Revoke and Let Live

A Secure Key Revocation API for Cryptographic Devices

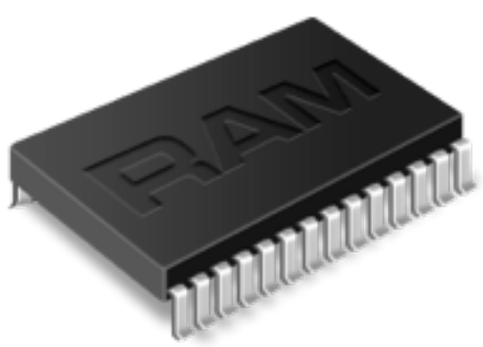
Véronique Cortier Graham Steel Cyrille Wiedling

LORIA-CNRS, Nancy (FR)

June 28th, 2012

Security APIs

Trusted devices



Host machines

Security API

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

Applications

- Smartphones,

- Bank Accounts,

- Electronic Ticketing Systems (e.g. Calypso),

- Vehicle-to-vehicle networking.

- ...

How does it work ?

Host machine

Trusted device

h_1	
h_2	

How does it work ?

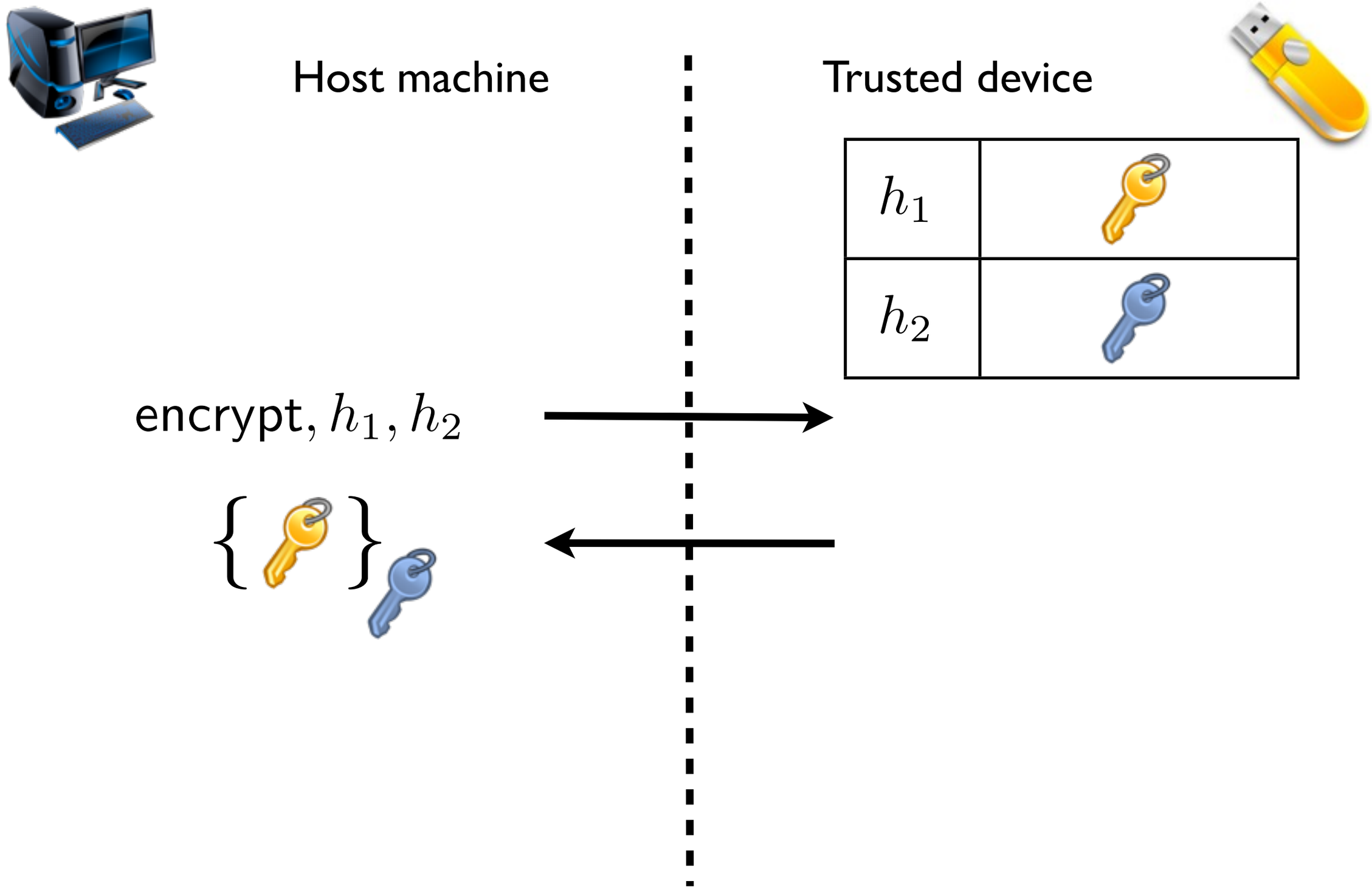
Host machine

Trusted device

h_1	
h_2	

encrypt, h_1, h_2 →

How does it work ?



How does it work ?

Host machine

Trusted device

h_1	
h_2	

encrypt, h_1, h_2 →

← $\{$  $\}$ 

decrypt, $\{$  $\}$  , h_2 →

How does it work ?

Host machine

Trusted device

h_1	
h_2	

encrypt, h_1, h_2 →

← $\{$  $\}$ 

decrypt, $\{$  $\}$  , h_2 →

h_3 ←

h_1	
h_2	
h_3	

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**.

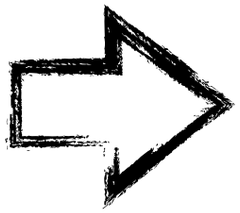
J. Courant, J.-F. Monin, WITS'06.

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 not verified)

Ideal Key Revocation API

The device should remain **functional** :

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

Ideal Key Revocation API

The device should remain **functional** :

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

Any key should be **revocable** :

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

Ideal Key Revocation API

The device should remain **functional** :

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

Any key should be **revocable** :

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

Keys must remains **confidential** :

Information about key should not be recovered by the intruder.

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,
- ...

Our Contributions

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

- A **formal proof of security** ensuring three properties :
  - Except is a **key** is lost then it **remains secret**,
  - 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

A **clock** supposed synchronized with a global clock

Description of the API

TRD

A **clock** supposed synchronized with a global clock

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

Description of the API

TRD

A **clock** supposed synchronized with a global clock

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

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

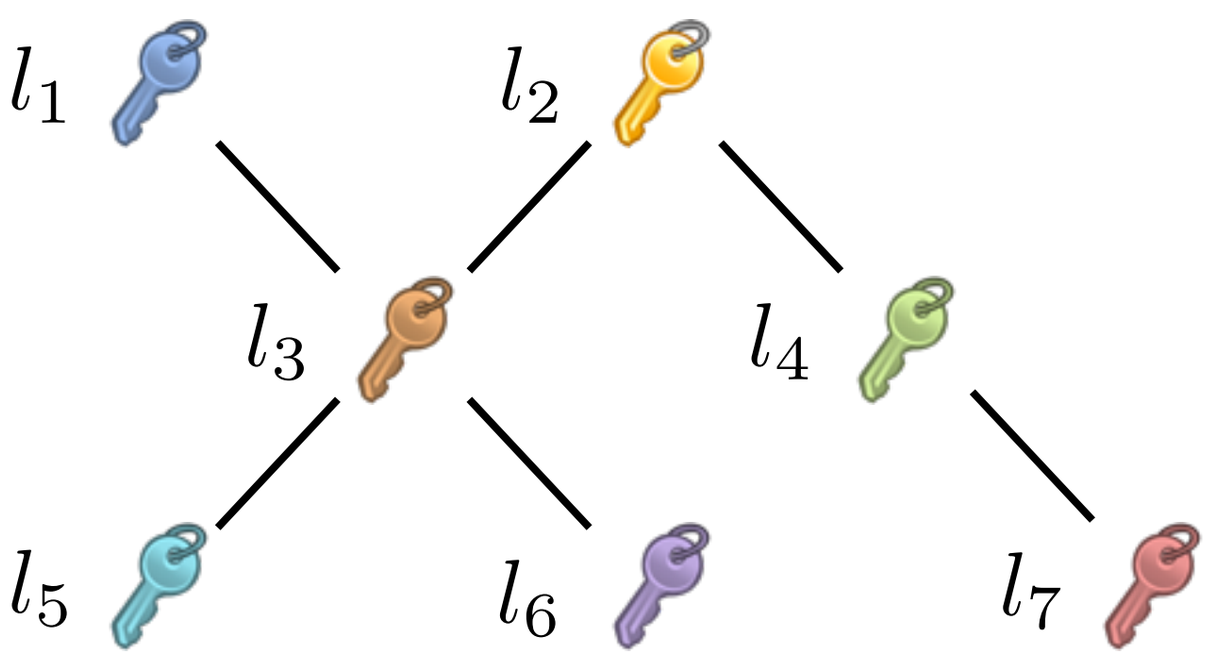
Description of the API

A **clock** supposed synchronized with a global clock

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

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

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



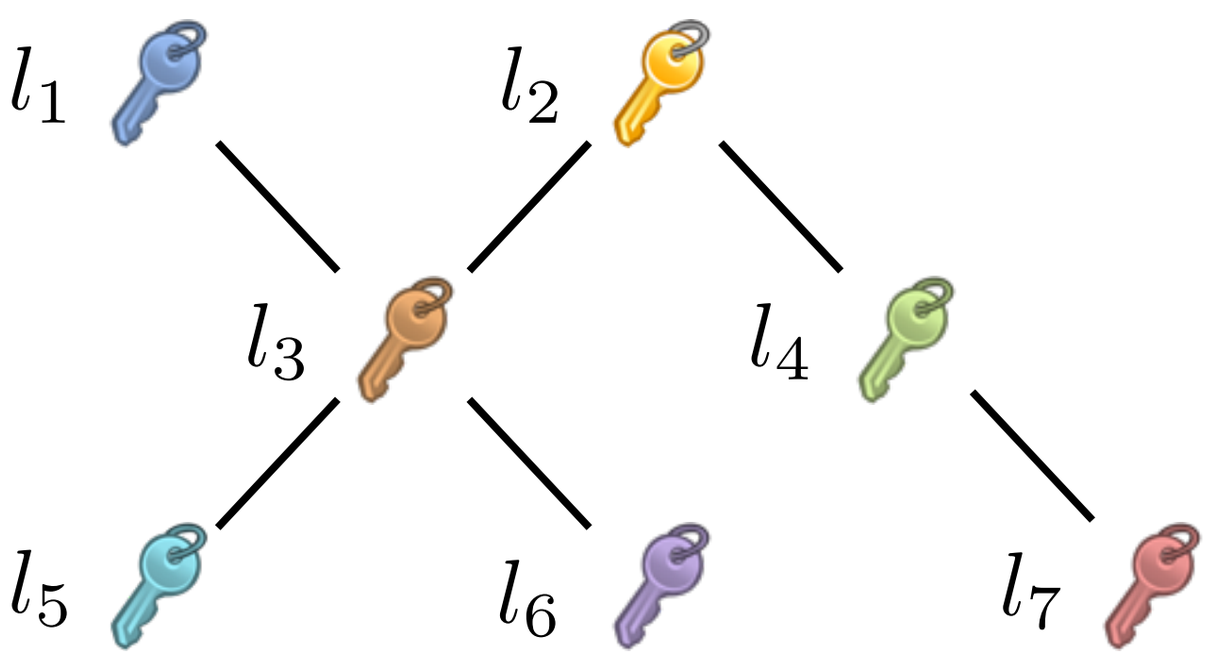
Description of the API

A **clock** supposed synchronized with a global clock

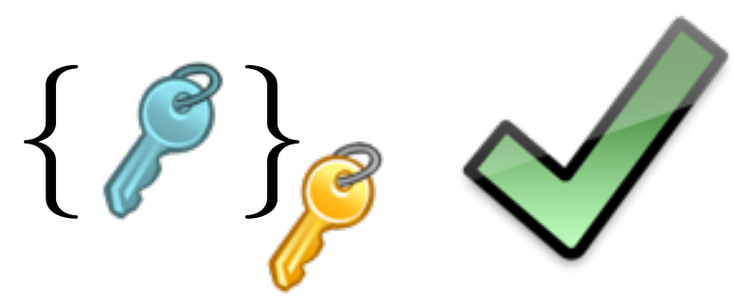
A **table** indexed by handles to store keys' information (level, validity, ...)

