A symbolic framework to analyse physical proximity in security protocols

Alexandre Debant, Stéphanie Delaune, Cyrille Wiedling

Univ Rennes - IRISA - CNRS

December 13, 2018

Introduction

Security protocols

Distributed programs that use cryptographic primitives to ensure **security properties.**

Secrecy Authentication

Integrity

Untraceability

Introduction

Security protocols

Distributed programs that use cryptographic primitives to ensure **security properties.**

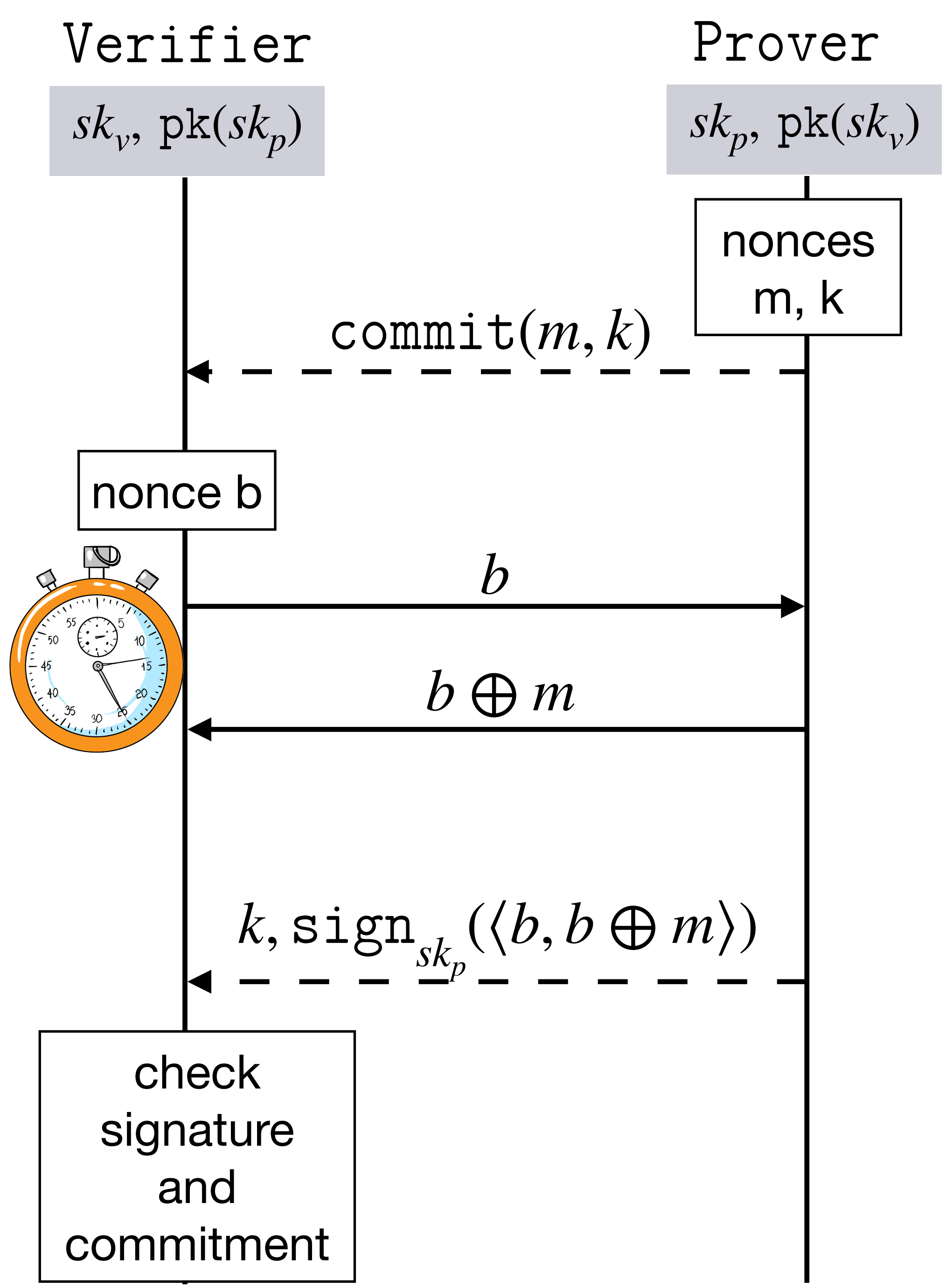
