

# Can we use proxy-readers to relieve back-end RFID systems?

**Gildas Avoine**

MIT, USA

<http://www.avoine.net>

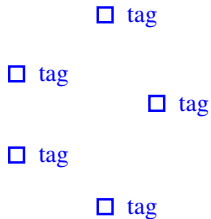
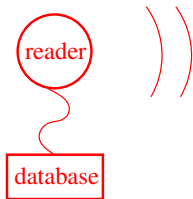


Framework

Challenge/Response Protocols

Proxy-Readers in Public Transportation

# Framework



- Tags are **low-cost** (passive, memory, comm. range, etc.)
- Tags can use **symmetric-key** cryptography only (vs public-key)
- Tags are **not tamper-resistant** (but attacks expensive)

- Security

- Prover authenticated iff he knows right key (auth vs ident)
- Resistant to compromised keys

- Privacy

- Identity is not revealed
- Resistant to malicious traceability

- Efficiency

- Lightweight i.e. implementable on low-capabilities provers
- Reasonable amount of computation (prover/verifier)

## Challenge/Response Protocols

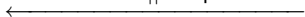
Verifier

database	
$Id_1$	$K_1$
$Id_2$	$K_2$
$\vdots$	$\vdots$
$Id_i$	$K_i$
$\vdots$	$\vdots$
$Id_n$	$K_n$

Challenge



Identifier || Response

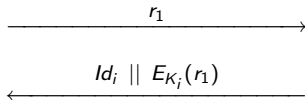


Prover ( $Id_i, K_i$ )

Verifier

database	
$Id_1$	$K_1$
$Id_2$	$K_2$
$\vdots$	$\vdots$
$Id_i$	$K_i$
$\vdots$	$\vdots$
$Id_n$	$K_n$

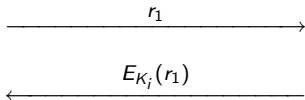
Prover ( $Id_i, K_i$ )



Verifier

database	
$Id_1$	$K_1$
$Id_2$	$K_2$
$\vdots$	$\vdots$
$Id_i$	$K_i$
$\vdots$	$\vdots$
$Id_n$	$K_n$

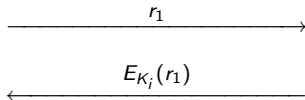
Prover ( $Id_i, K_i$ )



Verifier

database	
$Id_1$	$K_1$
$Id_2$	$K_2$
$\vdots$	$\vdots$
$Id_i$	$K_i$
$\vdots$	$\vdots$
$Id_n$	$K_n$

Prover ( $Id_i, K_i$ )

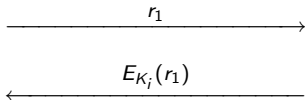


- The verifier must **check every key** of the database.

Verifier

database	
$Id_1$	$K_1$
$Id_2$	$K_2$
$\vdots$	$\vdots$
$Id_i$	$K_i$
$\vdots$	$\vdots$
$Id_n$	$K_n$

Prover ( $Id_i, K_i$ )



- The verifier must **check every key** of the database.
- The prover must **randomize** his response.

Verifier

database	
$Id_1$	$K_1$
$Id_2$	$K_2$
$\vdots$	$\vdots$
$Id_i$	$K_i$
$\vdots$	$\vdots$
$Id_n$	$K_n$

Prover ( $Id_i, K_i$ )

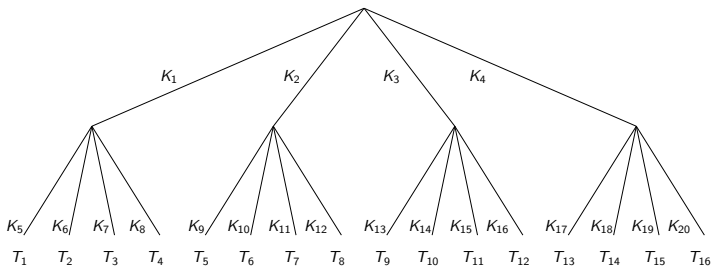
$\xrightarrow{r_1}$

$\xleftarrow{E_{K_i}(r_1, r_2)}$  pick  $r_2$

- The verifier must **check every key** of the database.
- The prover must **randomize** his response.

