Revoke and Let Live

A Secure Key Revocation API for Cryptographic Devices

Véronique Cortier
LORIA-CNRS, Nancy (FR)

Graham Steel
INRIA, Paris (FR)

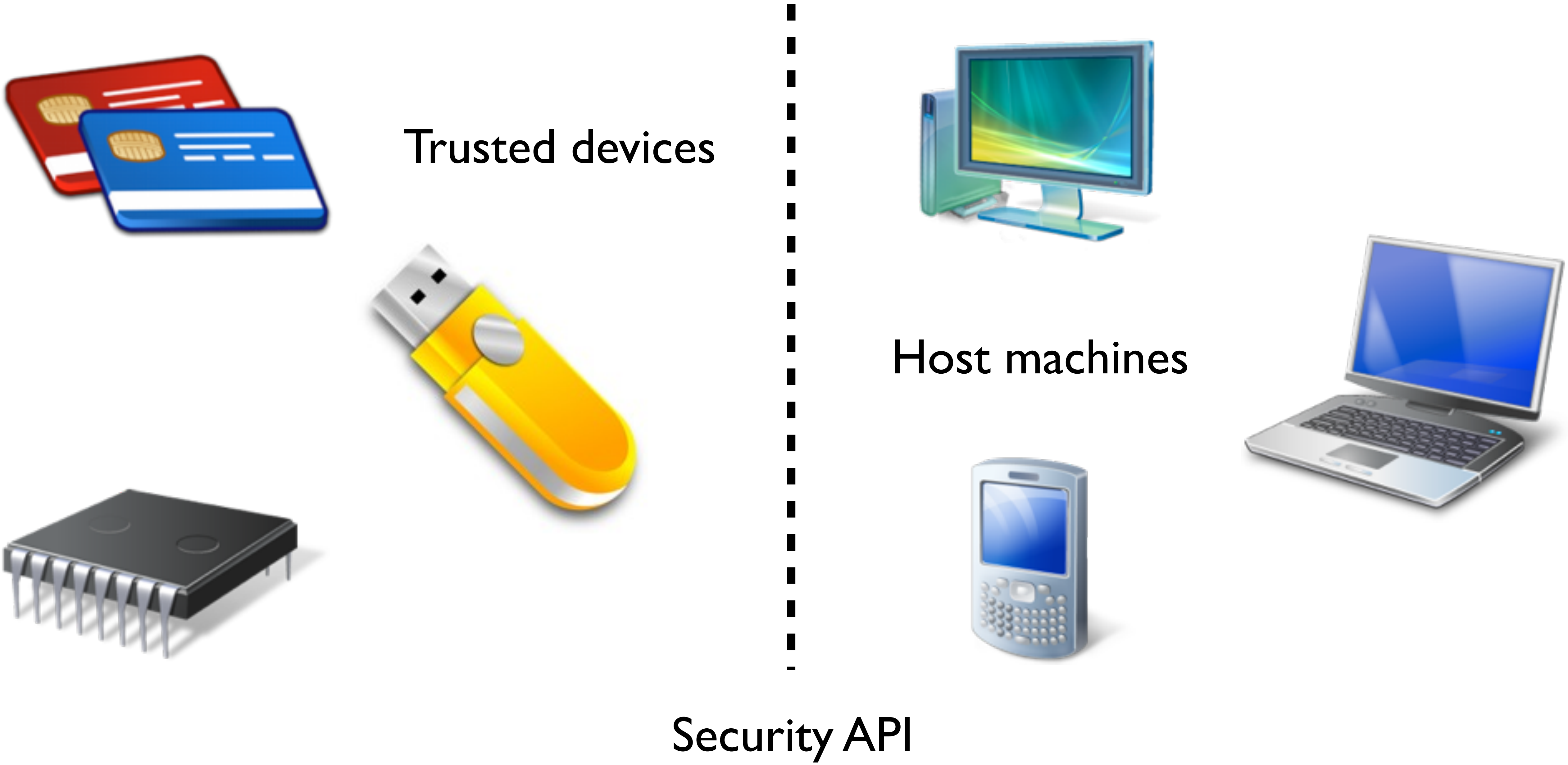
Cyrille Wiedling
LORIA-CNRS, Nancy (FR)

ACM CCS'12

October, 18th, 2012

Funded by erc

Security APIs



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.

- ...

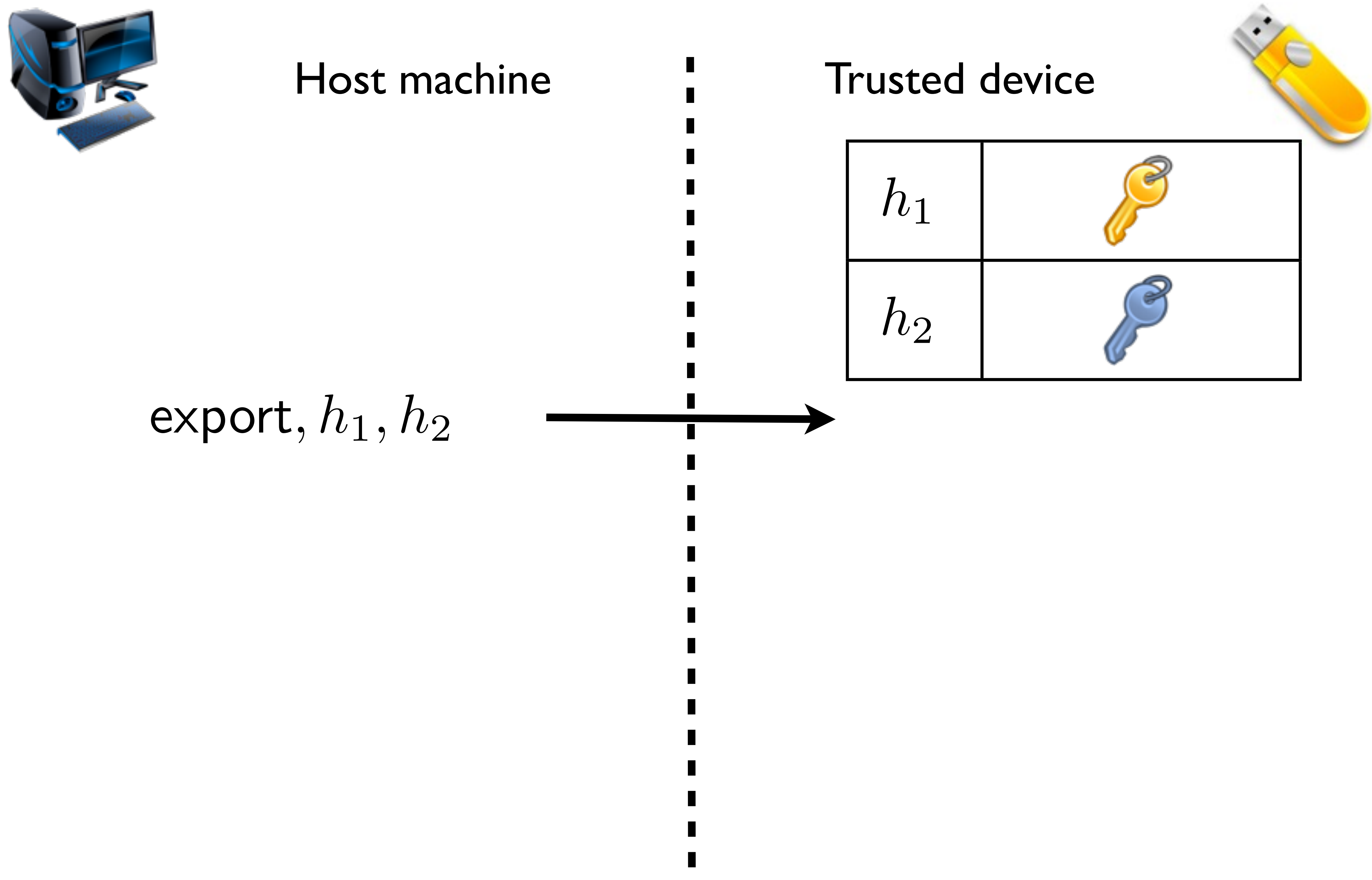
How does it work ?

Host machine

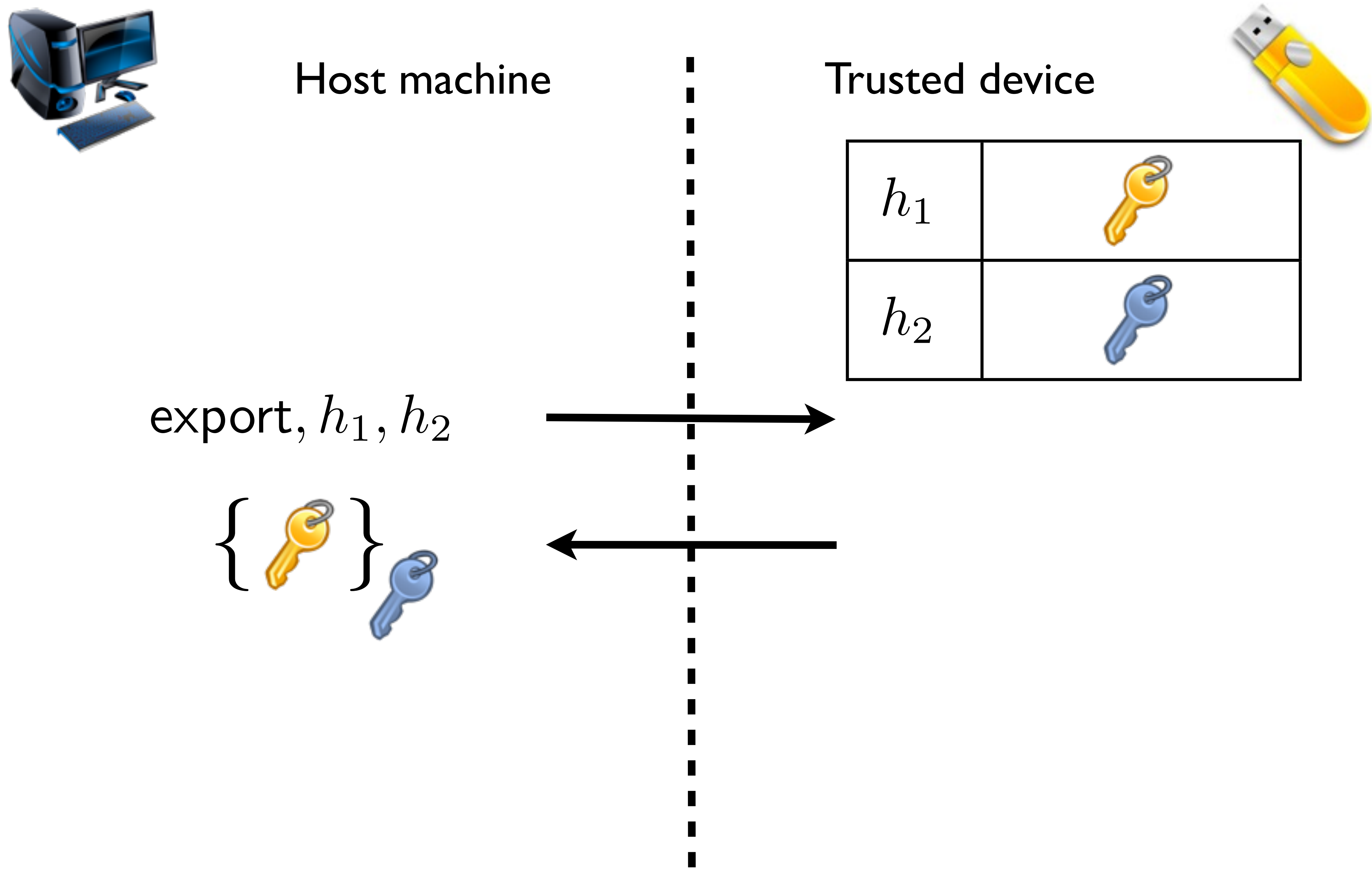
Trusted device

h_1	
h_2	

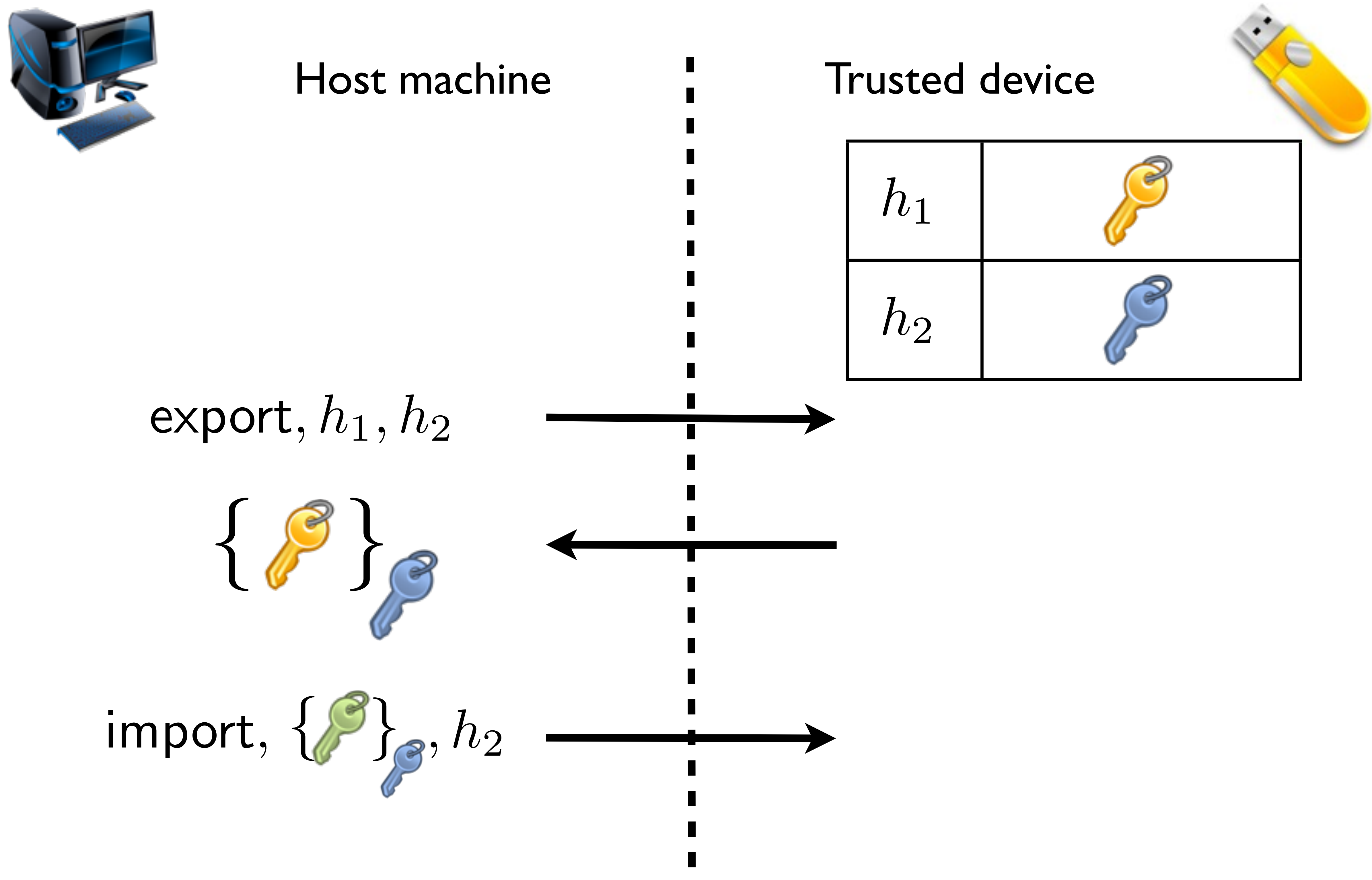
How does it work ?



