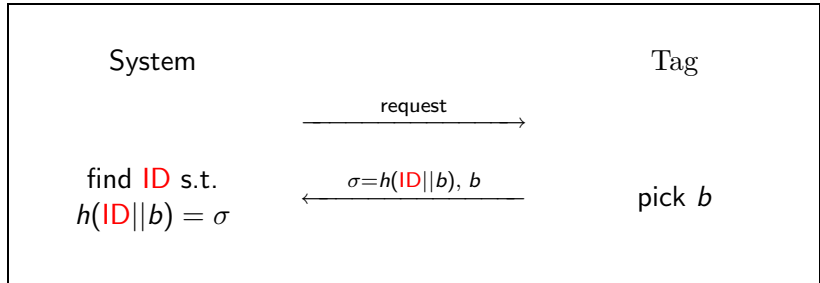
Is it really different when symmetric cryptography is used?



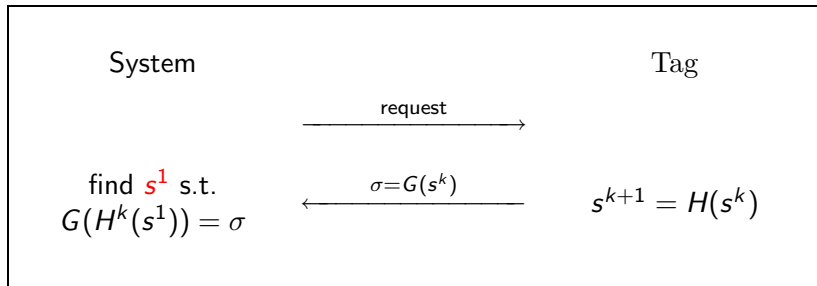
- Tag are **not tamper-resistant**: using the same key for all tags is not secure.
- Every tag should have a unique key:
 - One system / one tag (eg. automobile ignition key): Identifying one tag requires $O(1)$ operations
 - One system / n tags (eg. library): Identifying one tag requires $O(n)$ operations (exhaustive search) and identifying the whole system requires $O(n^2)$ operations.
This approach differs from all the other authentication protocols because we usually assume in cryptography that the verifier knows the identity of the prover before the protocol starts.

Existing Challenge-Response Protocols

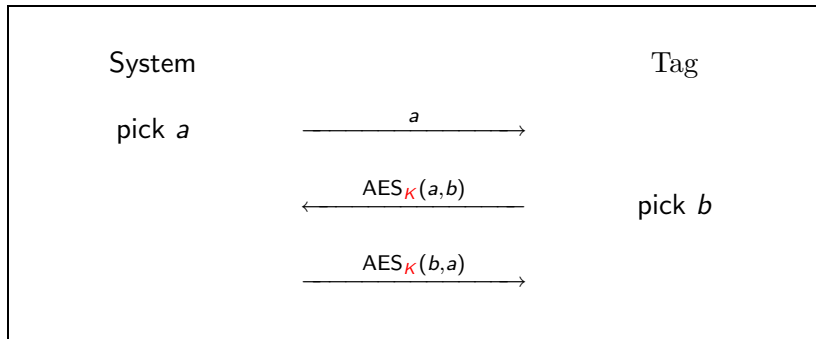
- When a tag is queried, it sends back an information which enables the system to retrieve its identifier.
- This information is refreshed each time the tag is queried.
- This information can be seen as a pseudonym.
- What differentiates the existing protocols is the technique which is used to refresh these pseudonyms.