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

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



$l_5 < l_2$



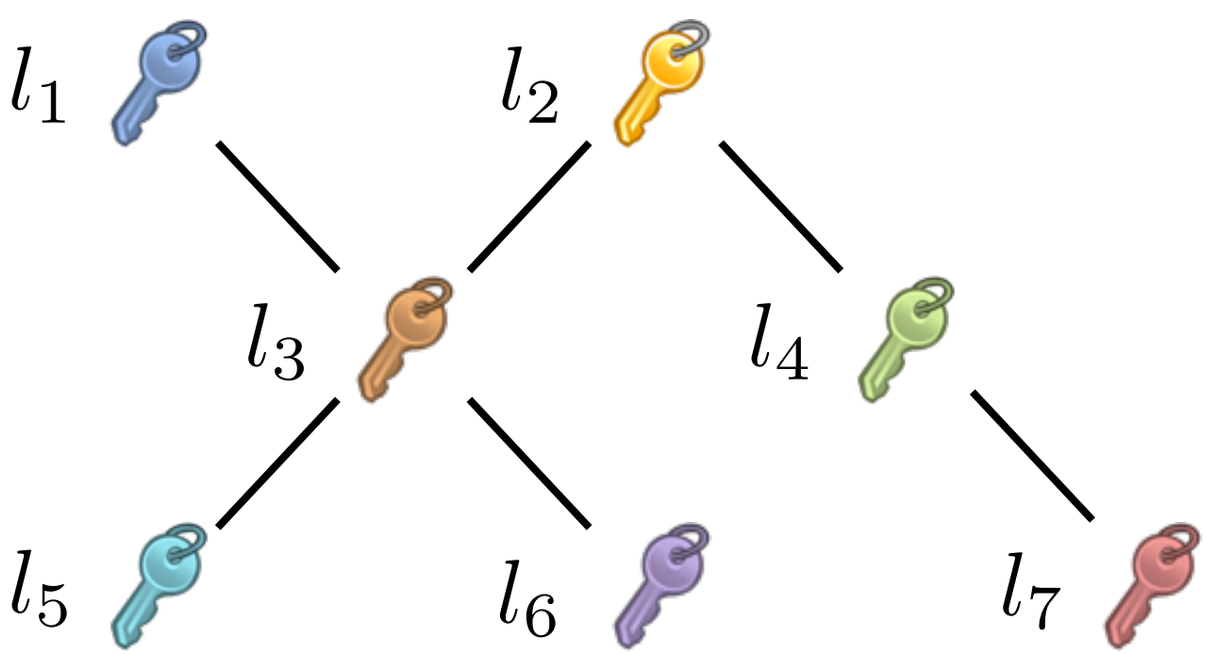
Description of the API

A **clock** supposed synchronized with a global clock

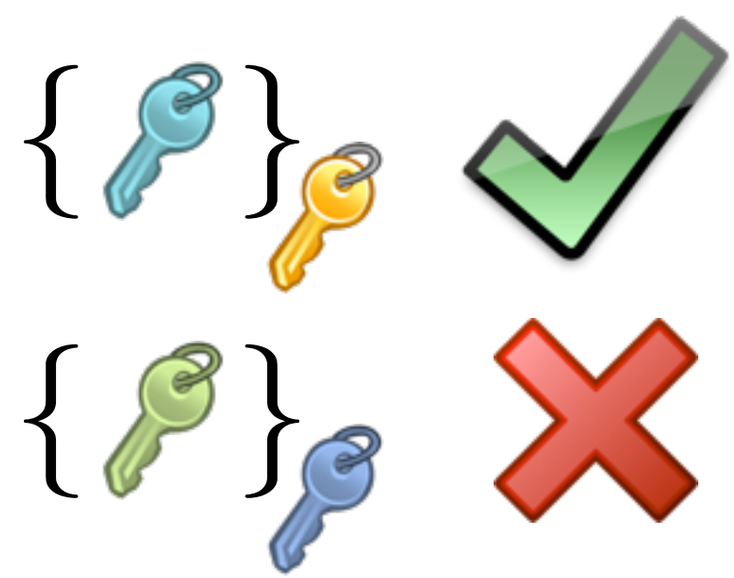
A **table** indexed by handles to store keys' information (level, validity, ...)

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

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



$l_5 < l_2$



$l_4 \not< l_1$

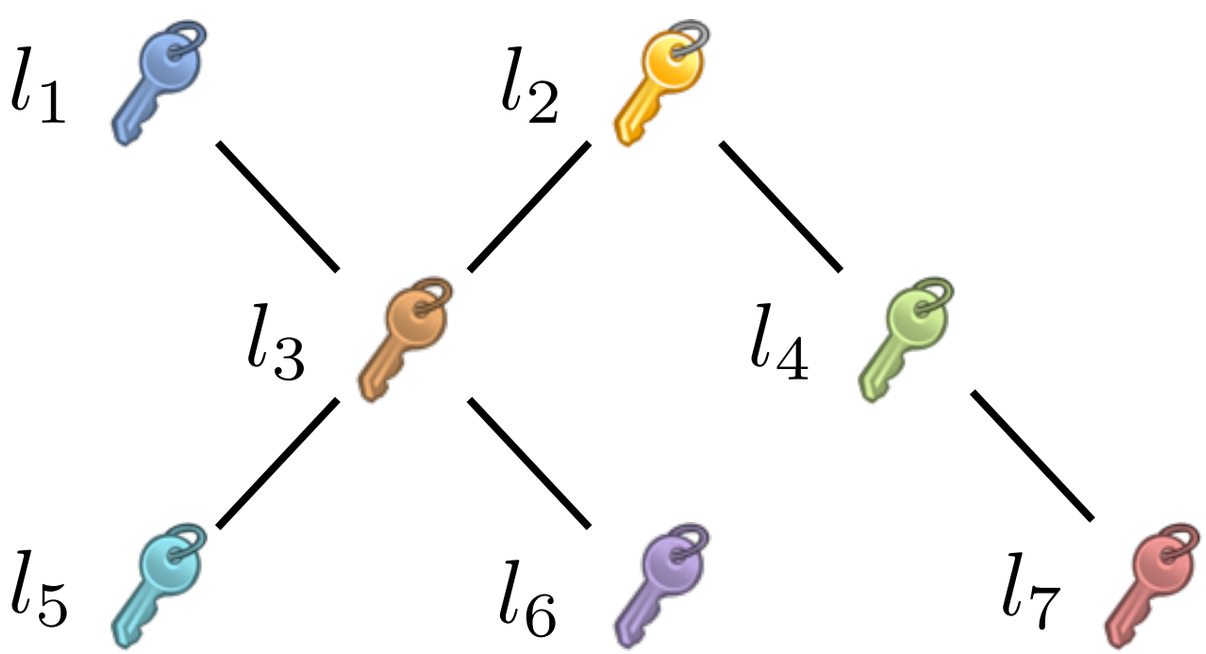
Description of the API

A **clock** supposed synchronized with a global clock

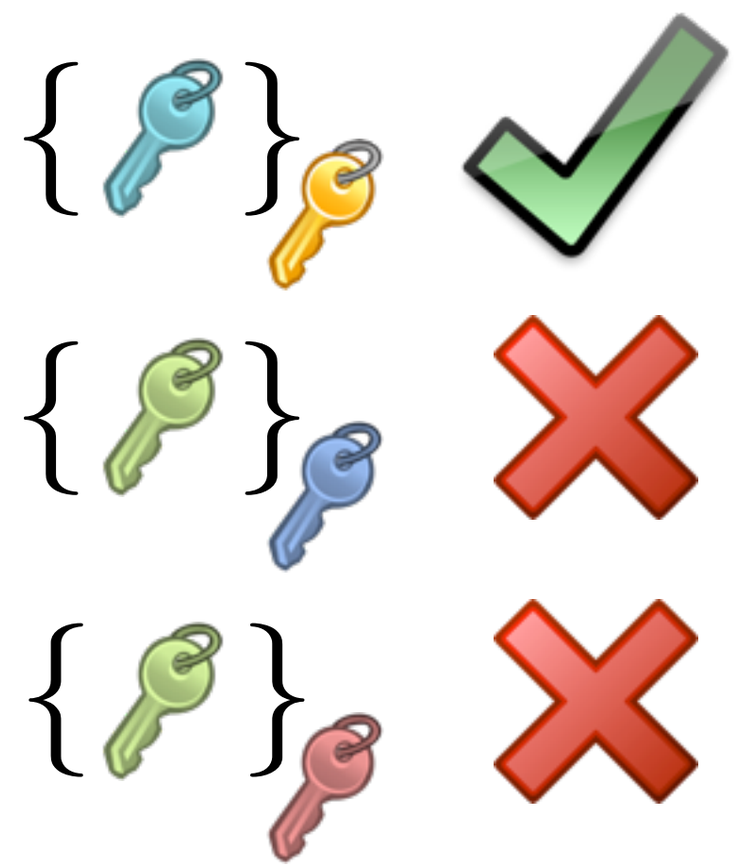
A **table** indexed by handles to store keys' information (level, validity, ...)

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

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



$l_5 < l_2$



$l_4 \not< l_1$

$l_4 \not< l_7$

User's Commands

A set of **basic commands** :

$\mathsf{generatePublic}(m)$

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

Store in a handle of the TRD, a generate nonce or key, the level and m.

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

User's Commands

A set of **basic commands** :

$\mathsf{generatePublic}(m)$

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

Store in a handle of the TRD, a generate nonce or key, the level and m.

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

$\mathsf{decrypt}(C, h)$

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

Decrypts C with the key stored in h and return a message or a handle.

User's Commands

A set of **basic commands** :

$\mathsf{generatePublic}(m)$

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

Store in a handle of the TRD, a generate nonce or key, the level and m.

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

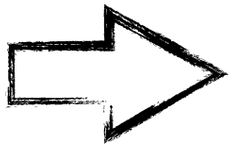
$\mathsf{decrypt}(C, h)$

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

Decrypts C with the key stored in h and return a message or a handle.

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

$X_i = h_i$ or $X_i = n_i$



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

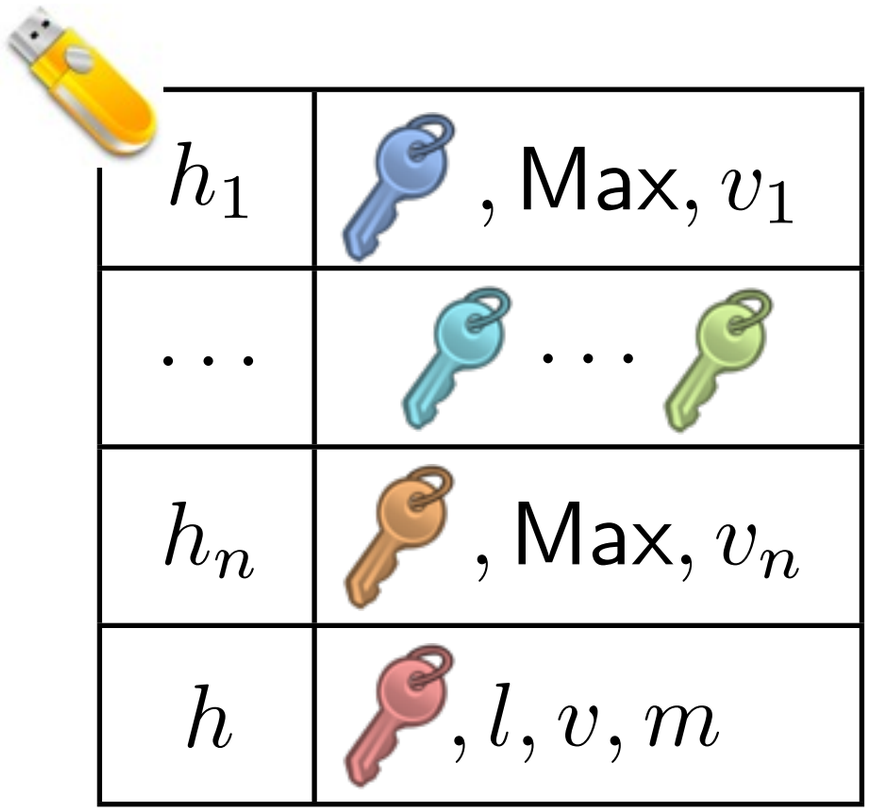
Lower Level Keys Management

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

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

Lower Level Keys Management

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



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

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 in $h_1, \ldots, h_n$.