Secrecy ~~Authentication~~

Integrity

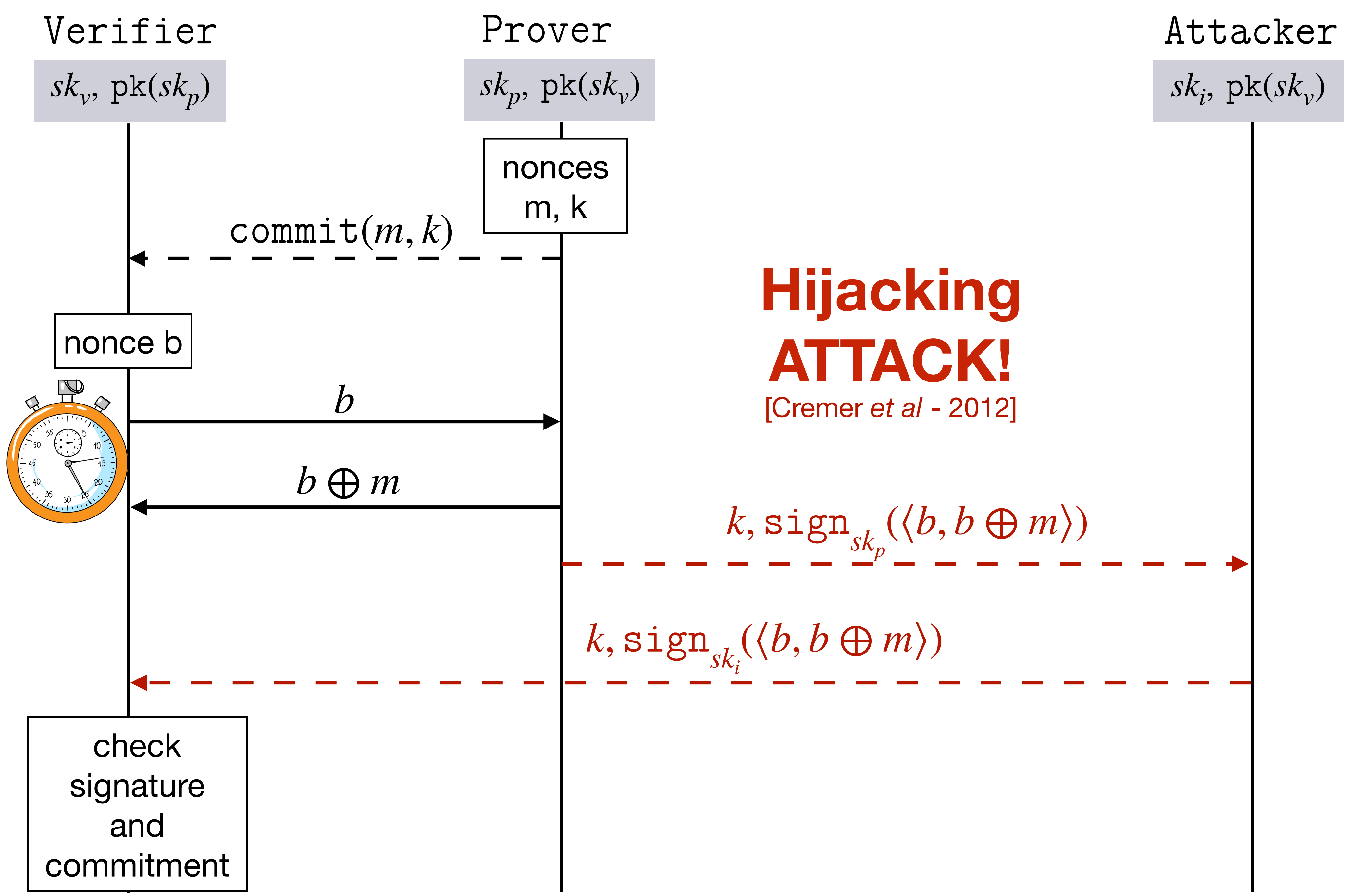
Untraceability

Authentication with physical proximity

Example: Brands and Chaum - 1993



Example: Brands and Chaum - 1993



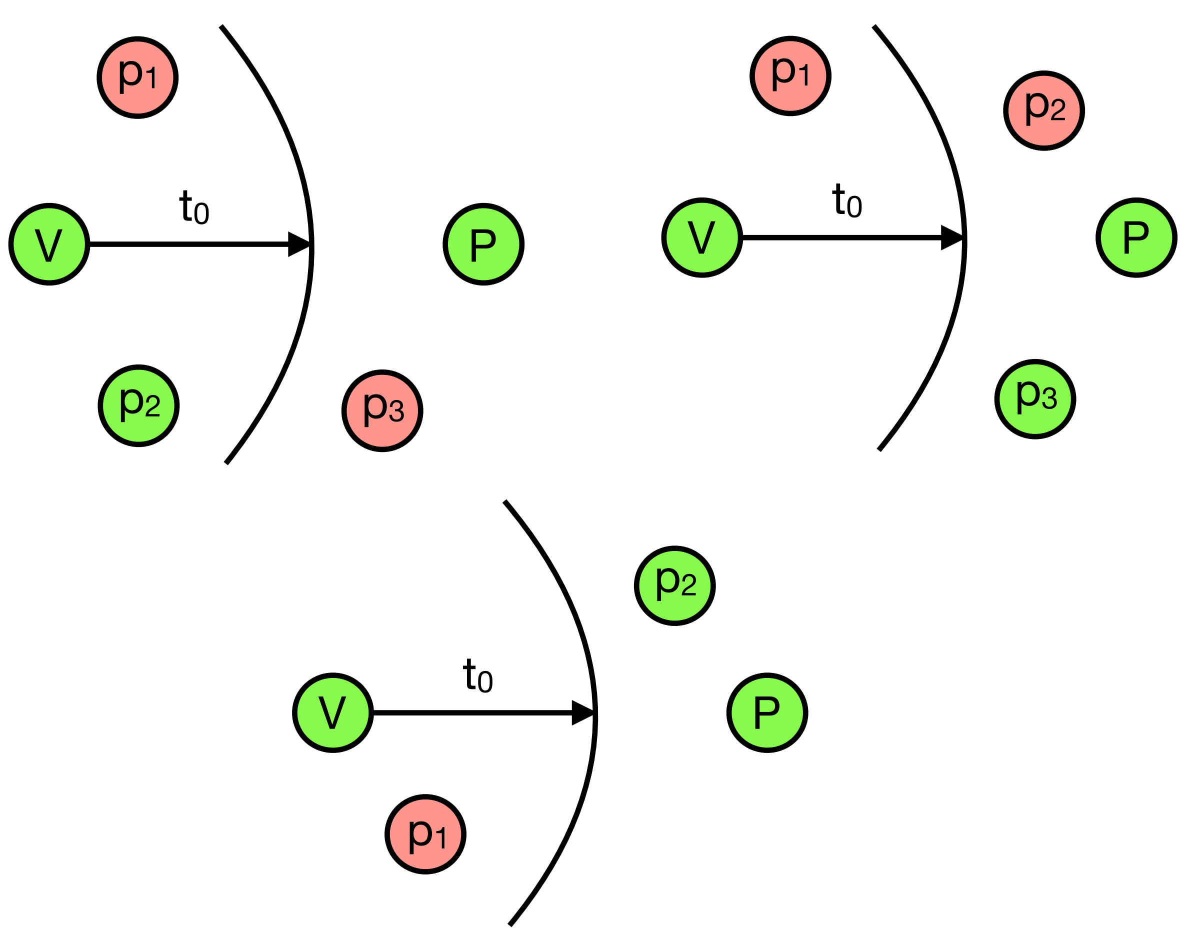
Classes of attacks

Mafia frauds

(or Man-in-the-Middle)

An attack in a topology such that:

- V is honest
- P is honest



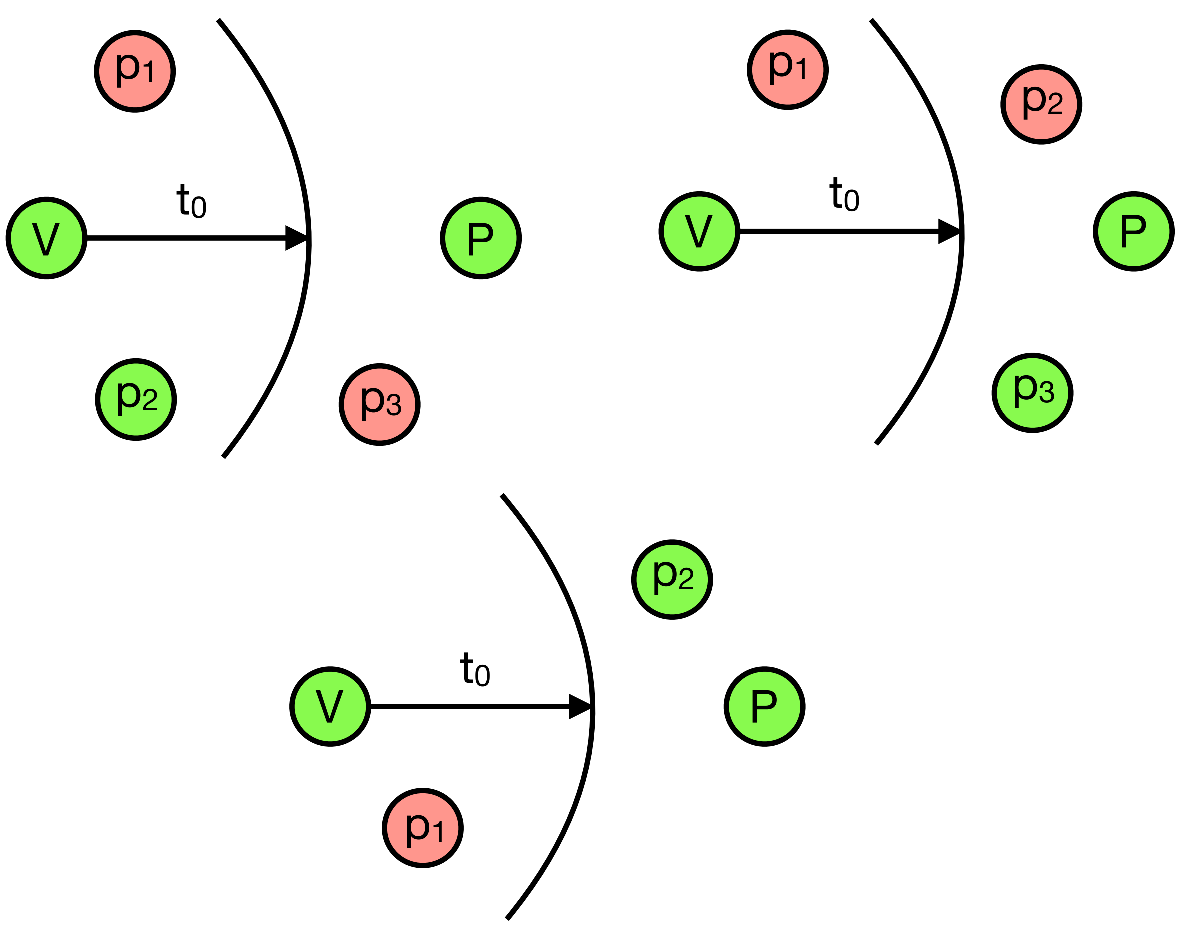
Classes of attacks

Mafia frauds

(or Man-in-the-Middle)

An attack in a topology such that:

