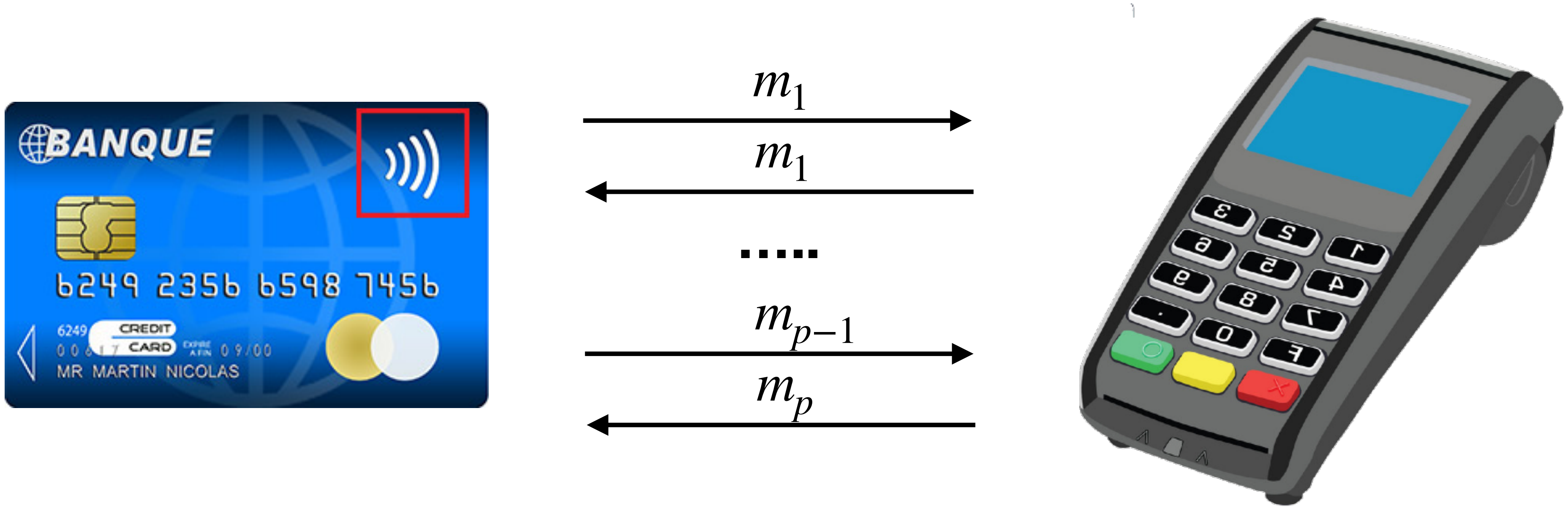
Symbolic verification of cryptographic protocols

Alexandre Debant

Univ Rennes - IRISA - CNRS

Rentrée ENS Rennes 2019
September 5th 2019

Distance bounding protocols

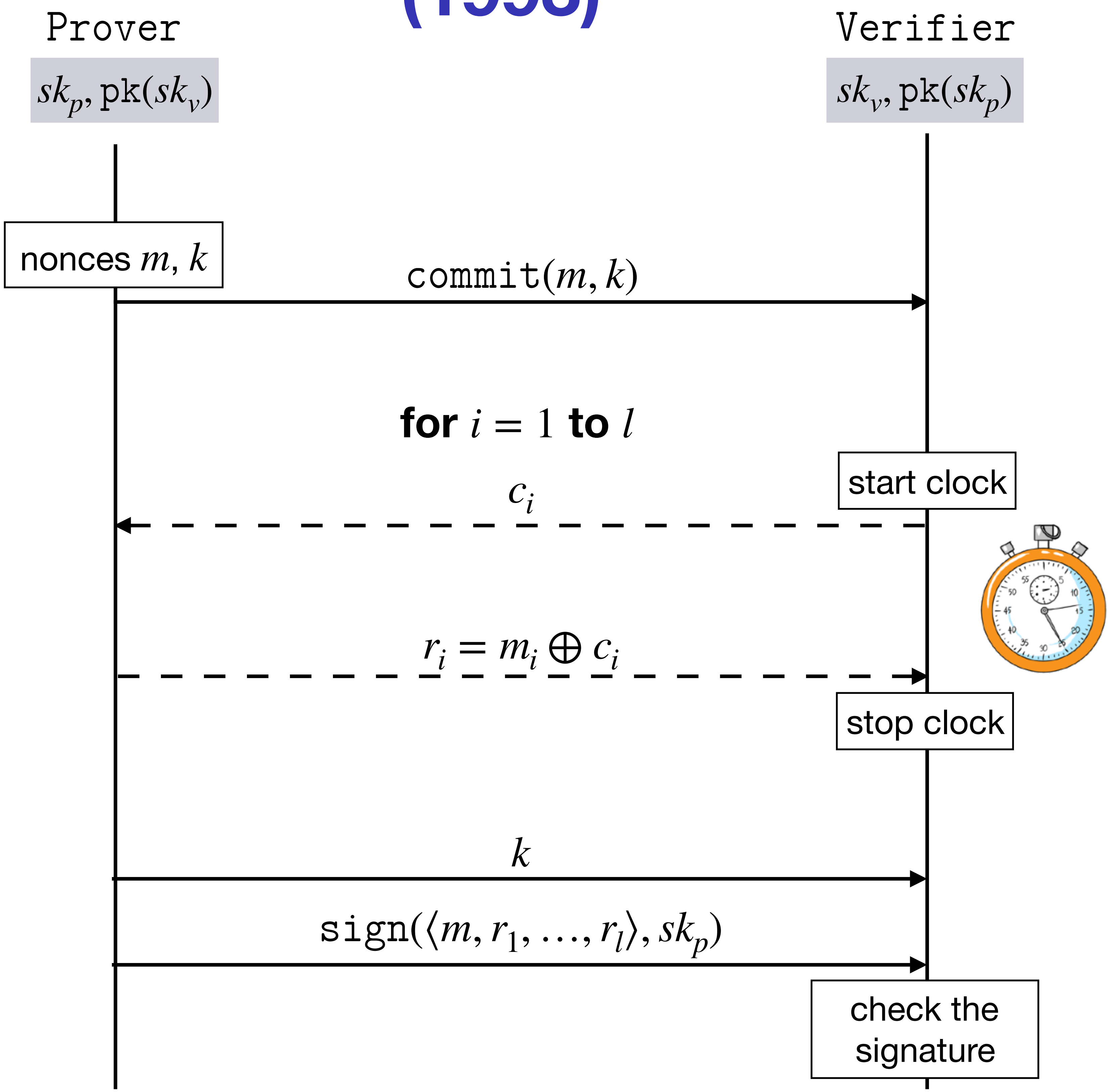


The payment reader must **authenticate**

AND

verify the proximity of the card.

Brands and Chaum (1993)



Prover

$sk_p, \mathtt{pk}(sk_v)$

Verifier

$sk_v, \mathtt{pk}(sk_p)$

nonces m, k

$\mathtt{commit}(m, k)$

for $i = 1$ **to** l

start clock

c_i

$r_i = m_i \oplus c_i$

stop clock

k

$\mathtt{sign}(\langle m, r_1, \ldots, r_l \rangle, sk_p)$

check the signature

Two attack scenarios

Mafia fraud (i.e. Man In the Middle)

An attacker, located in-between a verifier and a remote prover, tries to make the verifier think that they are close.