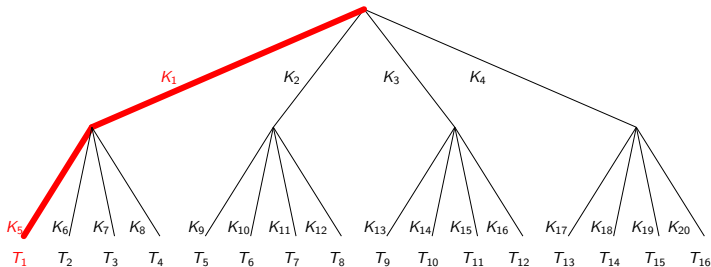
# Molnar and Wagner Tree-Based Authentication

- Molnar and Wagner [CCS'04] suggested a tree-based technique to reduce the complexity from  $O(n)$  to  $O(\log n)$ .
- The set of tags is recursively divided into **subsets**. Each subgroup has its own key.



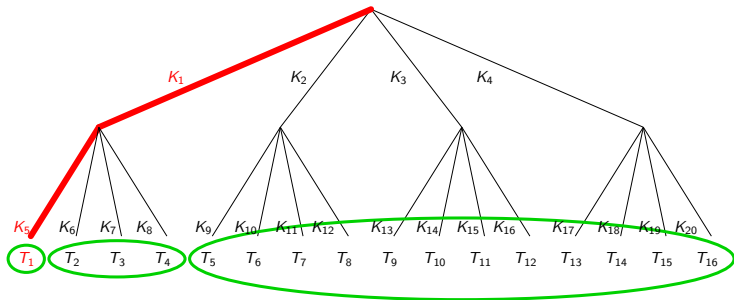
# Molnar and Wagner Tree-Based Authentication

- Molnar and Wagner [CCS'04] suggested a tree-based technique to reduce the complexity from  $O(n)$  to  $O(\log n)$ .
- The set of tags is recursively divided into **subsets**. Each subgroup has its own key.

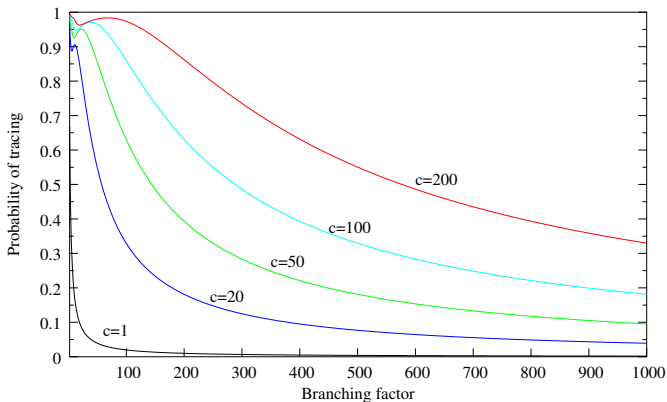


# Molnar and Wagner Tree-Based Authentication

- Molnar and Wagner [CCS'04] suggested a tree-based technique to reduce the complexity from  $O(n)$  to  $O(\log n)$ .
- The set of tags is recursively divided into **subsets**. Each subgroup has its own key.



Avoine, Dysli, and Oechslin [SAC'05] proved that the **tree-based** approach is not secure.



$$(n = 2^{20})$$

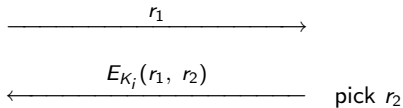
- Avoine and Oechslin [PerSec'05] suggested to replace the online calculations by offline calculations (**precalculations**).
- Space of  $r_2$  too large, so must be **predictable**.
- Too costly in terms of **storage** due to desynchronization.
- Time-memory **trade-off**.
- Challenge/Response, [OhkuboSK, 2003, MIT], and TMTO.