- V is honest
- P is honest

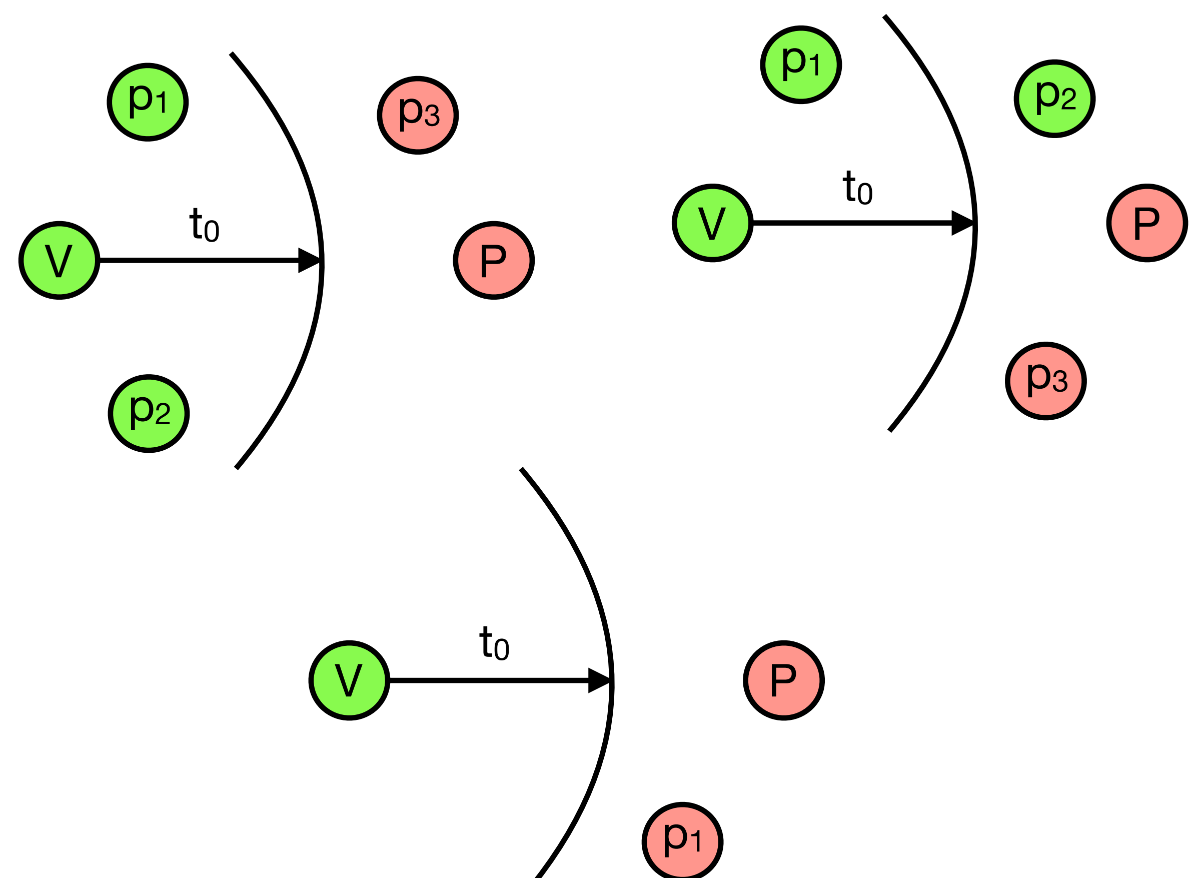


Distance hijacking

(or Man-in-the-Middle)

An attack in a topology such that:

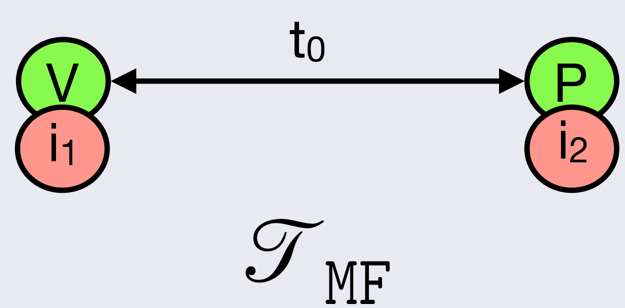
- V is honest
- P is **dishonest**
- No dishonest agents close to V

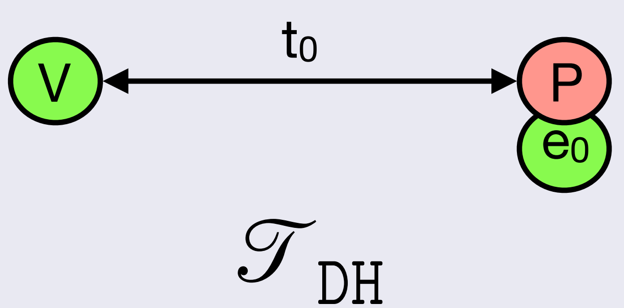


Contributions

Reduction results

Consider 1 topology is enough to prove Mafia fraud or Distance hijacking resistance!

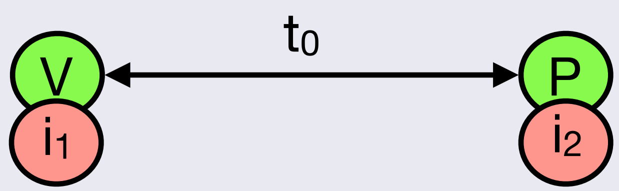




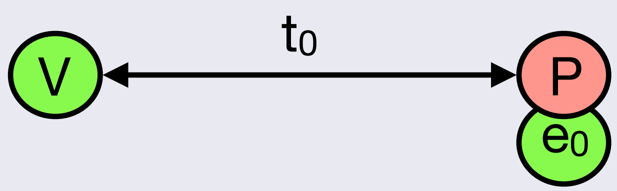
Contributions

Reduction results

Consider 1 topology is enough to prove Mafia fraud or Distance hijacking resistance!



$\mathscr{T}_{\mathtt{MF}}$



$\mathscr{T}_{\mathtt{DH}}$

Getting rid of topologies and time

- Modeling in ProVerif using phases
- Application to well-know DB protocols

Table of contents

Symbolic verification in a nutshell

Symbolic models:

(i) Terms: 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 but over a set of names $\mathcal{N}$ and a signature Σ given with either an equational theory $\mathtt{E}$ or a rewriting system.

Example

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

$$x \oplus 0 = x \qquad (x \oplus y) \oplus z = x \oplus (y \oplus z)$$
$$x \oplus x = 0 \qquad x \oplus y = y \oplus x$$

$$\mathtt{sdec}(\mathtt{senc}(x, y), y) \rightarrow x \qquad \mathtt{proj}_1(\mathtt{pair}(x, y)) \rightarrow x$$
$$\mathtt{proj}_2(\mathtt{pair}(x, y)) \rightarrow y$$

For example: $\mathtt{sdec}(\mathtt{senc}(n \oplus 0), k), k)$ is "equal" to n

Process algebra

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

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

Process algebra

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

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

Process algebra

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

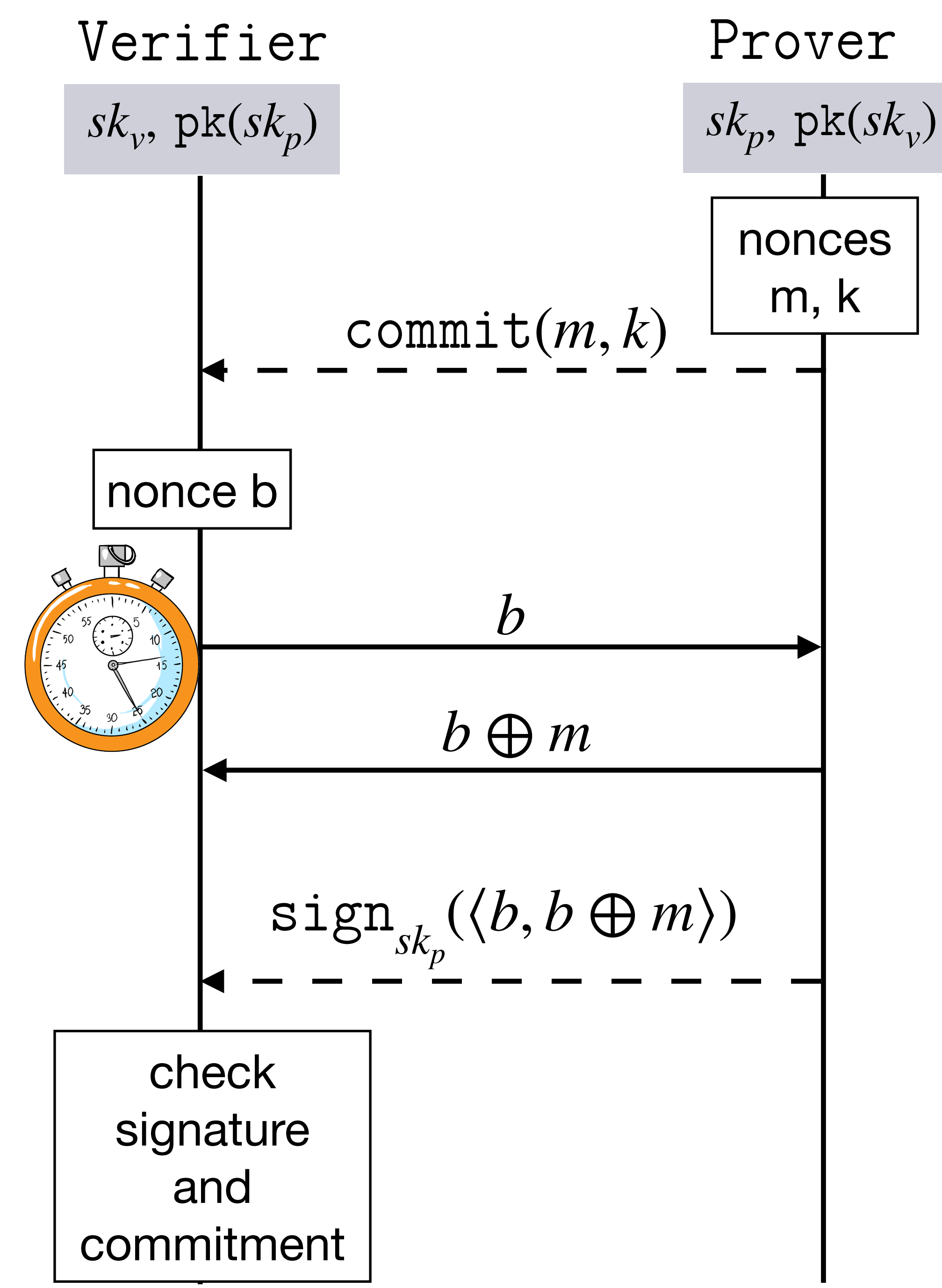
$P := 0$	null process
$\mid \ \mathtt{new}\ n\,.\,P$	name restriction
$\mid \ \mathtt{let}\ x = u\ \mathtt{in}\ P$	conditional declaration
$\mid \ \mathtt{out}(u)\,.\,P$	output
$\mid \ \mathtt{in}(x)\,.\,P$	input
$\mid \ \mathtt{in}^{<t}(x)\,.\,P$	guarded input
$\mid \ \mathtt{reset}\,.\,P$	personal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

Protocol

A protocol is a set of roles $(\Pi_1, \ldots, \Pi_k)$ describing the behavior of each honest agents.

