# A Survey on Protocol Verification and Security APIs

Weekly Seminar MAIS Darmstadt
(joint work with Steve Kremer and Graham Steel)

Robert Künnemann

# Outline

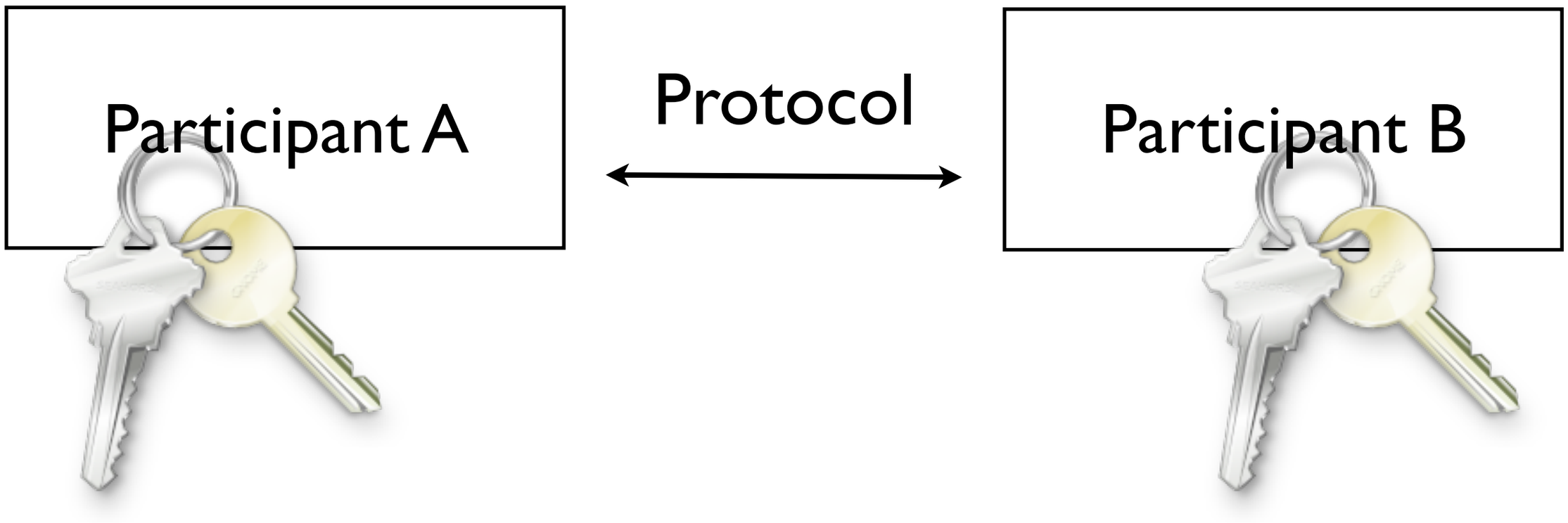
Nochmal durchsprechen

- Security APIs and the analysis of stateful protocols
- Horn clause resolution
- stateful analysis using multiset rewriting
- case study (MSR): the Yubikey protocol
- a translation from a high-level description to MSR and why it is needed

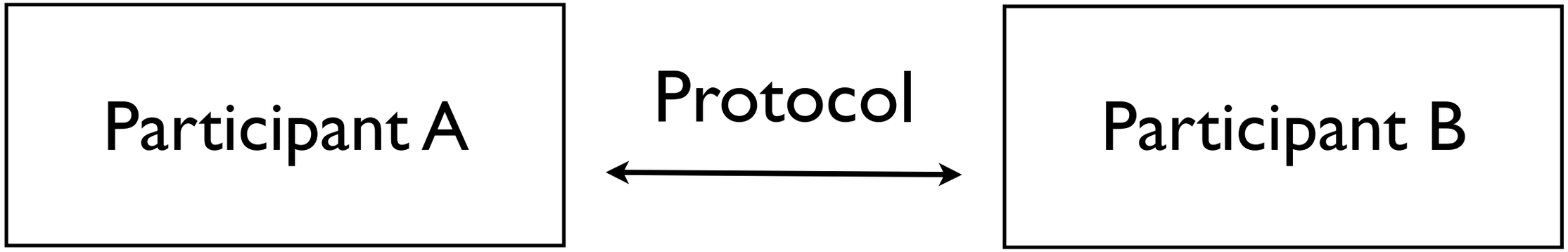
# Introduction

(Security APIs and the analysis of stateful protocols)