# Replacing Online calculations by Offline Calculations

Verifier

database	
$Id_1$	$K_1$
$Id_2$	$K_2$
$\vdots$	$\vdots$
$Id_i$	$K_i$
$\vdots$	$\vdots$
$Id_n$	$K_n$

Prover ( $Id_i, K_i$ )



# Replacing Online calculations by Offline Calculations

Verifier

database	
$Id_1$	$K_1$
$Id_2$	$K_2$
$\vdots$	$\vdots$
$Id_i$	$K_i$
$\vdots$	$\vdots$
$Id_n$	$K_n$

Prover ( $Id_i, K_i$ )

$\xrightarrow{r_1}$

$\xleftarrow{E_{K_i}(r_1, r_2) \parallel r_2}$

compute  
 $r_2 := PRNG(r_2)$

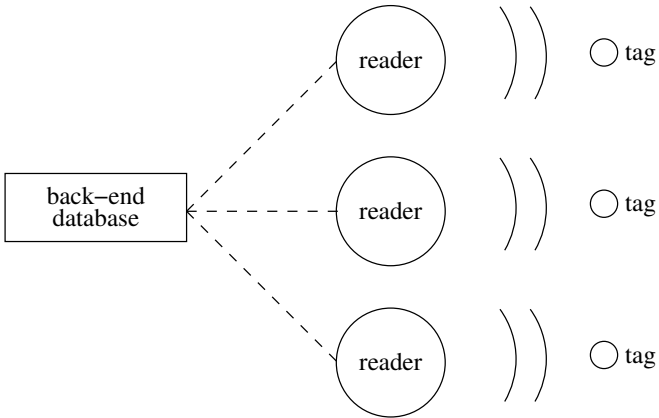
Method	Time (millisecond)
Without Precomputation	16'000
With Precomputation (342 MB)	0.122
With Precomputation (1.25 GB)	0.002

$(n = 2^{20}, m = 2^{10})$

There does not exist efficient private authentication protocols

Can we find a better protocol suited to some specific applications, which may allow to relax some assumptions?

## Proxy-Readers in Public Transportation



- Readers are **processing subsystem** by themselves
- Authentication based on **access passes** (eg monthly pass)
- Readers cannot be **permanently connected** to the backend (eg readers in buses, server failures)
- The system manager carries out **maintenance operations** on a regular basis (eg daily **period**)
- Readers can connect to the backend through a **secure channel** during the maintenance periods
- **Compromized readers** can be detected during during the maintenance operations or sooner.
- System manager is **trusted** by the client

## Property 1

## Authentication

When neither the readers nor the tags are corrupted, the system must provide authentication.

## Property 2

## Untraceability

When neither the readers nor the tags are corrupted, the system must ensure untraceability.

## Property 3

## Tag Compromise

If some tag is corrupted, this should not allow an adversary to impersonate or trace any other tag at any time.

## Property 4

## Tag Clone

If some tag is corrupted and cloned, the fake tag must be detected after a reasonable number of maintenance operations, depending on the frequency the fake tag is used.

## Property 5

## Efficiency

The system must be efficient meaning that the authentication processing time must be less than 100ms

## Property 6

## Reader Compromise

If some reader is corrupted during a given period, the system should be safe again after the maintenance operation following this period, without any replacement or update of the tags. This means that an adversary can impersonate and trace tags only during this period.

## Property 1

## Authentication

When neither the readers nor the tags are corrupted, the system must provide authentication.

- Challenge/response

## Property 2

## Untraceability

When neither the readers nor the tags are corrupted, the system must ensure untraceability.

- Private challenge/response i.e. randomized by the tag

### Property 3

### Tag Compromise

If some tag is corrupted, this should not allow an adversary to impersonate or trace any other tag at any time.

- Data stored by the tags are not linked (vs Molnar and Wagner approach)

## Property 4

## Tag Clone

If some tag is corrupted and cloned, the fake tag must be detected after a reasonable number of maintenance operations, depending on the frequency the fake tag is used.

- Statistical approach  
(cf body of literature about ad hoc networks)

## Property 5

## Efficiency

The system must be efficient meaning that the authentication processing time must be less than 100ms

- Load is shared between the readers
- Customers almost always take / get off the bus at the same stations
- Subsets of tags: the tag sends the identifier of its group
  - Variant of Molnar and Wagner (but more efficient)
  - If the number of subsets is small, is the privacy preserved?