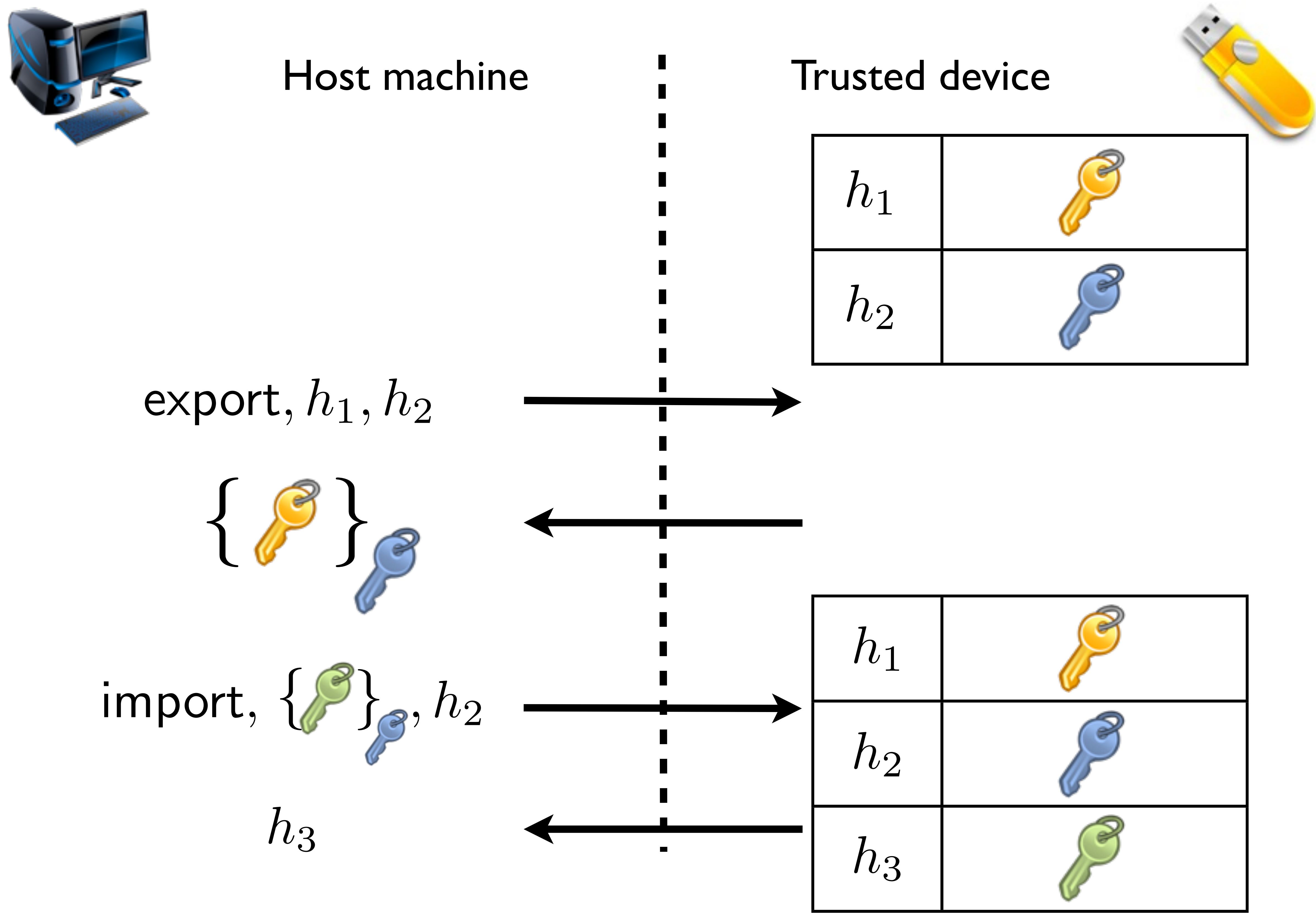
How does it work ?



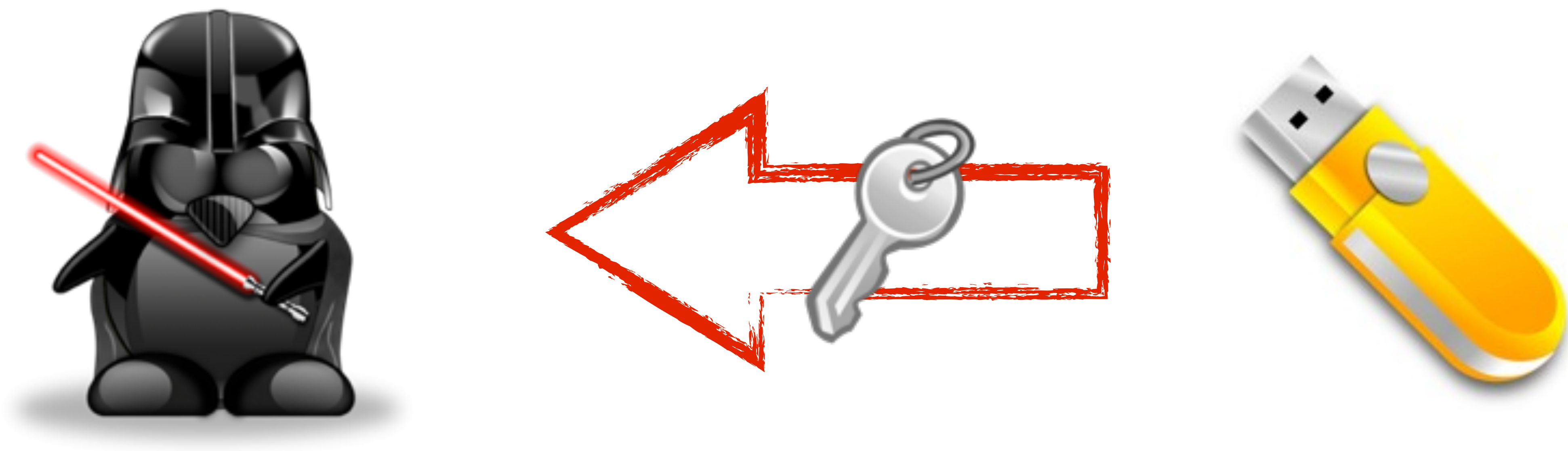
How does it work ?



How does it work ?



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,
- ...

Related Work

Proposals for key management APIs with security proofs but **without** adressing 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** adressing 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** adressing 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** adressing 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** adressing 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)

Ideal Key Revocation API

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

TRD

Description of the API

TRD

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

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, ...)

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)

Description of the API

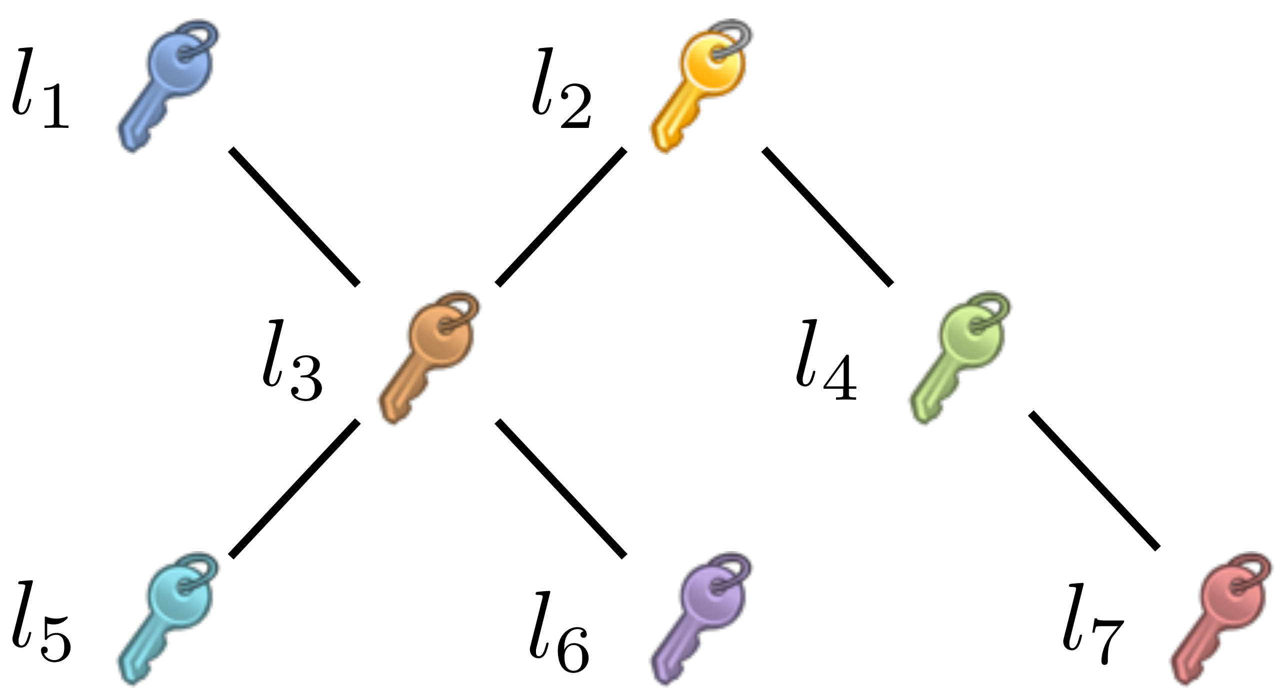
TRD

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.)



Description of the API

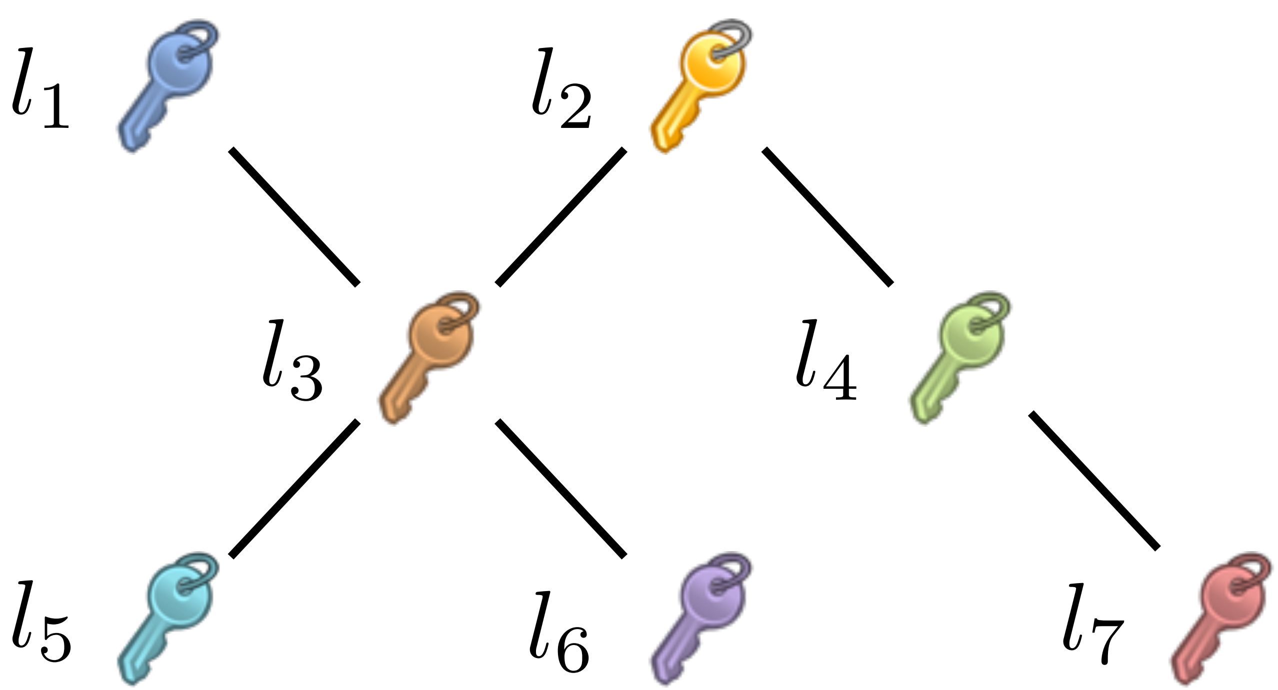
TRD

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

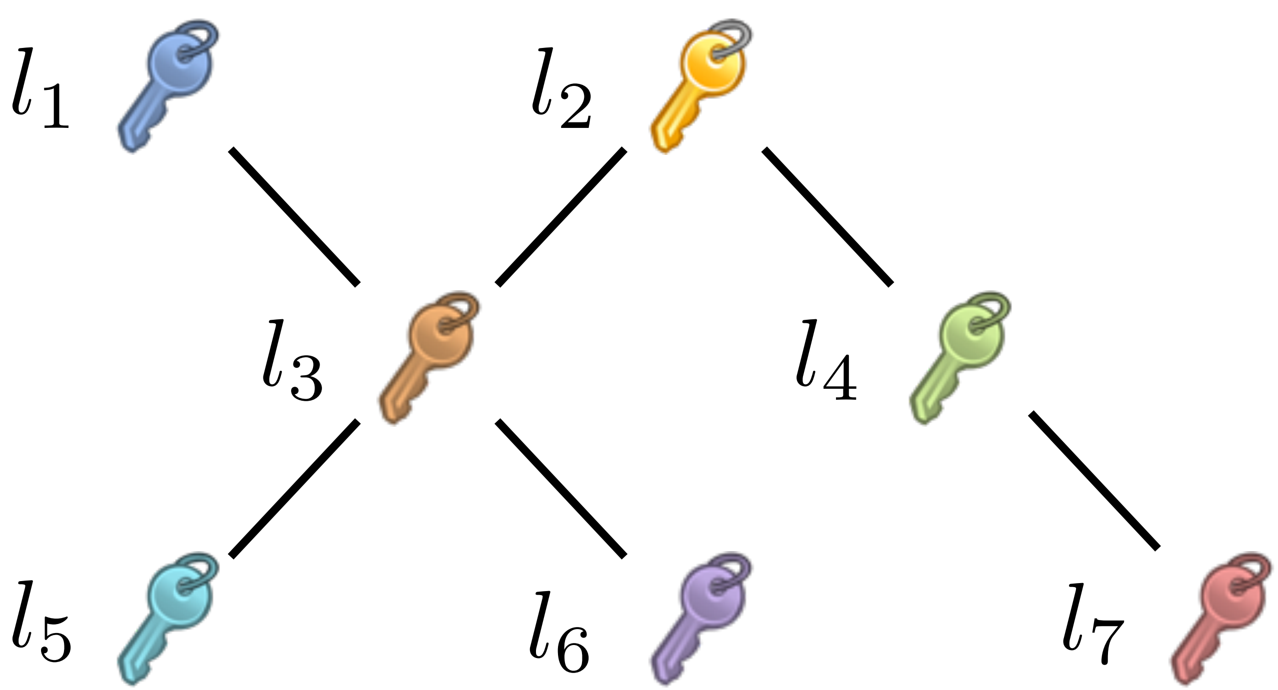
TRD

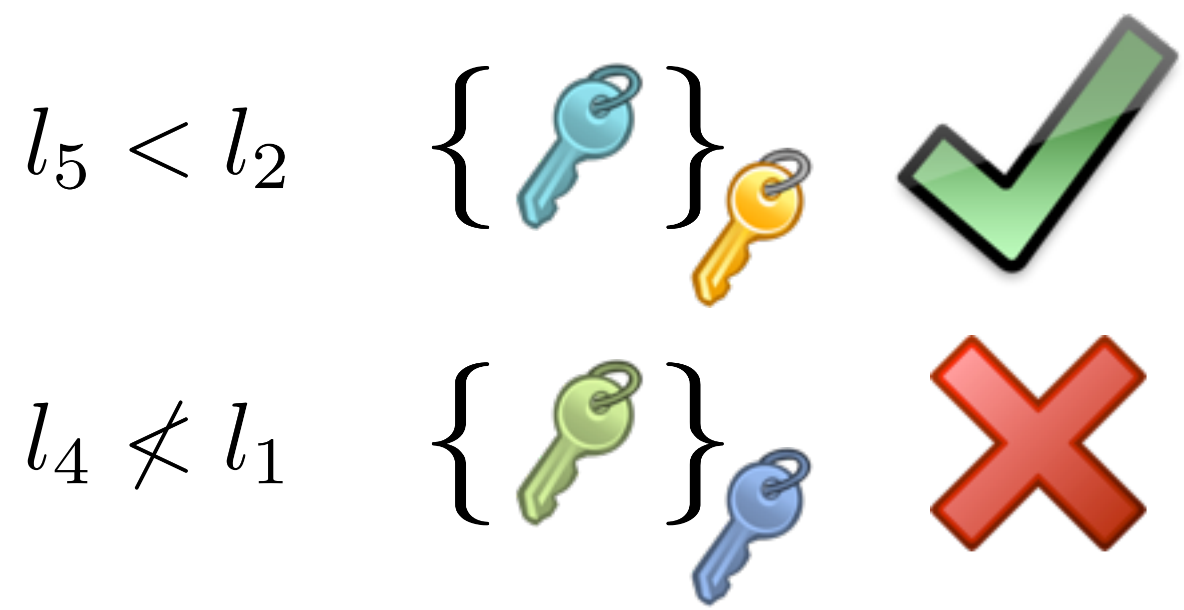
A **clock** assumed synchronized with a global clock

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.)