Two attack scenarios

Mafia fraud (i.e. Man In the Middle)

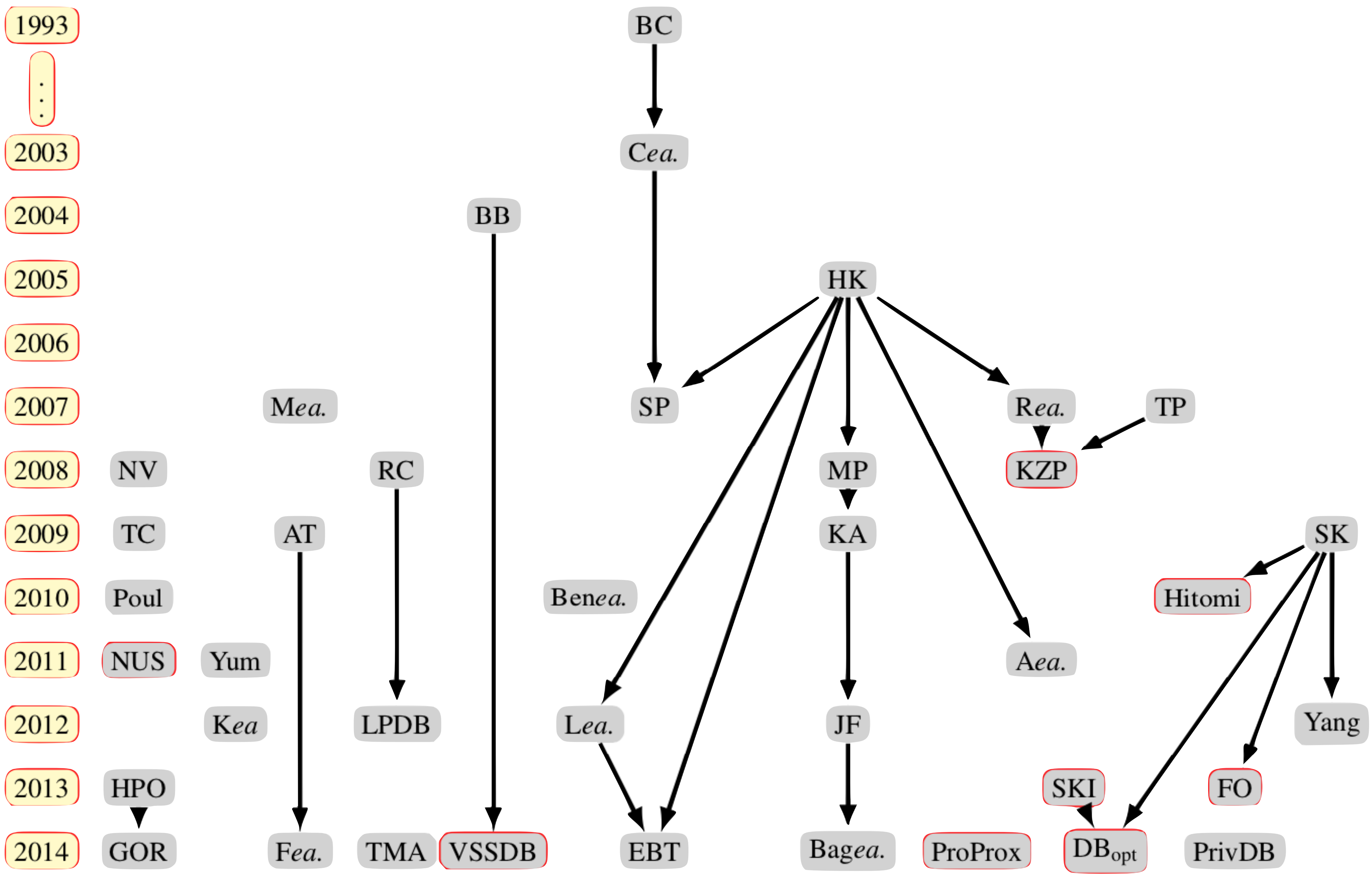
An attacker, located in-between a verifier and a remote prover, tries to make the verifier think that they are close.

Distance fraud (or distance hijacking)

An attacker tries to abuse honest provers to be authenticated by a remote verifier.

Survey of DB protocols

Brelurut *et al*. [FPS'15]



From 2003 to 2018: 40+ protocols proposed

Two major families of models…

… with some advantages and some drawbacks.

Computational models

\+ messages are bitstrings, a general and powerful attacker

– tedious proofs by hand and very error-prone

Symbolic models

– few abstractions (messages, attacker…)

\+ automatic procedures and existing tools

Some results make a link between these two models
[Abadi & Rogaway, 2000]

Table of contents

Table of contents

Overview of symbolic models

Symbolic models:

(i) Messages: abstracted with terms (e.g. $\mathtt{enc}(\langle n_1, n_2\rangle, k)$)

(ii) Protocols: specific logics, process algebra, multiset rewriting rules

(iii) Properties: trace property or equivalence property

Scyther

TAMARIN
Tamarin prover interactive mode

AVANTSSAR

ProVerif

Term algebra

Messages: terms built over a set of names $\mathcal{N}$ and a signature Σ given an equational theory $\mathtt{E}$.

Example

- Names: $\mathcal{N} = \{a, n, k\}$
- Signature: $\Sigma = \{\mathtt{senc}, \mathtt{sdec}, \mathtt{sign}, \mathtt{check\_sign}, \mathtt{pk}, \mathtt{pair}, \mathtt{proj}_1, \mathtt{proj}_2, \oplus\ \}$

$$\mathtt{proj}_1(\mathtt{pair}(x, y)) = x \qquad x \oplus 0 = x$$

$$\mathtt{proj}_2(\mathtt{pair}(x, y)) = y \qquad x \oplus x = 0$$

$$\mathtt{sdec}(\mathtt{senc}(x, y), y) = x \qquad x \oplus y = y \oplus x$$

$$\mathtt{check\_sign}(\mathtt{sign}(x, k), \mathtt{pk}(k)) = x \qquad (x \oplus y) \oplus z = x \oplus (y \oplus z)$$

For example: $\mathtt{sdec}(\mathtt{senc}(n \oplus 0), k), k) =_E n$

Process algebra

The role of an agent is described by a process following the grammar:

$P \ := \ 0$	null process
$\mid \ \texttt{new}\ n\,.\,P$	name restriction
$\mid \ \texttt{let}\ x = u\ \texttt{in}\ P$	declaration
$\mid \ \texttt{if}\ u = v\ \texttt{then}\ P$	condition
$\mid \ \texttt{out}(u)\,.\,P$	output
$\mid \ \texttt{in}(x)\,.\,P$	input

Process algebra

The role of an agent is described by a process following the grammar:

$P := 0$	null process
$\mid\ \texttt{new}\ n\,.\,P$	name restriction
$\mid\ \texttt{let}\ x = u\ \texttt{in}\ P$	declaration
$\mid\ \texttt{if}\ u = v\ \texttt{then}\ P$	condition
$\mid\ \texttt{out}(u)\,.\,P$	output
$\mid\ \texttt{in}(x)\,.\,P$	input
$\mid\ \texttt{in}^{<t}(x)\,.\,P$	guarded input
$\mid\ \texttt{reset}\,.\,P$	personal clock reset

Process algebra

The role of an agent is described by a process following the grammar:

$P \ := \ 0$	null process
$\mid \ \mathtt{new}\ n\,.\,P$	name restriction
$\mid \ \mathtt{let}\ x = u\ \mathtt{in}\ P$	declaration
$\mid \ \mathtt{if}\ u = v\ \mathtt{then}\ P$	condition
$\mid \ \mathtt{out}(u)\,.\,P$	output
$\mid \ \mathtt{in}(x)\,.\,P$	input
$\mid \ \mathtt{in}^{<t}(x)\,.\,P$	guarded input
$\mid \ \mathtt{reset}\,.\,P$	personal clock reset

Distance-bounding protocol

A distance-bounding protocol is a pair of roles (V, P) representing the verifier and the prover roles.