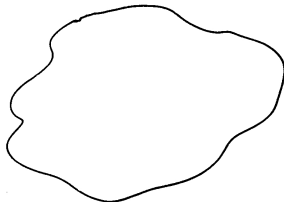
- Each reader stores the **full list** of identifiers but **sorts** them according to the number of times they interacted with **this** reader (or set of readers) eg within a sliding windows of one month.
- **Timing attack** if the adversary can eavesdrop **legitimate** communications.
- No problem if the adversary can only play with the tag or the reader **independently**.

**n=600 000 tags**



**Boston**

**n'=3 millions tags**



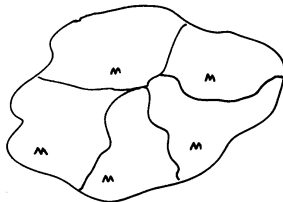
**London**

**n=600 000 tags**



**Boston**

**n'=3 millions tags**



**London**

- 1 The adversary **is challenged with** a target and interacts with it.
- 2 The set of tags is shuffled.
- 3 The adversary **must** propose a tag as being her target.

- 1 The adversary **chooses** a target and interacts with it.
- 2 The set of tags is shuffled.
- 3 The adversary **must** propose a tag as being her target.

- 1 The adversary **is challenged with** a target and interacts with it.
- 2 The set of tags is shuffled.
- 3 The adversary **can** propose a tag as being her target, **iff he definitely found her target**.

- 1 The adversary **chooses** a target and interacts with it.
- 2 The set of tags is shuffled.
- 3 The adversary **can** propose a tag as being her target, **iff he definitely found her target**.

- 1 The adversary **chooses** a target and interacts with it.
- 2 A challenger provides 2 tags to the attacker (one is the target).
- 3 The adversary **can** propose a tag as being her target, **iff he definitely found her target**.

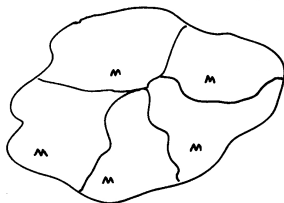
What happens if we consider the full metropolitan area, not only the RFID holders?

**n=600 000 tags**



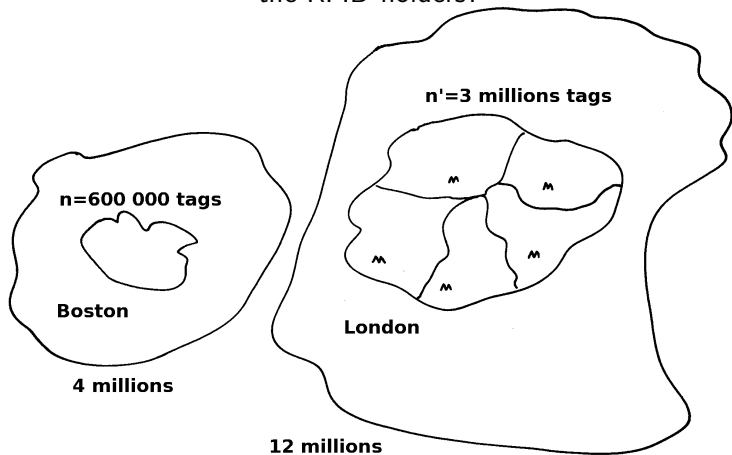
**Boston**

**n'=3 millions tags**



**London**

What happens if we consider the full metropolitan area, not only the RFID holders?



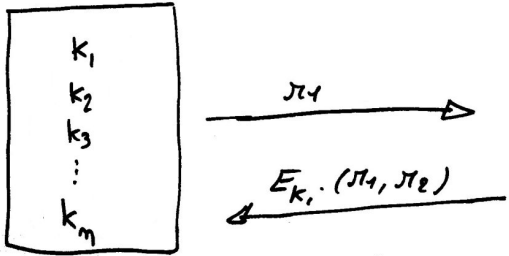
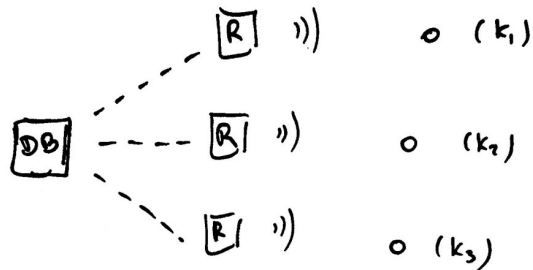
Model	Boston	London	London splitted
1-a/b (cha/cho must)	1 / 600k	1 / 3m	1 / 600k
1-c/d (cha/cho can)	–	–	1 (80%)
2-a/b (cha/cho must)	.5	.5	1 (80%), .5 (20%)
2-c/d (cha/cho can)	–	–	1 (80%)
2-d (cho can variant)	1 (85%)	1 (75%)	1 (95%)
2-d (cha can variant)	1 (25%)	1 (37%)	1 (42%)