Example: Brands and Chaum - 1993

$$
\begin{array}{l}
V(z_v, z_p) := \\
\quad \mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,. \\
\quad \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,. \\
\quad \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,. \\
\quad \mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in} \\
\quad \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{getmsg}(y_{\mathtt{sign}})\ \mathtt{in} \\
\quad \mathtt{let}\ y_{\mathtt{eq}} = \mathtt{eq}(\langle b, b \oplus y_m \rangle, y_{\mathtt{msg}})\ \mathtt{in} \\
\quad \mathtt{let}\ y_{\mathtt{eq'}} = \mathtt{eq}(b \oplus y_m, y_0)\ \mathtt{in} \\
\quad 0
\end{array}
$$



Example: Brands and Chaum - 1993

$V(z_v, z_p) :=$

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

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

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

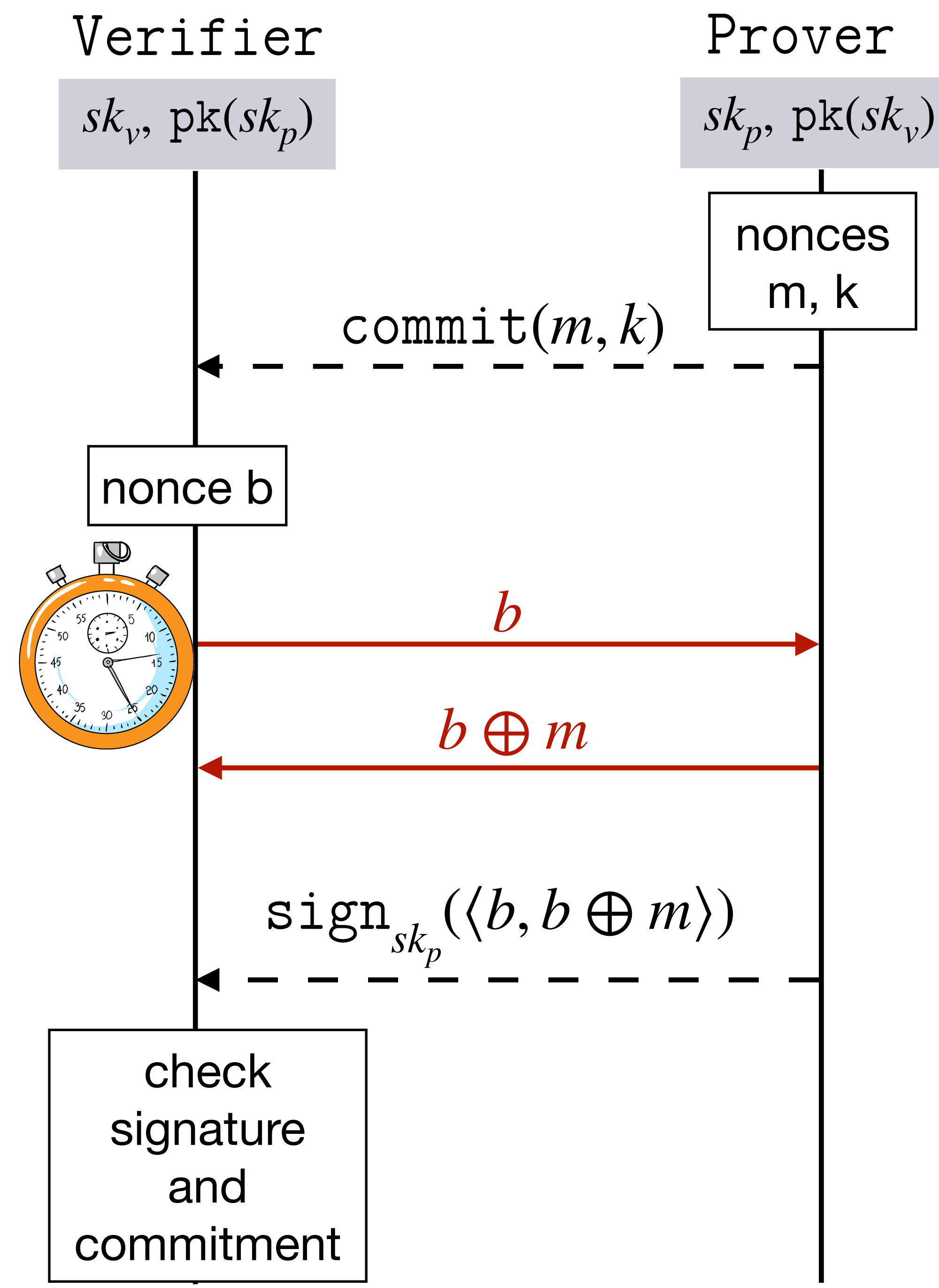
$\quad \mathtt{let}\ y_m = \mathsf{open}(y_c, y_k)\ \mathtt{in}$

$\quad \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{getmsg}(y_{\mathtt{sign}})\ \mathtt{in}$

$\quad \mathtt{let}\ y_{\mathsf{eq}} = \mathsf{eq}(\langle b, b \oplus y_m \rangle, y_{\mathtt{msg}})\ \mathtt{in}$

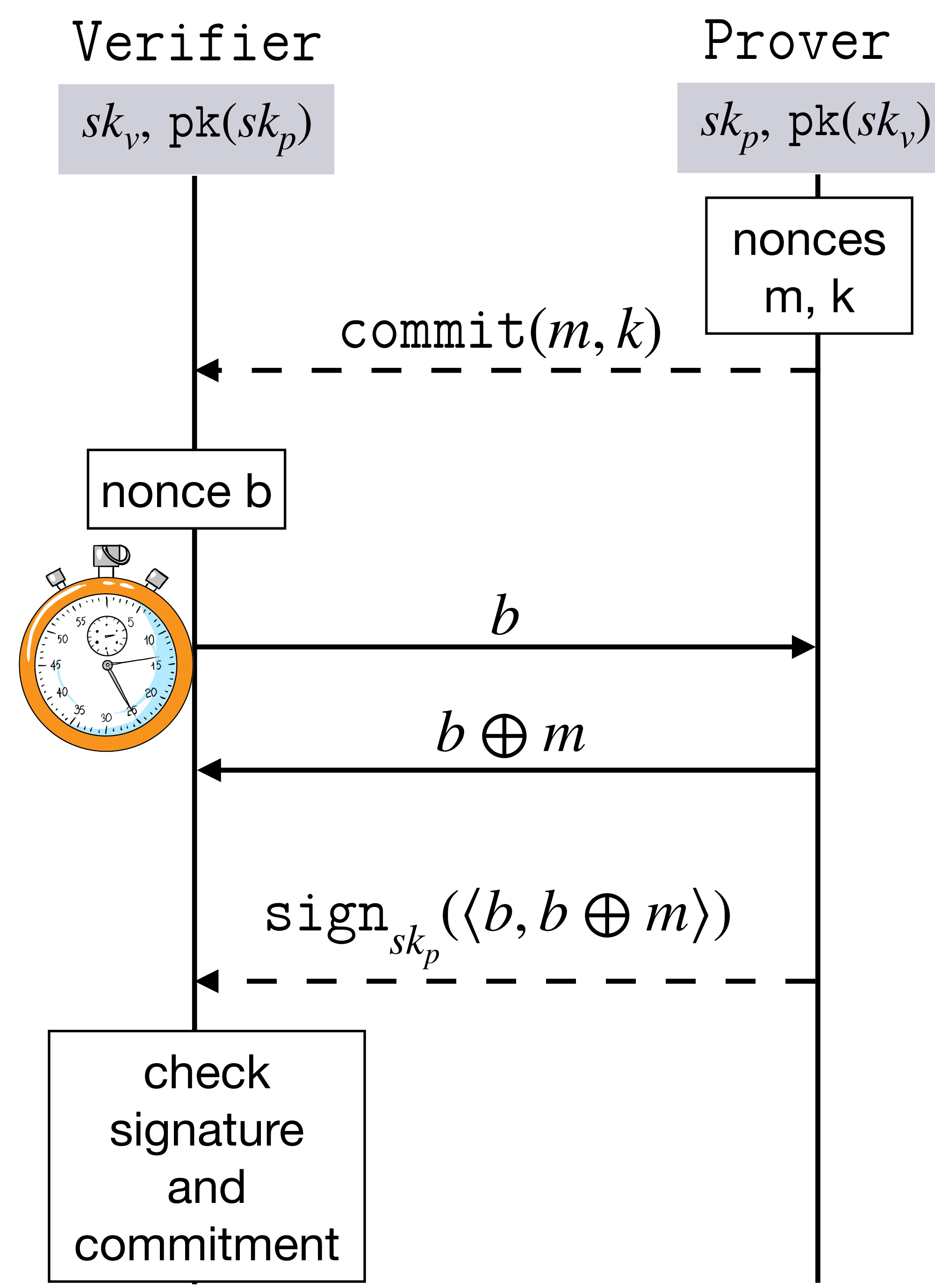
$\quad \mathtt{let}\ y_{\mathsf{eq}'} = \mathsf{eq}(b \oplus y_m, y_0)\ \mathtt{in}$