Example: Brands and Chaum - 1993

$V(sk_v, pk_p) :=$

$\mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,.$

$\mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,.$

$\mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,.$

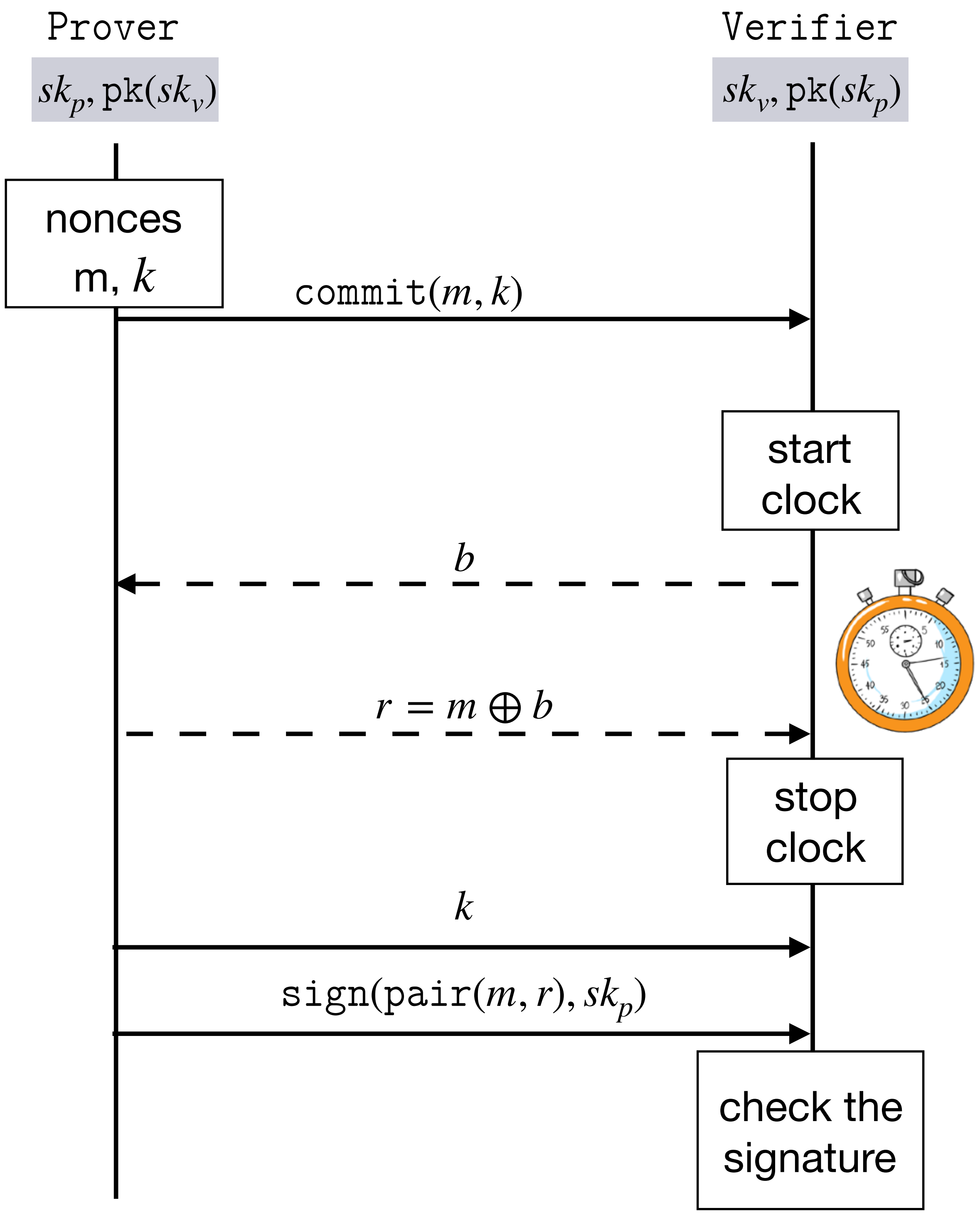
$\mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in}$

$\mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}}, \mathtt{pk}(sk_p))\ \mathtt{in}$

$\mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b) = y_{\mathtt{msg}}\ \mathtt{then}$

$\mathtt{if}\ b \oplus y_m = y_0\ \mathtt{in}$

0



Prover

$sk_p, \mathtt{pk}(sk_v)$

Verifier

$sk_v, \mathtt{pk}(sk_p)$

nonces m, k

$\mathtt{commit}(m, k)$

start clock

b

$r = m \oplus b$

stop clock

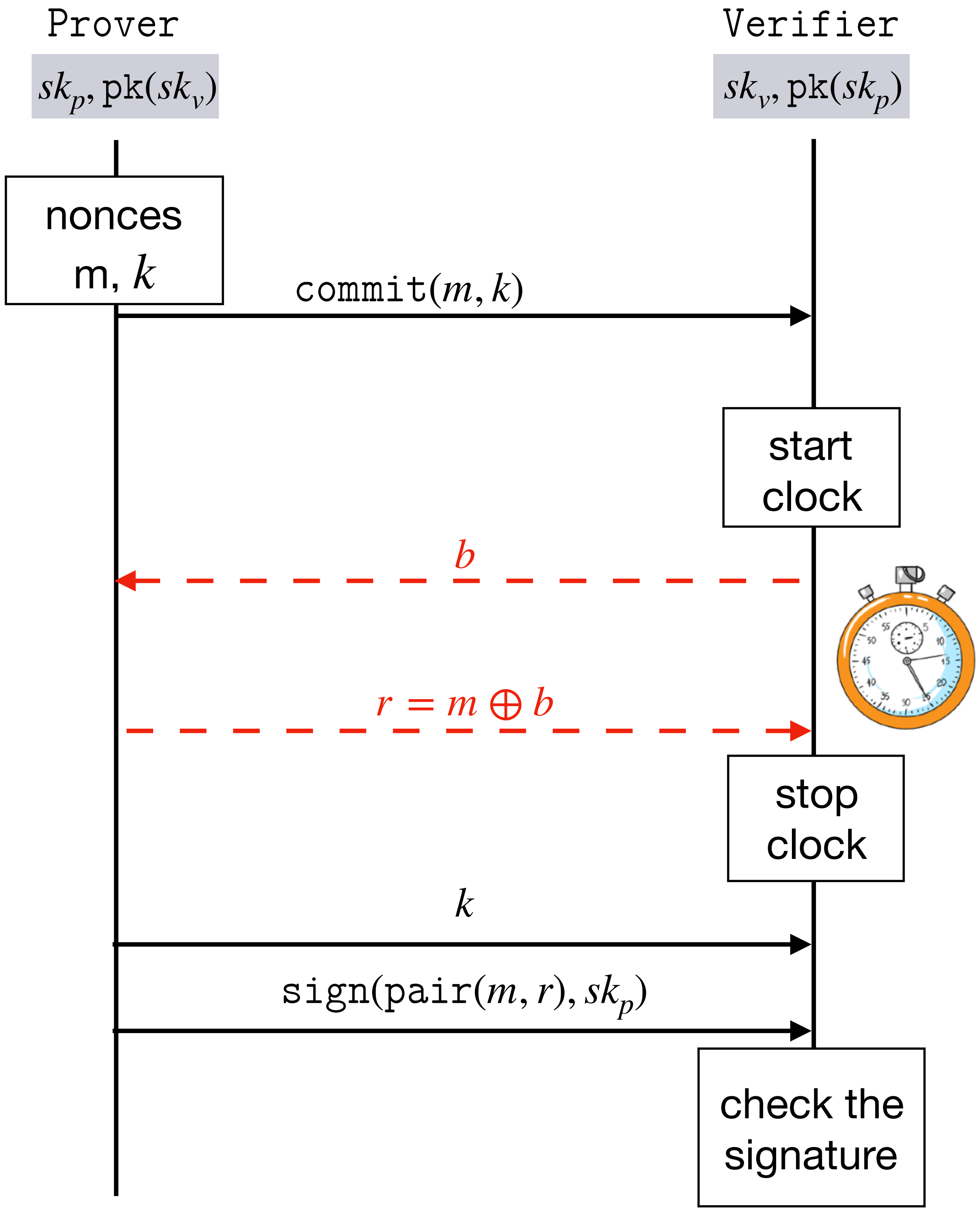
k

$\mathtt{sign}(\mathtt{pair}(m, r), sk_p)$

check the signature

Example: Brands and Chaum - 1993

$$
\begin{array}{l}
V(sk_v, pk_p) := \\
\quad \mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,. \\
\quad \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,. \\
\quad \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,. \\
\quad \mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in} \\
\quad \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}}, \mathtt{pk}(sk_p))\ \mathtt{in} \\
\quad \mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b) = y_{\mathtt{msg}}\ \mathtt{then} \\
\quad \mathtt{if}\ b \oplus y_m = y_0\ \mathtt{in} \\
\quad 0
\end{array}
$$



