Secret Shuffling: A Novel Approach to RFID Private Identification

Claude CASTELLUCCIA, Mate SOOS

INRIA team PLANETE, INRIA Rhône-Alpes

September 5, 2010

Table of Contents

Identification, Authentication, Private communication

What and why?

- Identification: Helps to choose the correct key(certificate, etc.) to authenticate the other party

Identification, Authentication, Private communication

What and why?

- Identification: Helps to choose the correct key(certificate, etc.) to authenticate the other party
- Authentication: Helps to be sure who we are talking to

Identification, Authentication, Private communication

What and why?

- Identification: Helps to choose the correct key(certificate, etc.) to authenticate the other party
- Authentication: Helps to be sure who we are talking to
- Private communication: Helps to hide messages' content

Our solution is a private identification solution. Private identification solutions until now:

Identification, Authentication, Private communication

What and why?

- Identification: Helps to choose the correct key(certificate, etc.) to authenticate the other party
- Authentication: Helps to be sure who we are talking to
- Private communication: Helps to hide messages' content

Our solution is a private identification solution. Private identification solutions until now:

- Hash-lock based: tree-like, synchronisation-type, mixed

Identification, Authentication, Private communication

What and why?

- Identification: Helps to choose the correct key(certificate, etc.) to authenticate the other party
- Authentication: Helps to be sure who we are talking to
- Private communication: Helps to hide messages' content

Our solution is a private identification solution. Private identification solutions until now:

- Hash-lock based: tree-like, synchronisation-type, mixed
- Intelligent systems outside the tag: non-authorised readers are not permitted to send identification requests. E.g. RFID blocker tag

Identification, Authentication, Private communication

What and why?

- Identification: Helps to choose the correct key(certificate, etc.) to authenticate the other party
- Authentication: Helps to be sure who we are talking to
- Private communication: Helps to hide messages' content

Our solution is a private identification solution. Private identification solutions until now:

- Hash-lock based: tree-like, synchronisation-type, mixed
- Intelligent systems outside the tag: non-authorised readers are not permitted to send identification requests. E.g. RFID blocker tag
- Ultra-lightweight crypto-primitives: lightweight implementations of ECC, AES, and totally new primitives (e.g. Vajda&Buttyán)

Protocol description

Protocol setup:

- Each tag has a constant, random K long key, k_i, that is a unique bitstring($k_i[1] \ldots k_i[K]$) for each tag $\mathcal{T}_i$

Protocol description

Protocol setup:

- Each tag has a constant, random K long key, k_i, that is a unique bitstring($k_i[1] \dots k_i[K]$) for each tag $\mathcal{T}_i$
- The reader knows all tag's keys

Protocol description

Protocol setup:

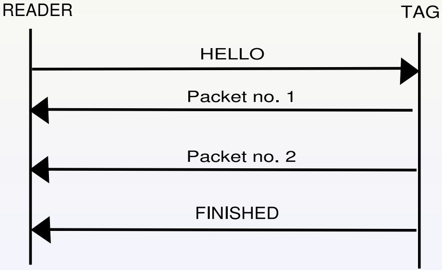
- Each tag has a constant, random K long key, k_i, that is a unique bitstring($k_i[1] \dots k_i[K]$) for each tag $\mathcal{T}_i$
- The reader knows all tag's keys
- There are far less tags, n, in the system than there are possible keys, 2^K: $n \ll 2^K$

Protocol description

Protocol setup:

- Each tag has a constant, random K long key, k_i, that is a unique bitstring$(k_i[1] \ldots k_i[K])$ for each tag $\mathcal{T}_i$
- The reader knows all tag's keys
- There are far less tags, n, in the system than there are possible keys, 2^K: $n \ll 2^K$

How it works:



Description of a packet

Description of a packet:

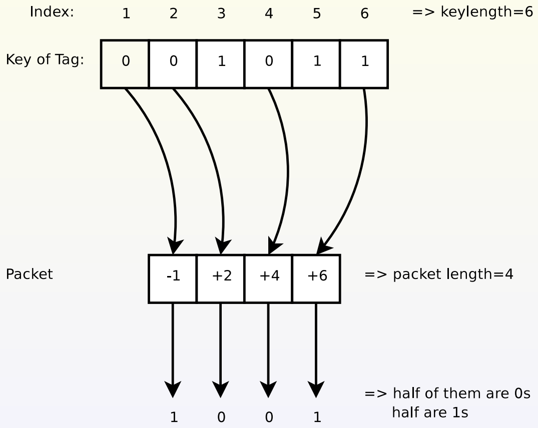
- Consists of L number of indexes from the key of the tag. Each index can be either inverted or not. No indexes are repeated