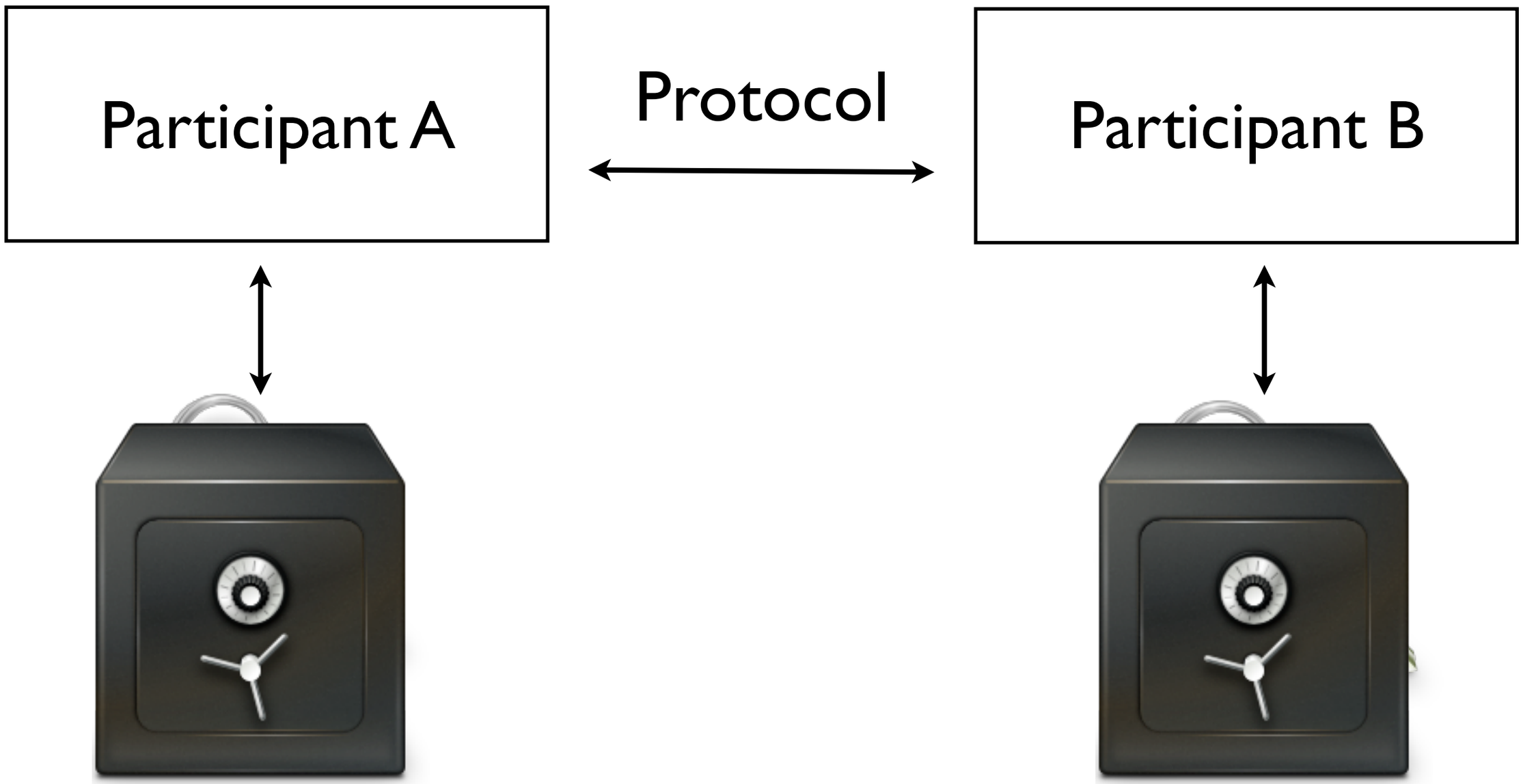
# Where do we want to go?



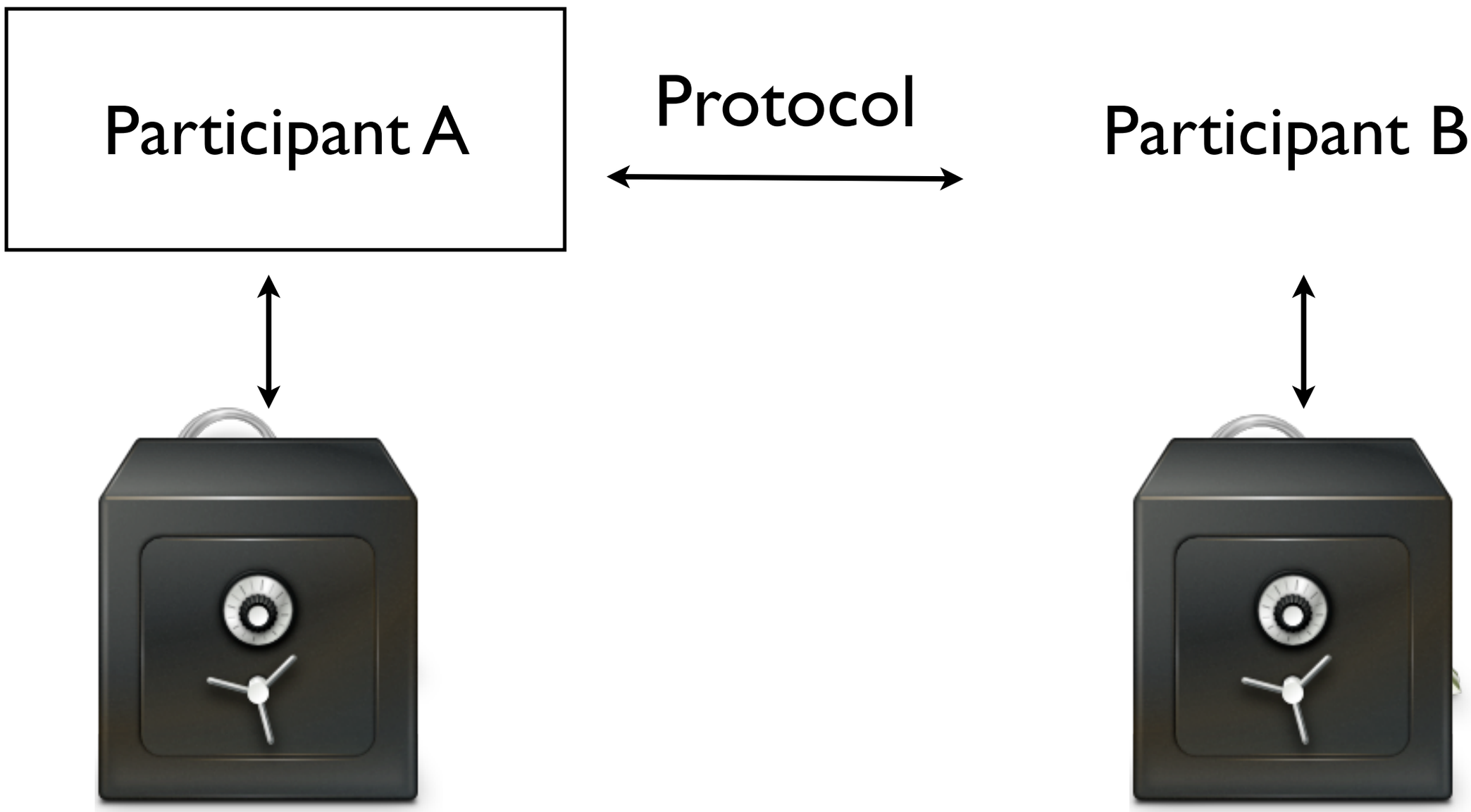
# Where do we want to go?