Prover

$sk_p, \mathtt{pk}(sk_v)$

Verifier

$sk_v, \mathtt{pk}(sk_p)$

nonces
m, k

$\mathtt{commit}(m, k)$

start
clock

b

$r = m \oplus b$

stop
clock

k

$\mathtt{sign}(\mathtt{pair}(m, r), sk_p)$

check the
signature

Example: Brands and Chaum - 1993

$V(sk_v, pk_p) :=$

$\mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,.$

$\mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,.$

$\mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,.$

$\mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in}$

$\mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}}, \mathtt{pk}(sk_p))\ \mathtt{in}$

$\mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b) = y_{\mathtt{msg}}\ \mathtt{then}$

$\mathtt{if}\ b \oplus y_m = y_0\ \mathtt{in}$

0

$P(sk_p, pk_v) :=$

$\mathtt{new}\ m\,.\,\mathtt{new}\ k\,.$

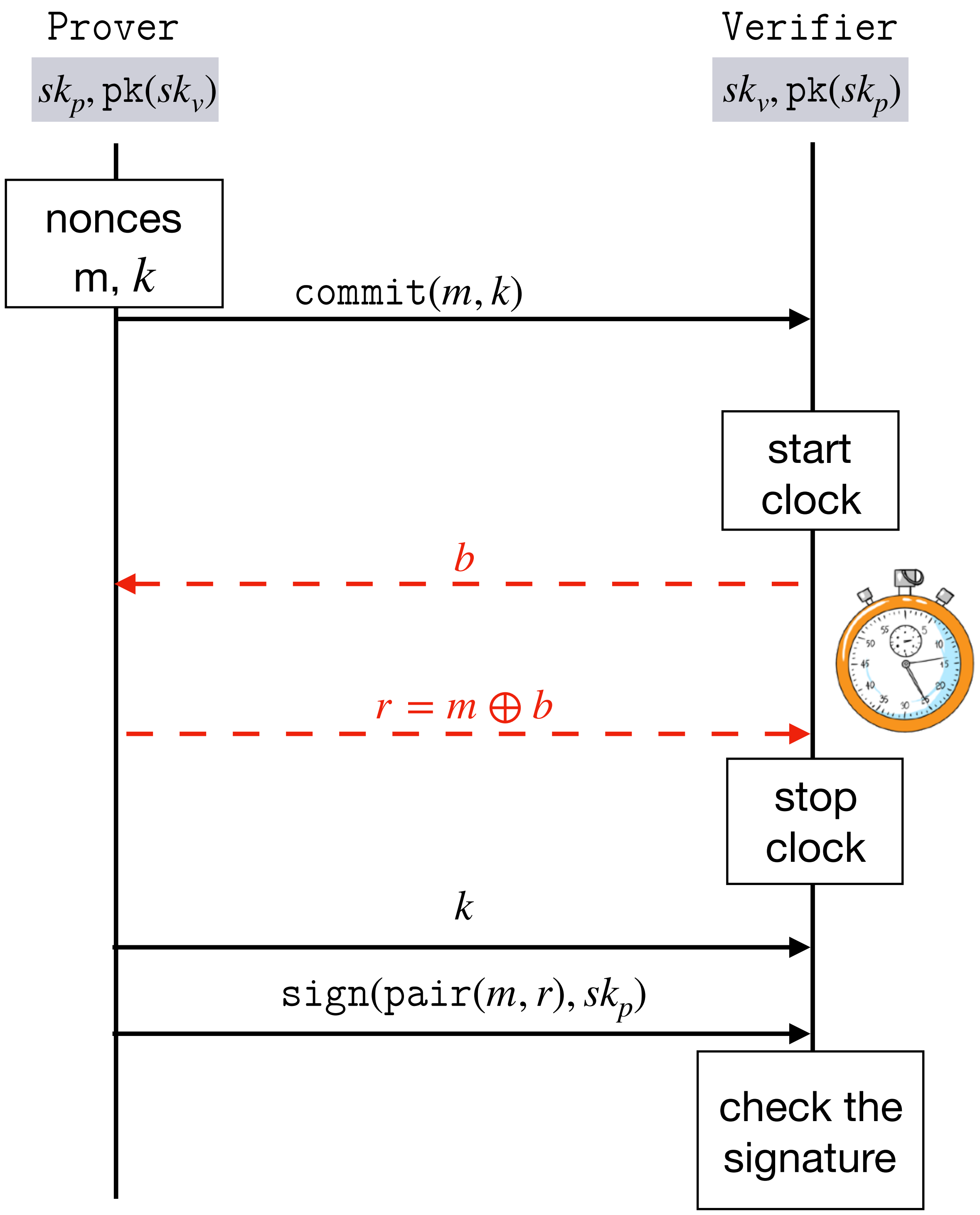
$\mathtt{out}(\mathtt{commit}(m, k))\,.$

$\mathtt{in}(y_b)\,.$

$\mathtt{out}(m \oplus y_b)\,.$

$\mathtt{out}(k)\,.\,\mathtt{out}(\mathtt{sign}(\mathtt{pair}(\mathtt{m}, \mathtt{m} \oplus \mathtt{y_b}), \mathtt{sk_p}))\,.$

0



Prover

Verifier

$sk_p, \mathtt{pk}(sk_v)$

$sk_v, \mathtt{pk}(sk_p)$

nonces
m, k

$\mathtt{commit}(m, k)$

start
clock

b

$r = m \oplus b$

stop
clock

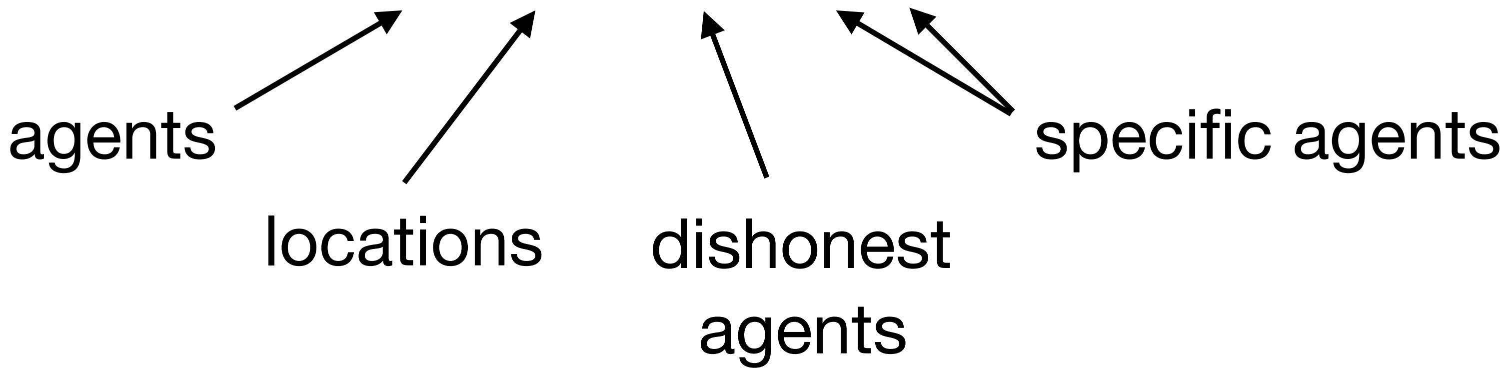
k

$\mathtt{sign}(\mathtt{pair}(m, r), sk_p)$

check the
signature

Topology

A **topology** is a tuple $\mathscr{T} = (\mathscr{A}, \mathtt{Loc}, \mathscr{M}, v, p)$.