$\quad 0$



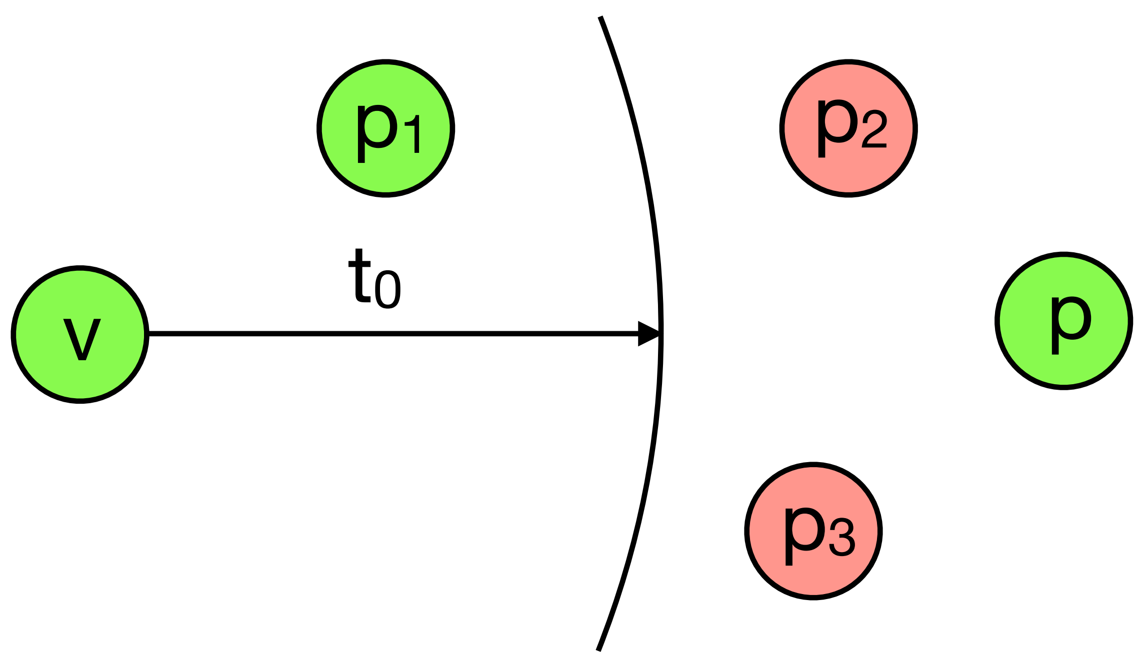
Example: Brands and Chaum - 1993

$$
\begin{aligned}
& V(z_v, z_p) := \\
& \qquad \mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,. \\
& \qquad \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,. \\
& \qquad \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,. \\
& \qquad \mathtt{let}\ y_m = \mathsf{open}(y_c, y_k)\ \mathtt{in} \\
& \qquad \mathtt{let}\ y_{\mathtt{msg}} = \mathsf{getmsg}(y_{\mathtt{sign}})\ \mathtt{in} \\
& \qquad \mathtt{let}\ y_{\mathsf{eq}} = \mathsf{eq}(\langle b, b \oplus y_m \rangle, y_{\mathtt{msg}})\ \mathtt{in} \\
& \qquad \mathtt{let}\ y_{\mathsf{eq}'} = \mathsf{eq}(b \oplus y_m, y_0)\ \mathtt{in} \\
& \qquad 0
\end{aligned}
$$



Topology

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

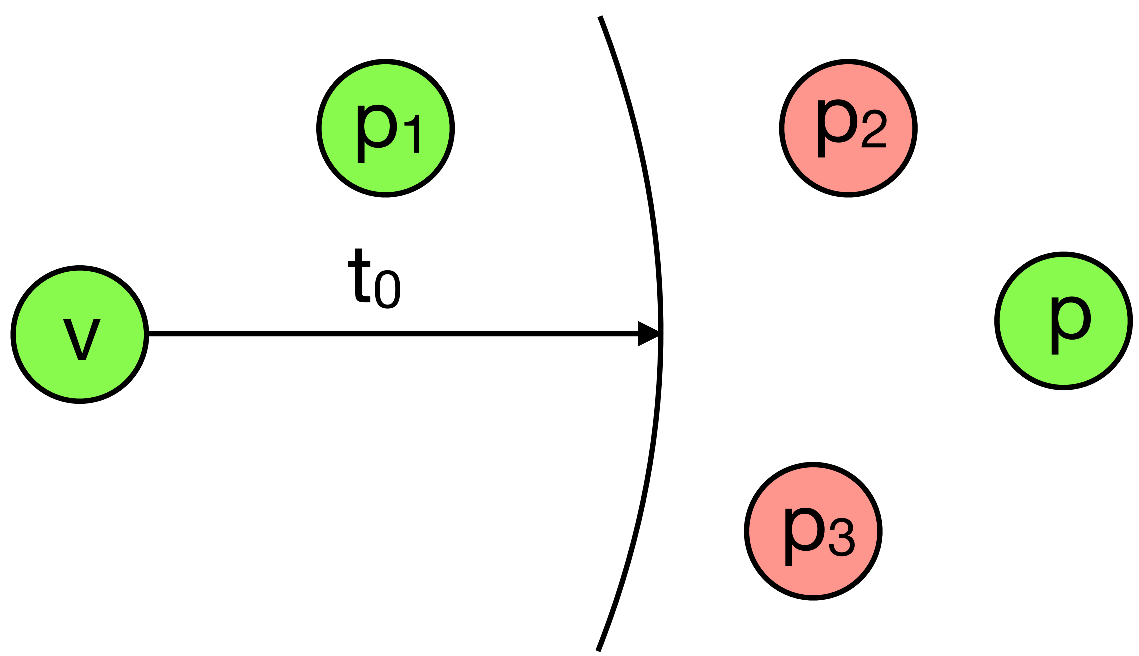


We define $\mathtt{Dist}_{\mathscr{T}}(a, b) = \dfrac{\|\mathtt{Loc}(a) - \mathtt{Loc}(b)\|}{c}$