## Property 6

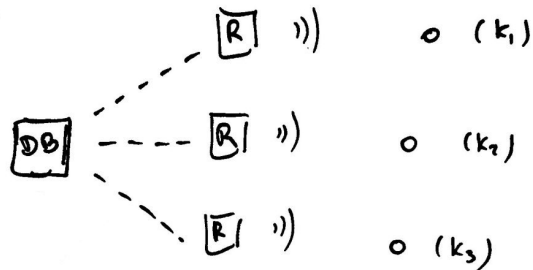
## Security Update

If some reader is corrupted during a given period, the system should be safe again after the maintenance operation following this period, without any replacement or update of the tags. This means that an adversary can impersonate and trace tags only during this period.

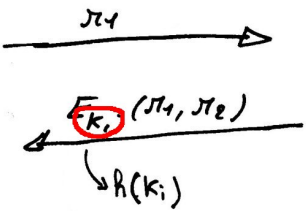
# Requirements Detailed



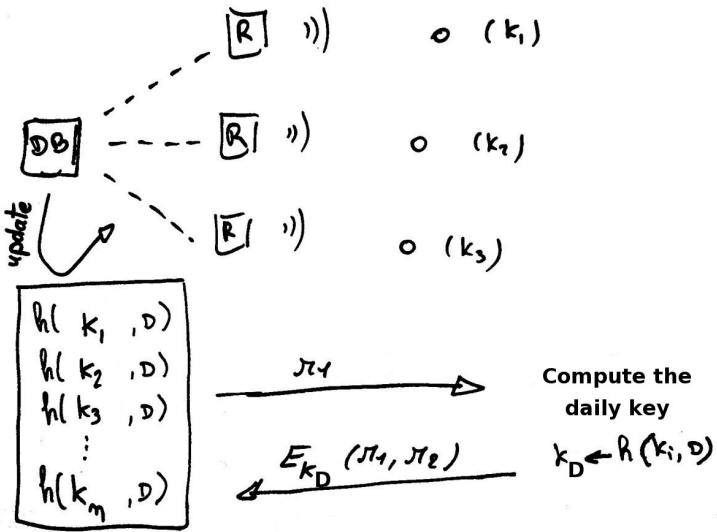
# Requirements Detailed



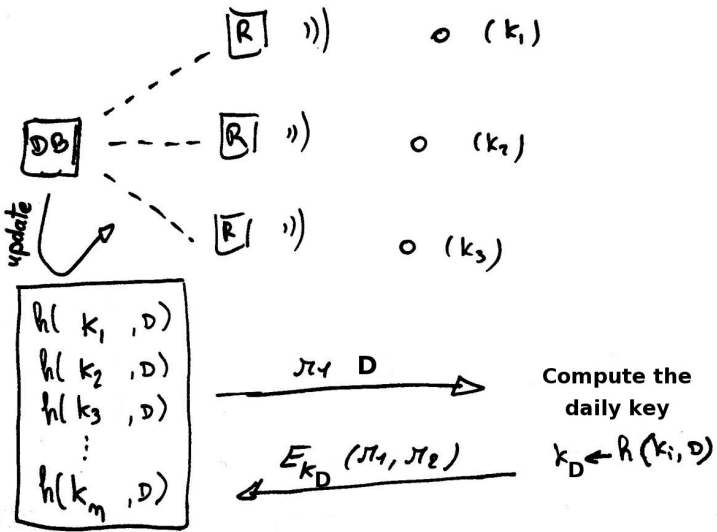
- h(k<sub>1</sub>)
- h(k<sub>2</sub>)
- h(k<sub>3</sub>)
- ⋮
- h(k<sub>m</sub>)



