Revoke and Let Live

A Secure Key Revocation API for Cryptographic Devices

Véronique Cortier
LORIA-CNRS, Nancy (FR)

Graham Steel
INRIA, Paris (FR)

Cyrille Wiedling
LORIA-CNRS, Nancy (FR)

ACM CCS’12
October, 18th, 2012

Funded by
Goal: Enforce security of data stored inside the trusted device, even when connected to untrusted host machines.
Applications

• Smartphones,

• Online Banking, Asynchronous Transfer Mode,

• Electronic Ticketing Systems,

• Vehicle-to-vehicle networking.

• ...

mardi 23 octobre 12
How does it work?

Host machine

Trusted device

\[
\begin{align*}
  h_1 & \\
  h_2 &
\end{align*}
\]
How does it work?

Host machine

Trusted device

\[ h_1, h_2 \]

export, \( h_1, h_2 \)
How does it work?

Host machine

Trusted device

Export, $h_1, h_2$

$mardi \ 23 \ octobre \ 12$
How does it work?

Host machine

Trusted device

\[
\begin{array}{|c|}
\hline
h_1 \\
\hline
h_2 \\
\hline
\end{array}
\]

export, \( h_1, h_2 \)

import, \( \{ \text{key1}, \text{key2} \} \), \( h_2 \)
How does it work?

Host machine

- **export**, $h_1, h_2$
- import, $h_3$

Trusted device

<table>
<thead>
<tr>
<th>$h_1$</th>
<th><img src="key1.png" alt="Key 1" /></th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_2$</td>
<td><img src="key2.png" alt="Key 2" /></td>
</tr>
<tr>
<td>$h_3$</td>
<td><img src="key3.png" alt="Key 3" /></td>
</tr>
</tbody>
</table>
Breaking keys in a TRD

There are ways for the attacker to break some keys of a Tamper-Resistant Device (TRD):

- Bruteforcing,
- Side-channel attack,
- ...

mardi 23 octobre 12
Related Work

Proposals for key management APIs with security proofs but without addressing the question of revocation.

J. Courant, J.-F. Monin, WITS’06.

C. Cachin, N. Chandran, CSF’09...
Related Work

**Proposals** for key management APIs with security proofs but **without** addressing the question of **revocation**.

J. Courant, J.-F. Monin, WITS’06.

C. Cachin, N. Chandran, CSF’09...

**Proposals** for key management APIs **with revocation**:

L. Eschenauer, V. D. Gligor, CCS’02.  
(Using a control server)

X. Z. Yong Wan, B. Ramamurthy, ICC’07.  
(Secret sharing scheme)
Related Work

Proposals for key management APIs with security proofs but without addressing the question of revocation.

J. Courant, J.-F. Monin, WITS’06.
C. Cachin, N. Chandran, CSF’09...

Proposals for key management APIs with revocation:

L. Eschenauer, V. D. Gligor, CCS’02. (Using a control server)
X. Z. Yong Wan, B. Ramamurthy, ICC’07. (Secret sharing scheme)

Use of long-term keys implying unrecoverable loss of devices if keys are lost
Related Work

Proposals for key management APIs with security proofs but without addressing the question of revocation.

J. Courant, J.-F. Monin, WITS’06.
C. Cachin, N. Chandran, CSF’09...

Proposals for key management APIs with revocation:

L. Eschenauer, V. D. Gligor, CCS’02. (Using a control server)
X. Z. Yong Wan, B. Ramamurthy, ICC’07. (Secret sharing scheme)

Use of long-term keys implying unrecoverable loss of devices if keys are lost

F. E. Kargl, Sevecom, 2009... (Two root keys)
Related Work

Proposals for key management APIs with security proofs but **without** addressing the question of revocation.

J. Courant, J.-F. Monin, WITS’06.
C. Cachin, N. Chandran, CSF’09...

Proposals for key management APIs **with revocation**:

L. Eschenauer, V. D. Gligor, CCS’02. (Using a control server)
X. Z. Yong Wan, B. Ramamurthy, ICC’07. (Secret sharing scheme)

**Use of long-term keys implying unrecoverable loss of devices if keys are lost**

F. E. Kargl, Sevecom, 2009... (Two root keys)

**Attacked** by S. Möderschein & P. Modesti
(solution proposed but no security proof)
Keys must remain **confidential**:

Information about key should not be recovered by the intruder.
Ideal Key Revocation API

Keys must remain **confidential**:

Information about key should not be recovered by the intruder.

Any key should be **revocable**:

The more sensitive a key is, the more an attacker will try to break it.
Ideal Key Revocation API

Keys must remain **confidential**:

Information about key should not be recovered by the intruder.