Description of the API

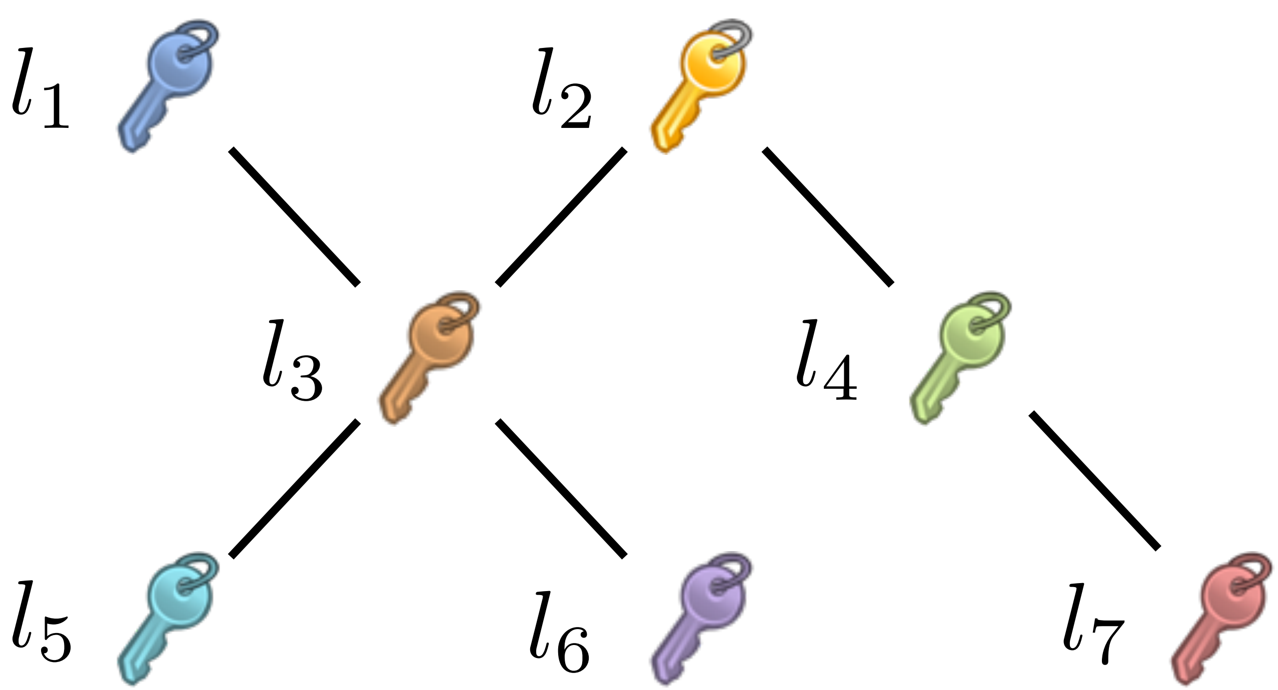
TRD

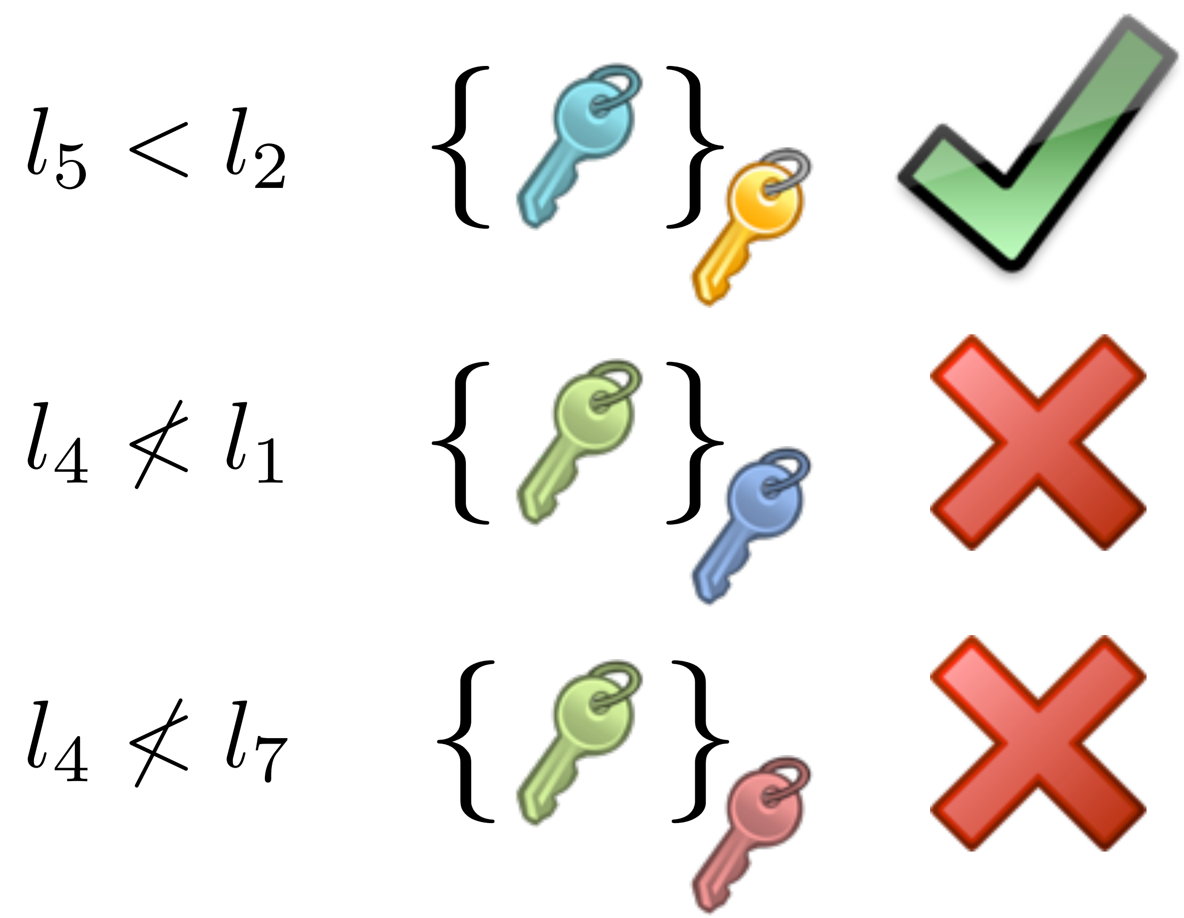
A **clock** assumed synchronized with a global clock

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.)





User's Commands

A set of **basic commands** :

«command» se prononce «commaaande»

$\text{generatePublic}(m)$

$\text{generateSecret}(l, m)$

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

Ex : $h \leftarrow (k, l, v, m)$

User's Commands

A set of **basic commands** :

«command» se prononce «commaaande»

generatePublic(m)

generateSecret(l, m)

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

Ex : $h \leftarrow (k, l, v, m)$

decrypt(C, h)

Ex : $C = \{\langle\langle$  $, l, v, m\rangle, \langle n, 0, v', m'\rangle\rangle\}$ 

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

User's Commands

A set of **basic commands** :

«command» se prononce «commaaande»

generatePublic(m)

generateSecret(l, m)

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

Ex : $h \leftarrow (k, l, v, m)$

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

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

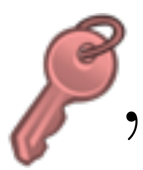
encrypt$(\langle X_1, \ldots, X_n\rangle, h)$

$X_i = h_i$ or $X_i = n_i$

$\{Y_1, \ldots, Y_n\}$

Lower Level Keys Management

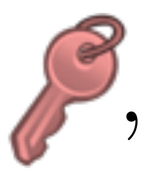
$\mathsf{update}(C, h_1, \ldots, h_n)$

h_1	 , Max, v_1
$\ldots$	 $\ldots$ 
h_n	 , Max, v_n
h	 $, l, v, m$

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

Lower Level Keys Management

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

h_1	 , Max, v_1
$\ldots$	 $\ldots$ 
h_n	 , Max, v_n
h	 $, l, v, m$

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

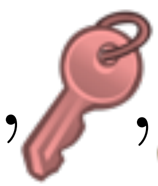
Attention, le «rouge» peut être vu comme du «rose» !

Lower Level Keys Management

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

h_1	 , Max, v_1
$\dots$	 $\dots$ 
h_n	 , Max, v_n
h	 , l, v, m

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

$C = \{$update ,  ,  , $l', v', m'\}$  $\dots$ 

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

Lower Level Keys Management

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

h_1	, Max, v_1
$\ldots$	$\ldots$
h_n	, Max, v_n
h	$, l, v, m$

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» !

Lower Level Keys Management

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

h_1	, Max, v_1
$\ldots$	$\ldots$
h_n	, Max, v_n
h	, l, v, m

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

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

h_1	, Max, v_1
$\ldots$	$\ldots$
h_n	, Max, v_n
h	, 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

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

h_1	, Max, v_1
$\ldots$	$\ldots$
h_n	, Max, v_n
h	, l, v, m

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

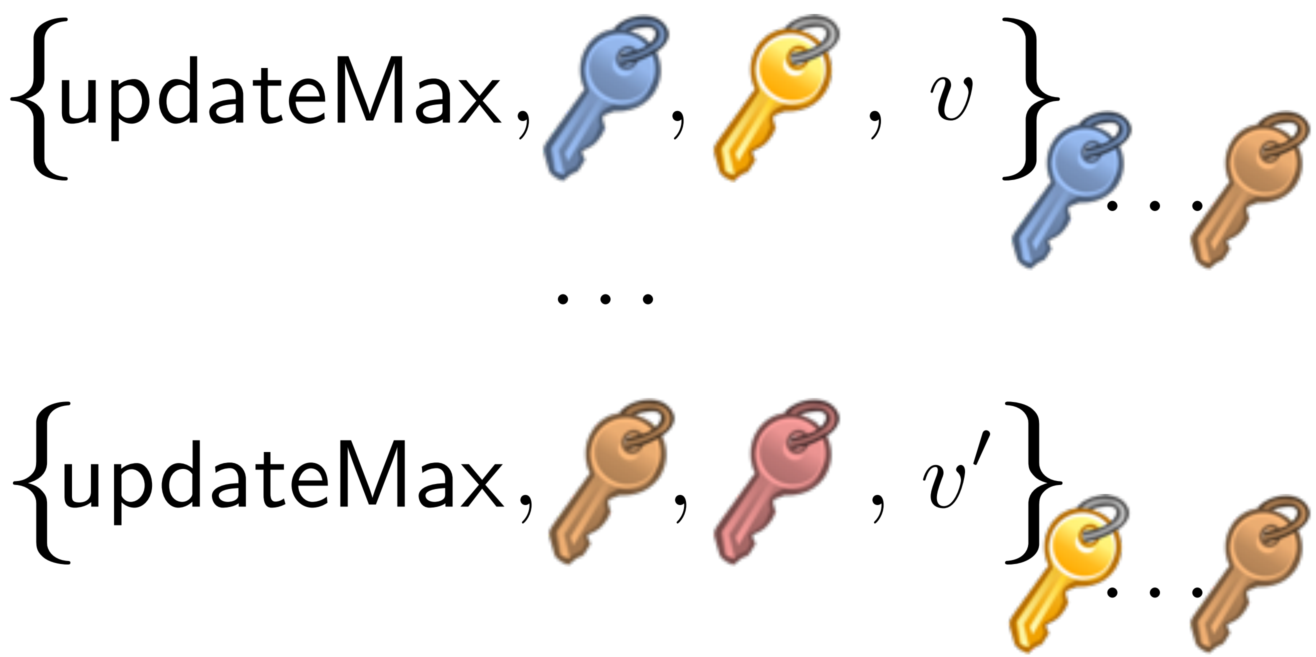
$$C = \{\text{update}, , , l', v', m'\} \ldots$$

3. Tests on the new attributes l', v'.
4. Table update with the new values.

h_1	, Max, v_1
$\ldots$	$\ldots$
h_n	, Max, v_n
h	, l', v', m'

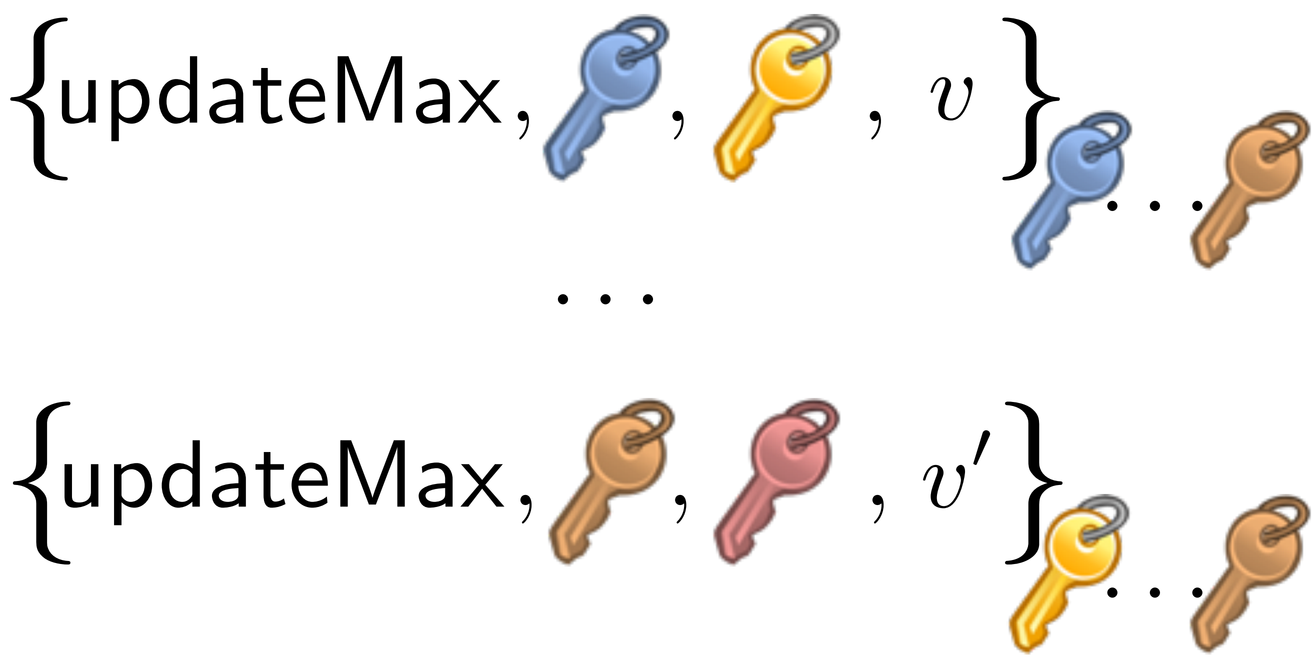
Attention, le «rouge» peut être vu comme du «rose» !