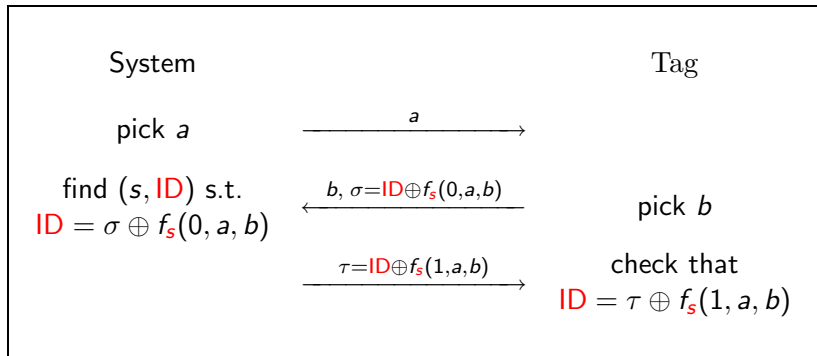
Protocol of Weis, Sarma, Rivest, and Engels
(CHES 2002)



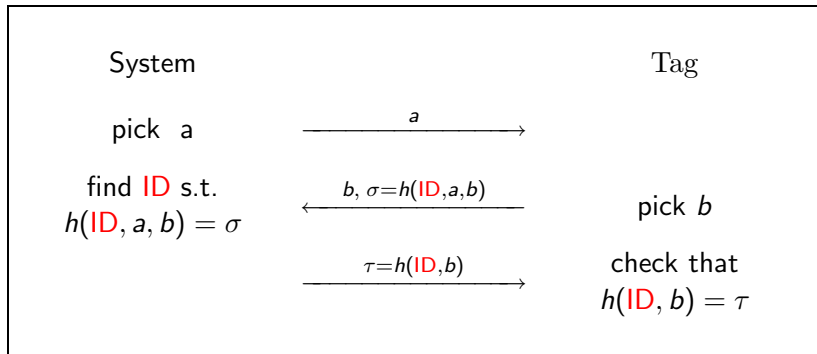
Protocol of Ohkubo, Suzuki, and Kinoshita
(RFID Privacy Workshop 2003)



Protocol of Feldhofer, Dominikus, and Wolkerstorfer
(CHES 2004)



Protocol of Molnar and Wagner
(CCS 2004)

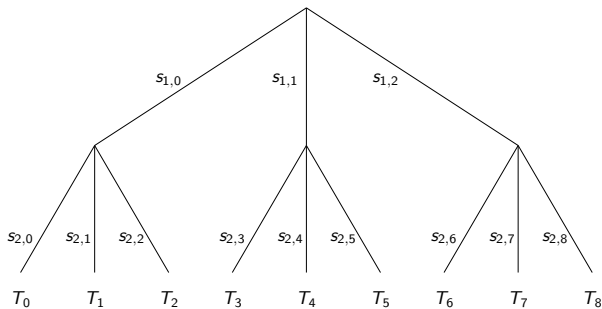


Protocol of Rhee, Kwak, Kim, and Won
(ESAS 2005)

(Almost) All these protocols suffer from a linear complexity (in order to identify one tag).

Reducing Computation Complexity

Molnar and Wagner
Tree-Based Technique



n : number of tags in the system

δ : branching factor

ℓ : depth of the tree = $\log_{\delta}(n)$

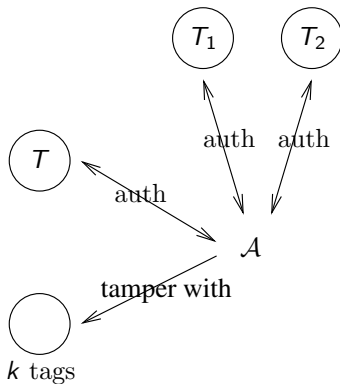
- Rely on an underlying (almost) generic mutual authentication protocol (eg. those of Molnar and Wagner presented a few slides ago)
- The underlying protocol is applied ℓ times (one per level of the tree)
- If the authentication succeeds in a given level, it goes onto the next level

Identifying one tag requires $\delta \log_{\delta}(n)$ operations instead of n .

Identifying the whole system requires $n\delta \log_{\delta}(n)$ operations instead of n^2 .

Example: in a library, we consider 2^{20} tagged books. We assume that the system can carry out 2^{23} operations per second. Identifying one tag requires **0.1 milliseconds** ($\delta = 1024$) and identifying the whole system requires about **2 seconds** with $\delta = 2$ or **2 minutes** with $\delta = 1024$.

- 1 The tags share parts of secrets.
- 2 If an attacker tampers with tags, she therefore obtains some parts of secrets.