Any key should be **revocable**:

The more sensitive a key is, the more an attacker will try to break it.

The device should remain **functional**:

A revocation of a key should not prevent the user from using another.
Our Contributions

- **Design** of an API satisfying previous properties with:
  - update functionality,
  - revocation functionality.

- A **formal proof of security** ensuring three properties:
  - A key remains secret unless it is explicitly lost,
  - the system is able to **recover itself** from a loss,
  - a revocation immediately secures the device.
Description of the API
A clock assumed synchronized with a global clock
Description of the API

- A clock assumed synchronized with a global clock
- A table indexed by handles to store keys’ information (level, validity date, ...)

TRD
Description of the API

A **clock** assumed synchronized with a global clock

A **table** indexed by handles to store keys’ information (level, validity date, ...)

A **blacklist** of elements of the form \((l, t)\)
Description of the API

A **clock** assumed synchronized with a global clock

A **table** indexed by handles to store keys’ information (level, validity date, ...)

A **blacklist** of elements of the form \((l, t)\)

**Hierarchy of levels:** (We consider an upper bound for levels.)

\[ l_1 \quad l_2 \quad l_3 \quad l_4 \quad l_5 \quad l_6 \quad l_7 \]
Description of the API

TRD

A **clock** assumed synchronized with a global clock

A **table** indexed by handles to store keys’ information (level, validity date, ...)

A **blacklist** of elements of the form \((l, t)\)

Hierarchy of levels:

We consider an upper bound for levels.

\[
l_5 < l_2
\]

\[
\{\}
\]
Description of the API

A clock assumed synchronized with a global clock

A table indexed by handles to store keys’ information (level, validity date, ...)

A blacklist of elements of the form \((l, t)\)

Hierarchy of levels:

We consider an upper bound for levels.

\[ l_5 < l_2 \] 
\[ l_4 \not< l_1 \]

mardi 23 octobre 12
Description of the API

TRD

- A **clock** assumed synchronized with a global clock
- A **table** indexed by handles to store keys’ information (level, validity date, ...)
- A **blacklist** of elements of the form \((l, t)\)

**Hierarchy of levels:**

\( l_1 \) \( l_2 \) \( l_3 \) \( l_4 \) \( l_5 \) \( l_6 \) \( l_7 \)

- \( l_5 < l_2 \)
- \( l_4 \not< l_1 \)
- \( l_4 \not< l_7 \)

(We consider an upper bound for levels.)

mardi 23 octobre 12
User’s Commands

A set of **basic commands**:

\[
\begin{align*}
generatePublic(m) \\
generateSecret(l, m)
\end{align*}
\]

Generate a nonce or a key, and store under a handle the information.

**Example**: \( h \leftarrow (k, l, v, m) \)
A set of basic commands:

- **generatePublic**($m$)
- **generateSecret**($l, m$)

Generate a nonce or a key, and store under a handle the information.

**Example:**

```plaintext
h \leftarrow (k, l, v, m)
```

- **decrypt**($C, h$)

**Example:**

```plaintext
C = \left\{ \langle \text{key}, l, v, m \rangle, \langle n, 0, v', m' \rangle \right\}
```

Decrypt $C$ with the key stored under $h$ and return a message or a handle.
A set of **basic commands**:

- **generatePublic**(\(m\))
- **generateSecret**(\(l, m\))

**decrypt**\((C, h)\)  
**Ex**:  
\[ C = \{ \langle l, v, m \rangle, \langle n, 0, v', m' \rangle \} \]

Decrypt \(C\) with the key stored under \(h\) and return a message or a handle.

**encrypt**\((\langle X_1, \ldots, X_n \rangle, h)\)  
\[ X_i = h_i \quad \text{or} \quad X_i = n_i \]

**Ex**:

\[ C = \{ h \} \]

\( h \leftarrow (k, l, v, m) \)
Lower Level Keys Management

update($C, h_1, \ldots, h_n$)

<table>
<thead>
<tr>
<th>$h_1$</th>
<th>$\text{Max, } v_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$h_n$</td>
<td>$\text{Max, } v_n$</td>
</tr>
<tr>
<td>$h$</td>
<td>$l, v, m$</td>
</tr>
</tbody>
</table>

Attention, le «rouge» peut être vu comme du «rose» !
Lower Level Keys Management

update($C, h_1, \ldots, h_n$)

<table>
<thead>
<tr>
<th>$h_1$</th>
<th>, Max, $v_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$h_n$</td>
<td>, Max, $v_n$</td>
</tr>
<tr>
<td>$h$</td>
<td>, $l, v, m$</td>
</tr>
</tbody>
</table>