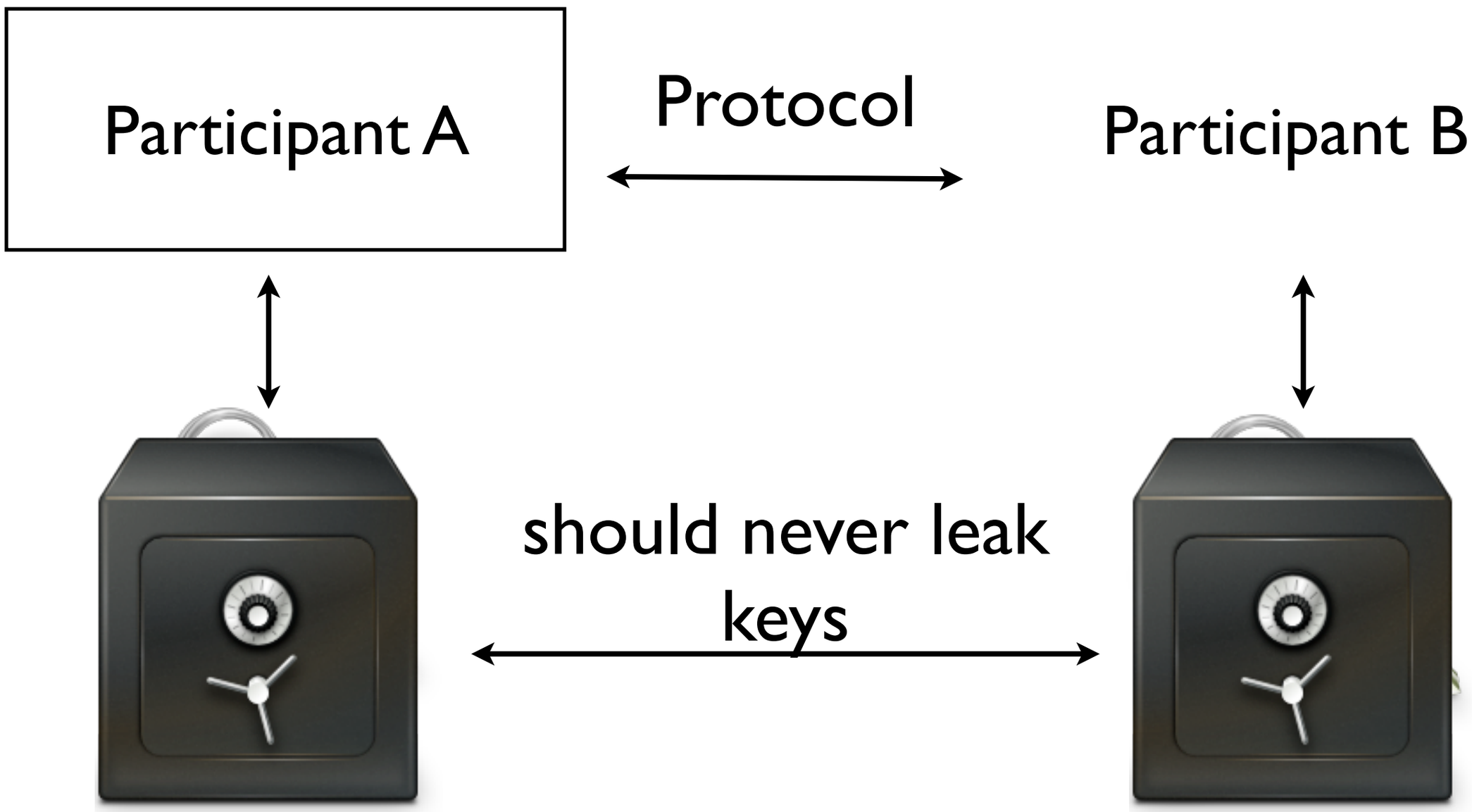

# Where do we want to go?



# Where do we want to go?

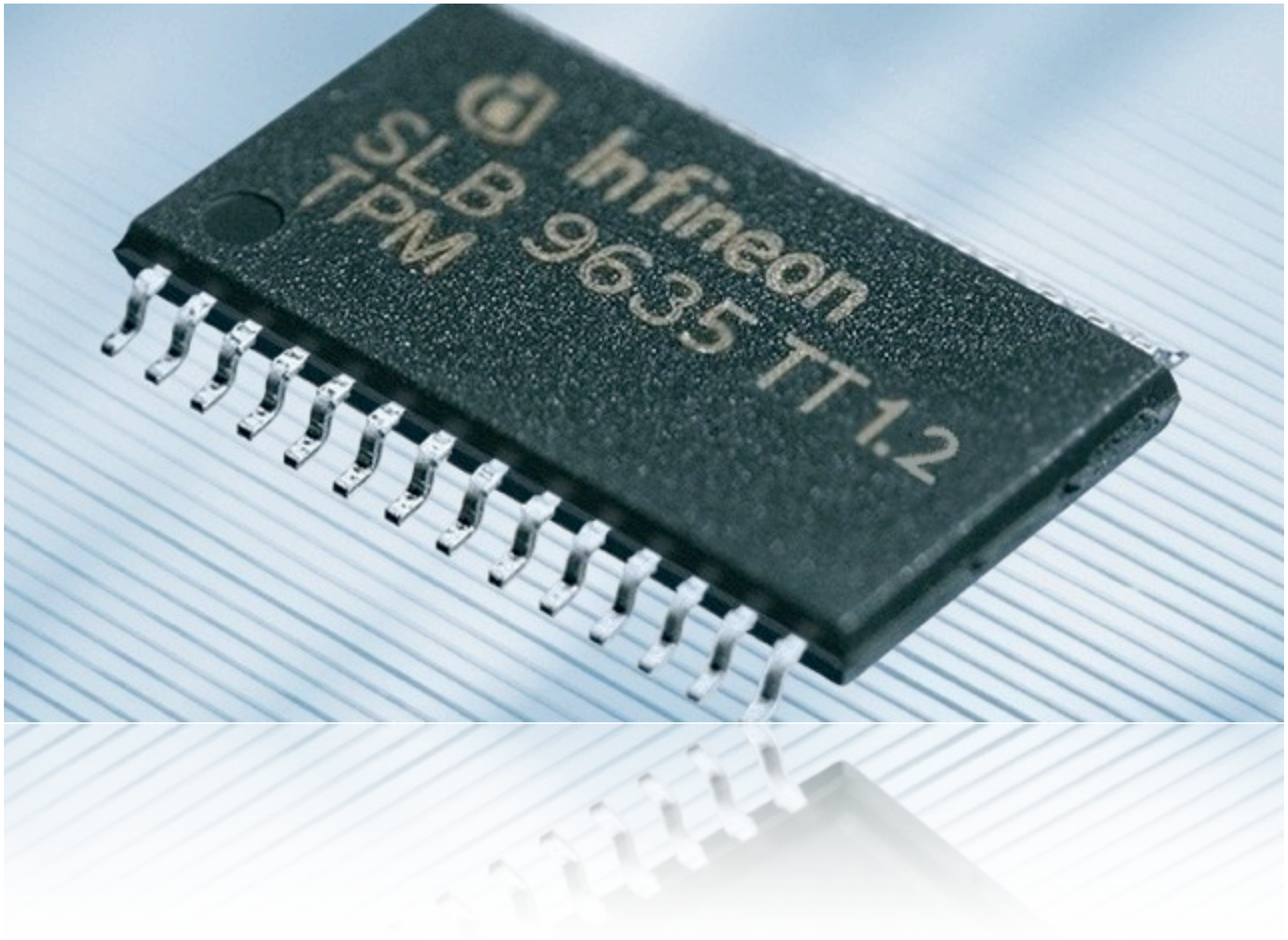


# Where do we want to go?




Vitale
carte d'assurance maladie
émise le 15/10/2007
PATRICIA
XXXXXX

# smart cards



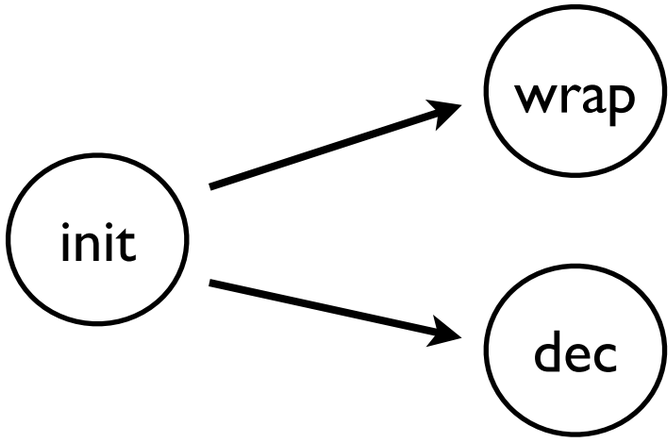

# TPM

# A really safe computer

# Commands

1. generate keys: $h \rightarrow (k, \mathrm{attr})$

2. change attributes




3. commands with pre-conditions

$$h_1 \rightarrow (k_1, \mathrm{wrap}), h_2 \rightarrow (k_2, a) : \{|k_2|\}_{k_1}$$