Topology

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

agents



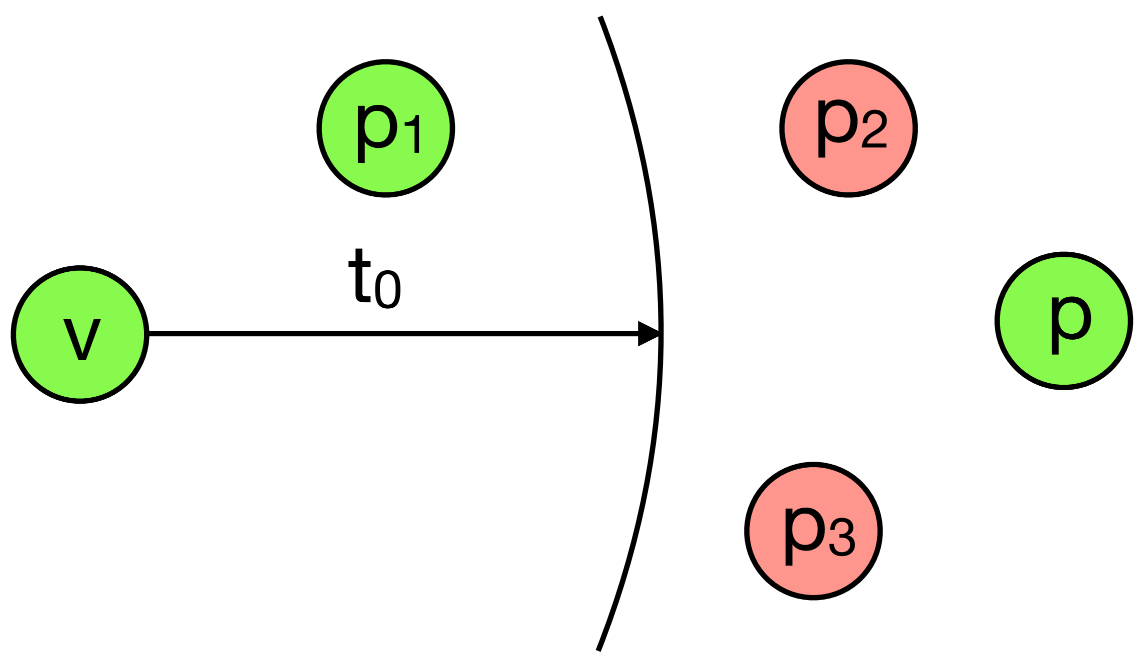
We define $\mathtt{Dist}_{\mathscr{T}}(a, b) = \dfrac{\|\mathtt{Loc}(a) - \mathtt{Loc}(b)\|}{c}$

Topology

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

agents

locations



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

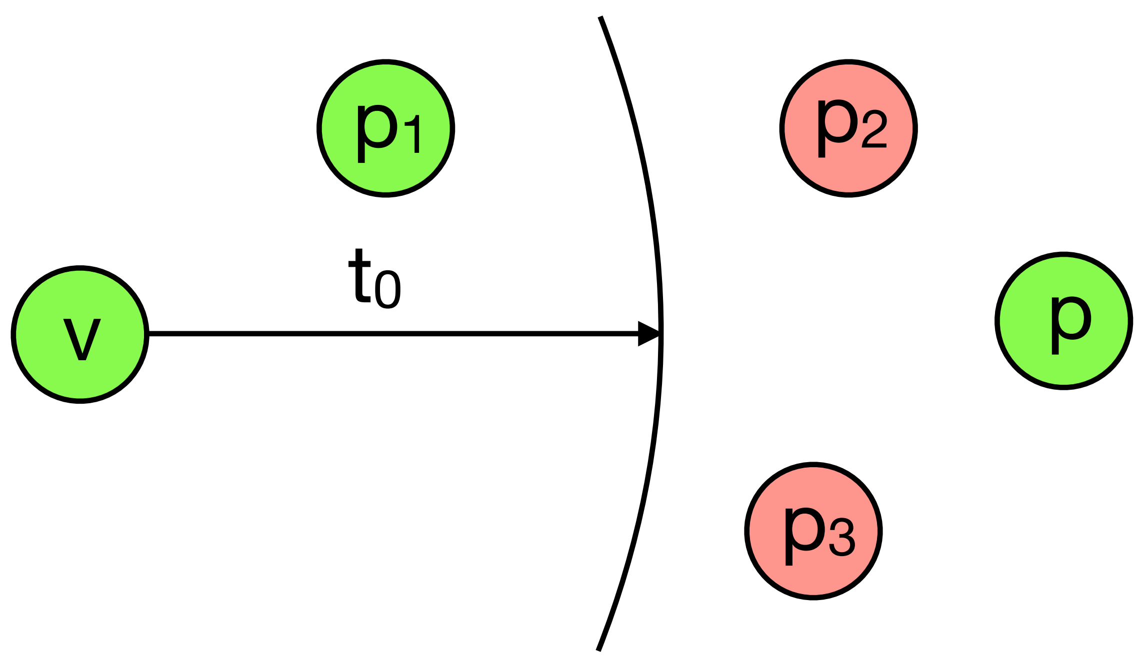
Topology

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

agents

locations

dishonest agents



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

Topology

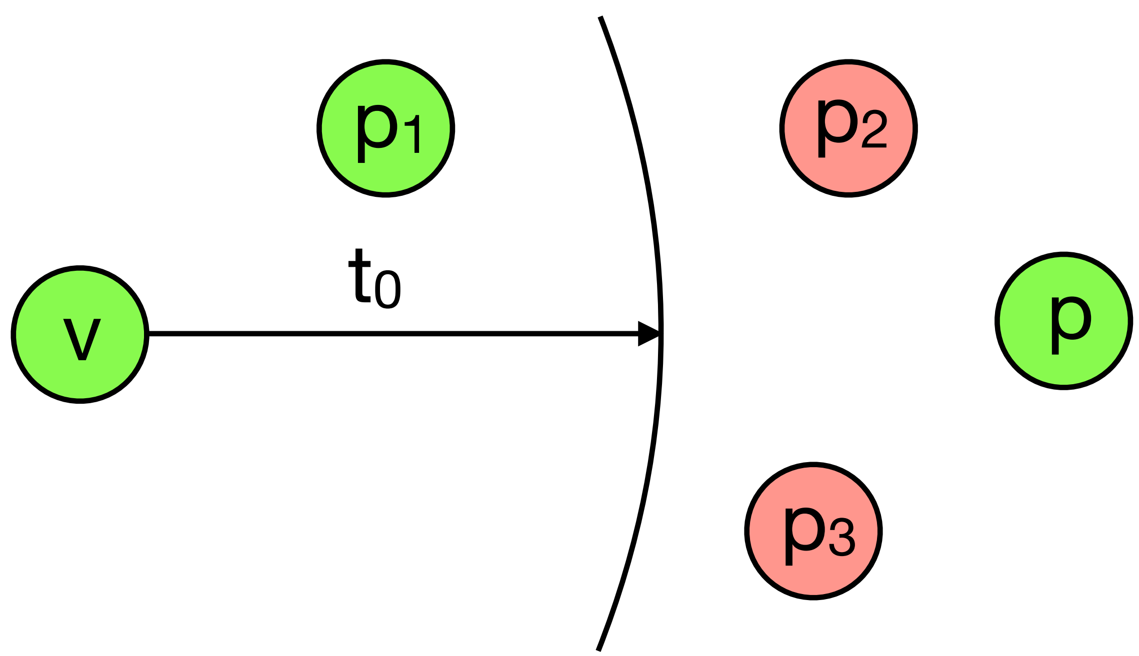
A **topology** is a tuple $\mathcal{T} = (\mathcal{A}, \mathtt{Loc}, \mathcal{M}, v, p)$.

agents

locations

dishonest agents

specific agents



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

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 $t_a \in \mathcal{R}_+$
- $\Phi = \{\mathtt{w}_1 \xrightarrow{a_1, t_1} m_1, \ldots, \mathtt{w}_n \xrightarrow{a_n, t_n} m_n\}$ is a frame
- $t \in \mathcal{R}_+$ is the global time

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

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

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

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

Configuration and semantics

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

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

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…

Security property: physical proximity

Mafia frauds (resp. Distance hijacking attacks)

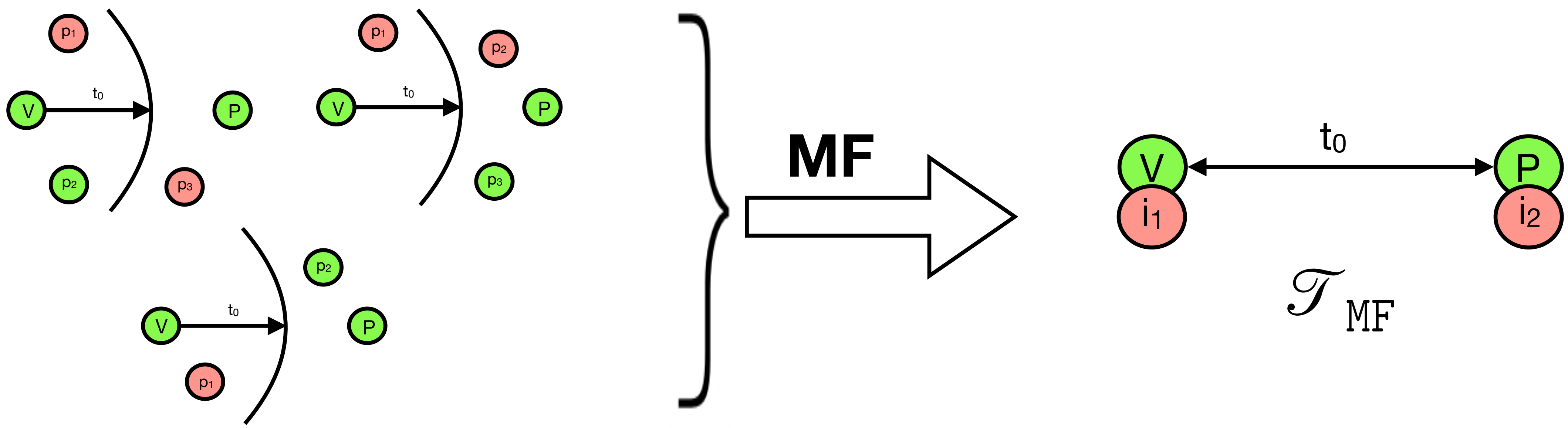
A protocol $\mathscr{P}_{\texttt{prox}}$ is resistant against Mafia frauds (resp. Distance hijacking attacks) if for all topologies $\mathscr{T} \in \mathscr{C}_{\texttt{MF}}$ (resp. $\mathscr{C}_{\texttt{DH}}$) and initial configuration K:

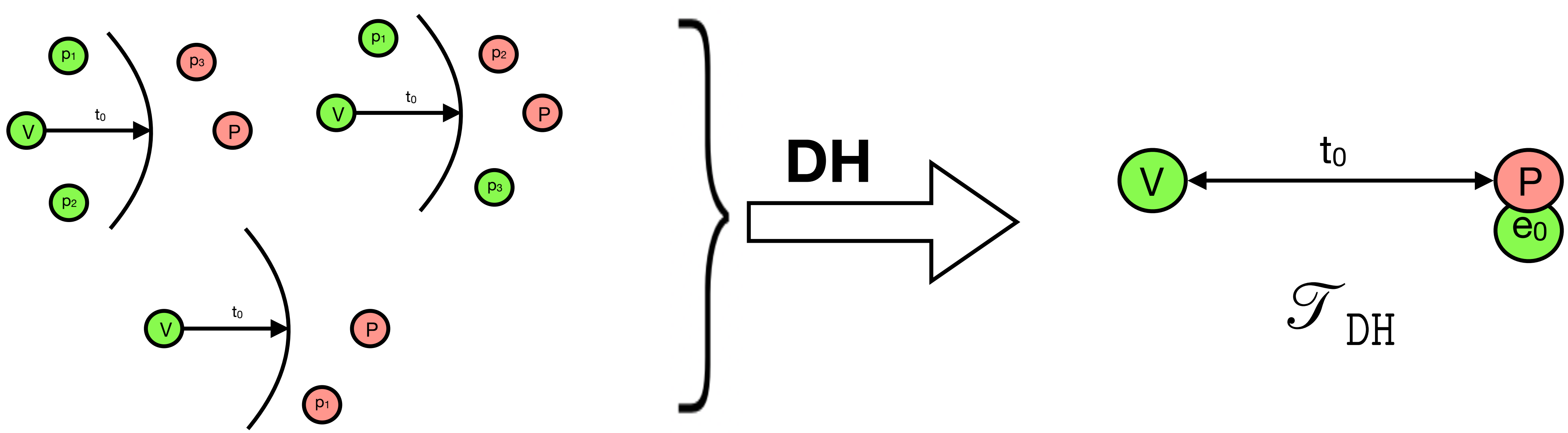
$$K \xrightarrow{tr} (\lfloor \texttt{end}(v_0, p_0) \rfloor_{v_0}^{t_{v_0}} \; ; \; \Phi \; ; \; t) \Rightarrow \texttt{Dist}_{\mathscr{T}}(v_0, p_0) < t_0$$

Table of contents

Reduction results

Only one topology is sufficient!



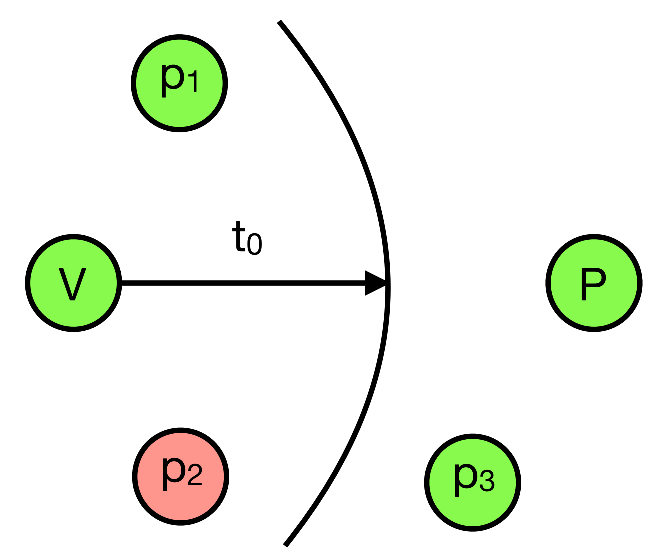


Mafia frauds

Theorem

Let $\mathscr{P}_{\texttt{prox}}$ be an **executable** protocol.
$\mathscr{P}_{\texttt{prox}}$ admits a Mafia fraud attack w.r.t. t_0-proximity, if and only if, there is an attack against t_0-proximity in the topology $\mathscr{T}_{\texttt{MF}}$.

Sketch of proof:



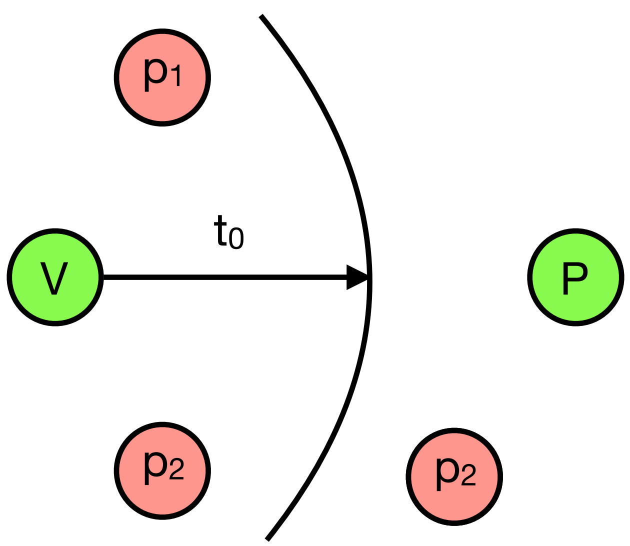
Mafia frauds

Theorem

Let $\mathscr{P}_{\mathtt{prox}}$ be an **executable** protocol.
$\mathscr{P}_{\mathtt{prox}}$ admits a Mafia fraud attack w.r.t. t_0-proximity, if and only if, there is an attack against t_0-proximity in the topology $\mathscr{T}_{\mathtt{MF}}$.

Sketch of proof:

1. The honest agents become malicious
   —> no executed processes



Mafia frauds

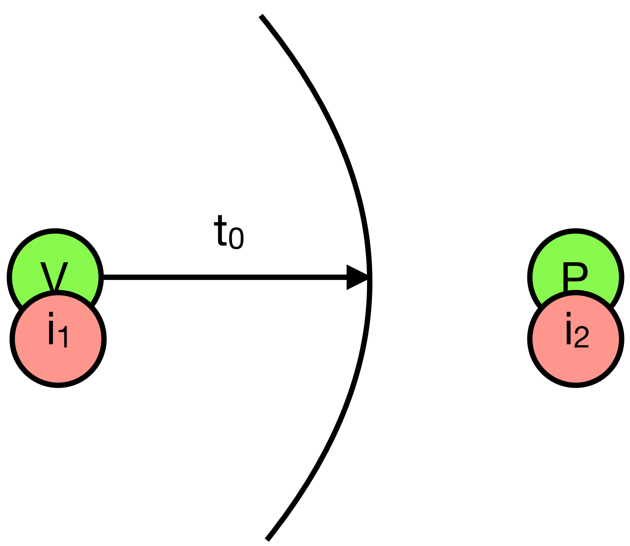
Theorem

Let $\mathscr{P}_{\texttt{prox}}$ be an **executable** protocol.
$\mathscr{P}_{\texttt{prox}}$ admits a Mafia fraud attack w.r.t. t_0-proximity, if and only if, there is an attack against t_0-proximity in the topology $\mathscr{T}_{\texttt{MF}}$.

Sketch of proof:

1. The honest agents become malicious
   —> no executed processes

2. We place them ideally
   (following [Nigam et al., ESORICS'16])



Mafia frauds

Theorem

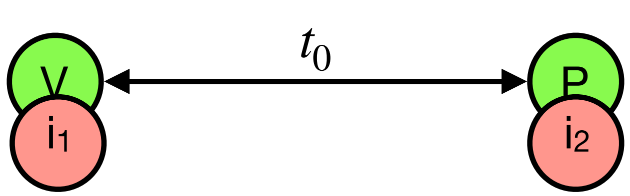
Let $\mathscr{P}_{\texttt{prox}}$ be an executable protocol.
$\mathscr{P}_{\texttt{prox}}$ admits a Mafia fraud attack w.r.t. t_0-proximity, if and only if, there is an attack against t_0-proximity in the topology $\mathscr{T}_{\texttt{MF}}$.