Probability that the attack succeeds is therefore

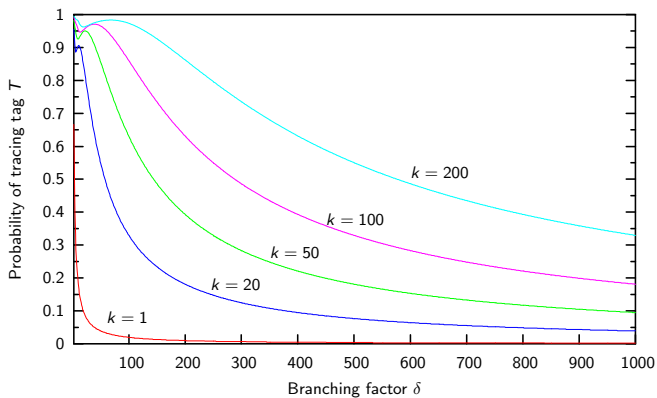
$$\frac{k_1}{\delta^2} (2\delta - k_1 - 1) + \sum_{i=2}^{\ell} \left(\frac{k_i}{\delta^2} (2\delta - k_i - 1) \prod_{j=1}^{i-1} \frac{k_j}{\delta^2} \right),$$

where

$$k_1 = \delta \left(1 - \left(1 - \frac{1}{\delta} \right)^k \right) \quad k_{i>1} = \delta \left(1 - \left(1 - \frac{1}{\delta} \right)^{g(k_i)} \right)$$

and

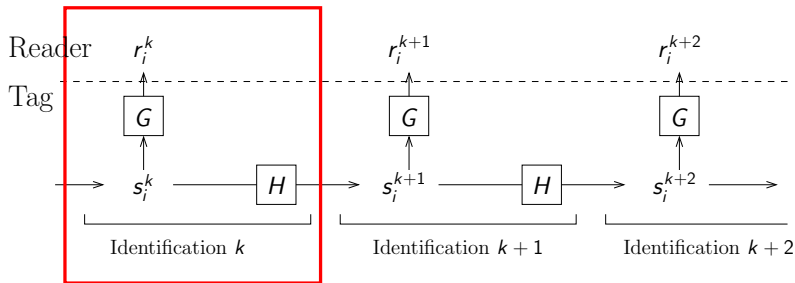
$$g(k_i) = k \prod_{j=1}^{i-1} \frac{1}{k_j}.$$



$k \backslash \delta$	2	20	100	500	1000
1	66.6%	9.5%	1.9%	0.3%	0.1%
20	95.5%	83.9%	32.9%	7.6%	3.9%
50	98.2%	94.9%	63.0%	18.1%	9.5%
100	99.1%	95.4%	85.0%	32.9%	18.1%
200	99.5%	96.2%	97.3%	55.0%	32.9%

Avoine and Oechslin
Time-Memory Trade-Off Technique

- Each tag needs 2 **hash functions** G and H (in theory).
- Each tag needs an **EEPROM** capable of storing an identifier.
- The personalisation of a tag T_i consists in storing in its memory a random identifier s_i^1 , which is also recorded by the database of the system.
- Thus, the database initially contains the set $\{s_i^1 \mid 1 \leq i \leq n\}$.



From each n initial identifiers s_i^1 , the system computes the hash chains until it finds r_i^k or until it reaches a given maximum limit m on the chain length.

$$\begin{array}{ccccccc}
 s_1^1 & \rightarrow & r_1^1 & r_1^2 & \dots & \dots & r_1^{m-1} & r_1^m \\
 s_2^1 & \rightarrow & r_2^1 & r_2^2 & \dots & \dots & r_2^{m-1} & r_2^m \\
 \vdots & \rightarrow & \dots & \dots & \dots & \dots & \dots & \vdots \\
 s_i^1 & \rightarrow & \dots & \dots & \boxed{r_i^k = G(H^{k-1}(s_i^1))} & \dots & \dots & r_i^m \\
 \vdots & \rightarrow & \dots & \dots & \dots & \dots & \dots & \vdots \\
 s_n^1 & \rightarrow & r_n^1 & r_n^2 & \dots & \dots & r_n^{m-1} & r_n^m
 \end{array}$$

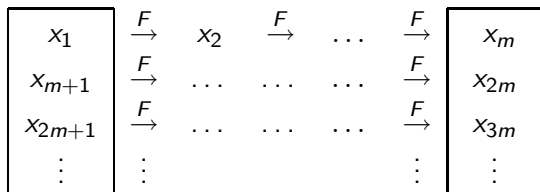
The **average** complexity in terms of **hash operation** is mn (if the tag is owned by the system) because two hash operations are carried out $mn/2$ times.