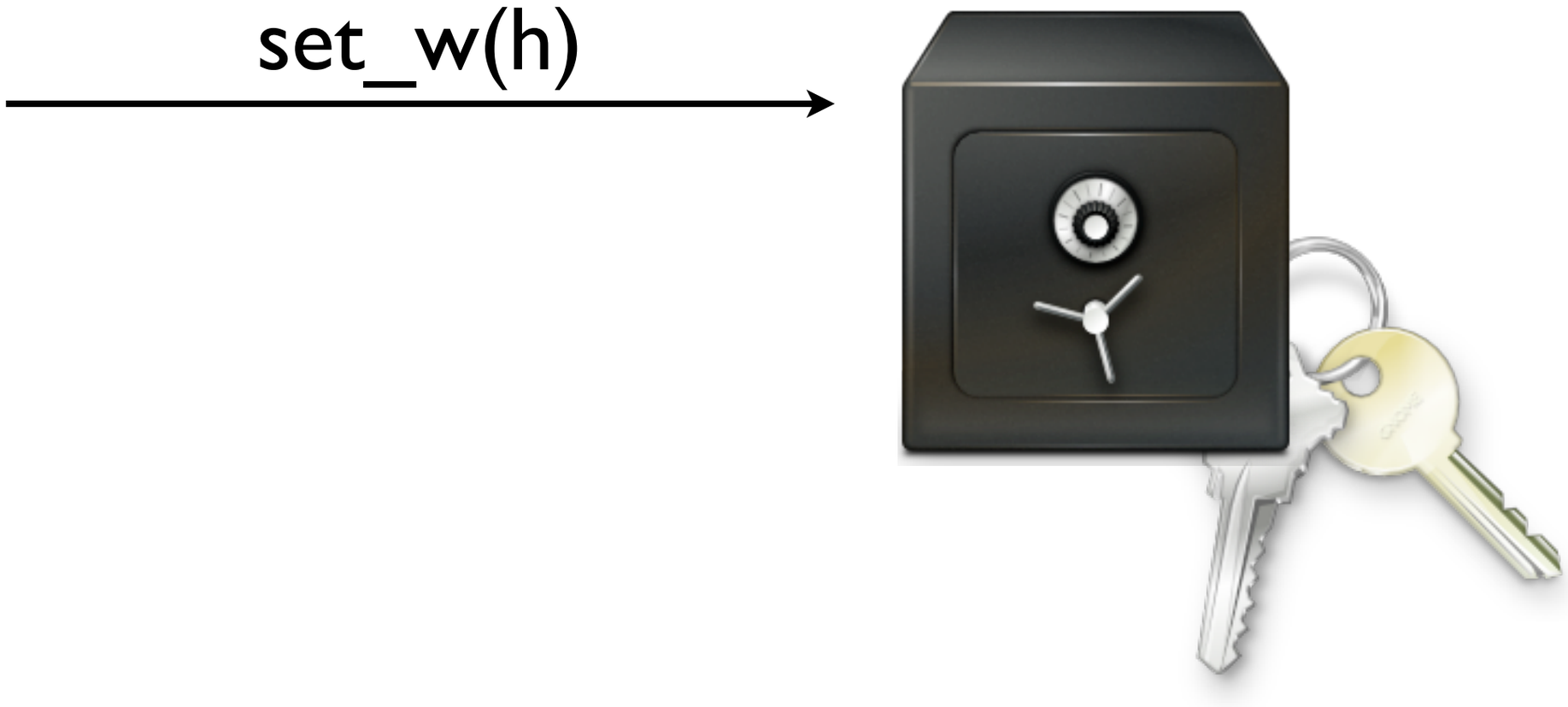
# Running Example: Wrap/Dec

1. Generate Key
2. Change Attributes
3. Wrap: $h_1 \rightarrow k_1, h_2 \rightarrow k_2 : \{|k_2|\}_{k_1}$
4. Dec: $h_1 \rightarrow k_1, \{|m|\}_{k_1} : m$

# A simple attack

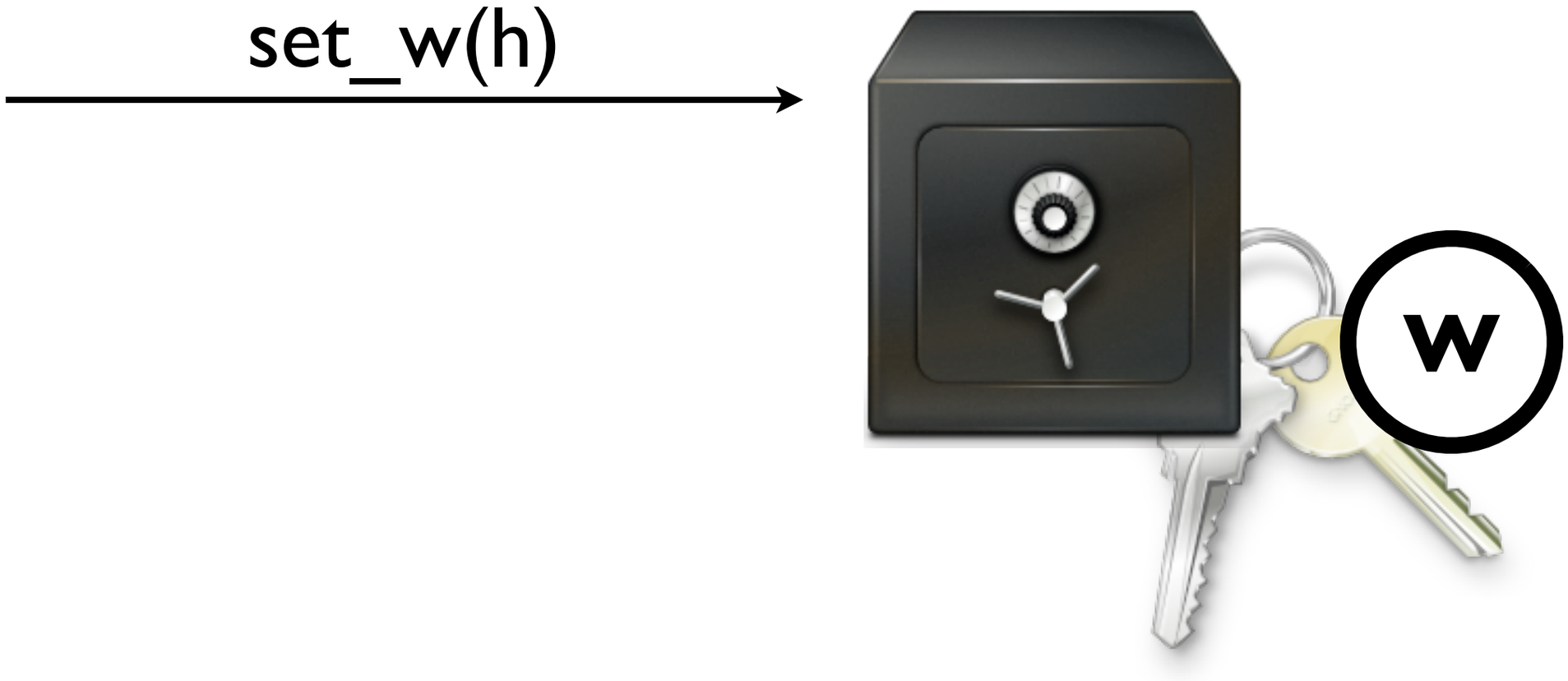
# A simple attack

set_w(h)



# A simple attack

set_w(h)



# A simple attack

set_w(h)

wrap(h,h)