Revocation Keys Management



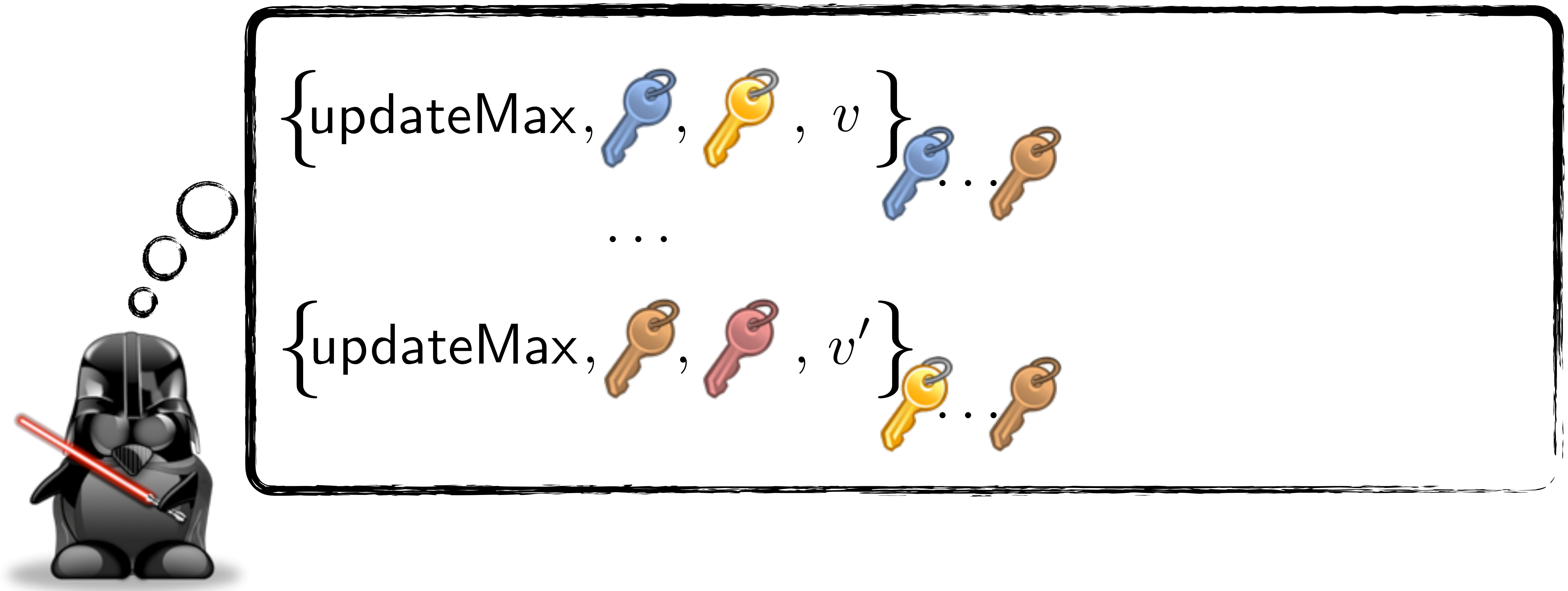
Revocation Keys Management

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



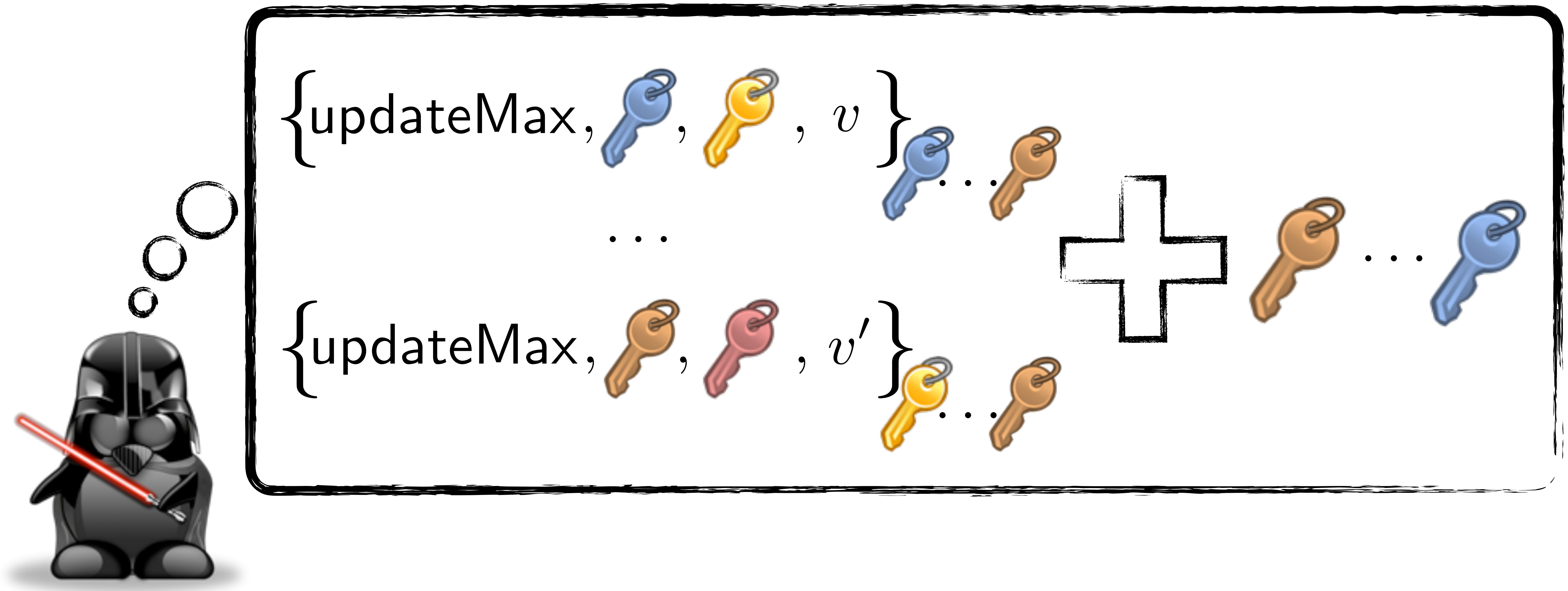
Revocation Keys Management

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



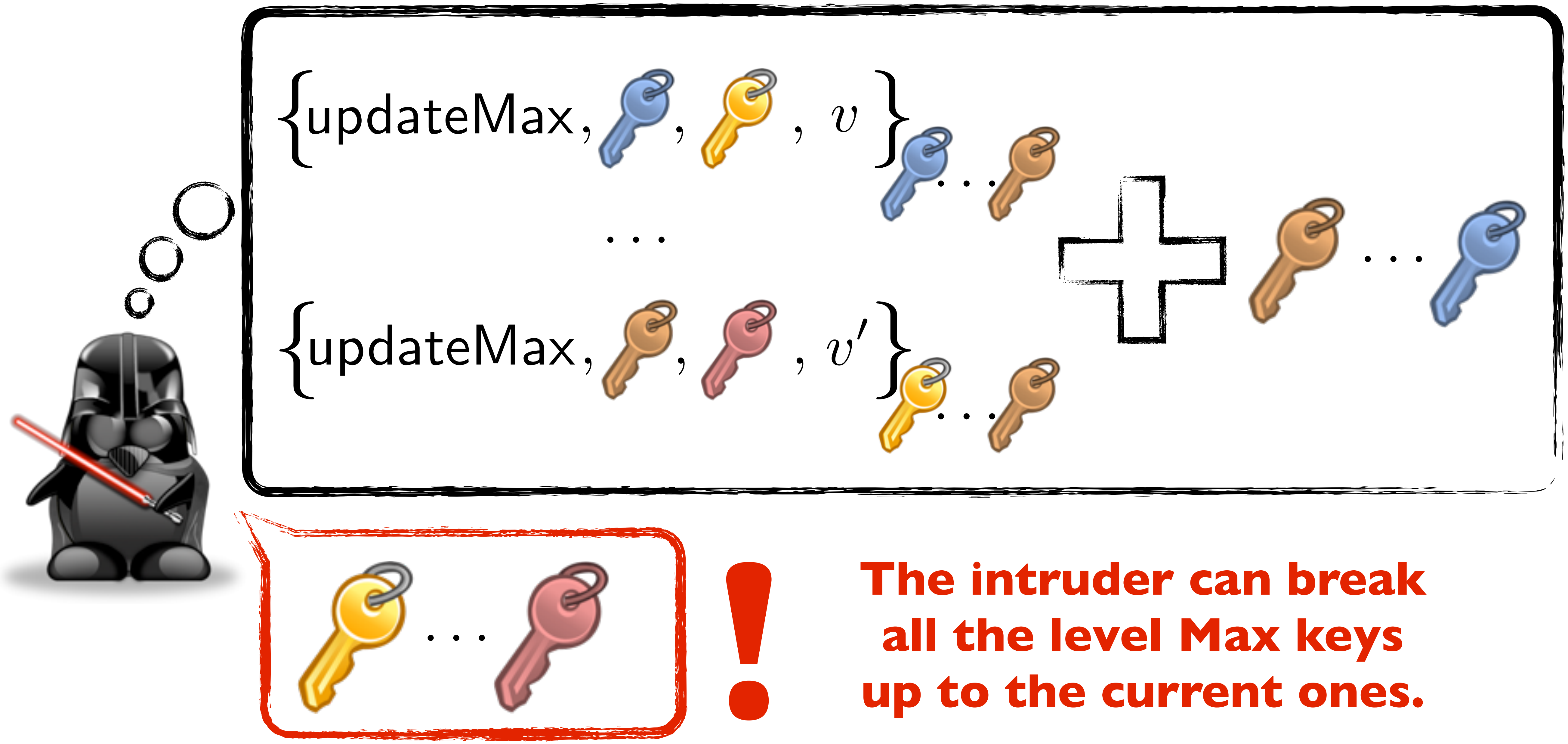
Revocation Keys Management

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 ?



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

Revocation Keys Management

Hypothesis :

Level Max commands are sent over a secure channel.

Revocation Keys Management

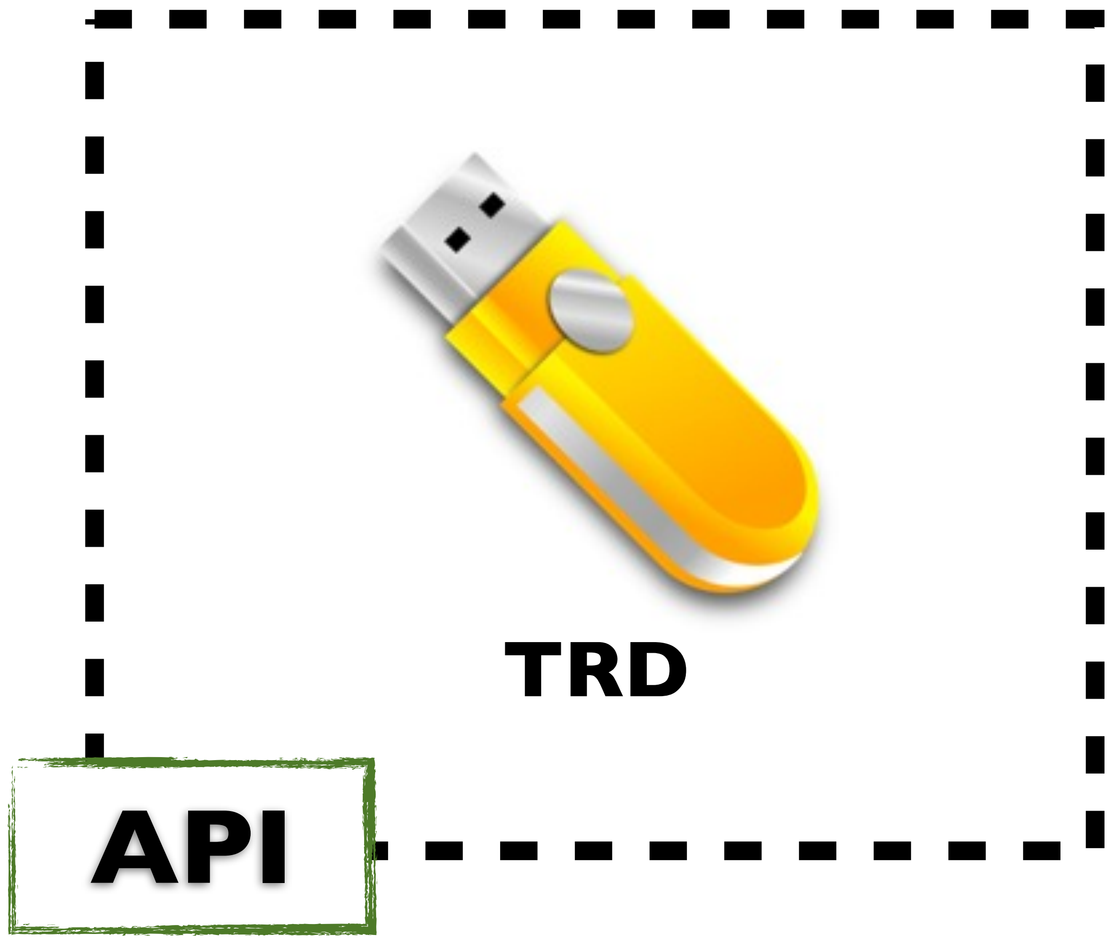
Hypothesis :

Level Max commands are sent over a secure channel.