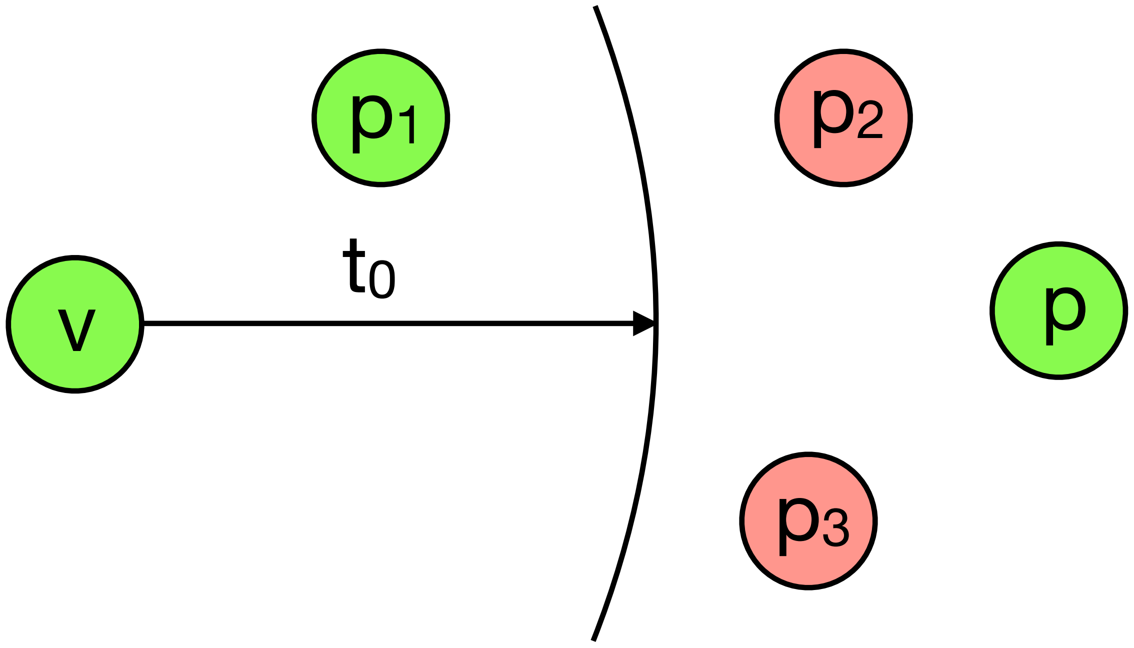


agents

locations

dishonest agents

specific agents



We define $\mathtt{Dist}_{\mathscr{T}}(a, b) = \dfrac{\|\mathtt{Loc}(a) - \mathtt{Loc}(b)\|}{c}$

Configuration and semantics

A **configuration** is a tuple $(\mathscr{P}; \Phi; t)$ where:

- $\mathscr{P}$ is a multiset of $\lfloor P \rfloor_a^{t_a}$ with $a \in \mathscr{A}$ and $t_a \in \mathscr{R}_+$
- $\Phi = \{\mathtt{w}_1 \xrightarrow{a_1, t_1} m_1, \ldots, \mathtt{w}_n \xrightarrow{a_n, t_n} m_n\}$ is a frame
- $t \in \mathscr{R}_+$ is the global time

Configuration and semantics

A **configuration** is a tuple $(\mathcal{P}; \Phi; t)$ where:

- $\mathcal{P}$ is a multiset of $\lfloor P \rfloor_a^{t_a}$ with $a \in \mathcal{A}$ and $t_a \in \mathcal{R}_+$
- $\Phi = \{\mathtt{w}_1 \xrightarrow{a_1, t_1} m_1, \dots, \mathtt{w}_n \xrightarrow{a_n, t_n} m_n\}$ is a frame
- $t \in \mathcal{R}_+$ is the global time

TIME $(\mathcal{P}; \Phi; t) \longrightarrow_{\mathcal{T}_0} (\mathcal{P}'; \Phi; t')$

- $t' > t$
- $\mathcal{P}' = \{ \lfloor P \rfloor_a^{t_a + (t'-t)} \mid \lfloor P \rfloor_a^{t_a} \in \mathcal{P} \}$

Configuration and semantics

A **configuration** is a tuple $(\mathscr{P}; \Phi; t)$ where:

- $\mathscr{P}$ is a multiset of $\lfloor P \rfloor_a^{t_a}$ with $a \in \mathscr{A}$ and $t_a \in \mathscr{R}_+$
- $\Phi = \{\mathtt{w}_1 \xrightarrow{a_1, t_1} m_1, \ldots, \mathtt{w}_n \xrightarrow{a_n, t_n} m_n\}$ is a frame
- $t \in \mathscr{R}_+$ is the global time

OUT $$(\lfloor \mathtt{out}(u) . P \rfloor_a^{t_a} \uplus \mathscr{P}; \Phi; t) \xrightarrow{a, \mathtt{out}(u)}_{\mathscr{T}_0} (\lfloor P \rfloor_a^{t_a} \uplus \mathscr{P}; \Phi'; t)$$

with $\Phi' = \Phi \cup \{w \xrightarrow{a, t} u\}$

Configuration and semantics

A **configuration** is a tuple $(\mathscr{P}; \Phi; t)$ where:

- $\mathscr{P}$ is a multiset of $\lfloor P \rfloor_a^{t_a}$ with $a \in \mathscr{A}$ and $t_a \in \mathscr{R}_+$
- $\Phi = \{\mathtt{w}_1 \xrightarrow{a_1, t_1} m_1, \ldots, \mathtt{w}_n \xrightarrow{a_n, t_n} m_n\}$ is a frame
- $t \in \mathscr{R}_+$ is the global time

IN $$(\lfloor \mathtt{in}^*(x).P \rfloor_a^{t_a} \uplus \mathscr{P}; \Phi; t) \xrightarrow{a, \mathtt{in}^*(u)}_{\mathscr{T}_0} (\lfloor P\{x \mapsto u\} \rfloor_a^{t_a} \uplus \mathscr{P}; \Phi; t)$$

if u is deducible from Φ

Configuration and semantics

A **configuration** is a tuple $(\mathcal{P}; \Phi; t)$ where:

- $\mathcal{P}$ is a multiset of $\lfloor P \rfloor_a^{t_a}$ with $a \in \mathcal{A}$ and $t_a \in \mathcal{R}_+$
- $\Phi = \{\mathtt{w}_1 \xrightarrow{a_1, t_1} m_1, \ldots, \mathtt{w}_n \xrightarrow{a_n, t_n} m_n\}$ is a frame
- $t \in \mathcal{R}_+$ is the global time

IN

$$(\lfloor \mathtt{in}^*(x) . P \rfloor_a^{t_a} \uplus \mathcal{P}; \Phi; t) \xrightarrow{a, \mathtt{in}^*(u)}_{\mathcal{T}_0} (\lfloor P\{x \mapsto u\} \rfloor_a^{t_a} \uplus \mathcal{P}; \Phi; t)$$

if $\exists b \in \mathcal{A}, t_b \in \mathcal{R}_+$ such that $t_b \leq t - \mathtt{Dist}_{\mathcal{T}_0}(b, a)$ and:

- if $b \notin \mathcal{M}$ then $u \in img(\lfloor \Phi \rfloor_b^{t_b})$
- if $b \in \mathcal{M}$ then u is deducible from $\bigcup_{c \in \mathcal{A}} \lfloor \Phi \rfloor_c^{t_b - \mathtt{Dist}_{\mathcal{T}_0}(c,b)}$

Configuration and semantics

A **configuration** is a tuple $(\mathscr{P}; \Phi; t)$ where:

- $\mathscr{P}$ is a multiset of $\lfloor P \rfloor_a^{t_a}$ with $a \in \mathscr{A}$ and $t_a \in \mathscr{R}_+$
- $\Phi = \{\mathtt{w}_1 \xrightarrow{a_1, t_1} m_1, \ldots, \mathtt{w}_n \xrightarrow{a_n, t_n} m_n\}$ is a frame
- $t \in \mathscr{R}_+$ is the global time