Description of a packet

Description of a packet:

- Consists of L number of indexes from the key of the tag. Each index can be either inverted or not. No indexes are repeated
- Has the following interesting property: $\sum_{j=1}^{L} k_i[a_j] \oplus b_j = L/2$ where $a_j \xleftarrow{r} [1, K]$ is a random index, and $b_j \xleftarrow{r} \{0, 1\}$ is a random bit $b_j \xleftarrow{r} \{0, 1\}$

Description of a packet



Description of a packet

From a computational complexity point of view:

- The packet is a constraint satisfaction problem (specifically, a linear pseudo-boolean constraint satisfaction problem)

Description of a packet

From a computational complexity point of view:

- The packet is a constraint satisfaction problem (specifically, a linear pseudo-boolean constraint satisfaction problem)
- The packet is an $L/2$-in-L LSAT problem

Description of a packet

From a computational complexity point of view:

- The packet is a constraint satisfaction problem (specifically, a linear pseudo-boolean constraint satisfaction problem)
- The packet is an $L/2$-in-L LSAT problem
- These problems are equivalent and NP-hard (Shaefer's dichotomy theorem)

Number of packets per identification

How many packets will let the reader identify the tag?

- Number of solutions possible for the reader: n

Number of packets per identification

How many packets will let the reader identify the tag?

- Number of solutions possible for the reader: n
- One packet reduces the solution space by a factor of

$$R \approx \frac{\binom{K}{L/2}^2}{\binom{2*K}{L}}$$

Number of packets per identification

How many packets will let the reader identify the tag?

- Number of solutions possible for the reader: n
- One packet reduces the solution space by a factor of

$$R \approx \frac{\binom{K}{L/2}^2}{\binom{2*K}{L}}$$

- We want to reduce the solution space to 0 - the only possible solution must be the *inherent* solution, i.e. k

Number of packets per identification

How many packets will let the reader identify the tag?

- Number of solutions possible for the reader: n
- One packet reduces the solution space by a factor of

$$R \approx \frac{\binom{K}{L/2}^2}{\binom{2*K}{L}}$$

- We want to reduce the solution space to 0 - the only possible solution must be the *inherent* solution, i.e. k
- The number of packets needed for a given false positive rate is then: $fp \approx \frac{log(fp/n)}{log(R)}$

Number of packets per identification

How many packets will let the reader identify the tag?

- Number of solutions possible for the reader: n
- One packet reduces the solution space by a factor of

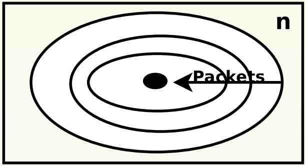
$$R \approx \frac{\binom{K}{L/2}^2}{\binom{2*K}{L}}$$

- We want to reduce the solution space to 0 - the only possible solution must be the *inherent* solution, i.e. k
- The number of packets needed for a given false positive rate is then: $fp \approx \frac{log(fp/n)}{log(R)}$
- For $fp = 0.1$, i.e. for 90% identification chance, if $K = 400, L = 10$ and $n = 1\ million$, $P = 13$

Graphic example

From the point of view of the size of the solution space:

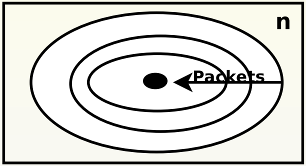
- Reader's point of view:



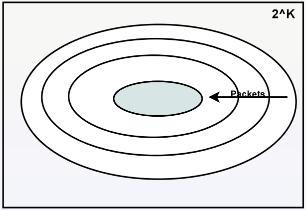
Graphic example

From the point of view of the size of the solution space:

- Reader's point of view:



- Attacker's point of view:



Algorithm to find the tag

What is the difference between a reader and an attacker?

- Caching $n = 1\ million$ keys takes as much as storing the keys

Algorithm to find the tag

What is the difference between a reader and an attacker?

- Caching $n = 1\ million$ keys takes as much as storing the keys
- Caching 2^K keys in memory is impossible

Caching:

Algorithm to find the tag

What is the difference between a reader and an attacker?

- Caching $n = 1\ million$ keys takes as much as storing the keys
- Caching 2^K keys in memory is impossible

Caching:

- Pre-construct look-up lists for all key's indexes:

	List[1]	List[2]	List[3]	List[4]
Key of Tag1	1	0	0	1
Key of Tag2	0	1	1	0
Key of Tag3	1	0	1	0

Algorithm to find the tag

What is the difference between a reader and an attacker?