1. Tests on keys stored under $h_1, \ldots, h_n$.

Attention, le «rouge» peut être vu comme du «rose» !

mardi 23 octobre 12
**Lower Level Keys Management**

<table>
<thead>
<tr>
<th>$h_1$</th>
<th>Max, $v_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$h_n$</td>
<td>Max, $v_n$</td>
</tr>
<tr>
<td>$h$</td>
<td>$l, v, m$</td>
</tr>
</tbody>
</table>

$\text{update}(C, h_1, \ldots, h_n)$

1. Tests on keys stored under $h_1, \ldots, h_n$.
2. Decryption of $C$.

$$C = \left\{ \text{update}, l', v', m' \right\}$$

---

**Attention, le «rouge» peut être vu comme du «rose» !**

---

**mardi 23 octobre 12**
Lower Level Keys Management

\[
\text{update}(C, h_1, \ldots, h_n)
\]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(h_1)</td>
<td>, Max, (v_1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(h_n)</td>
<td>, Max, (v_n)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(h)</td>
<td>, (l), (v), (m)</td>
<td></td>
</tr>
</tbody>
</table>

1. Tests on keys stored under \(h_1, \ldots, h_n\).

2. Decryption of \(C\).

\[
C = \{\text{update}, l', v', m'\}\ldots
\]

3. Tests on the new attributes \(l', v'\).

Attention, le «rouge» peut être vu comme du «rose» !

mardi 23 octobre 12
**Lower Level Keys Management**

update\((C, h_{1}, \ldots, h_{n})\)

<table>
<thead>
<tr>
<th>(h_{1})</th>
<th>, Max, (v_{1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>(h_{n})</td>
<td>, Max, (v_{n})</td>
</tr>
<tr>
<td>(h)</td>
<td>, (l, v, m)</td>
</tr>
</tbody>
</table>

1. Tests on keys stored under \(h_{1}, \ldots, h_{n}\).

2. Decryption of \(C\).

\[C = \{\text{update}, , l', v', m'\}\]

3. Tests on the new attributes \(l', v'\).

4. Table update with the new values.

*Attention, le «rouge» peut être vu comme du «rose»!*
Lower Level Keys Management

**update**\((C, h_1, \ldots, h_n)\)

<table>
<thead>
<tr>
<th>(h_1)</th>
<th>, Max, (v_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>(h_n)</td>
<td>, Max, (v_n)</td>
</tr>
<tr>
<td>(h)</td>
<td>, (l, v, m)</td>
</tr>
</tbody>
</table>

1. Tests on keys stored under \(h_1, \ldots, h_n\).
2. Decryption of \(C\).
3. Tests on the new attributes \(l', v'\).
4. Table update with the new values.

\[ C = \{\text{update}, \ldots, l', v', m'\} \]

**Attention, le «rouge» peut être vu comme du «rose» !**
Revocation Keys Management

\{\text{updateMax}, \text{ }, \text{ }, \text{ }, v \}\}

\ldots

\{\text{updateMax}, \text{ }, \text{ }, \text{ }, v' \}\}

\ldots
Revocation Keys Management

What if (old) revocation keys can be lost and if revocation messages are public?

\[
\begin{align*}
\{ \text{updateMax, } &\text{key1, key2, } v \} \ldots \\
\{ \text{updateMax, } &\text{key1', key2', } v' \} \ldots 
\end{align*}
\]
Revocation Keys Management

What if (old) revocation keys can be lost and if revocation messages are public?
Revocation Keys Management

What if (old) revocation keys can be lost and if revocation messages are public?
What if (old) revocation keys can be lost and if revocation messages are public?

The intruder can break all the level Max keys up to the current ones.
Hypothesis:

Level Max commands are sent over a secure channel.
Hypothesis:

Level Max commands are sent over a secure channel.

This can be achieved by several means:

- The administrator has a physical access to the TRD that needs to be updated,

- The user would connect his/her TRD to a trusted machine, on which a secure channel (e.g. via TLS) is established with the key administrator.
And now, what about Security?
And now, what about Security?
And now, what about Security?
Abstraction

Messages are represented by **terms**

**Nonces, keys:**

\[ n, m, \ldots, k_1, k_2, \ldots \]

**Primitives:**

\[ \{m\}_k, \langle m_1, m_2 \rangle, \ldots \]

**Modeling deduction rules:**

\[
\begin{array}{c c}
  x & y \\
  \hline
  \langle x, y \rangle & x & y \\
  \hline
  \langle x, y \rangle & \langle x, y \rangle & \langle x, y \rangle \\
  \hline
  \{x\}_y & y & \{x\}_y \\
  \hline
  \{x\}_y & y & \{x\}_y \\
  \hline
  \{x\}_y & y & \{x\}_y \\
\end{array}
\]

C'est une abstraction courante mais pas «classique».

mardi 23 octobre 12
Knowledge of the Intruder

TRD

API

Internet

API

TRD

TRD

API

TRD

mardi 23 octobre 12
A key in a TRD may be lost and known by the intruder.
Knowledge of the Intruder

A key in a TRD may be lost and known by the intruder

**Hypothesis:** At most a total of $N_{Max} - 1$ different « current » level Max keys for one TRD can be lost.
What about lost levels?
What about lost levels?
What about lost levels?