NEW, LET, RESET…

Example

$$\begin{aligned} V := {} & \mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,. \\ & \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,. \\ & \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,. \\ & \mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in} \\ & \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}}, \mathtt{pk}(sk_p))\ \mathtt{in} \\ & \mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b) = y_{\mathtt{msg}}\ \mathtt{then} \\ & \mathtt{if}\ b \oplus y_m = y_0\ \mathtt{then} \\ & 0 \end{aligned}$$

$$\begin{aligned} P := {} & \mathtt{new}\ m\,.\,\mathtt{new}\ k\,. \\ & \mathtt{out}(\mathtt{commit}(m,k))\,. \\ & \mathtt{in}(y_b)\,. \\ & \mathtt{out}(m \oplus y_b)\,. \\ & \mathtt{out}(k)\,. \\ & \mathtt{out}(\mathtt{sign}(\mathtt{pair}(m, m \oplus y_b), sk_p))\,. \\ & 0 \end{aligned}$$

$$(\lfloor V \rfloor_v^0 \uplus \lfloor P \rfloor_p^0\ ; \{\,\}; 0)$$

Example

$$
\begin{array}{l}
V := \mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,.\\
\qquad \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,.\\
\qquad \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,.\\
\qquad \mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in}\\
\qquad \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}}, \mathtt{pk}(sk_p))\ \mathtt{in}\\
\qquad \mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b) = y_{\mathtt{msg}}\ \mathtt{then}\\
\qquad \mathtt{if}\ b \oplus y_m = y_0\ \mathtt{then}\\
\qquad 0
\end{array}
$$

$$
\begin{array}{l}
P_1 := \qquad\qquad \mathtt{new}\ k\,.\\
\qquad \mathtt{out}(\mathtt{commit}(m', k))\,.\\
\qquad \mathtt{in}(y_b)\,.\\
\qquad \mathtt{out}(m' \oplus y_b)\,.\\
\qquad \mathtt{out}(k)\,.\\
\qquad \mathtt{out}(\mathtt{sign}(\mathtt{pair}(m', m' \oplus y_b), sk_p))\,.\\
\qquad 0
\end{array}
$$

$$
(\lfloor V \rfloor_v^0 \uplus \lfloor P \rfloor_p^0\ ; \{\}; 0) \longrightarrow_{\mathscr{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_1 \rfloor_p^0\ ; \{\}; 0)
$$

Example

$$\begin{array}{l} V := \mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,. \\ \qquad \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,. \\ \qquad \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,. \\ \qquad \mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in} \\ \qquad \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}}, \mathtt{pk}(sk_p))\ \mathtt{in} \\ \qquad \mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b) = y_{\mathtt{msg}}\ \mathtt{then} \\ \qquad \mathtt{if}\ b \oplus y_m = y_0\ \mathtt{then} \\ \qquad 0 \end{array}$$

$$\begin{array}{l} P_1 := \\ \qquad \mathtt{out}(\mathtt{commit}(m',k'))\,. \\ \qquad \mathtt{in}(y_b)\,. \\ \qquad \mathtt{out}(m' \oplus y_b)\,. \\ \qquad \mathtt{out}(k')\,. \\ \qquad \mathtt{out}(\mathtt{sign}(\mathtt{pair}(m', m' \oplus y_b), sk_p))\,. \\ \qquad 0 \end{array}$$

$$(\lfloor V \rfloor_v^0 \uplus \lfloor P \rfloor_p^0\ ; \{\,\}; 0) \longrightarrow_{\mathscr{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_1 \rfloor_p^0\ ; \{\,\}; 0) \longrightarrow_{\mathscr{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_2 \rfloor_p^0\ ; \{\,\}; 0)$$

Example

$$
\begin{aligned}
V := {} & \mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,.\\
& \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,.\\
& \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,.\\
& \mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in}\\
& \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}}, \mathtt{pk}(sk_p))\ \mathtt{in}\\
& \mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b) = y_{\mathtt{msg}}\ \mathtt{then}\\
& \mathtt{if}\ b \oplus y_m = y_0\ \mathtt{then}\\
& 0
\end{aligned}
$$

$$
\begin{aligned}
P_1 := {} & \\
& \mathtt{in}(y_b)\,.\\
& \mathtt{out}(m' \oplus y_b)\,.\\
& \mathtt{out}(k')\,.\\
& \mathtt{out}(\mathtt{sign}(\mathtt{pair}(m', m' \oplus y_b), sk_p))\,.\\
& 0
\end{aligned}
$$

$$
\begin{aligned}
(\lfloor V \rfloor_v^0 \uplus \lfloor P \rfloor_p^0 ; \{\}; 0) & \longrightarrow_{\mathcal{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_1 \rfloor_p^0 ; \{\}; 0) \longrightarrow_{\mathcal{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_2 \rfloor_p^0 ; \{\}; 0)\\
& \xrightarrow{p,\mathtt{out}(\mathtt{commit}(m',k'))}_{\mathcal{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_3 \rfloor_p^0 ; \Phi_1; 0)
\end{aligned}
$$

With $\Phi_1 = \{\mathtt{w}_1 \xrightarrow{p,0} \mathtt{commit}(m', k')\}$

Example

$$
\begin{aligned}
V := {} & \mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,. \\
& \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,. \\
& \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,. \\
& \mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in} \\
& \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}}, \mathtt{pk}(sk_p))\ \mathtt{in} \\
& \mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b) = y_{\mathtt{msg}}\ \mathtt{then} \\
& \mathtt{if}\ b \oplus y_m = y_0\ \mathtt{then} \\
& 0
\end{aligned}
$$

$$
\begin{aligned}
P_1 := {} & \\
& \mathtt{in}(y_b)\,. \\
& \mathtt{out}(m' \oplus y_b)\,. \\
& \mathtt{out}(k')\,. \\
& \mathtt{out}(\mathtt{sign}(\mathtt{pair}(m', m' \oplus y_b), sk_p))\,. \\
& 0
\end{aligned}
$$

$$
\begin{aligned}
& (\lfloor V \rfloor_v^0 \uplus \lfloor P \rfloor_p^0\,;\{\,\};0) \longrightarrow_{\mathscr{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_1 \rfloor_p^0\,;\{\,\};0) \longrightarrow_{\mathscr{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_2 \rfloor_p^0\,;\{\,\};0) \\
& \xrightarrow{p,\mathtt{out}(\mathtt{commit}(m',k'))}_{\mathscr{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_3 \rfloor_p^0\,;\Phi_1;0) \longrightarrow_{\mathscr{T}} (\lfloor V \rfloor_v^{t_1} \uplus \lfloor P_3 \rfloor_p^{t_1}\,;\Phi_1;t_1)
\end{aligned}
$$

With $\Phi_1 = \{\mathtt{w}_1 \xrightarrow{p,0} \mathtt{commit}(m', k')\}$

Example

$$
\begin{array}{ll}
V_1 := & \mathtt{new}\ b\,.\\
& \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,.\\
& \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,.\\
& \mathtt{let}\ y_m = \mathtt{open}(\mathtt{commit}(m',k'), y_k)\ \mathtt{in}\\
& \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}}, \mathtt{pk}(sk_p))\ \mathtt{in}\\
& \mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b) = y_{\mathtt{msg}}\ \mathtt{then}\\
& \mathtt{if}\ b \oplus y_m = y_0\ \mathtt{then}\\
& 0
\end{array}
$$

$$
\begin{array}{ll}
P_1 := & \\
& \mathtt{in}(y_b)\,.\\
& \mathtt{out}(m' \oplus y_b)\,.\\
& \mathtt{out}(k')\,.\\
& \mathtt{out}(\mathtt{sign}(\mathtt{pair}(m',m' \oplus y_b), sk_p))\,.\\
& 0
\end{array}
$$

$$
\begin{array}{l}
(\lfloor V \rfloor_v^0 \uplus \lfloor P \rfloor_p^0\ ; \{\}; 0) \longrightarrow_{\mathcal{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_1 \rfloor_p^0\ ; \{\}; 0) \longrightarrow_{\mathcal{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_2 \rfloor_p^0\ ; \{\}; 0)\\
\qquad \xrightarrow{p,\mathtt{out}(\mathtt{commit}(m',k'))}_{\mathcal{T}} (\lfloor V \rfloor_v^0 \uplus \lfloor P_3 \rfloor_p^0\ ; \Phi_1; 0) \longrightarrow_{\mathcal{T}} (\lfloor V \rfloor_v^{t_1} \uplus \lfloor P_3 \rfloor_p^{t_1}\ ; \Phi_1; t_1)\\
\qquad \xrightarrow{v,\mathtt{in}(\mathtt{commit}(m',k'))}_{\mathcal{T}} (\lfloor V \rfloor_v^{t_1} \uplus \lfloor P_3 \rfloor_p^{t_1}\ ; \Phi_1; t_1)
\end{array}
$$