The **general idea** of time-memory trade-offs is to reduce the time complexity by adding memory (or the inverse).

Example: exhaustive search on a one-way function $F: X \rightarrow X$ in order to find preimages. $N := |X|$

Extreme cases:

Storage: 0	Pre-computation: 0	On-line computation: N
Storage: N	Pre-computation: N	On-line computation: 0



Only the **first** and the **last** element of each chain is stored.

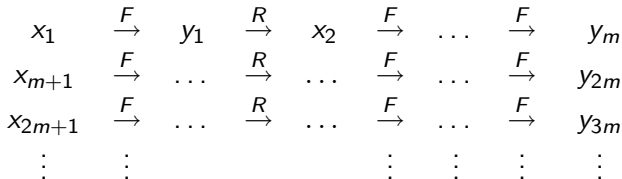
Given one output x_i of F that we need to invert, we generate a chain starting at x_i : $x_i \xrightarrow{F} x_{i+1} \xrightarrow{F} x_{i+2} \xrightarrow{F} \dots$

Each time we compute a new element, we check whether or not it is the **end of a chain** in the table.

We can then regenerate the complete chain and find x_{i-1} .

In practice, **inputs** and **outputs** of F are not in the same set X but $F: X \rightarrow Y$.

We need a **reduction** function $R: Y \rightarrow X$ that generates an arbitrary input of F from one of its outputs.



The computation time is $T \propto N^2/M^2$

In our case the function F is

$$F : (i, k) \mapsto r_i^k = G(H^{k-1}(s_i^1))$$

where $1 \leq i \leq n$ and $1 \leq k \leq m$.

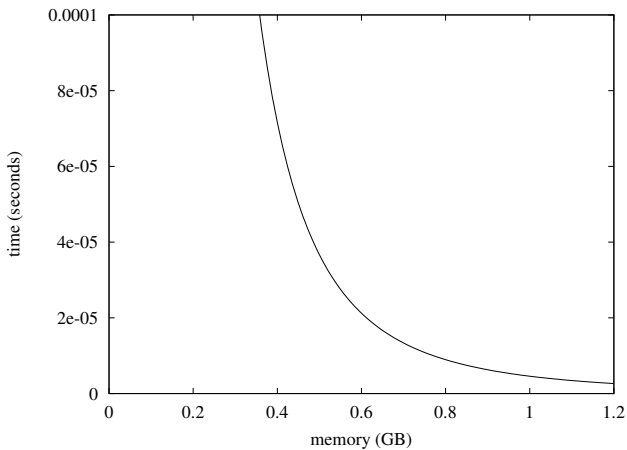
We need an arbitrary reduction function R ,

$$R : r_i^k \mapsto (i', k')$$

where $1 \leq i' \leq n, 1 \leq k' \leq m$. For example, we take

$$R(r) = (1 + (r \bmod n), 1 + (\lfloor \frac{r}{n} \rfloor \bmod m)).$$

Example: in a library, we consider 2^{20} tagged books. The length of the hash chains is 2^7 . We assume that the system can carry out 2^{23} hash operations per second. The system has **1.25 GB RAM**. Identifying one tag requires 0.002 milliseconds and identifying the whole system requires about **2 seconds**. Precomputations require 17 minutes.



Conclusion

- Designing cryptographic building blocks which are lightweight in terms of computation, communication, and memory.
- Thinking about alternative RFID protocols in order to reduce the computation complexity.

More information on Security and Privacy in RFID Systems

<http://lasecwww.epfl.ch/~gavoine/rfid/>