The intruder has control over whatever is under a level with a lost key.
What about lost levels?

The intruder has control over whatever is under a level with a lost key.

She may use an encrypt command to get a key with a lower level in a TRD containing a lost key.

Ex: Receive $\langle \text{key}, l_9, v, m \rangle$ with lost and of level $l_5$. 
Secrecy Result

Even if the intruder may:

• **control the network** and host machines,

• **break some keys** (but not too many revocation keys),

«I keep my secrets secret !»

mardi 23 octobre 12
Secrecy Result

Even if the intruder may:

- **control the network** and host machines,
- **break some keys** (but not too many revocation keys),

We have:

**Theorem 1**

Keys remain **secret** (not deducible) provided:

A valid expiration date & not « under a lost »
Theorem 2 (Stated for one level)

Assume that all keys are secret at time $t$ except those under a level $l$. Then at time $t + \Delta(l)$, all keys are secret except those under levels $l_1, \ldots, l_n$ such that $l_i < l$. 

«It’s just a flesh wound!»
Theorem 2 (Stated for one level)

Assume that all keys are secret at time $t$ except those under a level $l$.

Then at time $t + \Delta(l)$, all keys are secret except those under levels $l_1, \ldots, l_n$ such that $l_i < l$. 
Theorem 2  (Stated for one level)

Assume that all keys are secret at time $t$ except those under a level $l$.

Then at time $t + \Delta(l)$, all keys are secret except those under levels $l_1, \ldots, l_n$ such that $l_i < l$. 

\[ l_1 \quad l_2 \]
\[ l_3 \quad l_4 \]
\[ l_5 \quad l_6 \quad l_7 \]
\[ l_8 \quad l_9 \quad l_{10} \quad l_{11} \]
Theorem 2 (Stated for one level)

Assume that all keys are secret at time $t$ except those under a level $l$.

Then at time $t + \Delta(l)$, all keys are secret except those under levels $l_1, \ldots, l_n$ such that $l_i < l$. 

![Diagram](image-url)
Theorem 2 (Stated for one level)

Assume that all keys are secret at time $t$ except those under a level $l$. Then at time $t + \Delta(l)$, all keys are secret except those under levels $l_1, \ldots, l_n$ such that $l_i < l$.

It assumes that, during time $\Delta(l)$, you do not lose a level higher than the one you «try» to repair.
Blacklist Option

\[ \text{blacklist}(C, h_1, \ldots, h_n) \]

Ex: \[ C = \{ \langle \text{blacklist}, \langle l_3, t \rangle \} \]
Blacklist Option

blacklist \((C, h_1, \ldots, h_n)\)

Ex: \(C = \{\langle blacklist, \langle l_3, t \rangle \rangle \}\)

«For those who are in a hurry...»

mardi 23 octobre 12
Blacklist Option

\[
\text{blacklist}(C, h_1, \ldots, h_n) \quad \text{Ex: } C = \{\langle \text{blacklist}, \langle l_3, t \rangle \rangle \} \ldots
\]

«For those who are in a hurry...»

\[
(l_3, t_3) \rightarrow \]

mardi 23 octobre 12
Blacklist Option

\[
\text{blacklist}(C, h_1, \ldots, h_n)
\]

Ex: \( C = \{(\text{blacklist}, (l_3, t))\} \)

"For those who are in a hurry..."
**Theorem 3**  
*(Stated for one level)*

Assume that all keys are secret at time $t$ except those under a level $l$.  
If we blacklist level $l$ on a TRD, then, **immediately**, all keys are secret.
Future Work

• **Weaken assumptions**, especially on hidden level Max messages (maybe requiring more cryptographic primitives),

• **Extend** our API to **asymmetric encryption**,

• **Adapt** the result taking account of possible **clock drift**, or replacing the clock by some sort of nonce based freshness test,

• **Implement** the API in order to carry out some performance tests.
Thank you for your attention!

Public (Host Machines)

Speaker (Security API)

Truth (Trusted device)

CanYouHandle ClockSkew?

Sure!

CanYouHandle ClockSkew?

Perhaps

mardi 23 octobre 12