2. Decryption of C.

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

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 in $h_1, \ldots, h_n$.

2. Decryption of C.

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

3. Tests on the new attributes l', v'.

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 in $h_1, \ldots, h_n$.

2. Decryption of C.

$C = \{$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. Updating the table with the new values.

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 in $h_1, \ldots, h_n$.

2. Decryption of C.

$C = \{$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. Updating the table with the new values.

Revocation Keys Management

Since revocation keys are sensitive, we allow the admin to update them...

$\{\text{updateMax}, \text{🔑}, \text{🔑}, v\}_{\text{🔑}\ldots\text{🔑}}$

$\ldots$

$\{\text{updateMax}, \text{🔑}, \text{🔑}, v'\}_{\text{🔑}\ldots\text{🔑}}$

Revocation Keys Management

Since revocation keys are sensitive, we allow the admin to update them...

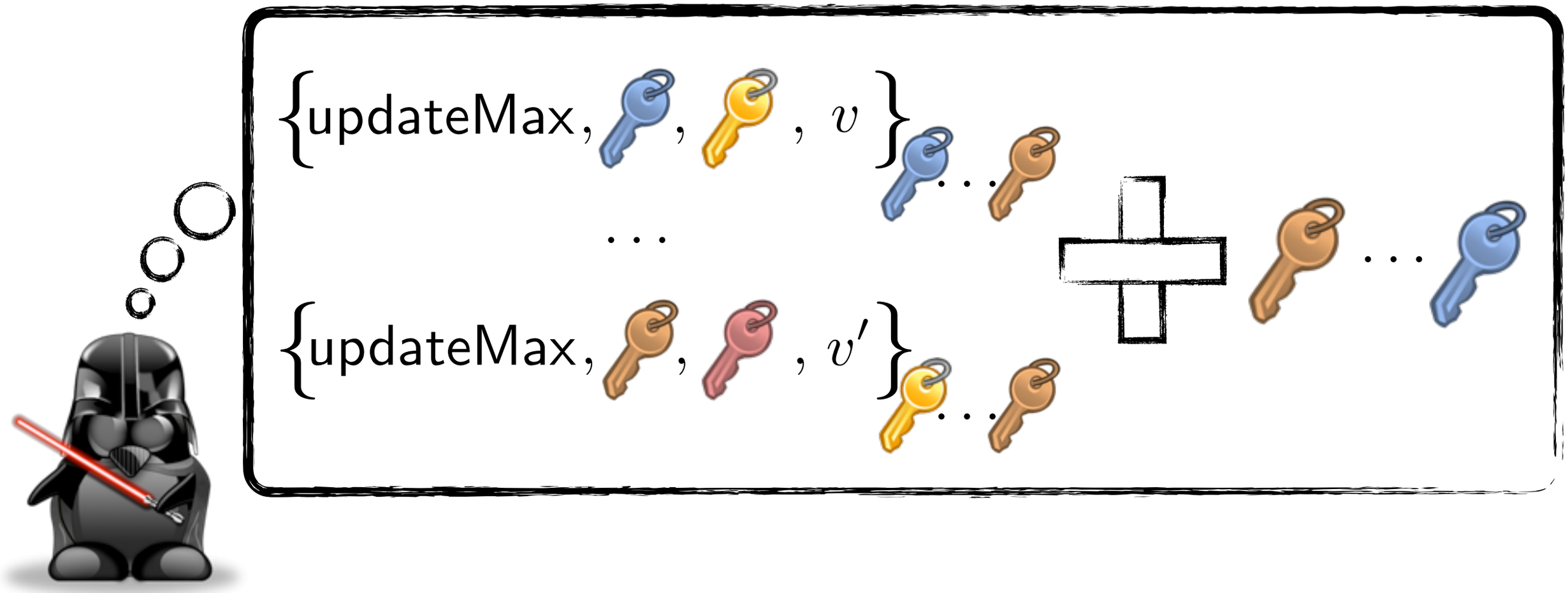
$\{\text{updateMax}, \text{🔑}, \text{🔑}, v\}_{\text{🔑}\ldots\text{🔑}}$

$\ldots$

$\{\text{updateMax}, \text{🔑}, \text{🔑}, v'\}_{\text{🔑}\ldots\text{🔑}}$

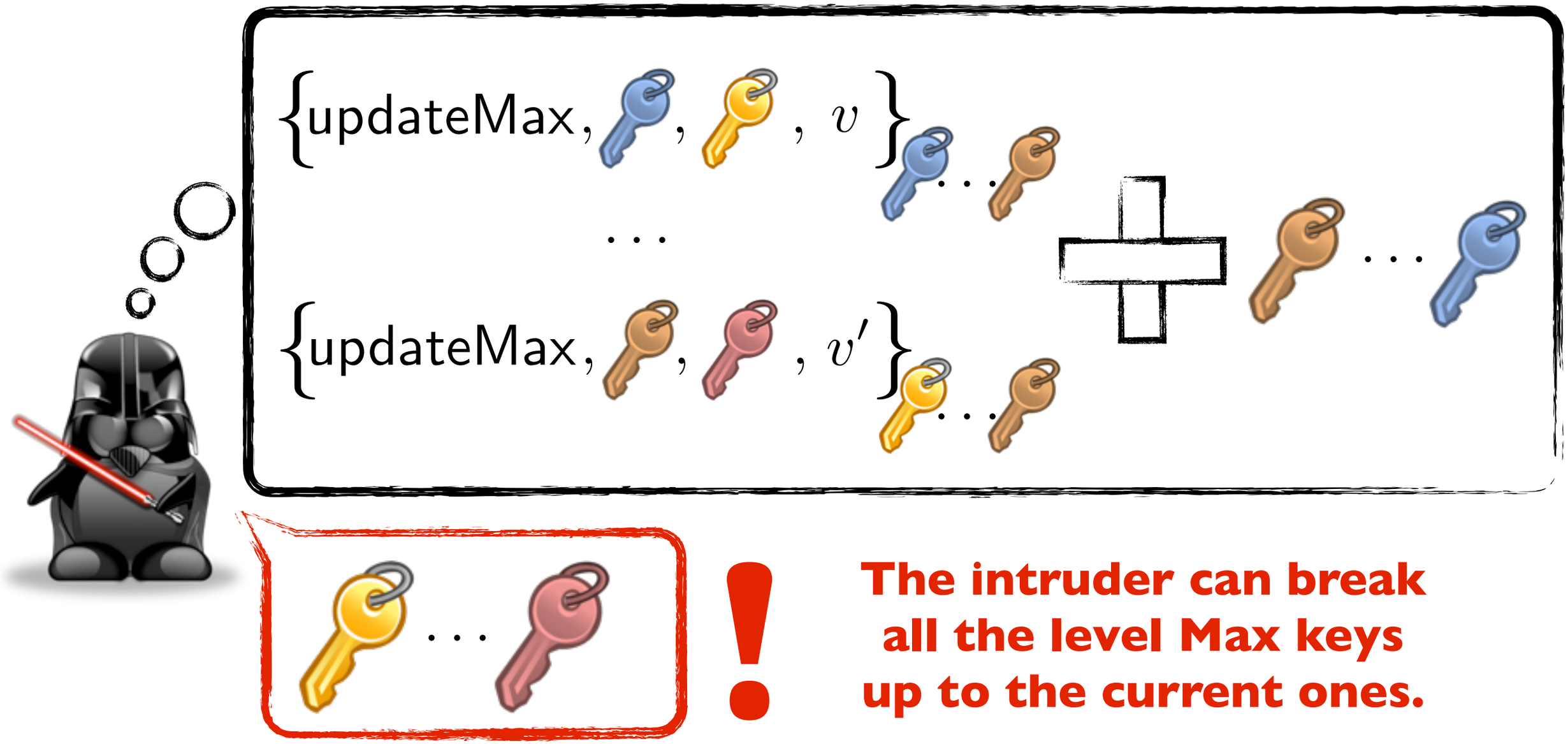
Revocation Keys Management

Since revocation keys are sensitive, we allow the admin to update them...



Revocation Keys Management

Since revocation keys are sensitive, we allow the admin to update them...



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 :

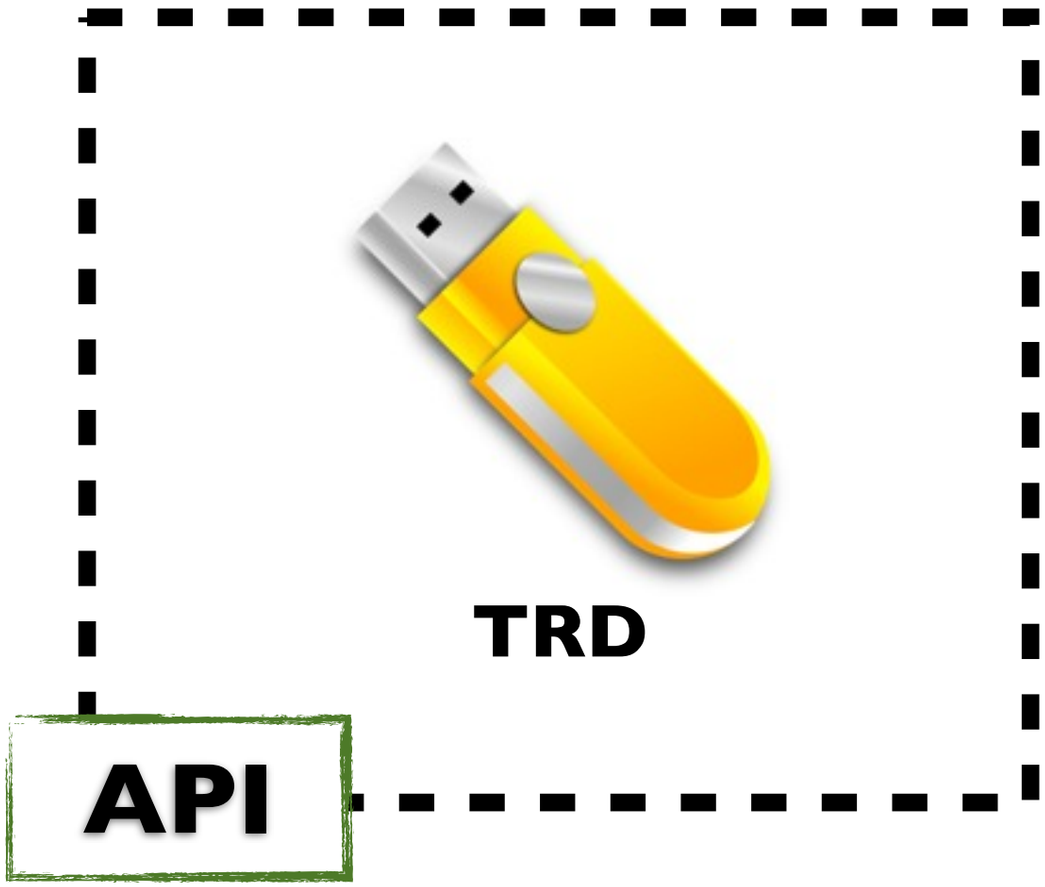
Level Max commands are sent over a secure channel.