- Caching $n = 1\ million$ keys takes as much as storing the keys
- Caching 2^K keys in memory is impossible

Caching:

- Pre-construct look-up lists for all key's indexes:

	List[1]	List[2]	List[3]	List[4]
Key of Tag1	1	0	0	1
Key of Tag2	0	1	1	0
Key of Tag3	1	0	1	0

- Go through the look-up table for the indexes in the packet, and calculate the shown sum for each packet. The tag that has $L/2$ for all packets is the one that is sending them

What do we mean by breaking the anonymity

We use Juels and Weis' "strong privacy" model:

* The attacker has q as a query limit and c as a calculation limit

What do we mean by breaking the anonymity

We use Juels and Weis' "strong privacy" model:

* The attacker has q as a query limit and c as a calculation limit

1. Give the attacker all n tags, let him query them without surpassing the q query limit

What do we mean by breaking the anonymity

We use Juels and Weis' "strong privacy" model:

* The attacker has q as a query limit and c as a calculation limit

1. Give the attacker all n tags, let him query them without surpassing the q query limit
2. Let the attacker do calculations within the limit of c

What do we mean by breaking the anonymity

We use Juels and Weis' "strong privacy" model:

* The attacker has q as a query limit and c as a calculation limit

1. Give the attacker all n tags, let him query them without surpassing the q query limit
2. Let the attacker do calculations within the limit of c
3. Let the attacker select 2 tags, $\mathcal{T}_A$ and $\mathcal{T}_B$

What do we mean by breaking the anonymity

We use Juels and Weis' "strong privacy" model:

- * The attacker has q as a query limit and c as a calculation limit

1. Give the attacker all n tags, let him query them without surpassing the q query limit
2. Let the attacker do calculations within the limit of c
3. Let the attacker select 2 tags, $\mathcal{T}_A$ and $\mathcal{T}_B$
4. Secretly and randomly select one of the two, let's call it $\mathcal{T}_C$

What do we mean by breaking the anonymity

We use Juels and Weis' "strong privacy" model:

* The attacker has q as a query limit and c as a calculation limit

1. Give the attacker all n tags, let him query them without surpassing the q query limit
2. Let the attacker do calculations within the limit of c
3. Let the attacker select 2 tags, $\mathcal{T}_A$ and $\mathcal{T}_B$
4. Secretly and randomly select one of the two, let's call it $\mathcal{T}_C$
5. Let the attacker query $\mathcal{T}_C$ without surpassing the q query limit

What do we mean by breaking the anonymity

We use Juels and Weis' "strong privacy" model:

* The attacker has q as a query limit and c as a calculation limit

1. Give the attacker all n tags, let him query them without surpassing the q query limit
2. Let the attacker do calculations within the limit of c
3. Let the attacker select 2 tags, $\mathcal{T}_A$ and $\mathcal{T}_B$
4. Secretly and randomly select one of the two, let's call it $\mathcal{T}_C$
5. Let the attacker query $\mathcal{T}_C$ without surpassing the q query limit
6. Let the attacker do calculations within the limit of c

What do we mean by breaking the anonymity

We use Juels and Weis' "strong privacy" model:

- * The attacker has q as a query limit and c as a calculation limit

1. Give the attacker all n tags, let him query them without surpassing the q query limit
2. Let the attacker do calculations within the limit of c
3. Let the attacker select 2 tags, $\mathcal{T}_A$ and $\mathcal{T}_B$
4. Secretly and randomly select one of the two, let's call it $\mathcal{T}_C$
5. Let the attacker query $\mathcal{T}_C$ without surpassing the q query limit
6. Let the attacker do calculations within the limit of c
7. The attacker must tell if $\mathcal{T}_C = \mathcal{T}_A$ or $\mathcal{T}_C = \mathcal{T}_B$ with sufficient probability

Best attacker strategy

- Since all tags are *totally independent*, only the two pre-selected ones will be examined, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$

Best attacker strategy

- Since all tags are *totally independent*, only the two pre-selected ones will be examined, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$
- Query only one of the two tags ($\mathcal{T}_A$,$\mathcal{T}_B$) for $q/2$ queries, e.g. $\mathcal{T}_A$, and obtain packets Run_A

Best attacker strategy

- Since all tags are *totally independent*, only the two pre-selected ones will be examined, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$
- Query only one of the two tags ($\mathcal{T}_A$,$\mathcal{T}_B$) for $q/2$ queries, e.g. $\mathcal{T}_A$, and obtain packets Run_A
- Query $\mathcal{T}_C$ for $q/2$ queries, and obtain the packets Run_C