With $\Phi_1 = \{\mathtt{w}_1 \xrightarrow{p,0} \mathtt{commit}(m',k')\}$

Example

$V_2 :=$

$\mathtt{reset}\,.\,\mathtt{out}(b')\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,.$
$\mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,.$
$\mathtt{let}\ y_m = \mathtt{open}(\mathtt{commit}(m',k'),y_k)\ \mathtt{in}$
$\mathtt{let}\ y_{\mathtt{msg}} = \mathtt{check\_sign}(y_{\mathtt{sign}},\mathtt{pk}(sk_p))\ \mathtt{in}$
$\mathtt{if}\ \mathtt{pair}(y_m, y_m \oplus b') = y_{\mathtt{msg}}\ \mathtt{then}$
$\mathtt{if}\ b' \oplus y_m = y_0\ \mathtt{then}$
0

$P_1 :=$

$\mathtt{in}(y_b)\,.$
$\mathtt{out}(m' \oplus y_b)\,.$
$\mathtt{out}(k')\,.$
$\mathtt{out}(\mathtt{sign}(\mathtt{pair}(m',m' \oplus y_b),sk_p))\,.$
0

$$
\begin{aligned}
&(\lfloor V\rfloor_v^0 \uplus \lfloor P\rfloor_p^0\,;\{\};0) \longrightarrow_{\mathcal{T}} (\lfloor V\rfloor_v^0 \uplus \lfloor P_1\rfloor_p^0\,;\{\};0) \longrightarrow_{\mathcal{T}} (\lfloor V\rfloor_v^0 \uplus \lfloor P_2\rfloor_p^0\,;\{\};0)\\
&\xrightarrow{p,\mathtt{out}(\mathtt{commit}(m',k'))}_{\mathcal{T}} (\lfloor V\rfloor_v^0 \uplus \lfloor P_3\rfloor_p^0\,;\Phi_1;0) \longrightarrow_{\mathcal{T}} (\lfloor V\rfloor_v^{t_1} \uplus \lfloor P_3\rfloor_p^{t_1}\,;\Phi_1;t_1)\\
&\xrightarrow{v,\mathtt{in}(\mathtt{commit}(m',k'))}_{\mathcal{T}} (\lfloor V_1\rfloor_v^{t_1} \uplus \lfloor P_3\rfloor_p^{t_1}\,;\Phi_1;t_1) \longrightarrow_{\mathcal{T}} (\lfloor V_2\rfloor_v^{t_1} \uplus \lfloor P_3\rfloor_p^{t_1}\,;\Phi_1;t_1)
\end{aligned}
$$

With $\Phi_1 = \{\mathtt{w}_1 \xrightarrow{p,0} \mathtt{commit}(m',k')\}$

Example

$V_2 := 0$

$P_1 := 0$

$$(\lfloor V\rfloor_v^0 \uplus \lfloor P\rfloor_p^0 ; \{\}; 0) \longrightarrow_{\mathscr{T}} (\lfloor V\rfloor_v^0 \uplus \lfloor P_1\rfloor_p^0 ; \{\}; 0) \longrightarrow_{\mathscr{T}} (\lfloor V\rfloor_v^0 \uplus \lfloor P_2\rfloor_p^0 ; \{\}; 0)$$

$$\xrightarrow{p,\mathtt{out}(\mathtt{commit}(m',k'))}_{\mathscr{T}} (\lfloor V\rfloor_v^0 \uplus \lfloor P_3\rfloor_p^0 ; \Phi_1; 0) \longrightarrow_{\mathscr{T}} (\lfloor V\rfloor_v^{t_1} \uplus \lfloor P_3\rfloor_p^{t_1} ; \Phi_1; t_1)$$

$$\xrightarrow{v,\mathtt{in}(\mathtt{commit}(m',k'))}_{\mathscr{T}} (\lfloor V_1\rfloor_v^{t_1} \uplus \lfloor P_3\rfloor_p^{t_1} ; \Phi_1; t_1) \longrightarrow_{\mathscr{T}} (\lfloor V_2\rfloor_v^{t_1} \uplus \lfloor P_3\rfloor_p^{t_1} ; \Phi_1; t_1)$$

$$\longrightarrow_{\mathscr{T}} \dots \longrightarrow_{\mathscr{T}} (\lfloor 0\rfloor_v^{2t_1} \uplus \lfloor 0\rfloor_p^{3t_1} ; \Phi_4; 3t_1)$$

With $\Phi_1 = \{\mathtt{w}_1 \xrightarrow{p,0} \mathtt{commit}(m',k')\}$ and
$\Phi_4 = \Phi_1 \cup \{\mathtt{w}_2 \xrightarrow{v,t_1} b'; \mathtt{w}_3 \xrightarrow{p,t_2} m' \oplus b'; \mathtt{w}_4 \xrightarrow{p,t_2} k'; \mathtt{w}_5 \xrightarrow{p,t_2} \mathtt{sign}(\dots)\}$

Mafia fraud resistance

[FSTTCS'18]

Mafia fraud resistance: A verifier never authenticates a far-away prover, even considering attackers located in-between.

Topologies: we denote by $\mathscr{C}_{\mathtt{MF}}$ the set of topologies $(\mathscr{A}_0, \mathtt{Loc}_0, \mathscr{M}_0, v_0, p_0)$ such that

$$\mathtt{Dist}_{\mathscr{T}}(v_0, p_0) = \frac{\|\mathtt{Loc}_0(v_0) - \mathtt{Loc}_0(p_0)\|}{c} > t_{\mathtt{prox}}$$

Mafia fraud resistance

A distance-bounding protocol $\mathscr{P}_{\mathtt{prox}}$ is mafia fraud resistant if for all topology $\mathscr{T} \in \mathscr{C}_{\mathtt{MF}}$, there is no initial configuration $(\mathscr{P}_0; \Phi_0; t_0)$ such that:

$$(\mathscr{P}_0; \Phi_0 \cup \Phi_{\mathsf{sd}}; t_0) \xrightarrow{tr}_{\mathscr{T}} (\lfloor \mathtt{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathscr{P}; \Phi; t)$$

Distance fraud resistance

[FSTTCS'18]

Distance fraud resistance: A verifier never authenticates a far-away attacker if this last has no accomplice in the verifier's vicinity.

Topologies: we denote by $\mathscr{C}_{\mathtt{DF}}$ the set of topologies $(\mathscr{A}_0, \mathtt{Loc}_0, \mathscr{M}_0, v_0, p_0)$ such that

$p_0 \in \mathscr{M}_0$ **and** for all $a \in \mathscr{M}_0, \mathtt{Dist}_{\mathscr{T}}(v_0, a) > t_{\mathtt{prox}}$

Distance fraud resistance

A distance-bounding protocol $\mathscr{P}_{\mathtt{prox}}$ is distance fraud resistant if for all topology $\mathscr{T} \in \mathscr{C}_{\mathtt{DF}}$, there is no initial configuration $(\mathscr{P}_0; \Phi_0; t_0)$ such that:

$$(\mathscr{P}_0; \Phi_0 \cup \Phi_{\mathsf{sd}}; t_0) \xrightarrow{tr}_{\mathscr{T}} (\lfloor \mathtt{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathscr{P}; \Phi; t)$$

Table of contents

Theoretical limitations

for analysing weak secrecy/authentication properties

Verifying trace properties is:

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]
- decidable for very restrictive classes [Lowe, 99]
  [Rammanujam & Suresh, 03] [D'Osualdo *et al*, 17]

Theoretical limitations

for analysing weak secrecy/authentication properties

Verifying trace properties is:

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]
- decidable for very restrictive classes [Lowe, 99] [Rammanujam & Suresh, 03] [D'Osualdo *et a*l, 17]

But efficient automatic tools exist:

TAMARIN Tamarin prover interactive mode ProVerif AVANTSSAR

Some success stories:

Automatic analysis of DB protocols

Underlying attacker model of existing tools

The attacker controls all the network; he can intercept, build and send messages without introducing any delay.

⟶ not suitable to analyse distance-bounding protocols…

Automatic analysis of DB protocols

Underlying attacker model of existing tools

The attacker controls all the network; he can intercept, build and send messages without introducing any delay.

⟶ not suitable to analyse distance-bounding protocols…

How to overcome this issue?