This can be achieved by several means :

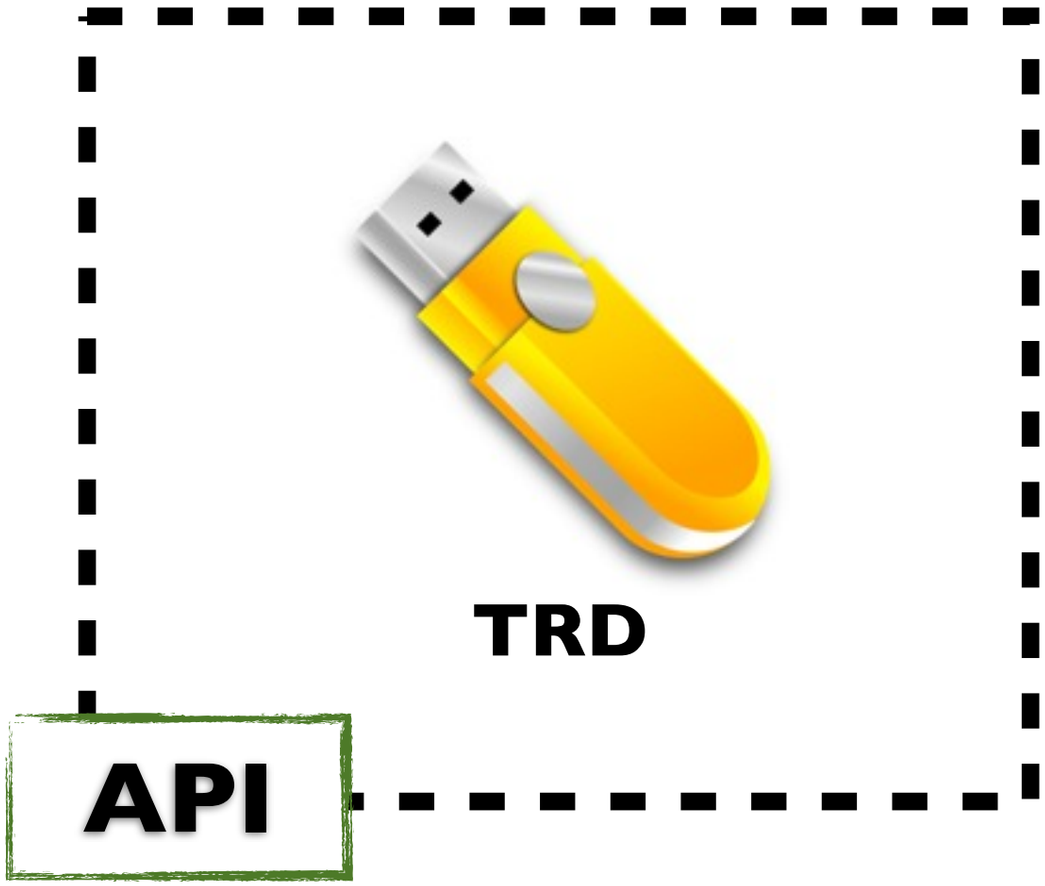
- The administrator has a physical access to the TRD that needs to be updated,

- 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.

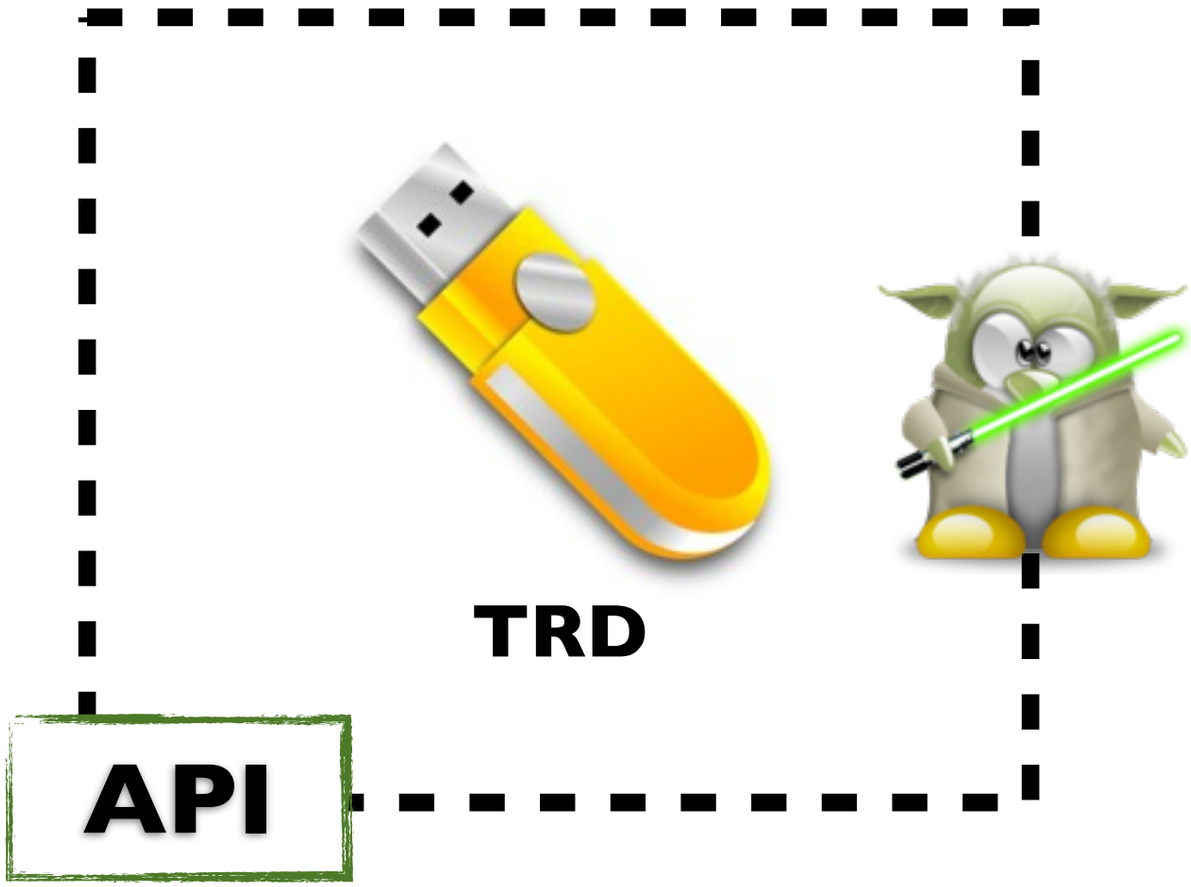
And now, what about Security ?



And now, what about Security ?



And now, what about Security ?



Abstraction

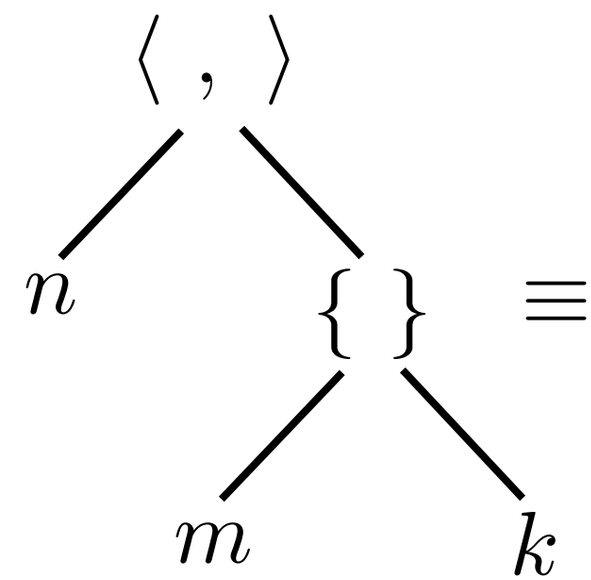
Messages are represented by **terms**

Nonces, keys :

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

Primitives :