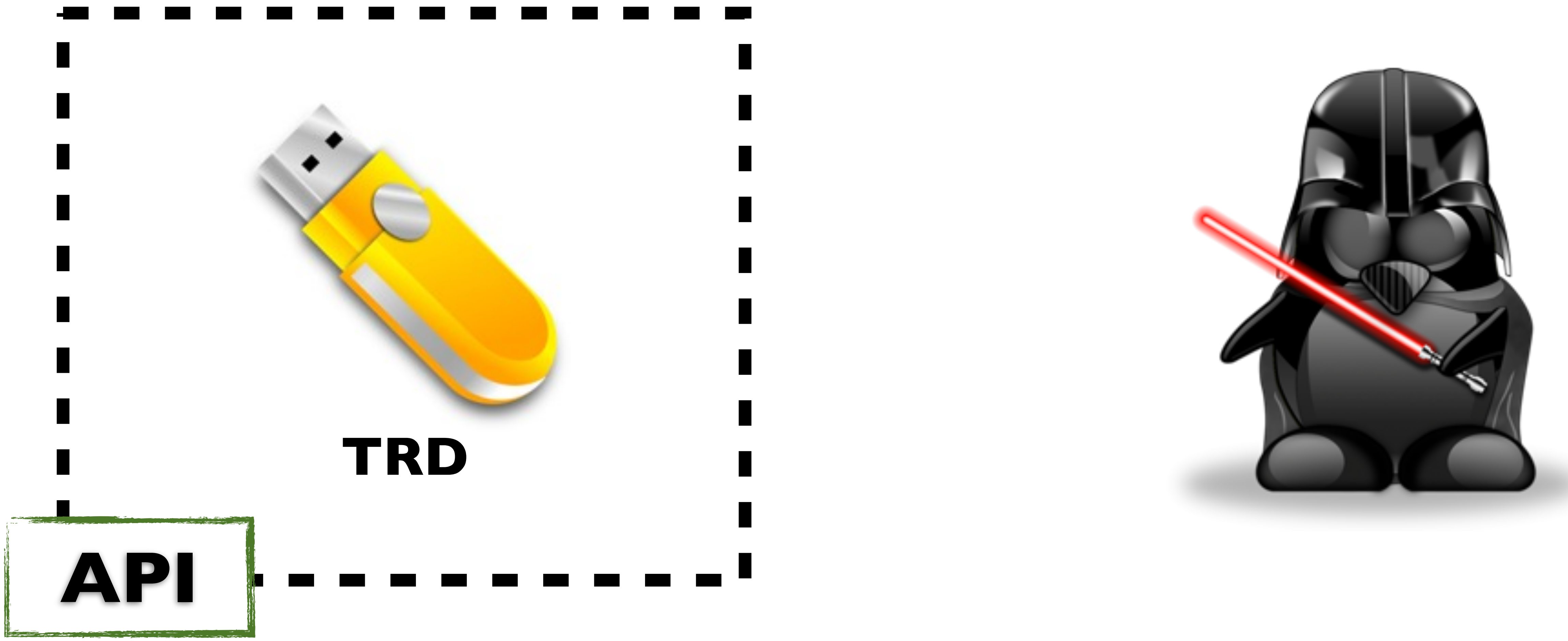
This can be achieved by several means :

- 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.

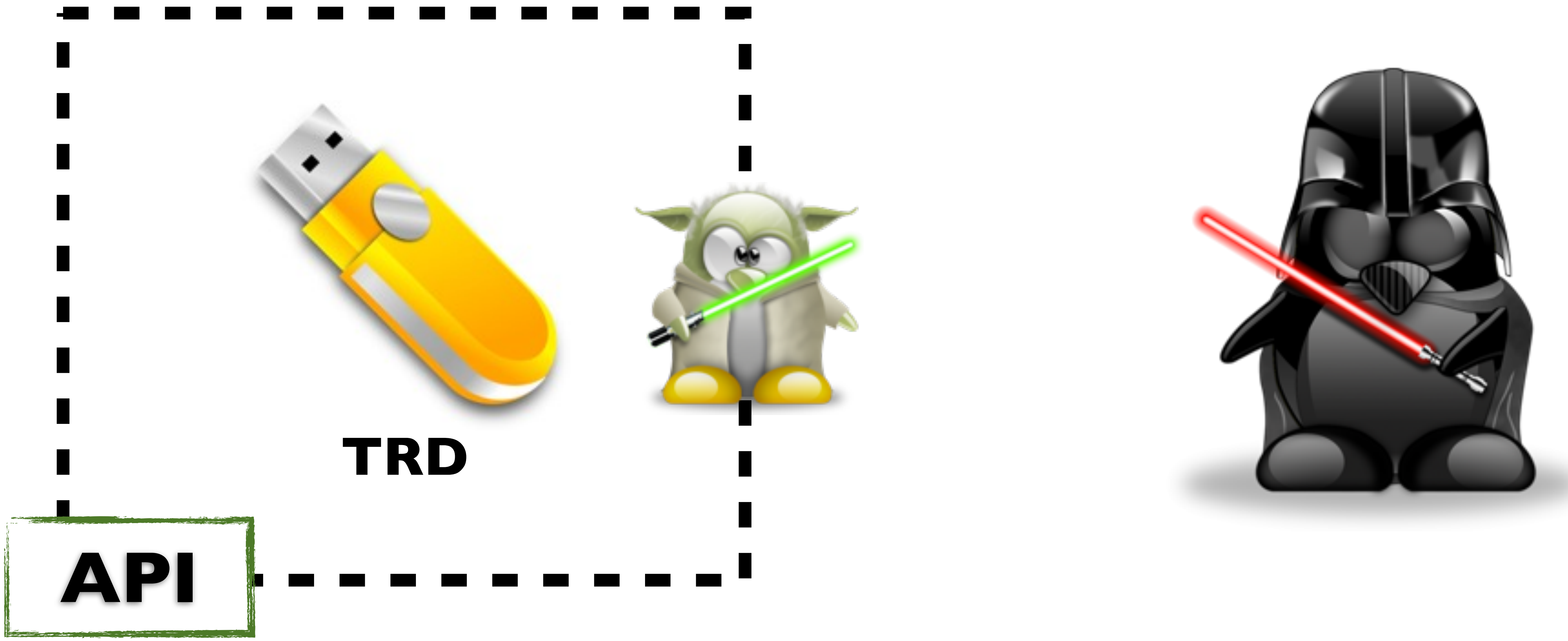
And now, what about Security ?



And now, what about Security ?



And now, what about Security ?



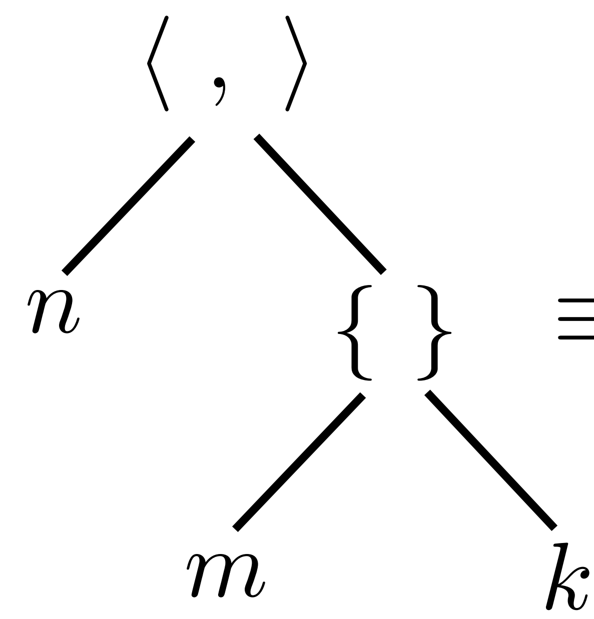
Abstraction

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

Messages are represented by **terms**

Nonces, keys :

$$n, m, \ldots, k_1, k_2, \ldots$$



$$\equiv \langle n, \{m\}_k \rangle$$

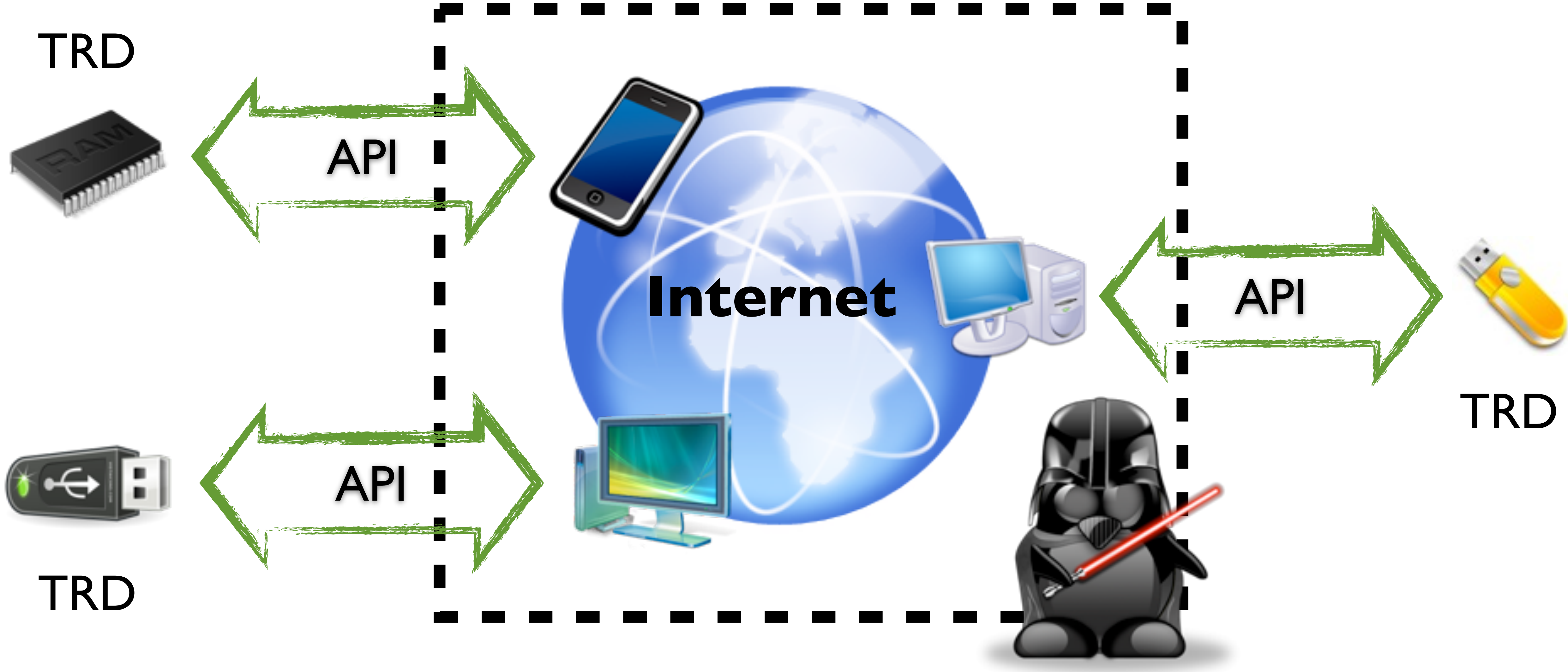
Primitives :

$$\{m\}_k,\ \langle m_1, m_2 \rangle, \ldots$$

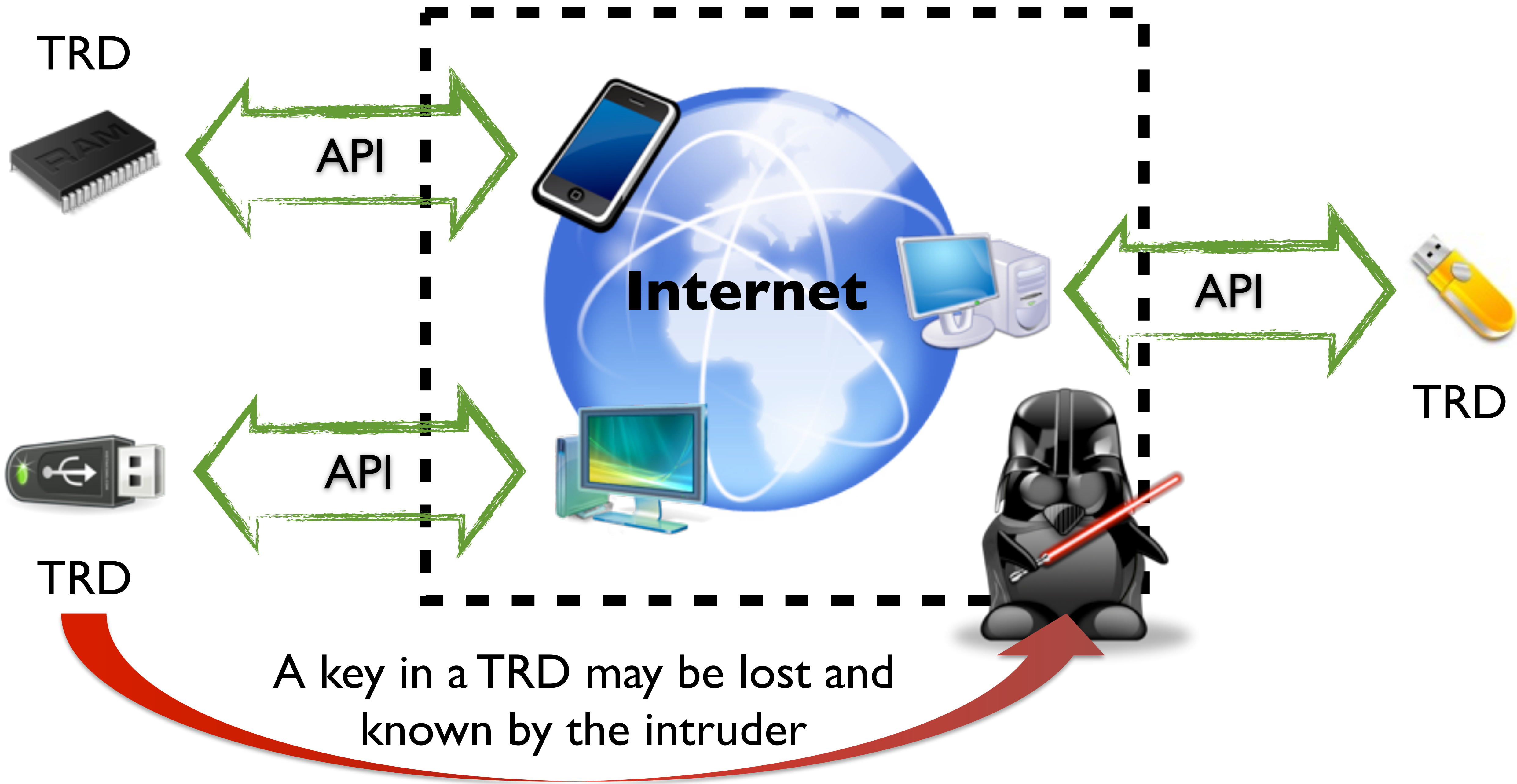
Modeling deduction rules :

$$\frac{x \quad y}{\langle x, y \rangle} \qquad \frac{\langle x, y \rangle}{x} \qquad \frac{\langle x, y \rangle}{y} \qquad \frac{x \quad y}{\{x\}_y} \qquad \frac{\{x\}_y \quad y}{x}$$