1. develop a new procedure and implement it in a new tool
2. establish theoretical results to get rid of topologies and time
   e.g. reduce the number of topologies or prove causality results

Mafia fraud / Distance fraud: one topology is enough

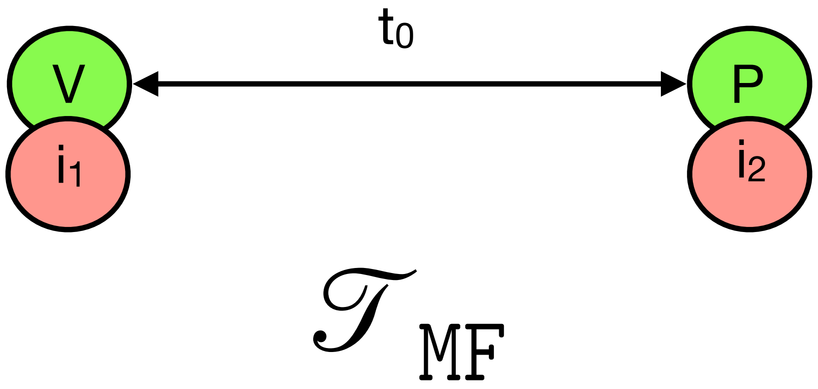
Theorem: one topology is enough

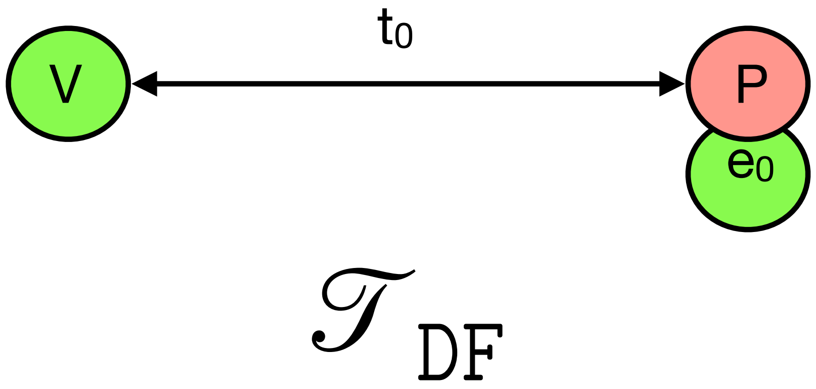
Let $\mathscr{P}_{\mathtt{prox}}$ be an **executable** distance-bounding protocol.

$\mathscr{P}_{\mathtt{prox}}$ is mafia fraud (resp. distance hijacking) resistant if and only if there is no initial configuration $(\mathscr{P}_0; \Phi_0; t_0)$ such that:

$$(\mathscr{P}_0; \Phi_0 \cup \Phi_{\mathsf{sd}}; t_0) \xrightarrow{tr}_{\mathscr{T}} (\lfloor \mathtt{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathscr{P}; \Phi; t)$$

with $\mathscr{T} = \mathscr{T}_{\mathtt{MF}}$ (resp. $\mathscr{T}_{\mathtt{DF}}$).





Encoding the reduced topologies

Up to now: we have reduced the number of topologies to only one

But: even a single topology **cannot be modeled** into existing tools

Encoding the reduced topologies

Up to now: we have reduced the number of topologies to only one

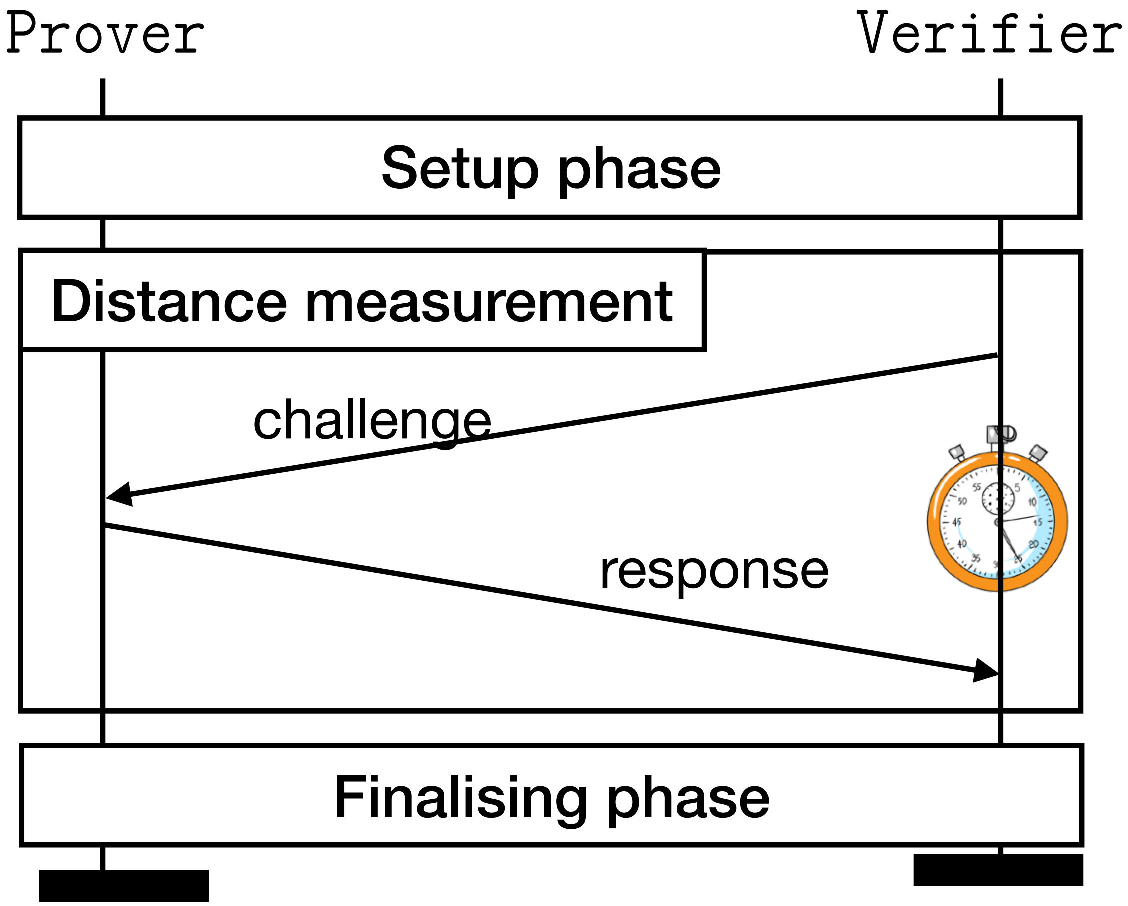
But: even a single topology **cannot be modeled** into existing tools

In **[FSTTCS'18]** proved that we can use the notion of phases available in the ProVerif tool to encode the reduced topologies.

Idea of the encoding

Use the notion of phases available in ProVerif

- ➡ *Phase 0* ⟶ *setup phase*
- ➡ *Phase 1* ⟶ *distance measurement*
- ➡ *Phase 2* ⟶ *finalising phase*



Case studies

Protocols	**Mafia fraud resistance**	**Distance fraud resistance**	**Terrorist fraud resistance**
Hancke and Kuhn	✓	✓	✗
Brands and Chaum	✓	✗	✗
Swiss-Knife	✓	✓	✓
SKI	✓	✓	✓
TREAD-Asymmetric	✗	✗	✓
TREAD-Asymmetric *fixed*	✓	✗	✓
TREAD-Symmetric	✓	✗	✓
Spade	✗	✗	✓
Spade *fixed*	✓	✗	✓
Munilla *et al.*	✓	✓	✗
MAD	✓	✗	✗
PaySafe	✓	✗	✗
NXP	✓	✗	✗

(✗: attack found, ✓: proved secure)
(we never obtained false attacks or non-termination)

Conclusion

Designing and analyzing cryptographic protocols is difficult!

But there exist automatic verification tools to:

- verify well-known security properties (e.g. confidentiality, authentication…)
- model standard cryptographic primitives
- analyse small protocols

Conclusion

Designing and analyzing cryptographic protocols is difficult!

But there exist automatic verification tools to:

- verify well-known security properties (e.g. confidentiality, authentication…)
- model standard cryptographic primitives
- analyse small protocols

Ongoing and future works:

- analyse new classes of protocols (e.g. DB protocols, stateful protocols…)
- verify new security properties (proximity, unlinkability…)
- model new primitives (homomorphic encryption, XOR…)
- …