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

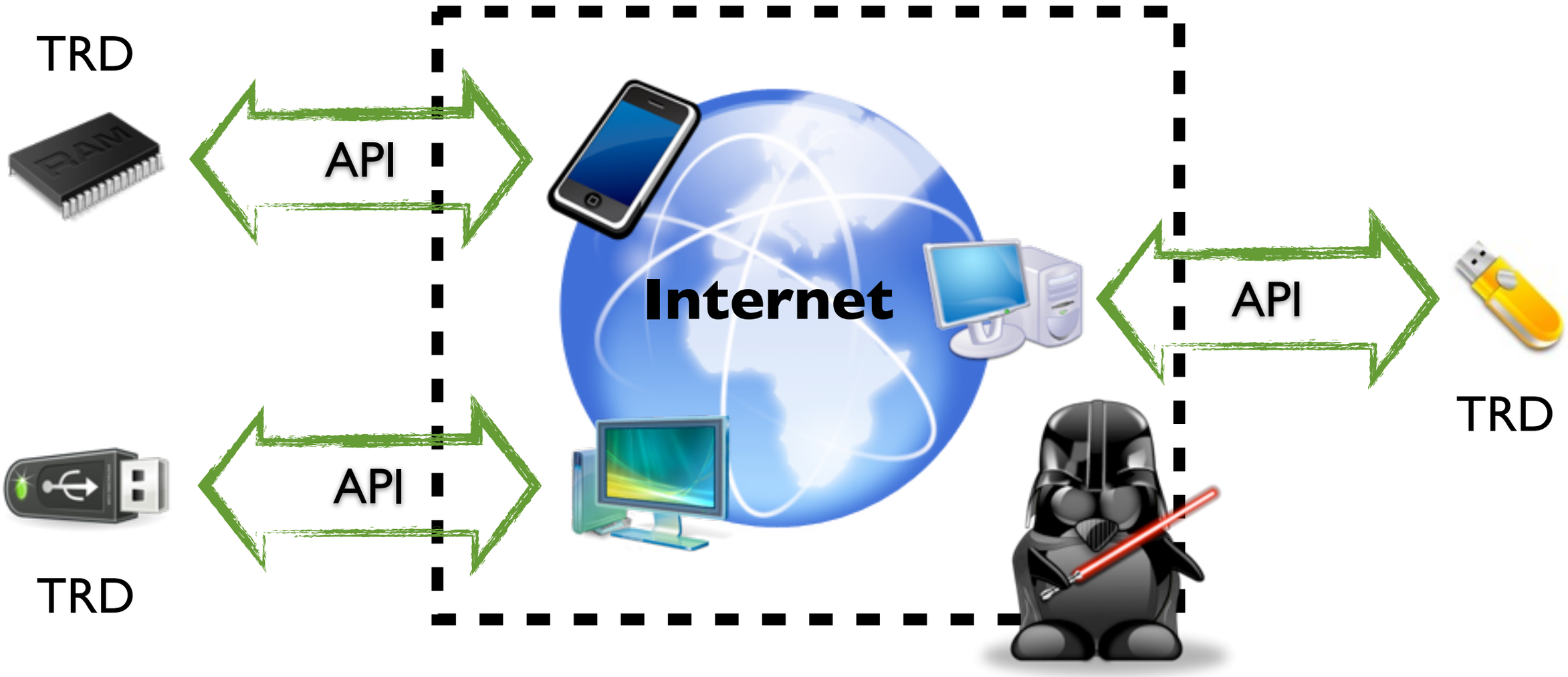


$$\equiv \langle n, \{m\}_k \rangle$$

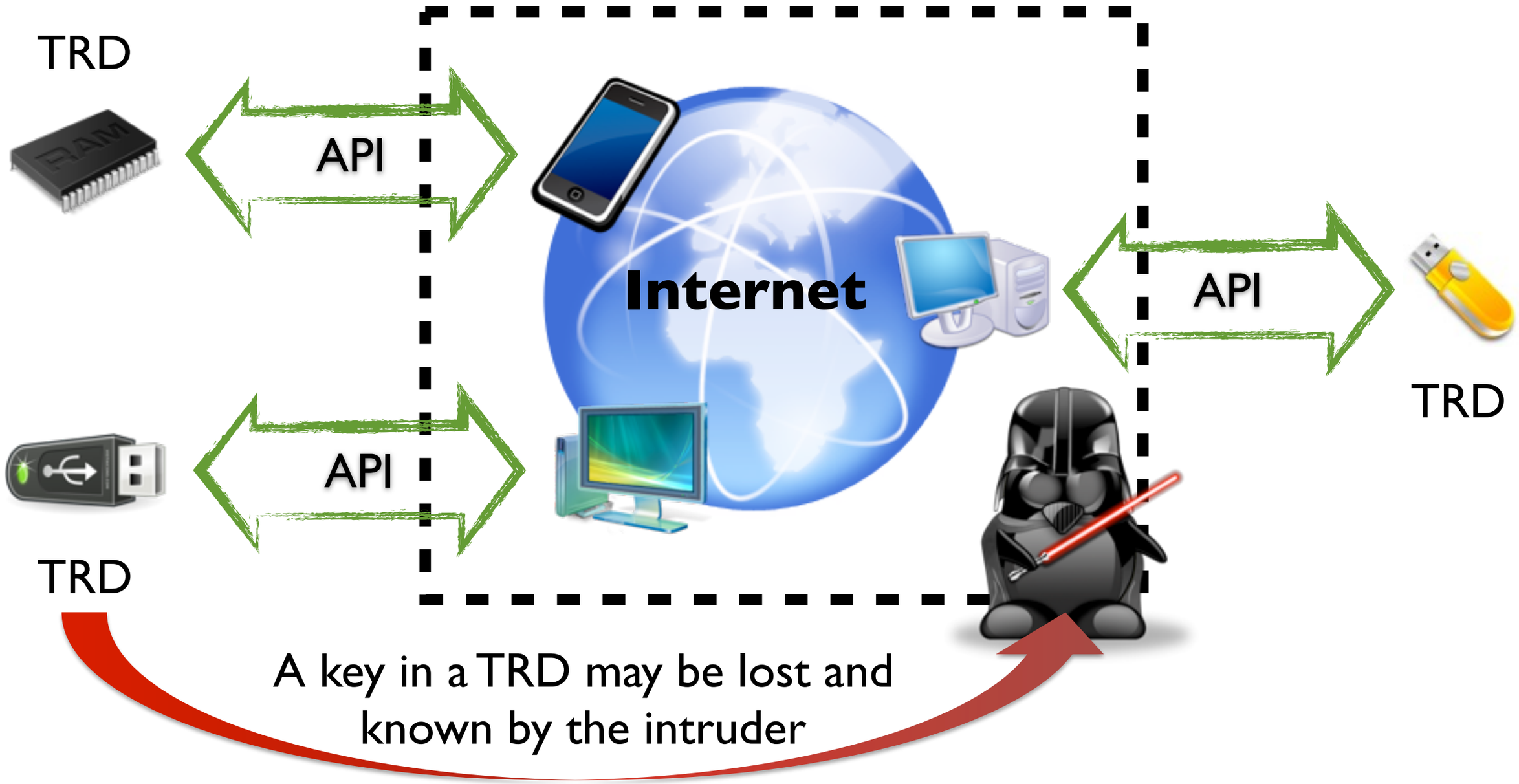
Modeling deduction rules :