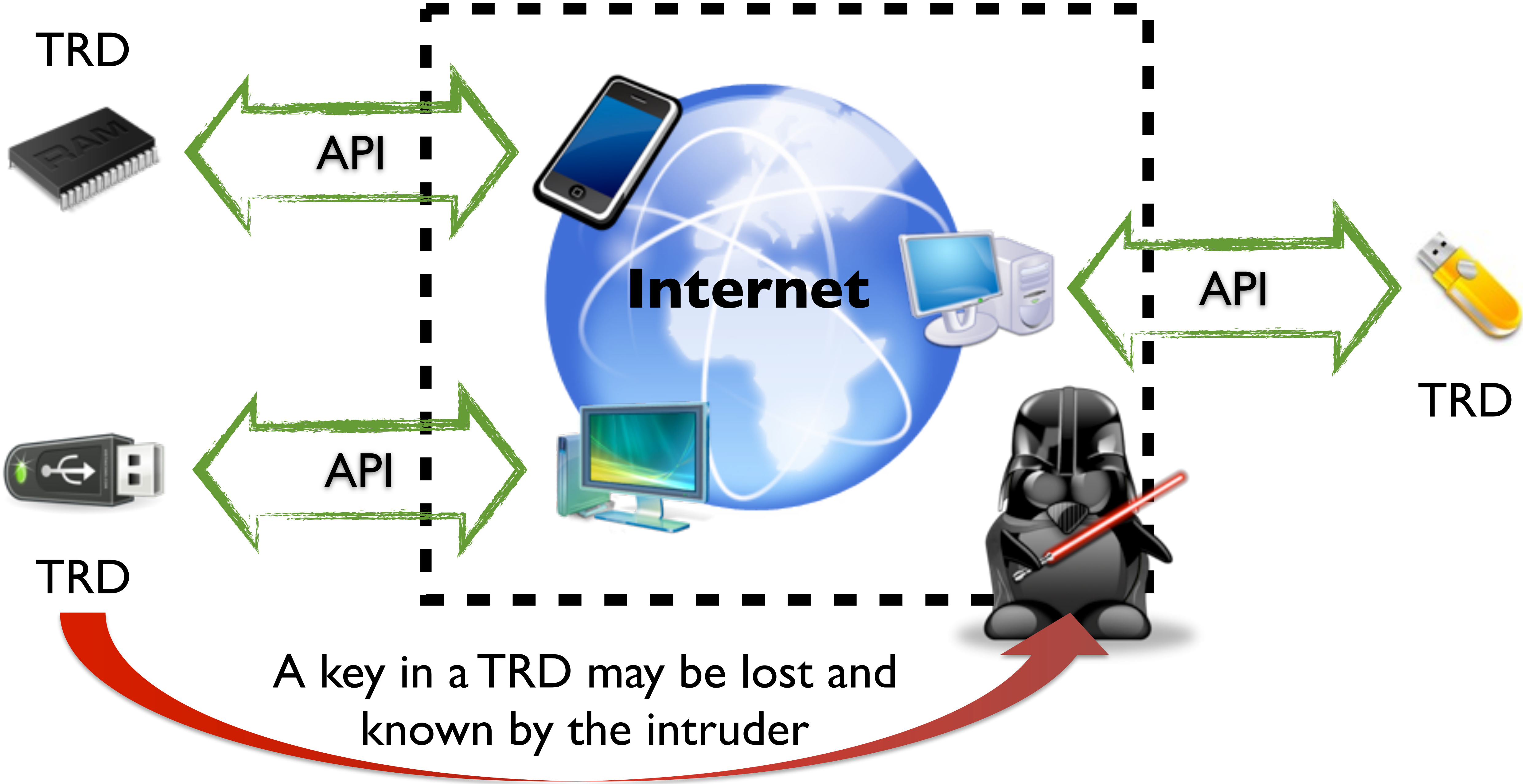
Knowledge of the Intruder



Knowledge of the Intruder

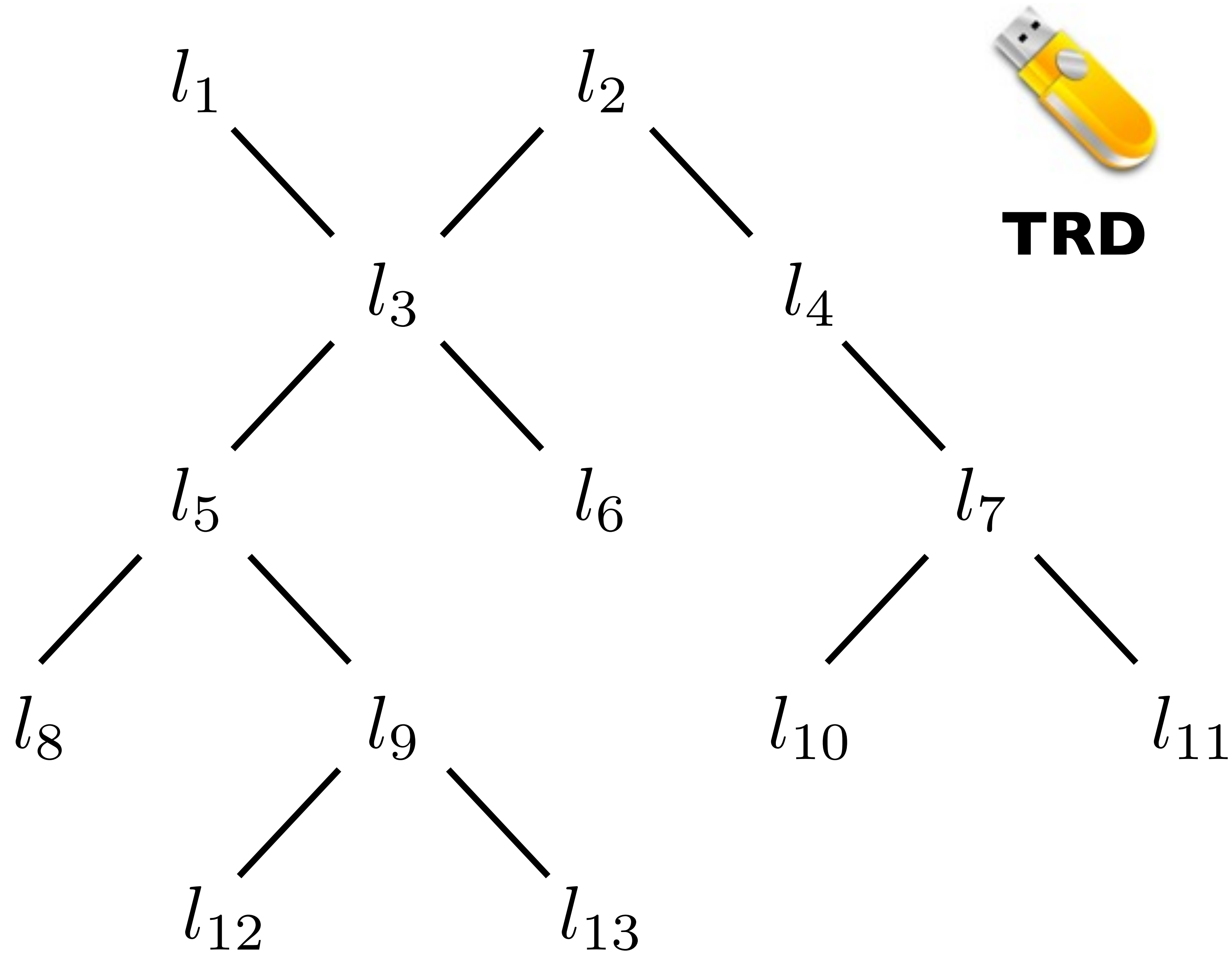


Knowledge of the Intruder



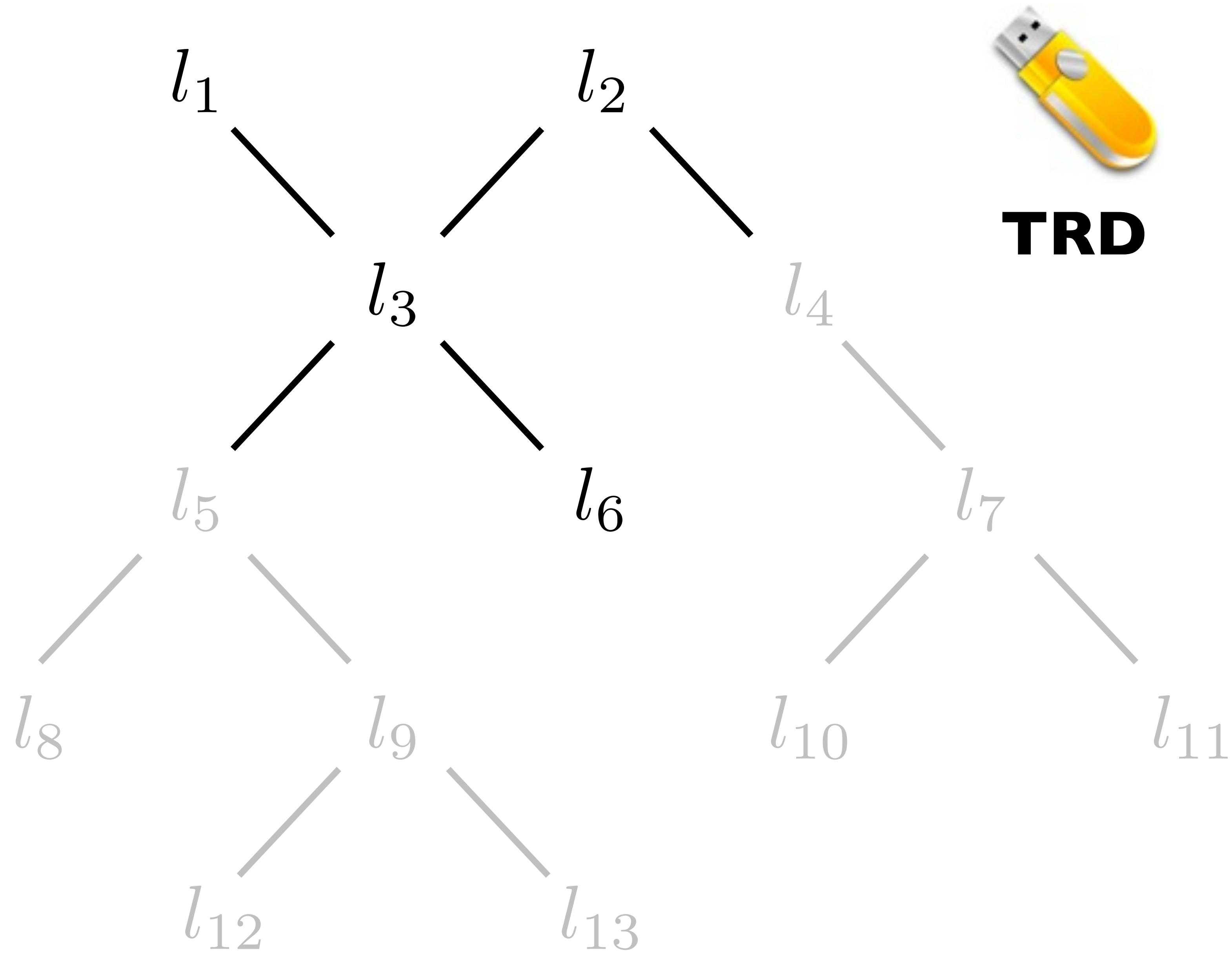
Hypothesis : At most a total of $N_{\text{Max}} - 1$ different « current » level Max keys for one TRD can be lost.

What about lost levels ?



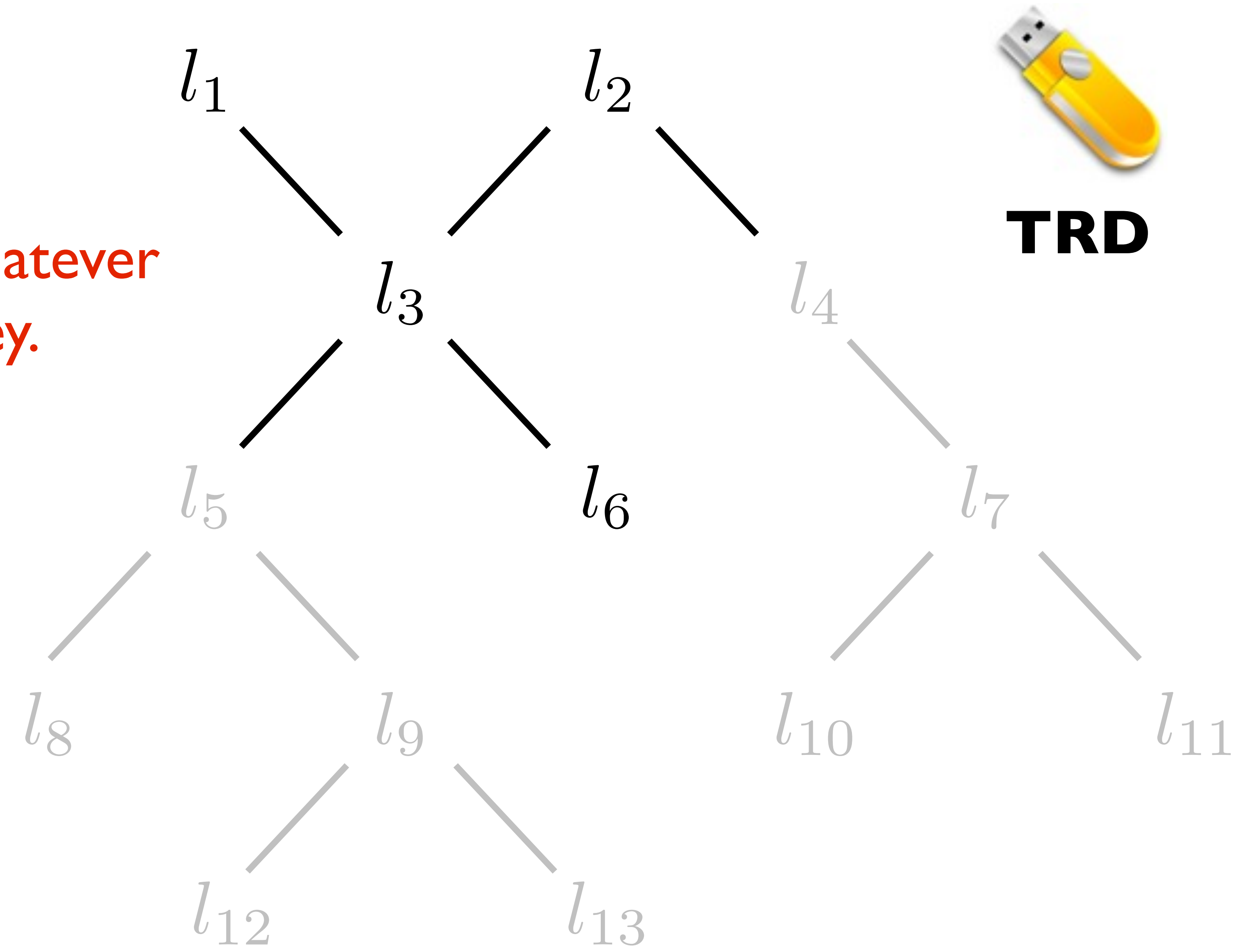
TRD

What about lost levels ?