Best attacker strategy

- Since all tags are *totally independent*, only the two pre-selected ones will be examined, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$
- Query only one of the two tags ($\mathcal{T}_A$,$\mathcal{T}_B$) for $q/2$ queries, e.g. $\mathcal{T}_A$, and obtain packets Run_A
- Query $\mathcal{T}_C$ for $q/2$ queries, and obtain the packets Run_C
- Find the solution to the constraint satisfaction problem defined by the packets $Run_A \cup Run_C$

Best attacker strategy

- Since all tags are *totally independent*, only the two pre-selected ones will be examined, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$
- Query only one of the two tags ($\mathcal{T}_A$,$\mathcal{T}_B$) for $q/2$ queries, e.g. $\mathcal{T}_A$, and obtain packets Run_A
- Query $\mathcal{T}_C$ for $q/2$ queries, and obtain the packets Run_C
- Find the solution to the constraint satisfaction problem defined by the packets $Run_A \cup Run_C$
- If the solution is UNSAT, then the two tags must be different – packets sent by $\mathcal{T}_A$ always have solution k_A

Best attacker strategy

- Since all tags are *totally independent*, only the two pre-selected ones will be examined, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$
- Query only one of the two tags ($\mathcal{T}_A$,$\mathcal{T}_B$) for $q/2$ queries, e.g. $\mathcal{T}_A$, and obtain packets Run_A
- Query $\mathcal{T}_C$ for $q/2$ queries, and obtain the packets Run_C
- Find the solution to the constraint satisfaction problem defined by the packets $Run_A \cup Run_C$
- If the solution is UNSAT, then the two tags must be different – packets sent by $\mathcal{T}_A$ always have solution k_A
- If the solution is SAT, then:

Best attacker strategy

- Since all tags are *totally independent*, only the two pre-selected ones will be examined, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$
- Query only one of the two tags ($\mathcal{T}_A$,$\mathcal{T}_B$) for $q/2$ queries, e.g. $\mathcal{T}_A$, and obtain packets Run_A
- Query $\mathcal{T}_C$ for $q/2$ queries, and obtain the packets Run_C
- Find the solution to the constraint satisfaction problem defined by the packets $Run_A \cup Run_C$
- If the solution is UNSAT, then the two tags must be different – packets sent by $\mathcal{T}_A$ always have solution k_A
- If the solution is SAT, then:
    - Either $\mathcal{T}_A \neq \mathcal{T}_C$ BUT we did not gather enough packets to show they are different

Best attacker strategy

- Since all tags are *totally independent*, only the two pre-selected ones will be examined, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$
- Query only one of the two tags ($\mathcal{T}_A$,$\mathcal{T}_B$) for $q/2$ queries, e.g. $\mathcal{T}_A$, and obtain packets Run_A
- Query $\mathcal{T}_C$ for $q/2$ queries, and obtain the packets Run_C
- Find the solution to the constraint satisfaction problem defined by the packets $Run_A \cup Run_C$
- If the solution is UNSAT, then the two tags must be different – packets sent by $\mathcal{T}_A$ always have solution k_A
- If the solution is SAT, then:
  - Either $\mathcal{T}_A \neq \mathcal{T}_C$ BUT we did not gather enough packets to show they are different
  - OR $\mathcal{T}_A = \mathcal{T}_C$. – if we have gathered enough for sure, we can safely say this. 'Enough' in this context is defined as P_{att}

Algorithm to attack

Best algorithm to attack the system:

- There are specialized solvers to find a solution to the problem described by the packets (LPBC solvers). But, these are slow for multiple reasons

Algorithm to attack

Best algorithm to attack the system:

- There are specialized solvers to find a solution to the problem described by the packets (LPBC solvers). But, these are slow for multiple reasons
- There are solvers to find a solution to general SAT problems (i.e. $a \vee \bar{b} \vee c \vee ...$). Packets must be converted to this representation. These solvers are fast

Algorithm to attack

Best algorithm to attack the system:

- There are specialized solvers to find a solution to the problem described by the packets (LPBC solvers). But, these are slow for multiple reasons
- There are solvers to find a solution to general SAT problems (i.e. $a \vee \bar{b} \vee c \vee ...$). Packets must be converted to this representation. These solvers are fast

We decided on Minisat (best of the 2005&2006 SAT competition). It is fast, open-source and readily modifiable

Threshold phenomenon

There is a so-called threshold phenomenon for all NP-hard problems. This states that when solving a *randomly* generated SAT problem, there are three phases in terms of the number of constraints:

- Solution is fast to find, chance to find one is nearly 100%

Threshold phenomenon

There is a so-called threshold phenomenon for all NP-hard problems. This states that when solving a *randomly* generated SAT problem, there are three phases in terms of the number of constraints:

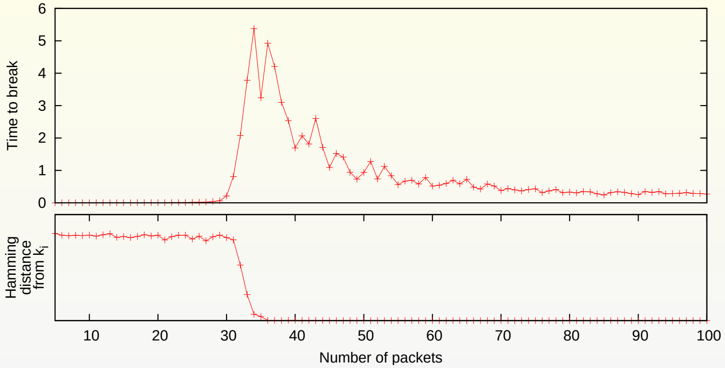
- Solution is fast to find, chance to find one is nearly 100%
- After a certain point, the chance to find solution changes very rapidly from 100% to 0%, and at the same time, the difficulty to find a solution jumps to very high levels. This is the *threshold point*.

Threshold phenomenon

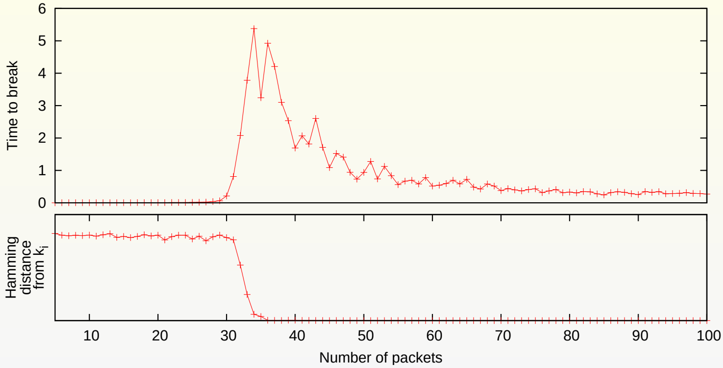
There is a so-called threshold phenomenon for all NP-hard problems. This states that when solving a *randomly* generated SAT problem, there are three phases in terms of the number of constraints:

- Solution is fast to find, chance to find one is nearly 100%
- After a certain point, the chance to find solution changes very rapidly from 100% to 0%, and at the same time, the difficulty to find a solution jumps to very high levels. This is the *threshold point*.
- After the threshold point, the chance to find a solution is almost 0%, but if there exists a solution (or if it does not), it becomes exponentially easier to find it (or find that it does not exist respectively) in respect to the number of constraints.

Graphically



Graphically



The attacker can only use the right side of the graph

Results

packets/K	100	200	400	1000
$1{*}P_{att}$	$1.47e2$ s	$3.17e11$ s	$1.46e28$ s	$1.46e78$ s
$3{*}P_{att}$	$3.33e1$ s	$7.41e5$ s	$3.67e14$ s	$4.49e40$ s
$9{*}P_{att}$	$6.31e0$ s	$4.54e3$ s	$2.35e9$ s	$3.27e26$ s
$27{*}P_{att}$	$4.27e0$ s	$6.37e2$ s	$1.42e7$ s	$1.57e20$ s
$64{*}P_{att}$	$4.02e0$ s	$4.87e2$ s	$7.15e6$ s	$2.27e19$ s
$192{*}P_{att}$	$5.34e0$ s	$7.31e1$ s	$1.37e4$ s	$9.01e10$ s
$576{*}P_{att}$	$1.00e1$ s	$7.28e1$ s	$3.86e3$ s	$5.74e8$ s

Table: Time to break the anonymity

Conclusion&Future work

- We have developed an RFID privacy solution that is suitable for cheap tags

Conclusion&Future work

- We have developed an RFID privacy solution that is suitable for cheap tags
- The developed protocol's fundamentals are such that it can potentially be a foundation for many protocols to come

Conclusion&Future work

- We have developed an RFID privacy solution that is suitable for cheap tags
- The developed protocol's fundamentals are such that it can potentially be a foundation for many protocols to come
- We are at the moment developing an improvement of the presented protocol

Thank you for your time

Are there any questions?