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

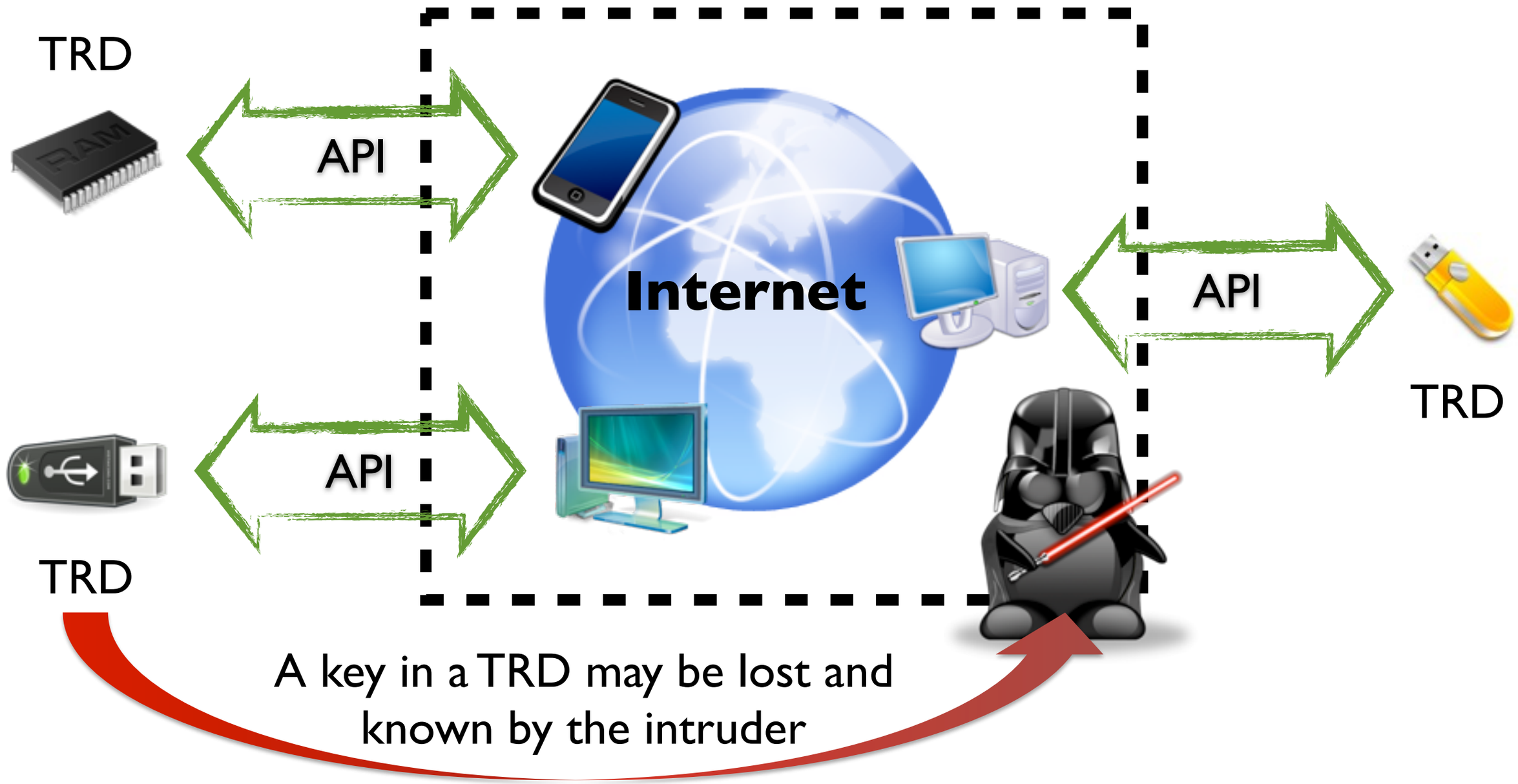
Knowledge of the Intruder



Knowledge of the Intruder

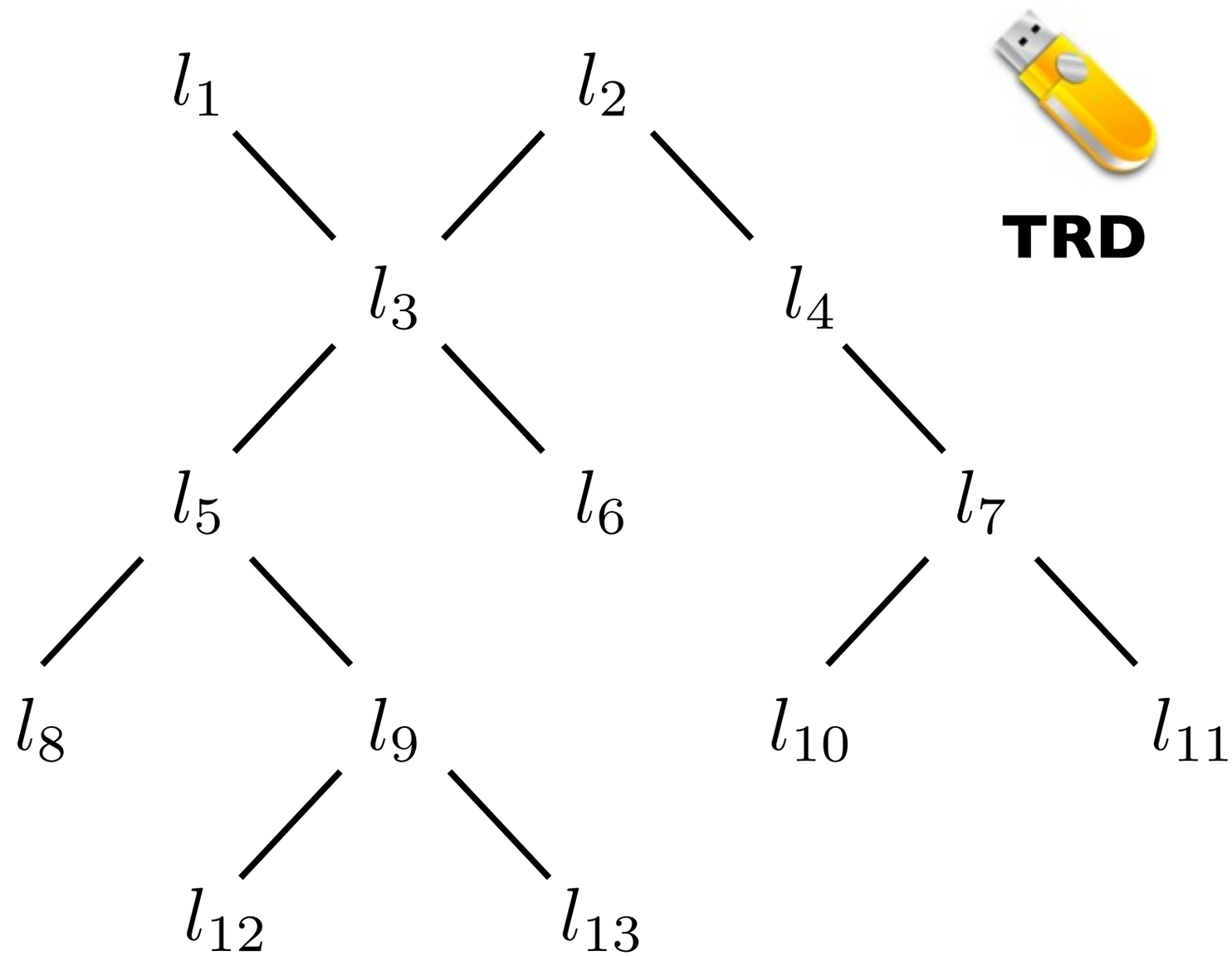


Knowledge of the Intruder

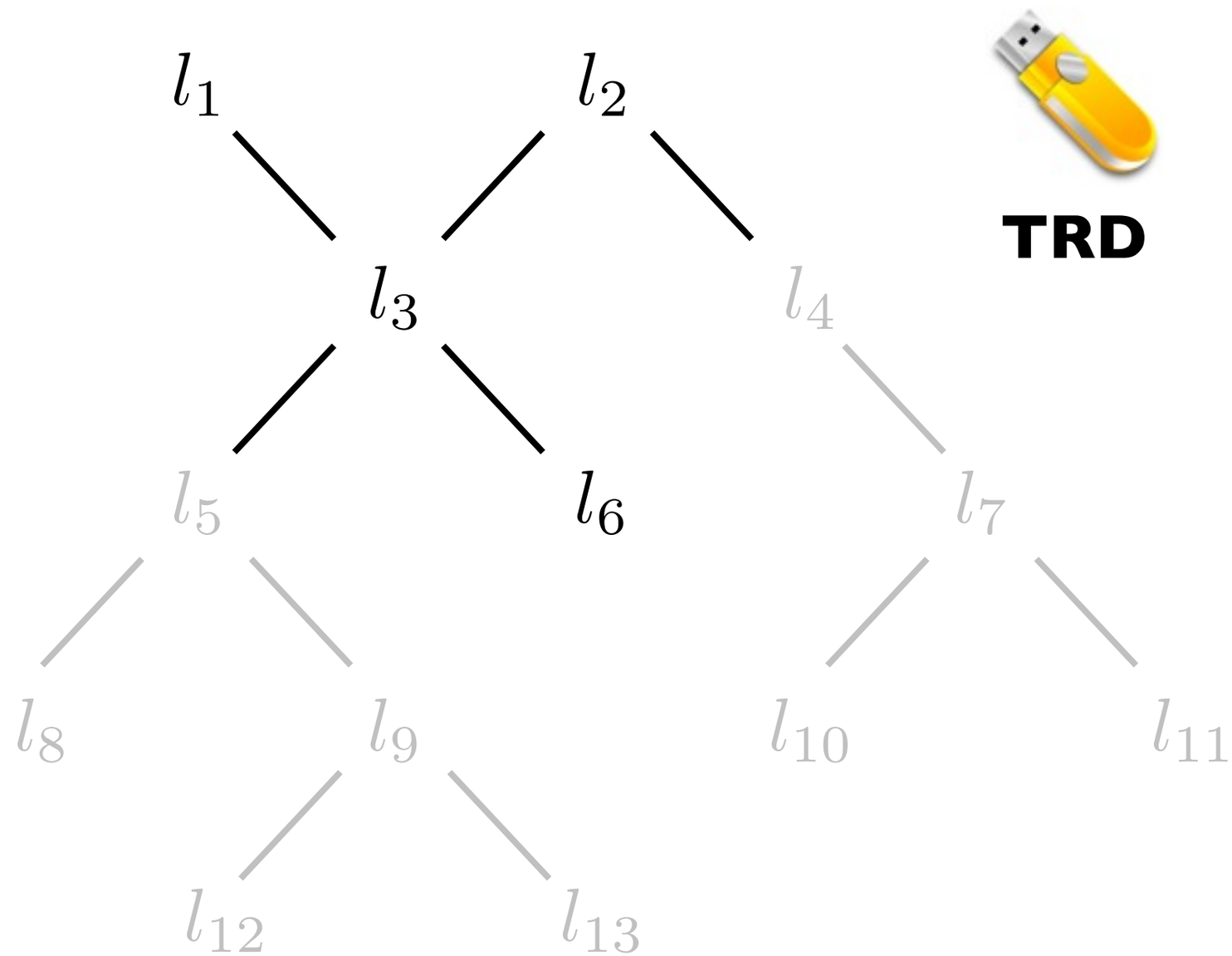


Hypothesis : Only a total of $N_{\text{Max}} - 1$ different « current » level Max keys for one TRD can be lost.

What about lost levels ?

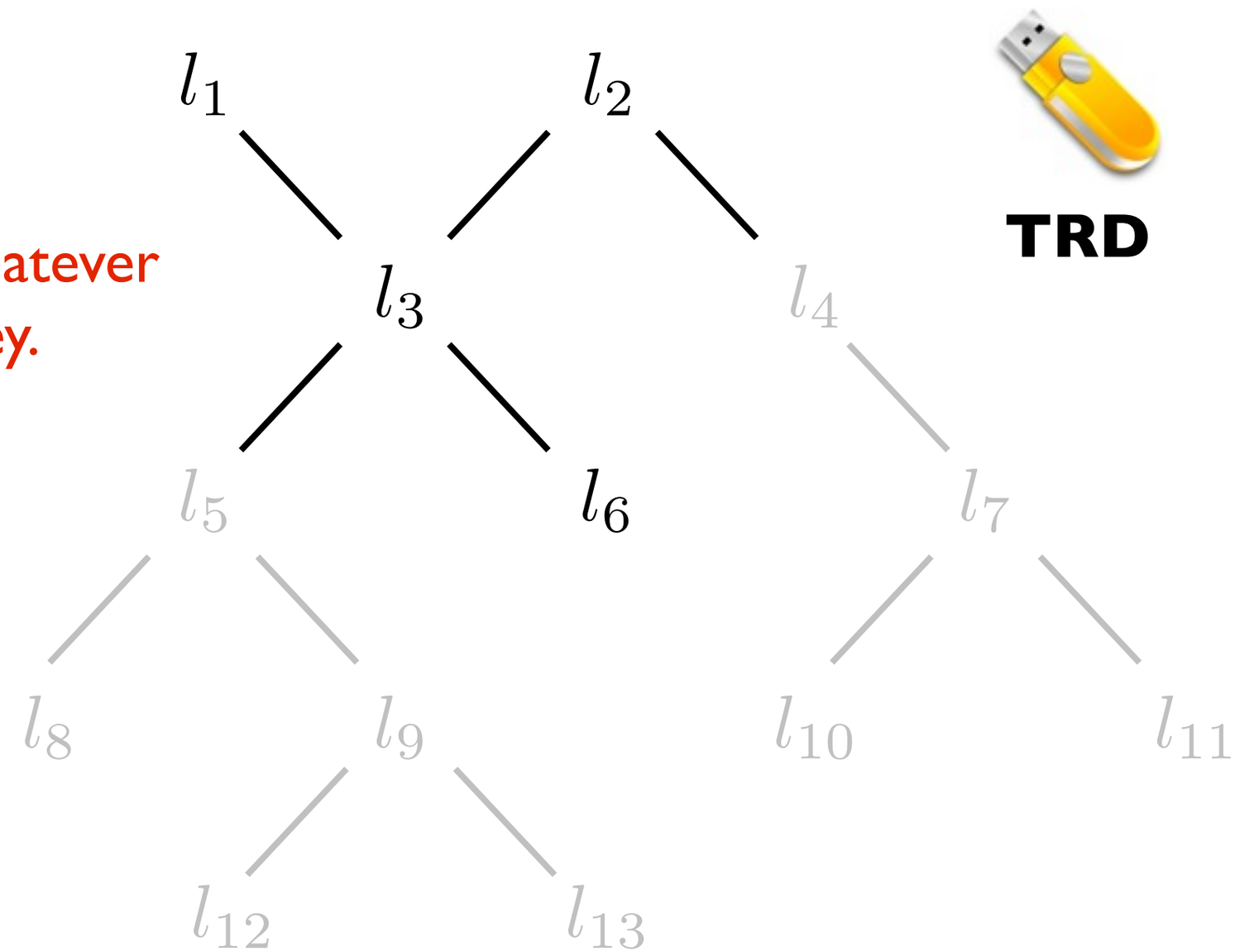


What about lost levels ?



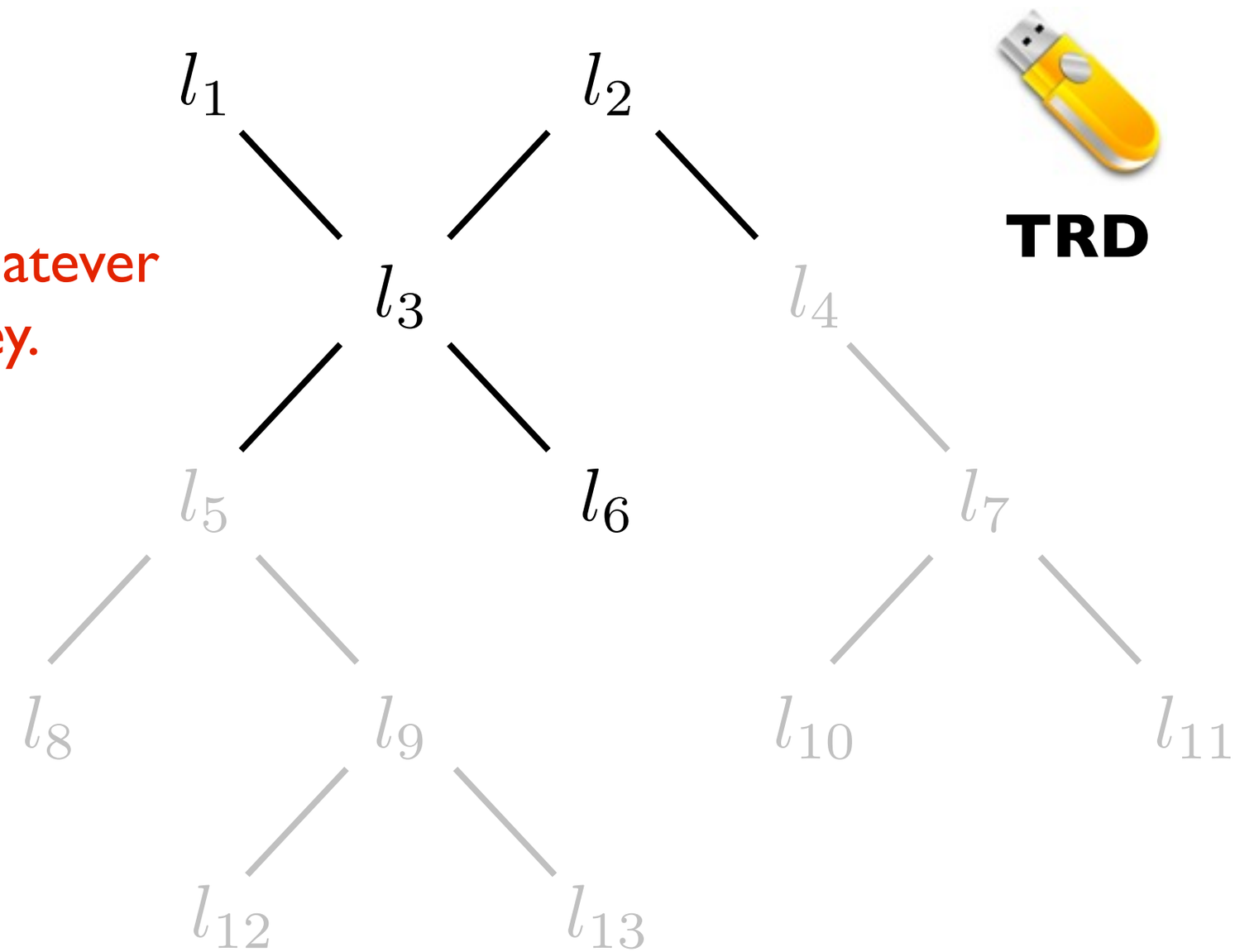
What about lost levels ?

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.



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

Ex : Obtaining $\{\langle$  $, l_9, v, m\rangle\}$  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

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

- A **valid expiration date**,
- They're **not « under a lost »**.

Self Repair Property

«It's just a flesh wound !»

Theorem (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 $l_i < l$.

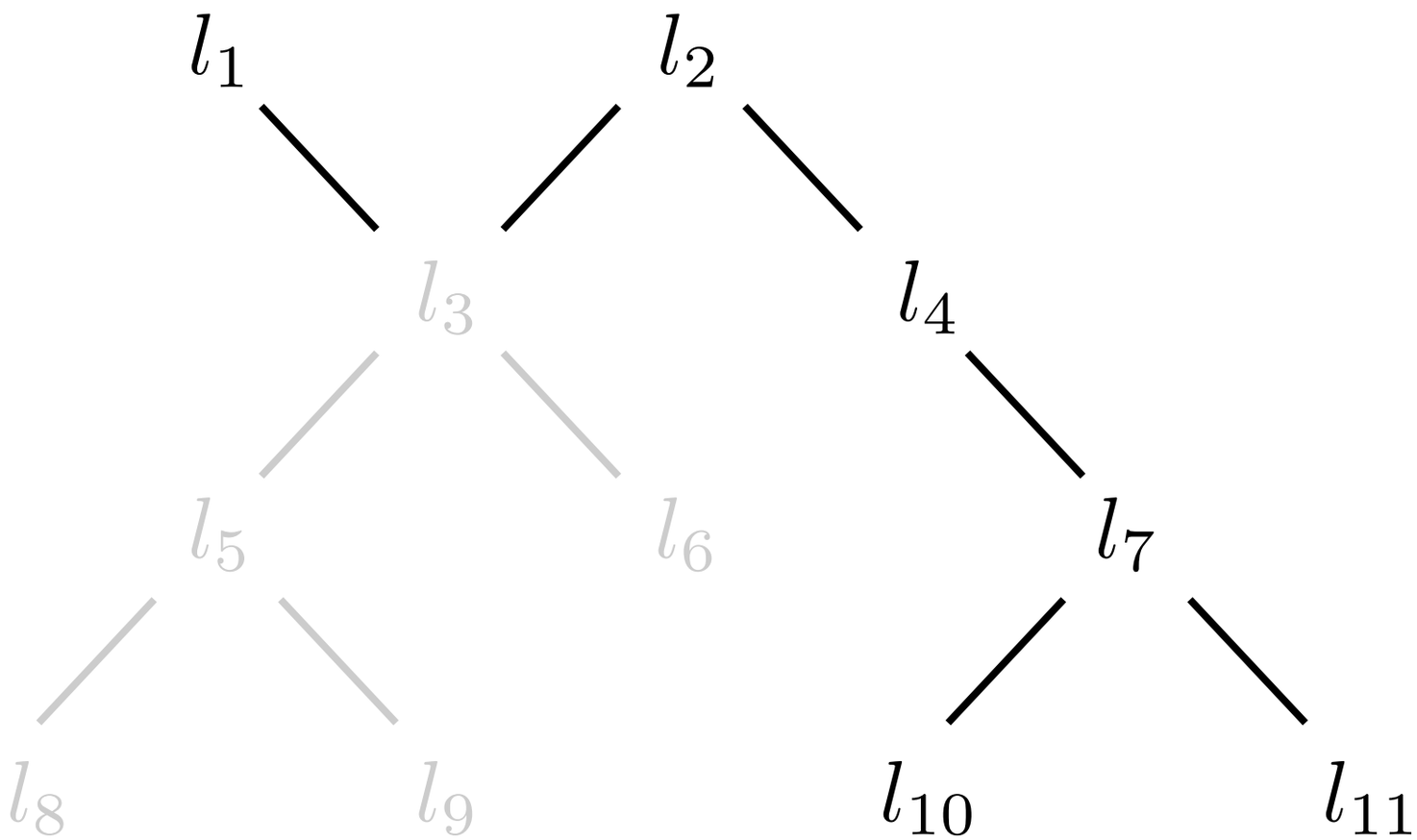
Self Repair Property

«It's just a flesh wound !»