TRD

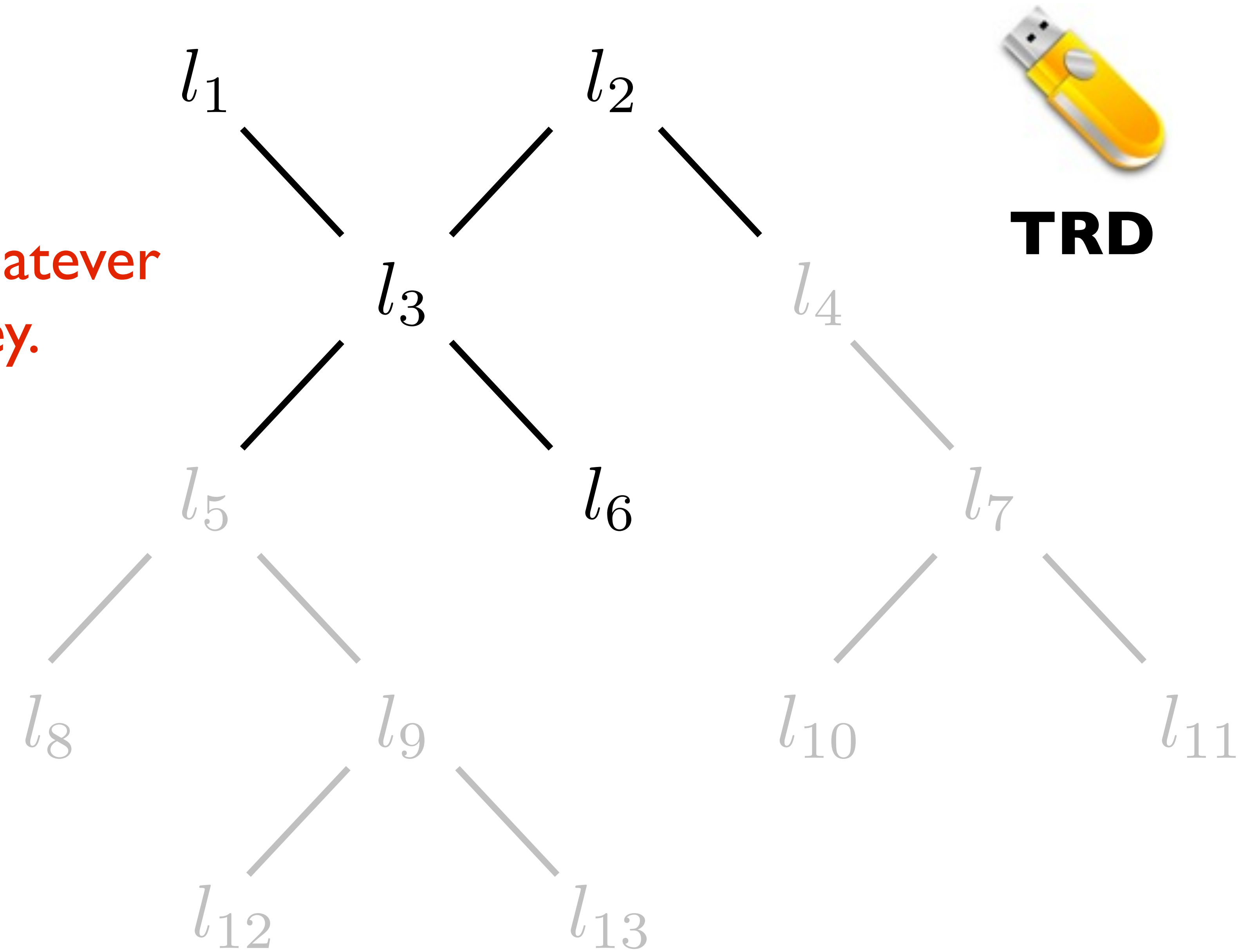
What about lost levels ?



TRD

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 $\{\langle$  $, l_9, v, m\rangle\}$  with  lost and of level l_5 .

Secrecy Result

«I keep my secrets secret !»

Even if the **intruder** may :

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

Secrecy Result

«I keep my secrets secret !»

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 »**

Self Repair Property

«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 $l_i < l$.

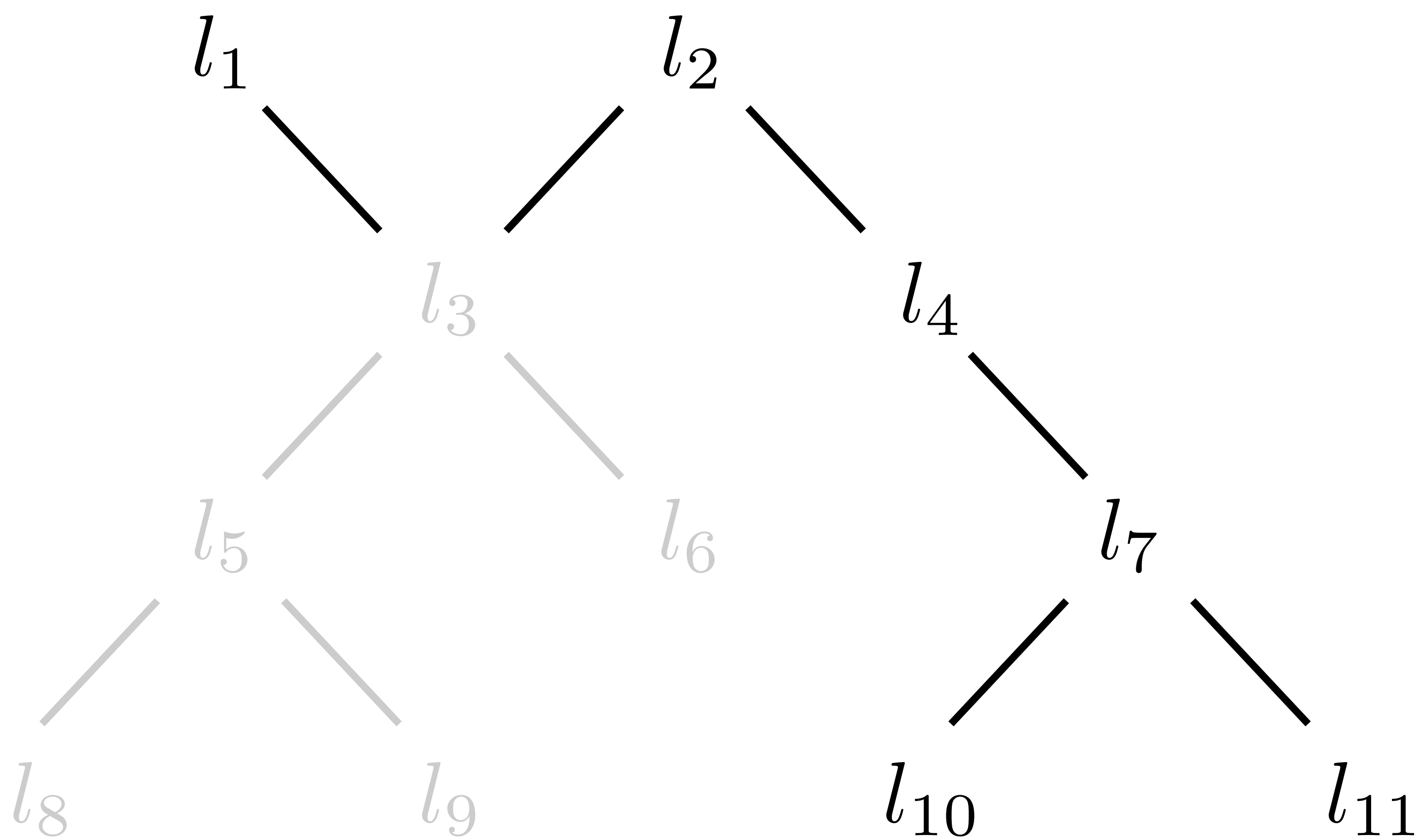
Self Repair Property

«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 $l_i < l$.



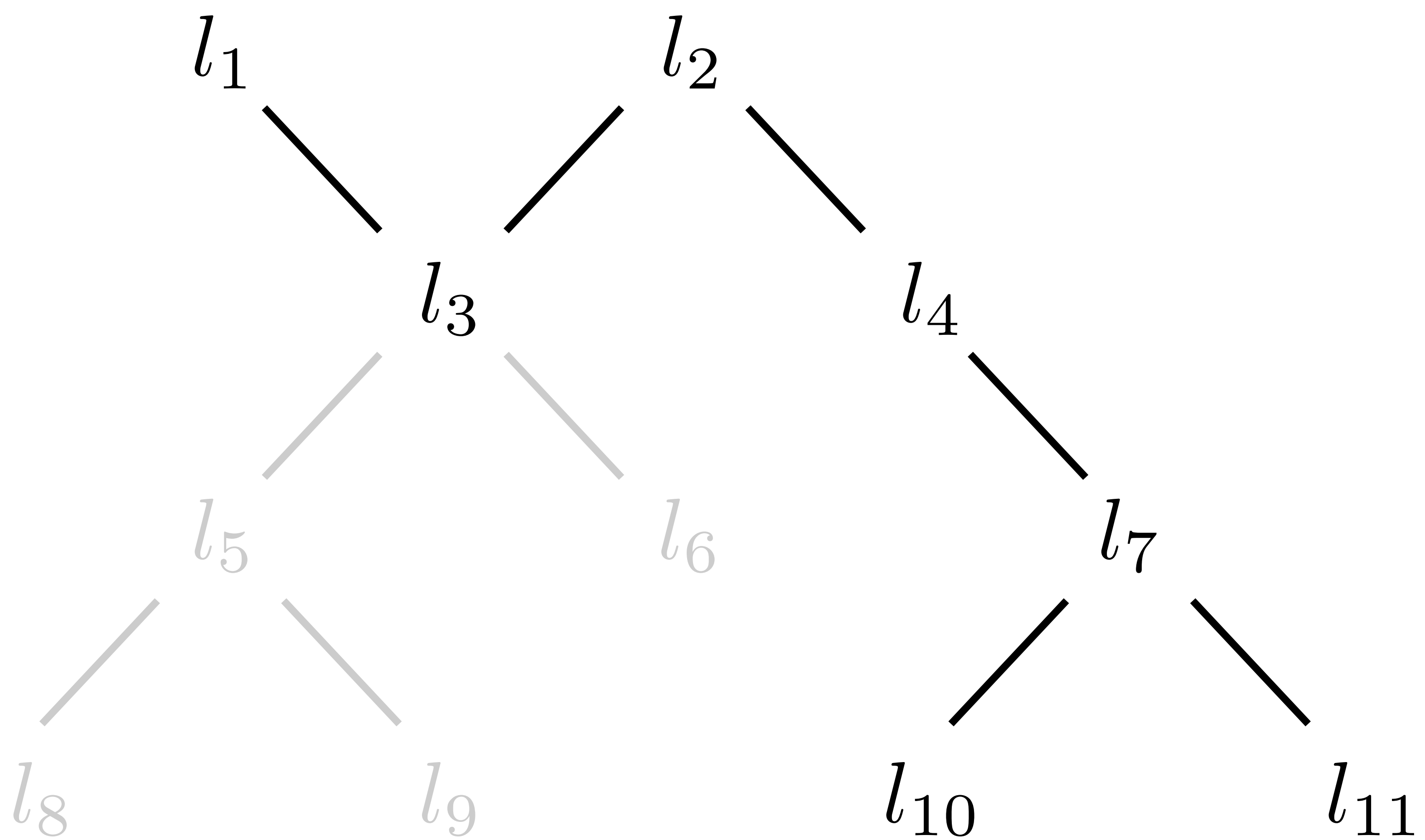
Self Repair Property

«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 $l_i < l$.



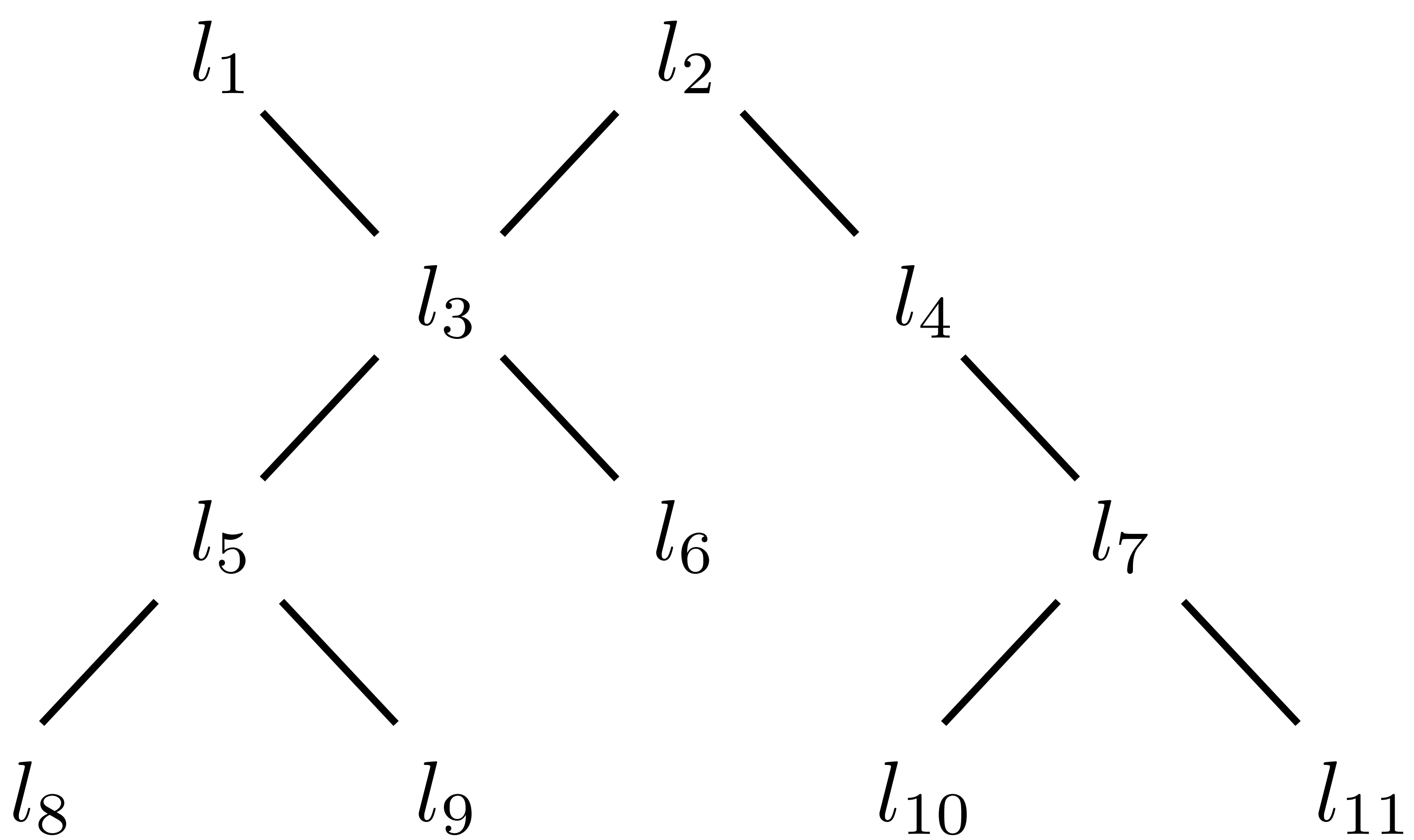
Self Repair Property

«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 $l_i < l$.



