Analysing the Molva and Di Pietro Private RFID Authentication Scheme

Mate Soos
INRIA team PLANETE, INRIA Rhône-Alpes

July 7, 2008

Table of Contents

Outline

Protocol

The protocol can be divided into three phases:

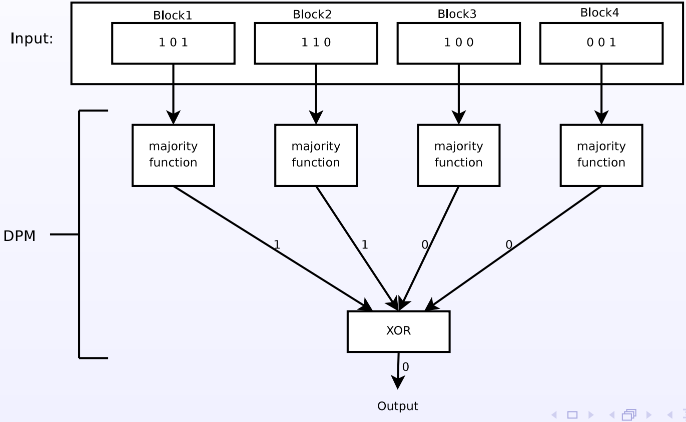
1. Private identification
2. Tag authentication
3. Reader authentication

Some specifics:

- There are n tags $\mathcal{T}_1 \ldots \mathcal{T}_n$ in the system
- Each tag has a unique l-bit long key k_i
- Each reader $\mathcal{R}_j$ has an ID ID_j
- Reader-specific key of a tag: $k_{i,j} = h(k_i||ID_j||k_i)$, where h is a hash function
- ID of a tag is its reader-specific key

Identification

Uses the function $DPM(x) = \bigoplus_{i=0}^{l/3} M(x[3i], x[3i+1], x[3i+2])$, where M is the majority function:



Identification

Steps of the identification:

1. $\mathcal{R}_j$ sends ID_j to the tag

Identification

Steps of the identification:

1. $\mathcal{R}_j$ sends ID_j to the tag
2. $\mathcal{T}_i$ computes $k_{i,j} = h(k_i||ID_j||k_i)$

Identification

Steps of the identification:

1. $\mathcal{R}_j$ sends ID_j to the tag
2. $\mathcal{T}_i$ computes $k_{i,j} = h(k_i||ID_j||k_i)$
3. $\mathcal{T}_i$ generates l-bit nonces $r_1 \dots r_q$:

Identification

Steps of the identification:

1. $\mathcal{R}_j$ sends ID_j to the tag
2. $\mathcal{T}_i$ computes $k_{i,j} = h(k_i||ID_j||k_i)$
3. $\mathcal{T}_i$ generates l-bit nonces $r_1 \ldots r_q$:
   - $\alpha_p = r_p \oplus k_{i,j}$

Identification

Steps of the identification:

1. $\mathcal{R}_j$ sends ID_j to the tag
2. $\mathcal{T}_i$ computes $k_{i,j} = h(k_i||ID_j||k_i)$
3. $\mathcal{T}_i$ generates l-bit nonces $r_1 \ldots r_q$:
   - $\alpha_p = r_p \oplus k_{i,j}$
   - $V[p] = DPM(r_p)$

Identification

Steps of the identification:

1. $\mathcal{R}_j$ sends ID_j to the tag
2. $\mathcal{T}_i$ computes $k_{i,j} = h(k_i||ID_j||k_i)$
3. $\mathcal{T}_i$ generates l-bit nonces $r_1 \ldots r_q$:
   - $\alpha_p = r_p \oplus k_{i,j}$
   - $V[p] = DPM(r_p)$
   - sends the $(\alpha_p, V[p])$ pairs

Identification

Steps of the identification:

1. $\mathcal{R}_j$ sends ID_j to the tag
2. $\mathcal{T}_i$ computes $k_{i,j} = h(k_i||ID_j||k_i)$
3. $\mathcal{T}_i$ generates l-bit nonces $r_1 \dots r_q$:
   - $\alpha_p = r_p \oplus k_{i,j}$
   - $V[p] = DPM(r_p)$
   - sends the $(\alpha_p, V[p])$ pairs
4. $\mathcal{R}_j$ computes $DPM(\alpha_p \oplus k_{i,j})$ for all keys $k_{i,j}$ it possesses and checks it against $V[p]$. This is called the *Lookup Process*

q is selected such that it is highly improbable that the Lookup Process fails

Tag authentication

Tag authentication is a simple challenge-response:

1. $\mathcal{R}_j$ sends a nonce n_j to the tag

Tag authentication

Tag authentication is a simple challenge-response:

1. $\mathcal{R}_j$ sends a nonce n_j to the tag
2. $\mathcal{T}_i$ computes and sends $\omega = h(k_{i,j}||n_j||r_1||k_{i,j})$ to the reader

Tag authentication

Tag authentication is a simple challenge-response:

1. $\mathcal{R}_j$ sends a nonce n_j to the tag
2. $\mathcal{T}_i$ computes and sends $\omega = h(k_{i,j}||n_j||r_1||k_{i,j})$ to the reader
3. $\mathcal{R}_j$ computes $r_1 = \alpha_1 \oplus k_{i,j}$ and checks ω against $h(k_{i,j}||n_j||r_1||k_{i,j})$

Reader authentication

Reader authentication is also a simple challenge-response:

1. $\mathcal{R}_j$ computes $r_1 = \alpha_1 \oplus k_{i,j}$ and sends $h(k_{i,j}||r_1||k_{i,j})$ to the tag

Reader authentication

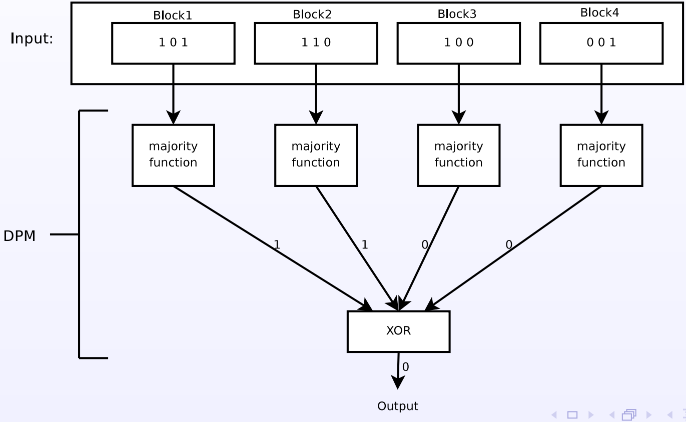
Reader authentication is also a simple challenge-response:

1. $\mathcal{R}_j$ computes $r_1 = \alpha_1 \oplus k_{i,j}$ and sends $h(k_{i,j}||r_1||k_{i,j})$ to the tag
2. $\mathcal{T}_i$ computes $h(k_{i,j}||r_1||k_{i,j})$ and checks it against the received hash. If they match, the reader is authenticated

Outline

Key-equivalences

- If an even number of key blocks are inverted, the resulting key will be indistinguishable by the reader from the original key



Key-equivalences

- So there are key-equivalence groups in the key space

Key-equivalences

- So there are key-equivalence groups in the key space
- Each key-equivalence group contains $2^{l/3-1}$ keys

Key-equivalences

- So there are key-equivalence groups in the key space
- Each key-equivalence group contains $2^{l/3-1}$ keys
- In a similar manner, there are pair-equivalences

Key-equivalences

- So there are key-equivalence groups in the key space
- Each key-equivalence group contains $2^{l/3-1}$ keys
- In a similar manner, there are pair-equivalences
- Key- and pair-equivalences cause a big headache for the Lookup Process

Key-equivalences

- An α_p-$V[p]$ pair essentially give (somewhat obscure) information about the key of the tag

Key-equivalences

- An α_p-$V[p]$ pair essentially give (somewhat obscure) information about the key of the tag
- Naturally, there is only so much different information that is possible to give

Key-equivalences

- An α_p-$V[p]$ pair essentially give (somewhat obscure) information about the key of the tag
- Naturally, there is only so much different information that is possible to give
- So, there is a chance to give the same information twice

Key-equivalences

- An α_p-$V[p]$ pair essentially give (somewhat obscure) information about the key of the tag
- Naturally, there is only so much different information that is possible to give
- So, there is a chance to give the same information twice
- Tautology is a set of x pairs that give the same information as $x - 1$ pairs

Key-equivalences

- An α_p-$V[p]$ pair essentially give (somewhat obscure) information about the key of the tag
- Naturally, there is only so much different information that is possible to give
- So, there is a chance to give the same information twice
- Tautology is a set of x pairs that give the same information as $x - 1$ pairs
- Tautologies are also possible and they cause further problems for the Lookup Process

Speed problems

Average time and RAM required by the Lookup Process to find one tag on a Xeon E5345@2.33GHz with all optimisations other than assembly-level coding:

Number of tags	10^6	10^7	10^8
Time (s)	0.1	1.1	12
Memory (MB)	9.6	96	965

Finding $k_{i,j}$

- If an attacker inverts one bit of a block in α_2 such that output of the majority function is not inverted, the Lookup Process will still find the key $k_{i,j}$

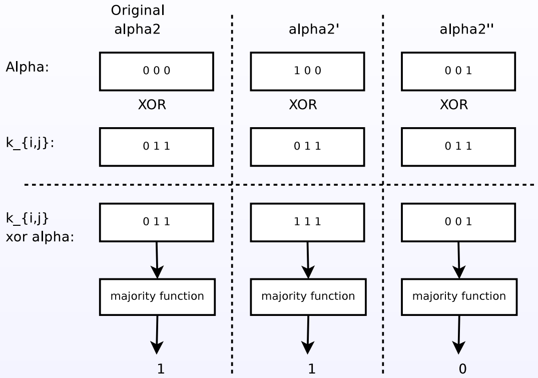
Finding $k_{i,j}$

- If an attacker inverts one bit of a block in α_2 such that output of the majority function is not inverted, the Lookup Process will still find the key $k_{i,j}$
- If the Lookup Process finds the correct key, the authentication will go through, since only α_1 is authenticated

Finding $k_{i,j}$

- If an attacker inverts one bit of a block in α_2 such that output of the majority function is not inverted, the Lookup Process will still find the key $k_{i,j}$
- If the Lookup Process finds the correct key, the authentication will go through, since only α_1 is authenticated
- So, by inverting one bit of a block in α_2 and checking the result of the authentication, the attacker can learn something very specific about that block

Finding $k_{i,j}$



There are only two bit-combinations for which:

1. inverting the fist bit does not change the majority
2. inverting the last bit changes the majority

These are: 011 and 100

Finding $k_{i,j}$

- Each MiM authentication attack gives 1 bit of block-specific information

Finding $k_{i,j}$

- Each MiM authentication attack gives 1 bit of block-specific information
- After $2/3 \cdot l - 1$ MiM attacks the attacker breaks the key to the key-equivalence level

Finding $k_{i,j}$

- Each MiM authentication attack gives 1 bit of block-specific information
- After $2/3 \cdot l - 1$ MiM attacks the attacker breaks the key to the key-equivalence level
- At this point, the tag is no longer private

Finding $k_{i,j}$

- Each MiM authentication attack gives 1 bit of block-specific information
- After $2/3 \cdot l - 1$ MiM attacks the attacker breaks the key to the key-equivalence level
- At this point, the tag is no longer private
- The attacker needs to brute-force the remaining $1/3 \cdot l + 1$ bits of the key using the authentication data

Finding $k_{i,j}$

- Each MiM authentication attack gives 1 bit of block-specific information
- After $2/3 \cdot l - 1$ MiM attacks the attacker breaks the key to the key-equivalence level
- At this point, the tag is no longer private
- The attacker needs to brute-force the remaining $1/3 \cdot l + 1$ bits of the key using the authentication data
- Therefore, for $l = 99$ the authentication can be broken easily

Finding $k_{i,j}$

- Each MiM authentication attack gives 1 bit of block-specific information
- After $2/3 \cdot l - 1$ MiM attacks the attacker breaks the key to the key-equivalence level
- At this point, the tag is no longer private
- The attacker needs to brute-force the remaining $1/3 \cdot l + 1$ bits of the key using the authentication data
- Therefore, for $l = 99$ the authentication can be broken easily
- For larger l-s, privacy is still lost and the scheme behaves as an authentication scheme that has a keyspace of 1/3rd+1 of available key-bits

Outline

Design flaws

- Identification and authentication boundaries should have been clearly defined

Design flaws

- Identification and authentication boundaries should have been clearly defined
- Identification and authentication keys should have been generated differently

Design flaws

- Identification and authentication boundaries should have been clearly defined
- Identification and authentication keys should have been generated differently
- Given that the identification was not cryptographically secured, the integrity of the data exchanged during identification should have been authenticated during authentication

Design flaws

- Identification and authentication boundaries should have been clearly defined
- Identification and authentication keys should have been generated differently
- Given that the identification was not cryptographically secured, the integrity of the data exchanged during identification should have been authenticated during authentication
- The choice of the DPM function was not clearly motivated and its design was not analysed in a separate paragraph

Thank you for your time

Any questions?