Theorem (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 $l_i < l$.



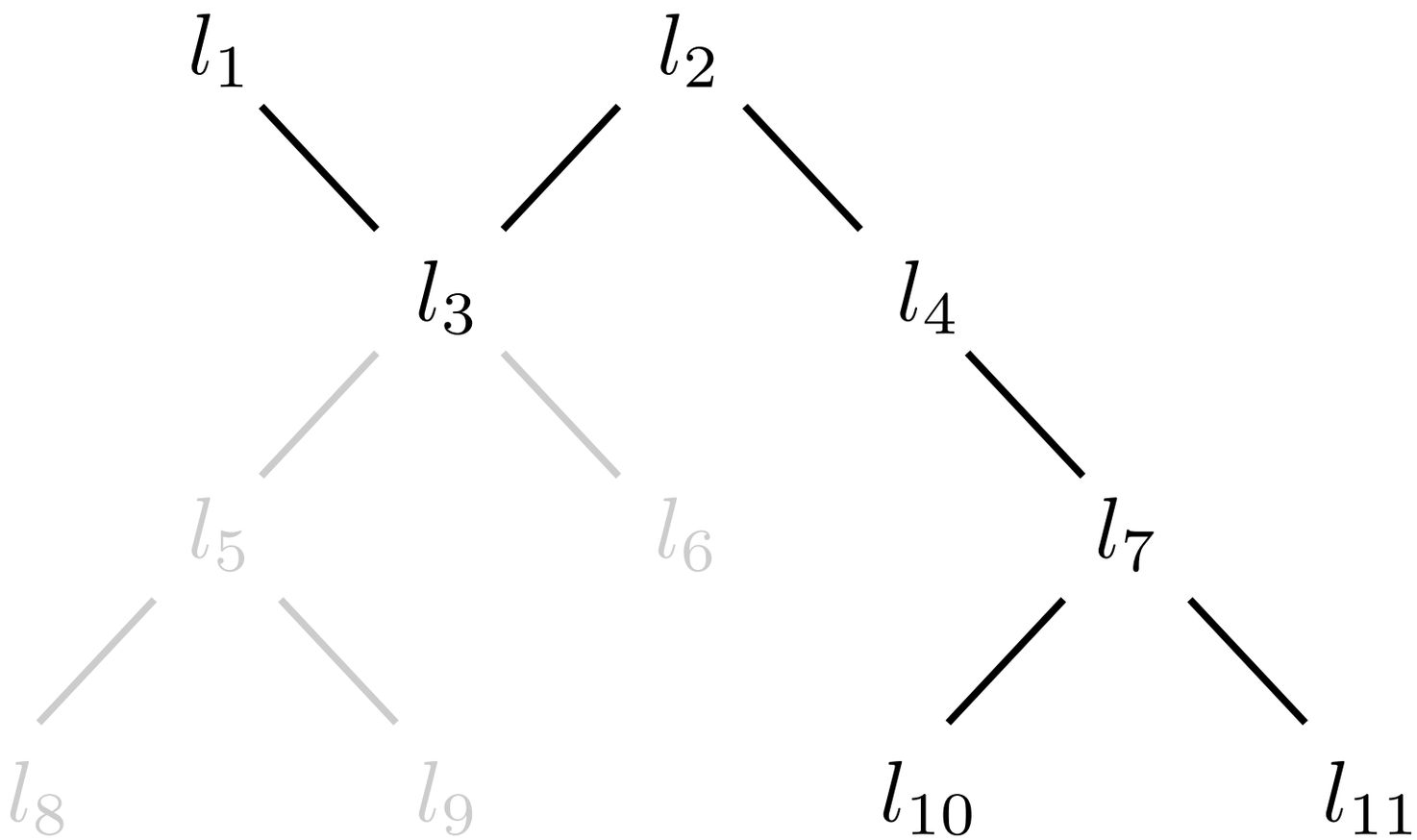
Self Repair Property

«It's just a flesh wound !»

Theorem (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 $l_i < l$.



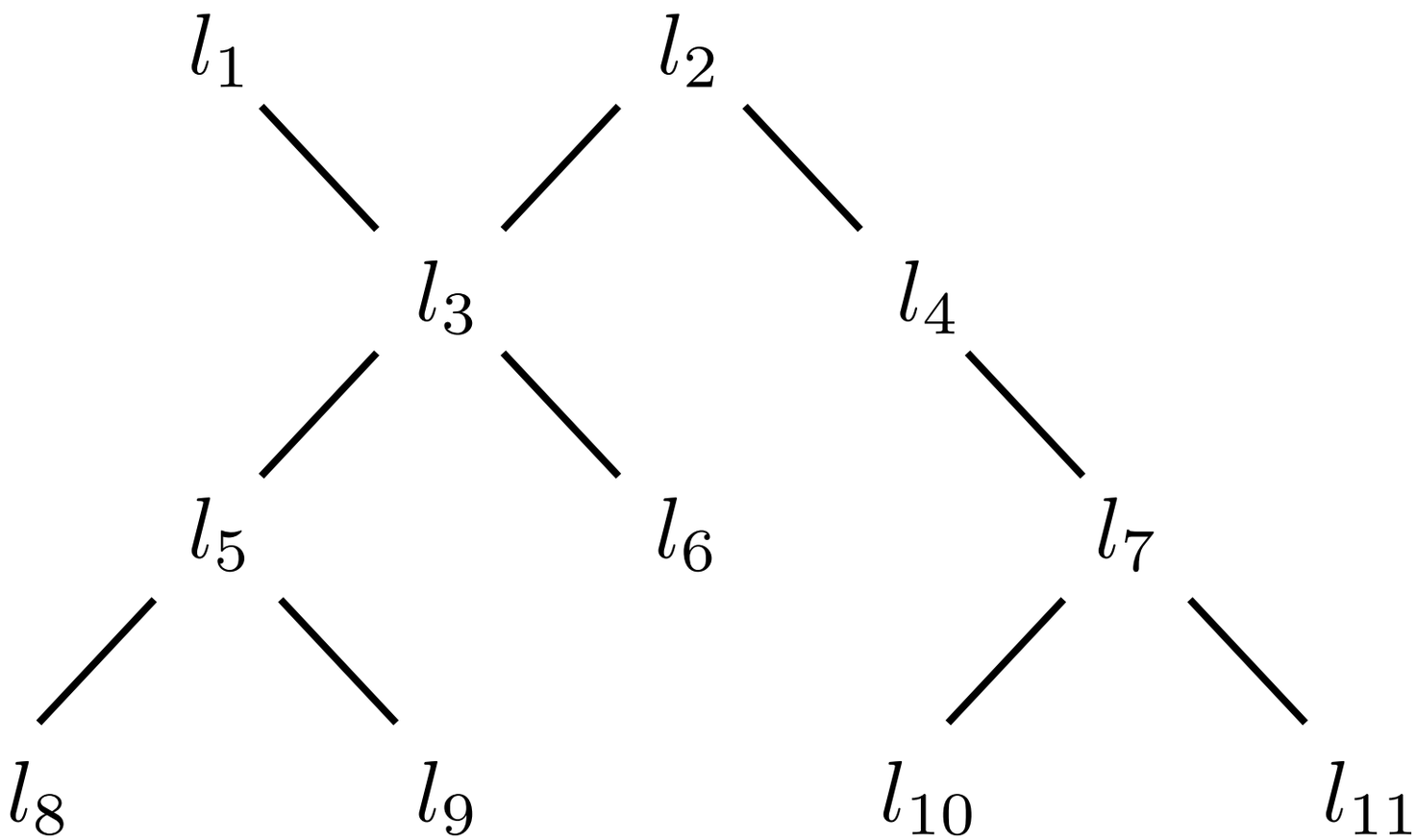
Self Repair Property

«It's just a flesh wound !»

Theorem (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 $l_i < l$.



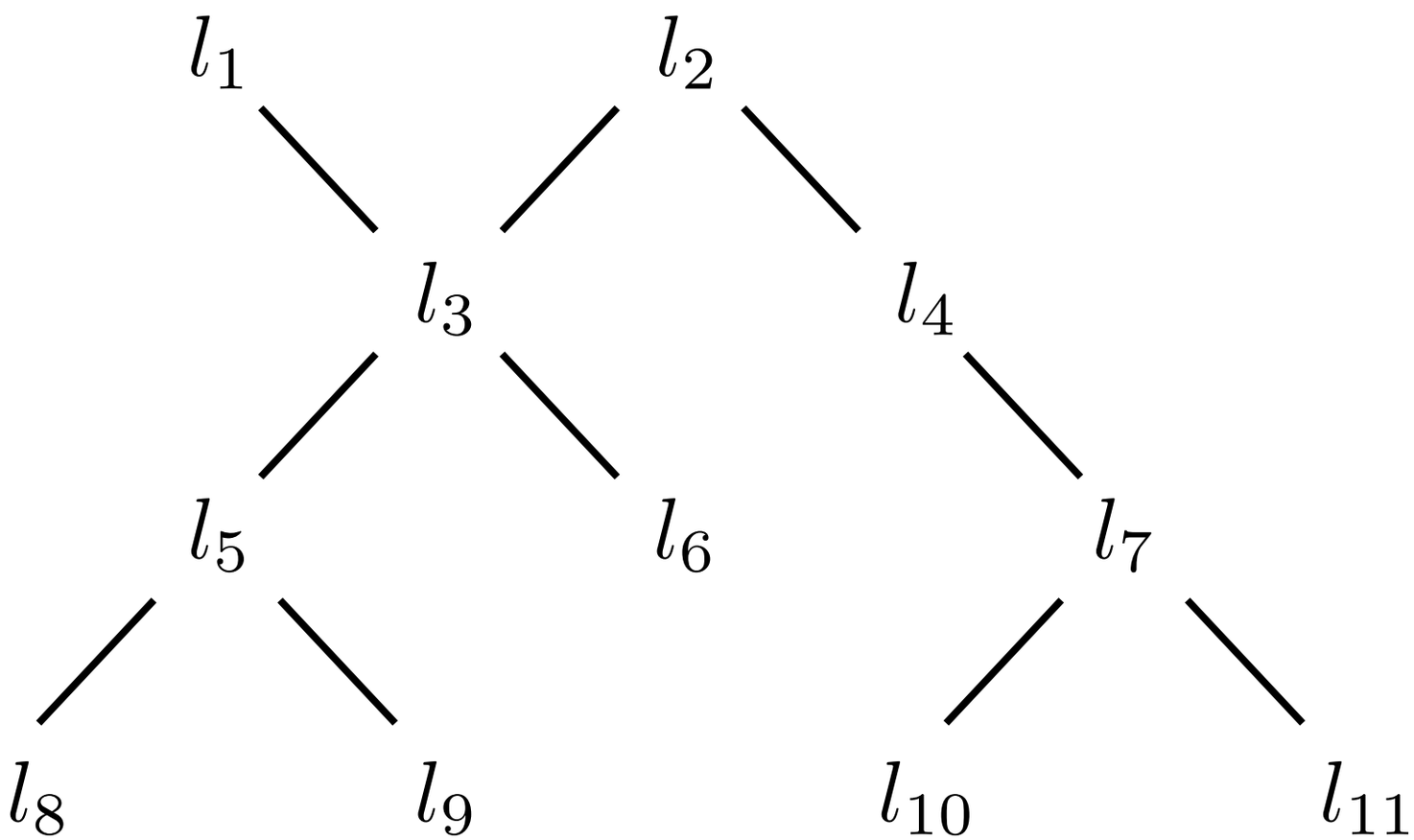
Self Repair Property

«It's just a flesh wound !»