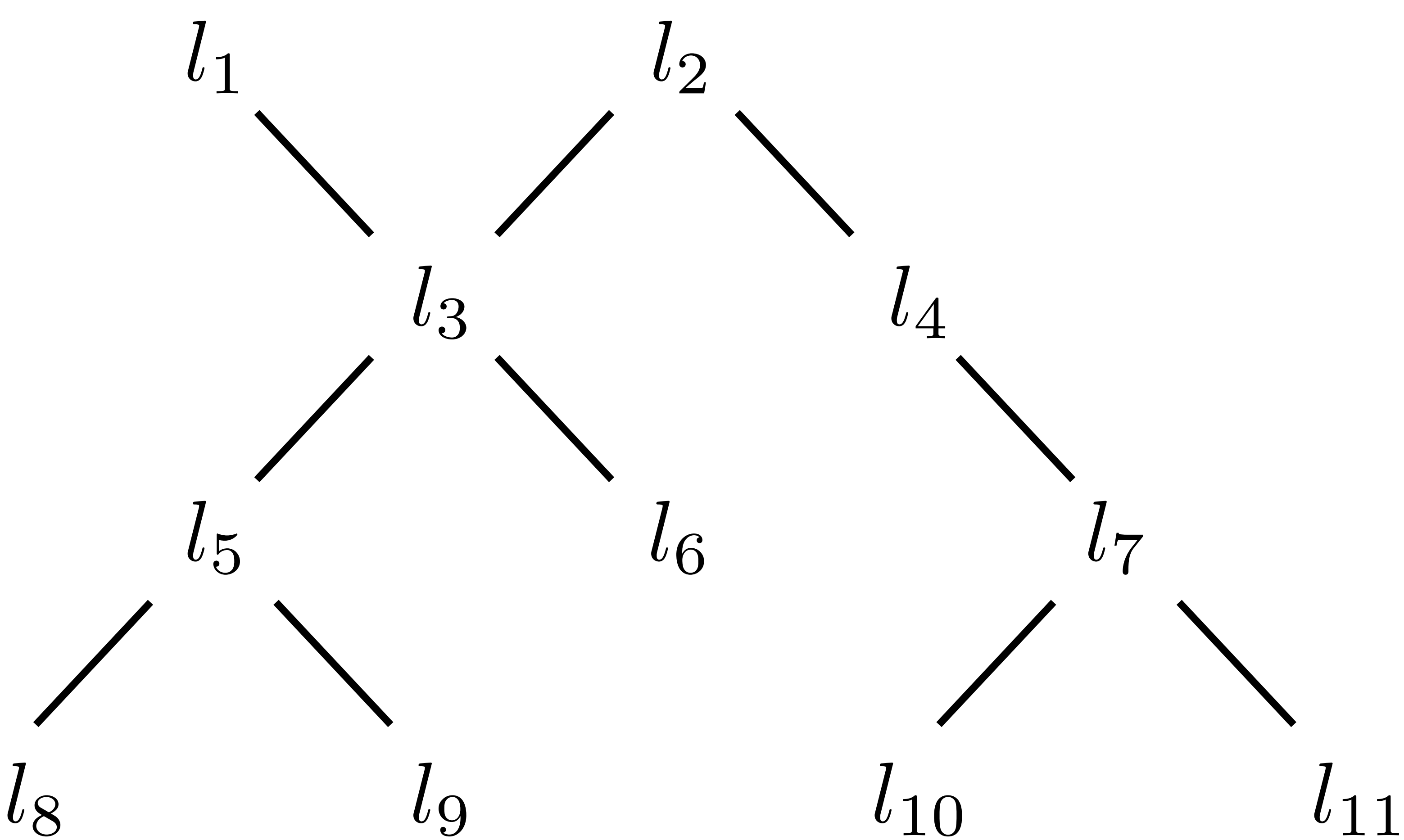
Self Repair Property

«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 $l_i < l$.



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

Blacklist Option

«For those who are in a hurry...»

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

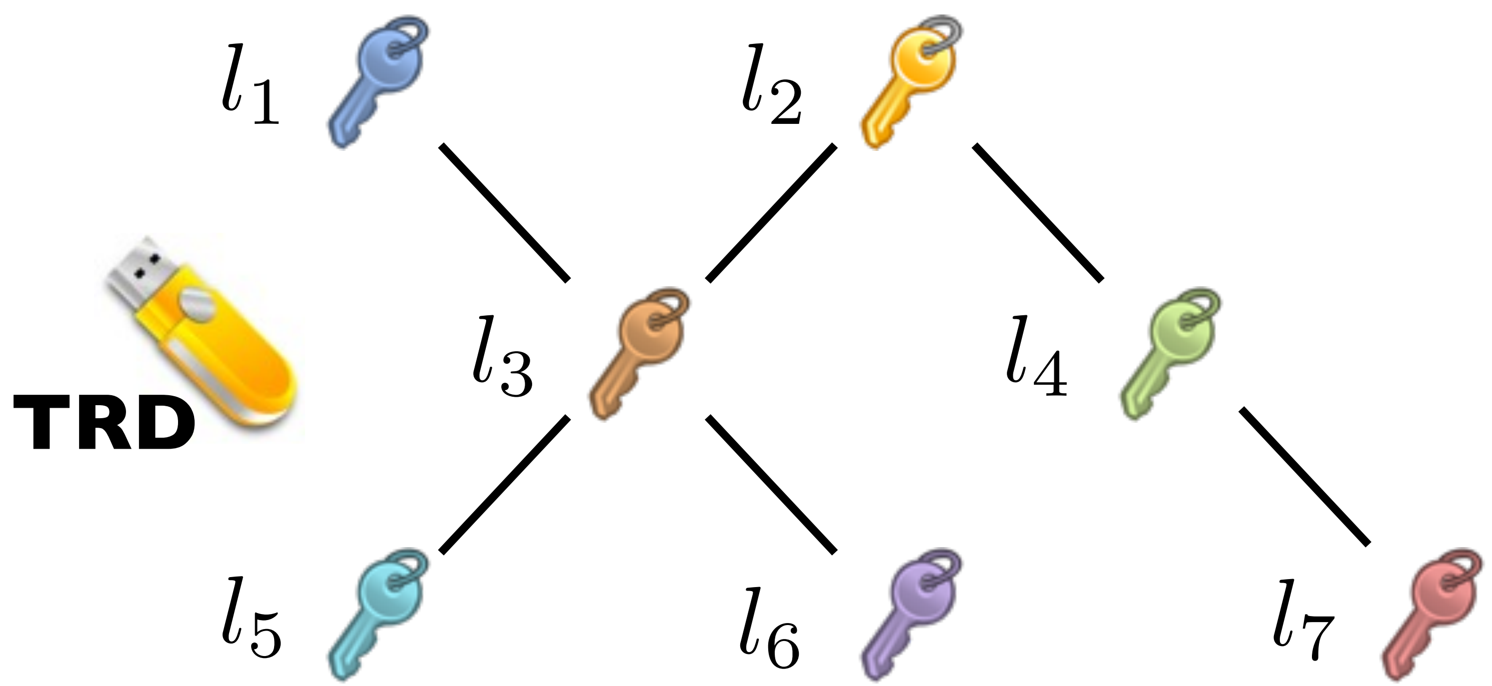
$\text{Ex} : C = \left\{ \langle \text{blacklist}, \langle l_3, t \rangle \rangle \right\}$  $\ldots$ 

Blacklist Option

«For those who are in a hurry...»

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

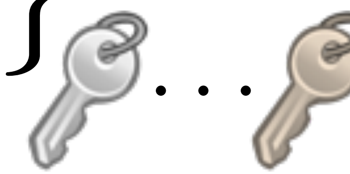
Ex : $C = \left\{ \langle \text{blacklist}, \langle l_3, t \rangle \rangle \right\}$

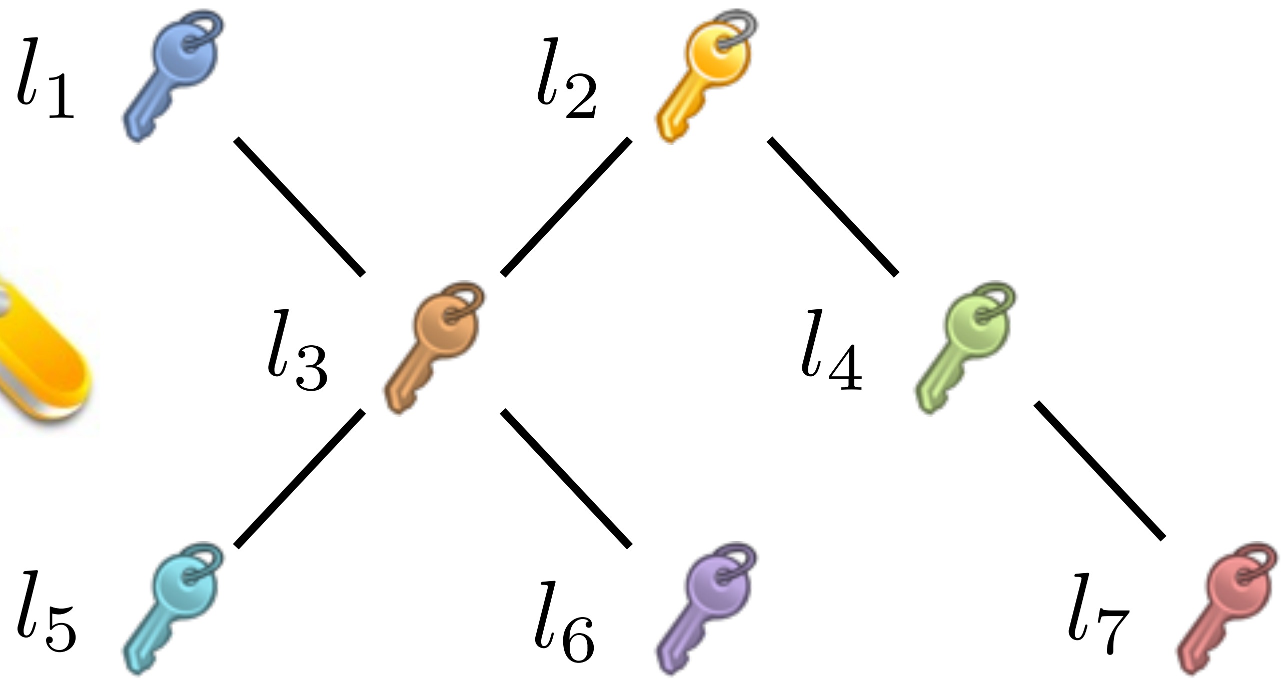


Blacklist Option

«For those who are in a hurry...»

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

Ex : $C = \left\{ \langle \text{blacklist}, \langle l_3, t \rangle \rangle \right\}$ 



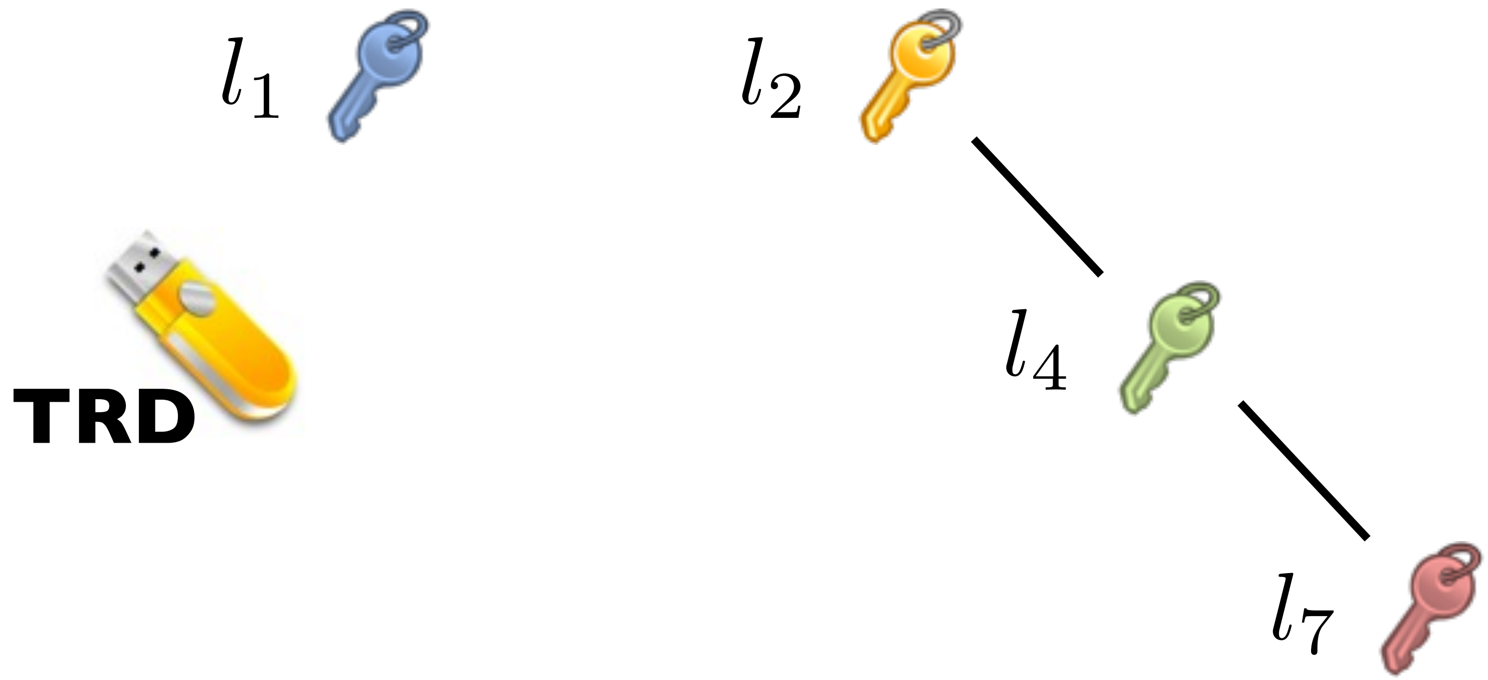
$(l_3, t_3) \longrightarrow$ 

Blacklist Option

«For those who are in a hurry...»

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

Ex : $C = \left\{ \langle \text{blacklist}, \langle l_3, t \rangle \rangle \right\} \ldots$



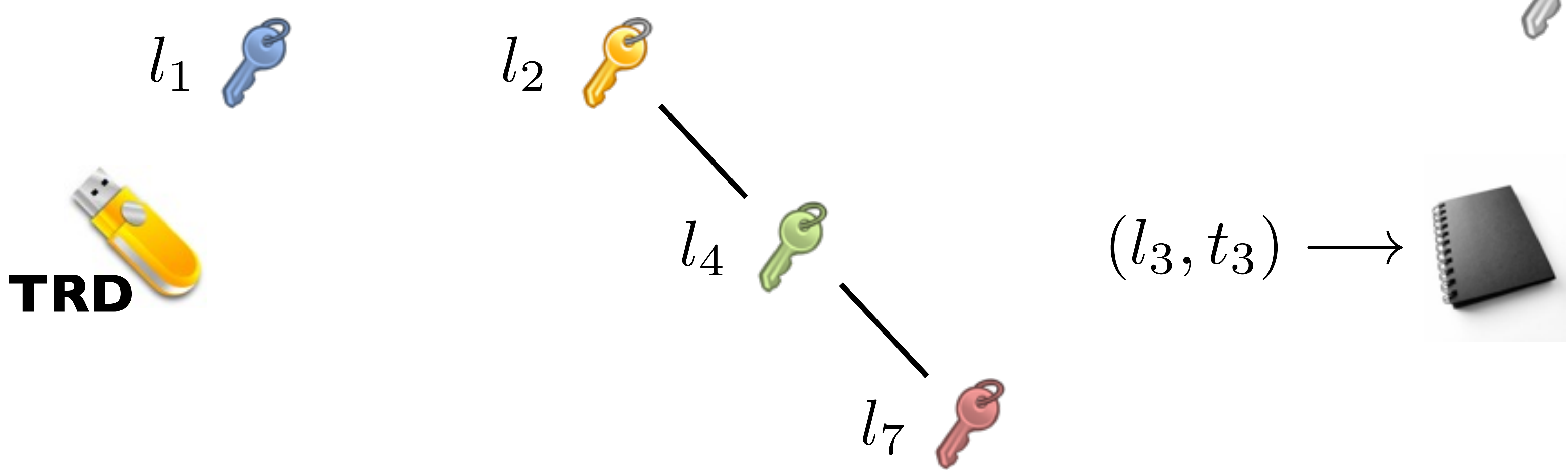
$(l_3, t_3) \longrightarrow$

Blacklist Option

«For those who are in a hurry...»

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

Ex : $C = \left\{ \langle \text{blacklist}, \langle l_3, t \rangle \rangle \right\}$ ...



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 !