# A simple attack

set_w(h)

wrap(h,h)

c= $\{|k|\}_k$

# A simple attack

set_w(h) →

wrap(h,h) →

← c= $\{|k|\}_k$

set_d(h) →

# A simple attack

set_w(h) →

wrap(h,h) →

← c= $\{|k|\}_k$

set_d(h) →

# A simple attack

set_w(h) →

wrap(h,h) →

← c= $\{|k|\}_k$

set_d(h) →

decrypt(h,c) →

# A simple attack

set_w(h) →

wrap(h,h) →

← c= $\{|k|\}_k$

set_d(h) →

decrypt(h,c) →

← k

# A simple attack

set_w(h) →

wrap(h,h) →

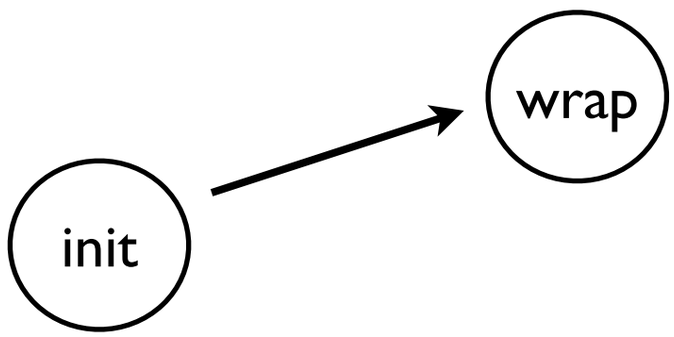
← c= $\{|k|\}_k$

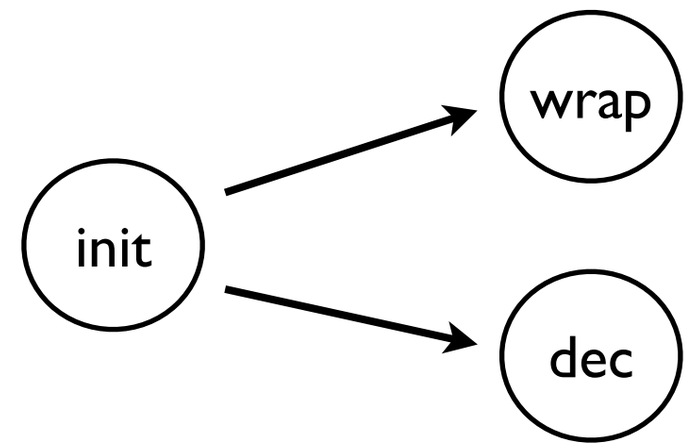
set_d(h) →

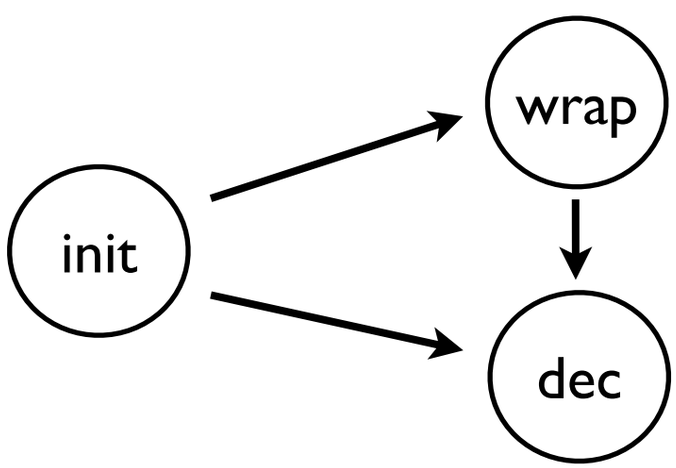
decrypt(h,c) →

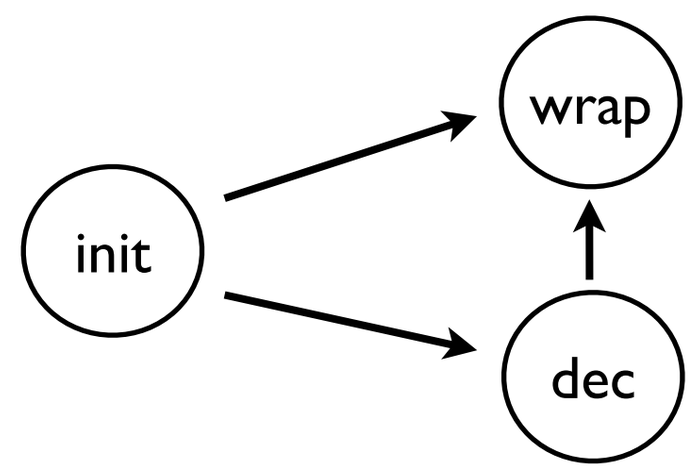
← k !!

# Policies



# Model as a protocol

Attacker

# Model as a protocol

Attacker

state (keys+attrs) grows non-monotonically
→ facts can disappear

# Model as a protocol

Attacker

state (keys+attrs) grows non-monotonically
→ facts can disappear

capture <u>causality</u>

# Model as a protocol

Attacker

state (keys+attrs) grows non-monotonically
→ facts can disappear

capture <u>causality</u>

avoid <u>state explosion</u>

# Existing Approaches

# Existing Approaches

- ProVerif[B01] (the most popular):
  - translate protocol in pi-calculus to Horn clauses, then perform resolution

$$mess(c, m_1) \wedge \ldots \wedge mess(c, m_n) \rightarrow mess(c, m_{out})$$

  - sound, but not complete
  - might diverge, but verifies most stateless protocols

# State and Horn Clauses

weniger details!

dreimal durchsprechen!