Theorem (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 you **do not lose** a level higher than the one you «try» to repair.

Blacklist Opportunity

«For those who are in a hurry...»

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

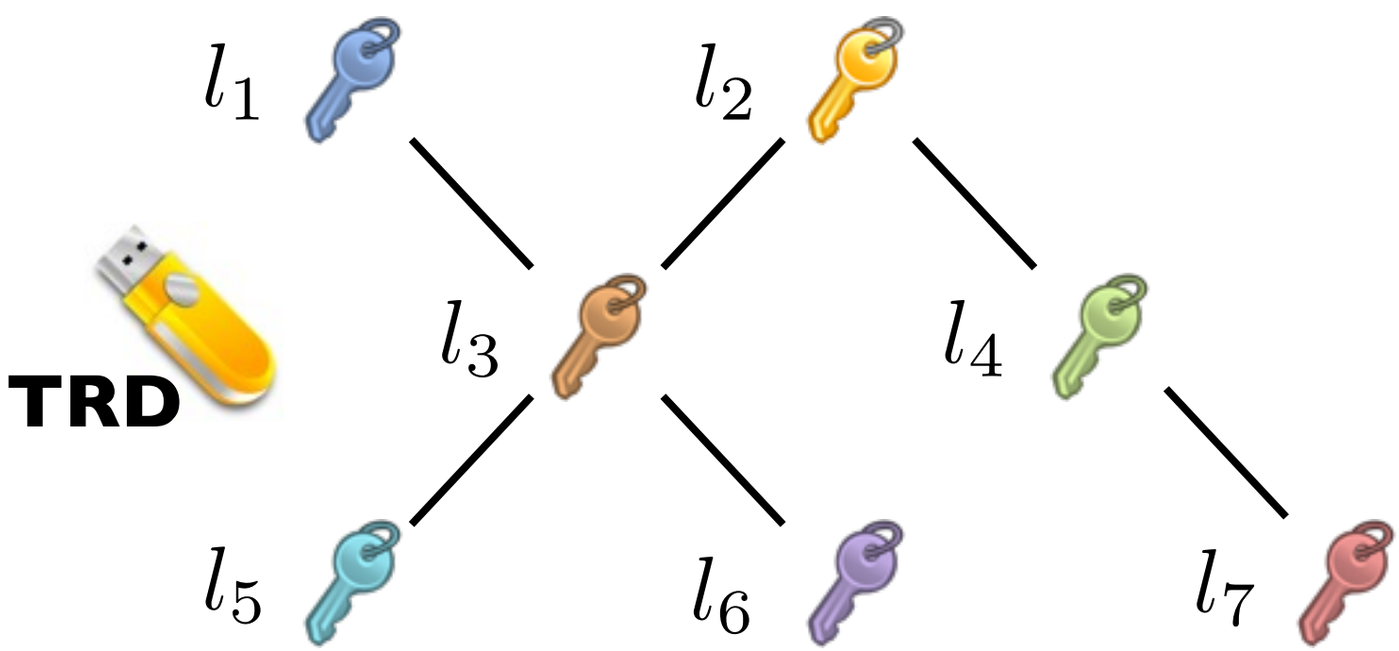
$\mathsf{Ex}: C = \{\langle \mathsf{blacklist}, \langle l_3, t_3 \rangle \rangle\}$ 

Blacklist Opportunity

«For those who are in a hurry...»

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

Ex : $C = \{\langle \mathsf{blacklist}, \langle l_3, t_3 \rangle \rangle\}$

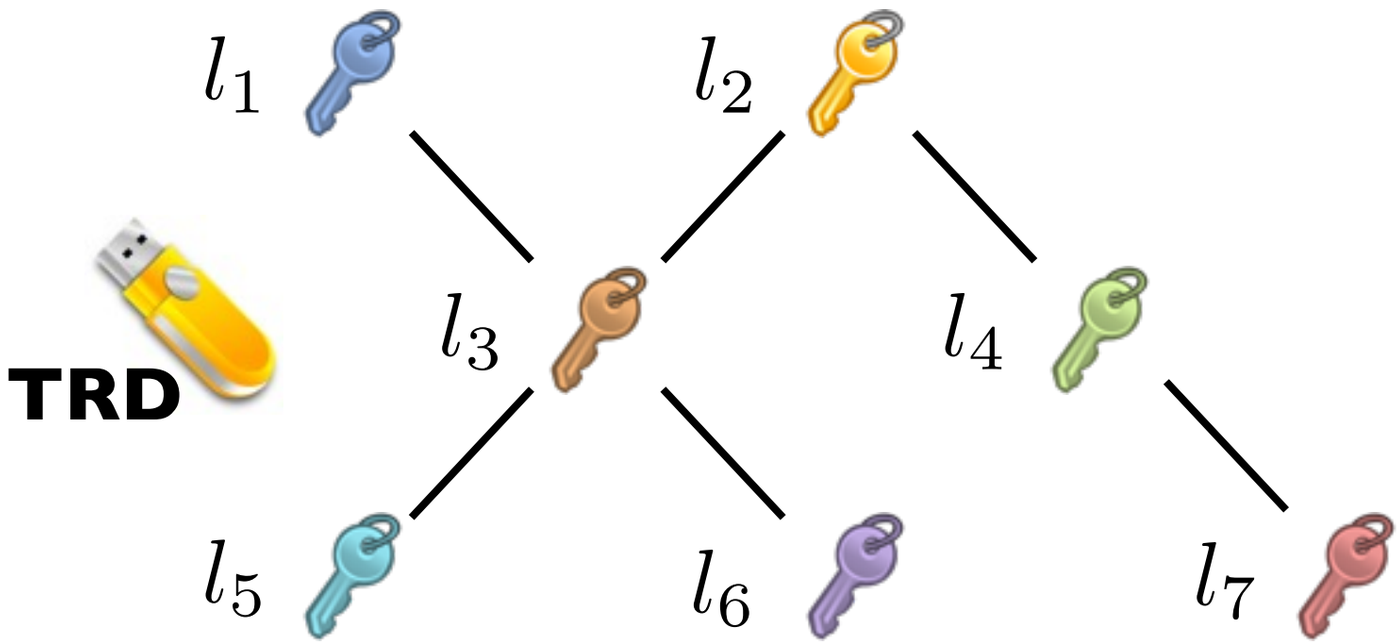


Blacklist Opportunity

«For those who are in a hurry...»

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

$\mathsf{Ex}: C = \{\langle \mathsf{blacklist}, \langle l_3, t_3 \rangle \rangle\}$



$(l_3, t_3) \longrightarrow$

Blacklist Opportunity

«For those who are in a hurry...»

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

Ex : $C = \{\langle \mathsf{blacklist}, \langle l_3, t_3 \rangle \rangle\}$

l_1

l_2

TRD

l_4

$(l_3, t_3) \longrightarrow$

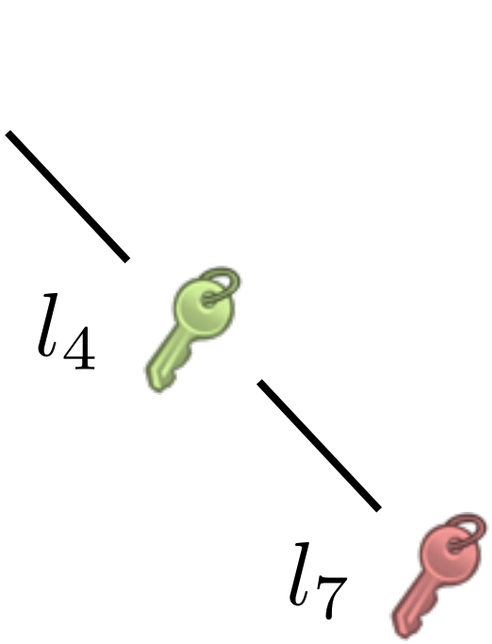
l_7

Blacklist Opportunity

«For those who are in a hurry...»

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

Ex : $C = \{\langle \mathsf{blacklist}, \langle l_3, t_3 \rangle \rangle\}$



$(l_3, t_3) \longrightarrow$

Theorem (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, **immediatly**, all keys are secret except those under levels $l_1, \ldots, l_n$ such that $l_i < l$, for the TRD.

Concluding by Future Work

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

- **Extending** our API to **asymmetric encryption**,

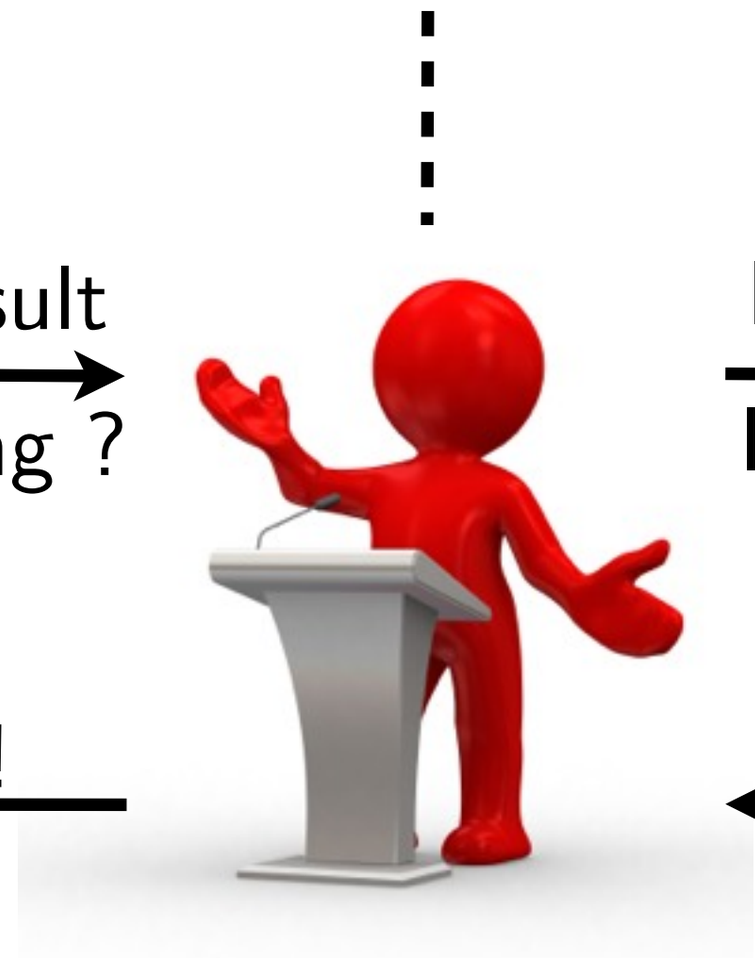
- **Adapting** the result taking account of possible **clock drift**, or replacing the clock by a sort of nonce based freshness test,

- **Implementing** the API in order to carry out some performance tests.

Thank you for your attention !

IsTheResult

Interesting ?



IsTheResult

Interesting ?

Sure !

Perhaps

Public
(Host Machines)

Speaker
(Security API)

Truth
(Trusted device)