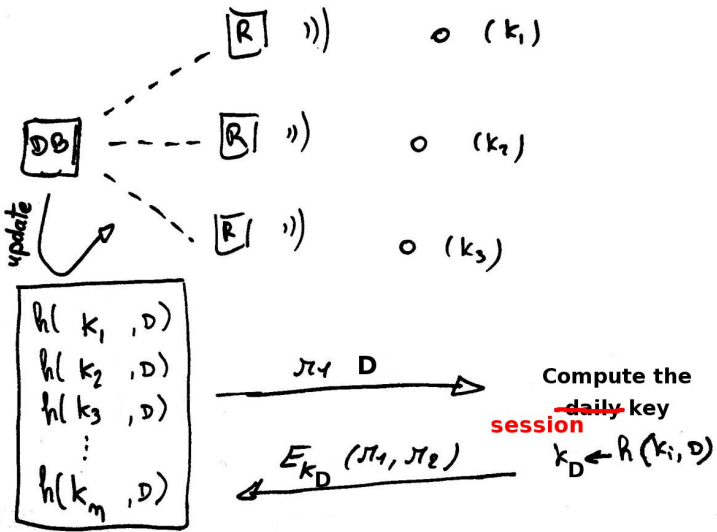
# Requirements Detailed



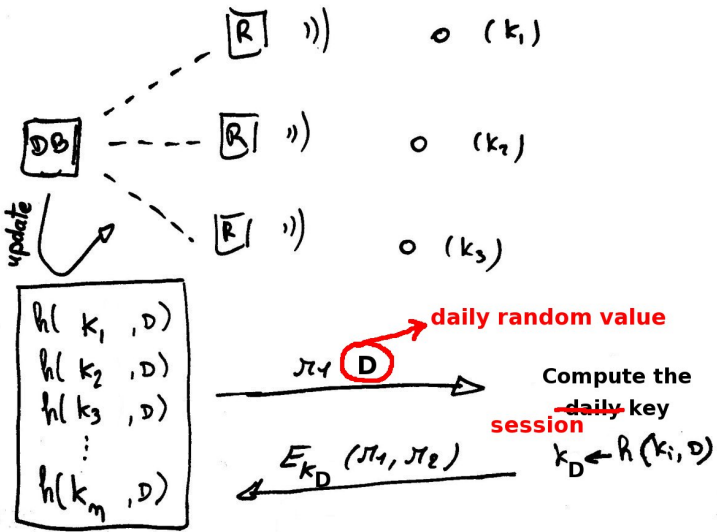
# Requirements Detailed



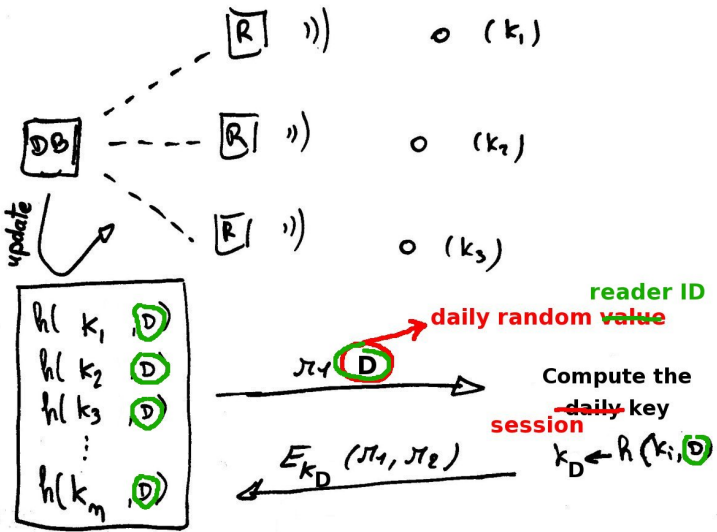
# Requirements Detailed



# Requirements Detailed



# Requirements Detailed



## Summary and Conclusion

- Proxy-readers may be a new approach to relieve the back-end database.
- Proxy-readers improve the resilience of the system.
- Does this approach make sense in practice?
- Do proxy-readers already exist?