- direct encoding: private channels
- Abstractions: translations into Horn clauses
  - StatVerif [ARR11]
  - abstraction by set-membership[M10]
- Strand Spaces w/ state [G12]

# State and Horn Clauses

- direct encoding: private channels
- Abstractions: translations i
  - StatVerif [ARR11]
  - abstraction by set-memb
- Strand Spaces w/ state [G

fact persists

$mess(c, M)$

# State and Horn Clauses

- direct encoding: private channels
- Abstractions: translations into Horn clauses
  - StatVerif [ARR11]
  - abstraction by set-membership[M10]
- Strand Spaces w/ state [G12]

# State and Horn Clauses

- direct encoding: private channels
- Abstractions: translations into Horn clauses
  - StatVerif [ARR11]
  - abstraction by set-m
- Strand Spaces w/ state

Put state into facts

$$mess(\sigma, c, M)$$

only fixed number of cells

# State and Horn Clauses

- direct encoding: private channels
- Abstractions: translations into Horn clauses
  - StatVerif [ARR11]
  - abstraction by set-membership[M10]
- Strand Spaces w/ state [G12]

# State and Horn Clauses

- direct encoding
- Abstractions: tr
  - StatVerif [ARF
  - abstraction by set-membership[M10]
- Strand Spaces w/ state [G12]

abstract only certain objects

$$senc(k, m) \rightarrow senc(val(0, 1, 0), m)$$
$$(k \notin Init, k \in Wrap, k \notin Dec)$$

no key-compromise, re-import

# State and Horn Clauses

- direct encoding: private channels
- Abstractions: translations into Horn clauses
  - StatVerif [ARR11]
  - abstraction by set-membership[M10]
- Strand Spaces w/ state [G12]

# State and Horn Clauses

- direct encoding
- Abstractions: tr
  - StatVerif [ARF
  - abstraction by set-membership[M10]
- Strand Spaces w/ state [G12]

captures causality (protocol runs as acylic graphs)

no tool support

# Tamarin

[SMCB10]

# How tamarin works

- State consists of permanent and linear facts
- Multiset Rewriting Rules for:
  - Protocol
  - Message Deduction
  - Equational Theory

# How tamarin works

Rules describing the adversary:

$$
\begin{aligned}
MD = \{\ & \mathsf{Out}(x) \text{–}[\,]\!\rightarrow \mathsf{!K}(x), && \mathsf{!K}(x) \text{–}[\ \mathsf{K}(x)\ ]\!\rightarrow \mathsf{In}(x) && \}\\
\cup \{\ & \text{–}[\,]\!\rightarrow \mathsf{!K}(x{:}pub), && \mathsf{Fr}(x{:}fresh) \text{–}[\,]\!\rightarrow \mathsf{!K}(x{:}fresh) && \}\\
\cup \{\ & \mathsf{!K}(x_1), \ldots, \mathsf{!K}(x_k) \text{–}[\,]\!\rightarrow \mathsf{!K}(f(x_1, \ldots, x_k)) \mid f \in \Sigma^k_{DH} && && \}
\end{aligned}
$$

# How tamarin works

$l -[a]\!\!\rightarrow r$ ground instantiation of a rule

$S$ contains facts from l

$S'$ is S minus old and plus new facts

$$S \xrightarrow{a} S'$$

$\mathrm{traces}(P)$ all possible sequences of actions

# constraint solving

- property is negated (counter example)
- translated into graph constraints
- refine constraints (case distinction)
- arrive at solved constraint system:
  - report attack
- no refinements to apply anymore:
  - successful verification

# constraint solving

- sound+complete, but might diverge
- example protocol:

$$\{ -[]\!\!\rightarrow A(1), A(x) -[]\!\!\rightarrow A(x) \}$$

# Yubikey Protocol

# Yubikey-protocol

- larger case study of a one-time password protocol

# Yubikey-protocol

usb →


enc(otp,k) →

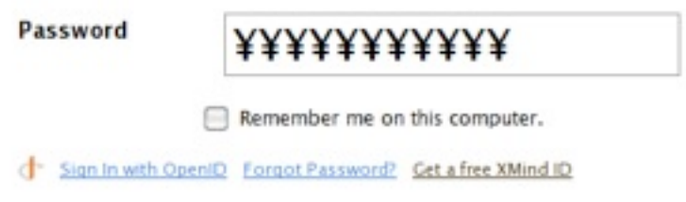

Network

enc(otp,k) ↓ ↑ ok

otp=<id, counter, ...>
counter larger than last one saved

# Yubikey-protocol

usb →




enc(otp,k) →

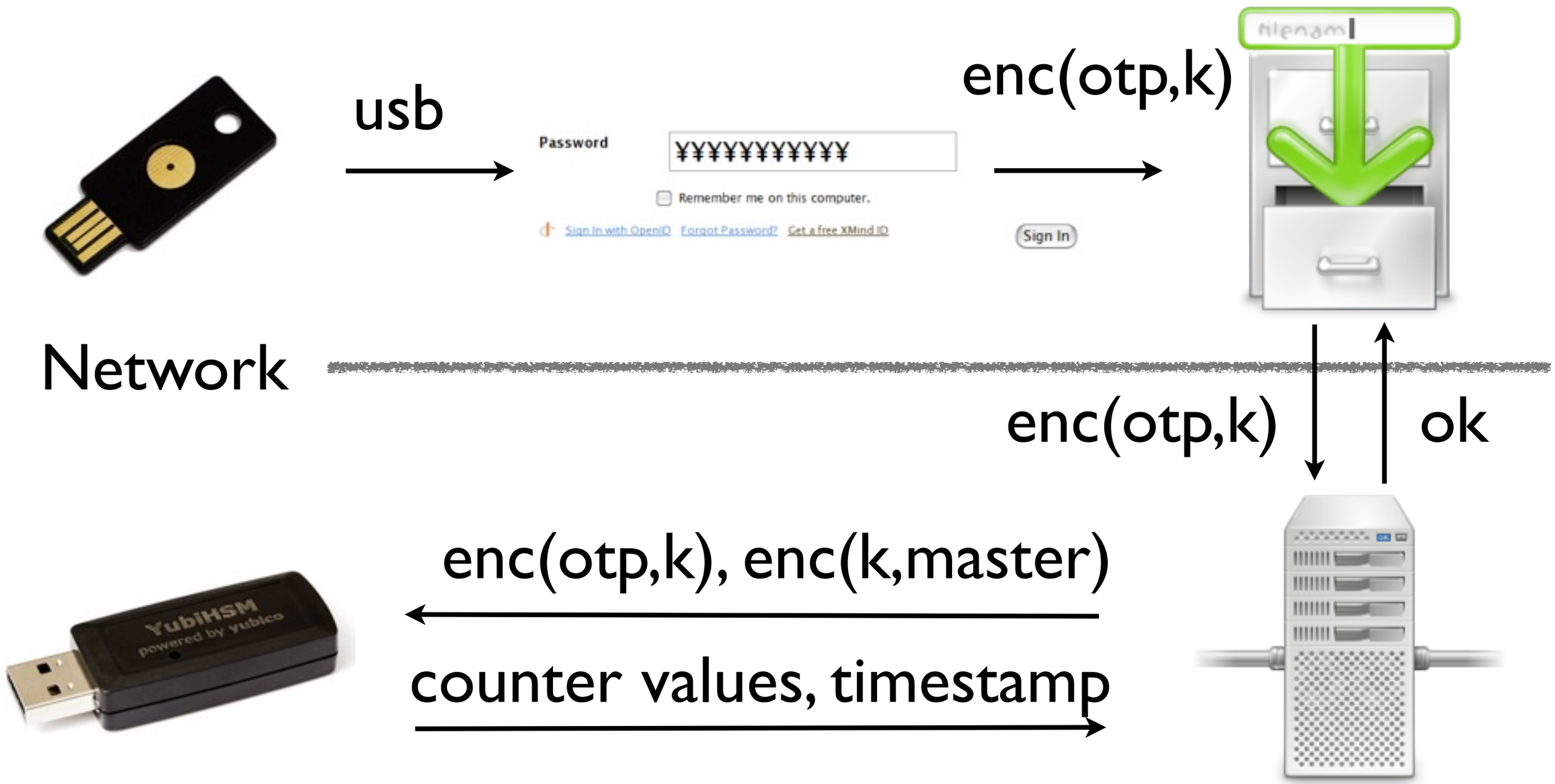
Network

are replay attacks possible?

enc(otp,k) ↓ ↑ ok

otp=<id, counter, ...>
counter larger than last one saved

# Yubikey-protocol



# Yubikey-protocol

usb →


enc(otp,k) →

Network

Is this an improvement?

enc(otp,k) ↓ ↑ ok


enc(otp,k), enc(k,master) ←

counter values, timestamp →

# Yubikey-protocol

- larger case study of a one-time password protocol
- tamarin can show (with manual intervention) that replay-attacks are impossible
- tamarin can show that the YubiHSM provides secrecy of the otp-keys in case of server compromise (if configured carefully, 2 servers necessary)

# A calculus for stateful protocols

# A calculus for stateful protocols

- extend ProVerif's input language with operators for state manipulation:
  - write to map-like structure
  - locking!

# Wrap/Dec

```
!(
  in('create');
  new h; new k;
  event NewKey(h,k);
  insert <'key',h>,k;
  insert <'att',h>, 'init';
  out(h)
)
```

# Wrap/Dec (ctd.)

```
!( in(<'set_dec',h>);
   lock <'att',h>;
   lookup <'att',h> as a in
        if a='init'  then
          delete <'att',h>;
          insert <'att',h>, 'dec';
   unlock <'att',h> )
```

# Why?

- easier for novices (and ProVerif experts)
- large protocols are tedious in MSR
- important issue: rules are atomic, locking issues are not in the model (i.e. race conditions could be overseen)
- Yubikey protocol: 49 (translated) VS 4 rules (manual)

# Challenges

- „Digestibility“ - translated rules should allow for easy verification in tamarin
- Completeness + Soundness

# A gist of the translation

- every node in process tree represented by (at least) one rule
- a fact $\mathsf{state}_p(v_1, \ldots, v_n)$ is present if statement at position p with can be executed next
- ex: $state_p(\ldots) \rightarrow state_{p \cdot 1}(\ldots), state_{p \cdot 2}(\ldots)$
- carefully crafted axioms

# Case studies

| Example | Typing Lemmas | Automated Run |
|---|---|---|
| Security API à la PKCS#11 | 1 | yes |
| Yubikey Protocol [19, 28] | 3 | no |
| GJM Contract-Signing protocol [3, 15] | 0 | yes |
| Mödersheim's Example (locks/inserts) [22] | 0 | little interaction |
| Mödersheim's Example (embedded MSRs) [22] | 0 | yes |
| Security Device Example [3] | 1 | yes |
| Needham-Schroeder-Lowe [21] | 1 | yes |

# Further Information

Robert Künnemann, Graham Steel. YubiSecure? Formal Security Analysis and Results for the Yubikey and YubiHSM. In: STM'2012

Steve Kremer, Robert Künnemann, Graham Steel. Universally Composable Key-Management. In: ESORICS'2013

Steve Kremer, Robert Künnemann. Automated analysis of security protocols with global state. (In submission)

# Conclusions

- protocols can be represented using Horn clauses
- very efficient (resolution) and sufficiently precise for most stateless protocols
- stateful protocols need more precision
- MSRs as workhorse, high-level representation to avoid mistakes

Thanks for your attention.

# References

[B01] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: CSFW'01

[G12] Joshua D. Guttman. State and Progress in Strand Spaces: Proving Fair Exchang. In: J. Automated Reasoning

[ARR11] Myrto Arapinis, Eike Ritter, Mark Ryan.StatVerif: Verification of Stateful Processes. In: CSF’11

[M10] Sebastian Mödersheim. Abstraction by set-membership: verifying security protocols and web services with databases. In: CCS‘2010

[SMCB12] Benedikt Schmidt, Simon Meier, Cas Cremers, David Basin. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Propertie. In: CSF‘2012