Sketch of proof:

1. The honest agents become malicious
   —> no executed processes

2. We place them ideally
   (following [Nigam et al., ESORICS'16])

3. We shorten the distance



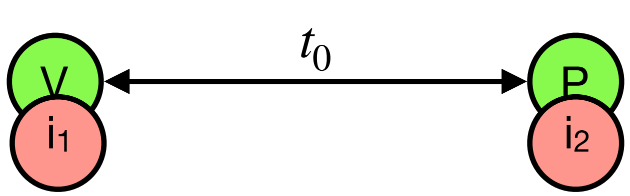
Mafia frauds

Theorem

Let $\mathscr{P}_{\texttt{prox}}$ be an **executable** protocol.
$\mathscr{P}_{\texttt{prox}}$ admits a Mafia fraud attack w.r.t. t_0-proximity, if and only if, there is an attack against t_0-proximity in the topology $\mathscr{T}_{\texttt{MF}}$.

Sketch of proof:

1. The honest agents become malicious
   —> no executed processes



2. We place them ideally
   (following [Nigam et al., ESORICS'16])

3. We shorten the distance

Remark. This proof cannot be adapted for distance hijacking attacks!

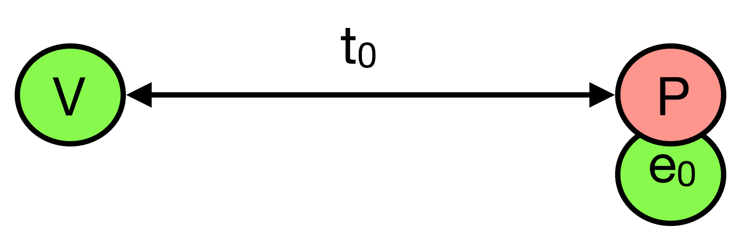
Distance hijacking attacks

Theorem

Let $\mathscr{P}_{\mathtt{prox}}$ be a protocol such that the Verifier role respects the following grammar:

$$\begin{array}{rl} P, Q := \mathtt{end}(z_0, z_1) & | \quad \mathtt{in}(x).P \quad | \quad \mathtt{let}\ x = v\ \mathtt{in}\ P \\ & | \quad \mathtt{new}\ n.P \quad | \quad \mathtt{out}(u).P \quad | \quad \mathtt{reset}.\mathtt{out}(u').\mathtt{in}^{<t}(x).P \end{array}$$

If $\mathscr{P}_{\mathtt{prox}}$ admits a Distance hijacking attack w.r.t. t_0-proximity, then $\overline{\mathscr{P}_{\mathtt{prox}}}$ admits an attack against t_0-proximity in the topology $\mathscr{T}_{\mathtt{DH}}$.



$\mathscr{T}_{\mathtt{DH}}$

In $\overline{\mathscr{P}_{\mathtt{prox}}}$ we only keel guards computed by v_0.

Table of contents

Getting rid of topologies and time

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

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

We propose a methodology to encode the two reduced topology in the ProVerif tool.

Overview of the encoding

➡ few assumptions on the protocol

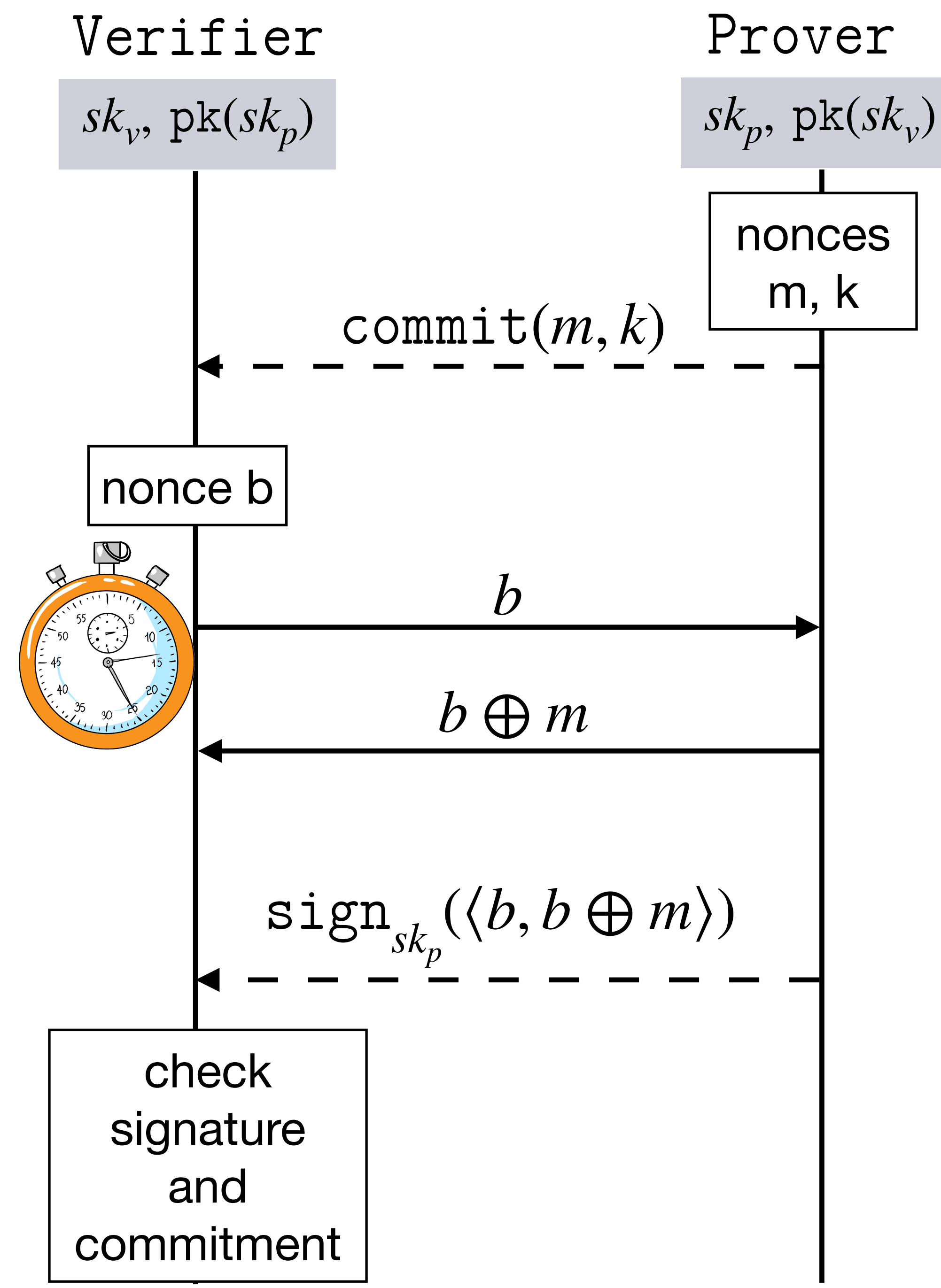
➡ it relies on the phases of ProVerif

e.g. in DB protocols:

- *Phase 0* ⟶ *slow initialization phase*
- *Phase 1* ⟶ *rapid phase*
- *Phase 2* ⟶ *slow verification phase*

Example: Brands and Chaum - 1993

$$
\begin{aligned}
&V(z_v, z_p) := \\
&\quad \mathtt{in}(y_c)\,.\,\mathtt{new}\ b\,. \\
&\quad \mathtt{reset}\,.\,\mathtt{out}(b)\,.\,\mathtt{in}^{<2\times t_0}(y_0)\,. \\
&\quad \mathtt{in}(y_k)\,.\,\mathtt{in}(y_{\mathtt{sign}})\,. \\
&\quad \mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in} \\
&\quad \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{getmsg}(y_{\mathtt{sign}})\ \mathtt{in} \\
&\quad \mathtt{let}\ y_{\mathtt{eq}} = \mathtt{eq}(\langle b, b \oplus y_m\rangle, y_{\mathtt{msg}})\ \mathtt{in} \\
&\quad \mathtt{let}\ y_{\mathtt{eq}'} = \mathtt{eq}(b \oplus y_m, y_0)\ \mathtt{in} \\
&\quad 0
\end{aligned}
$$



Example: Brands and Chaum - 1993

$\overline{V_0}(z_v, z_p) :=$

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

$\quad \mathtt{phase}\ 1.$

$\quad \mathtt{out}(b)\,.\,\mathtt{in}(y_0)\,.$

$\quad \mathtt{phase}\ 2.$

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

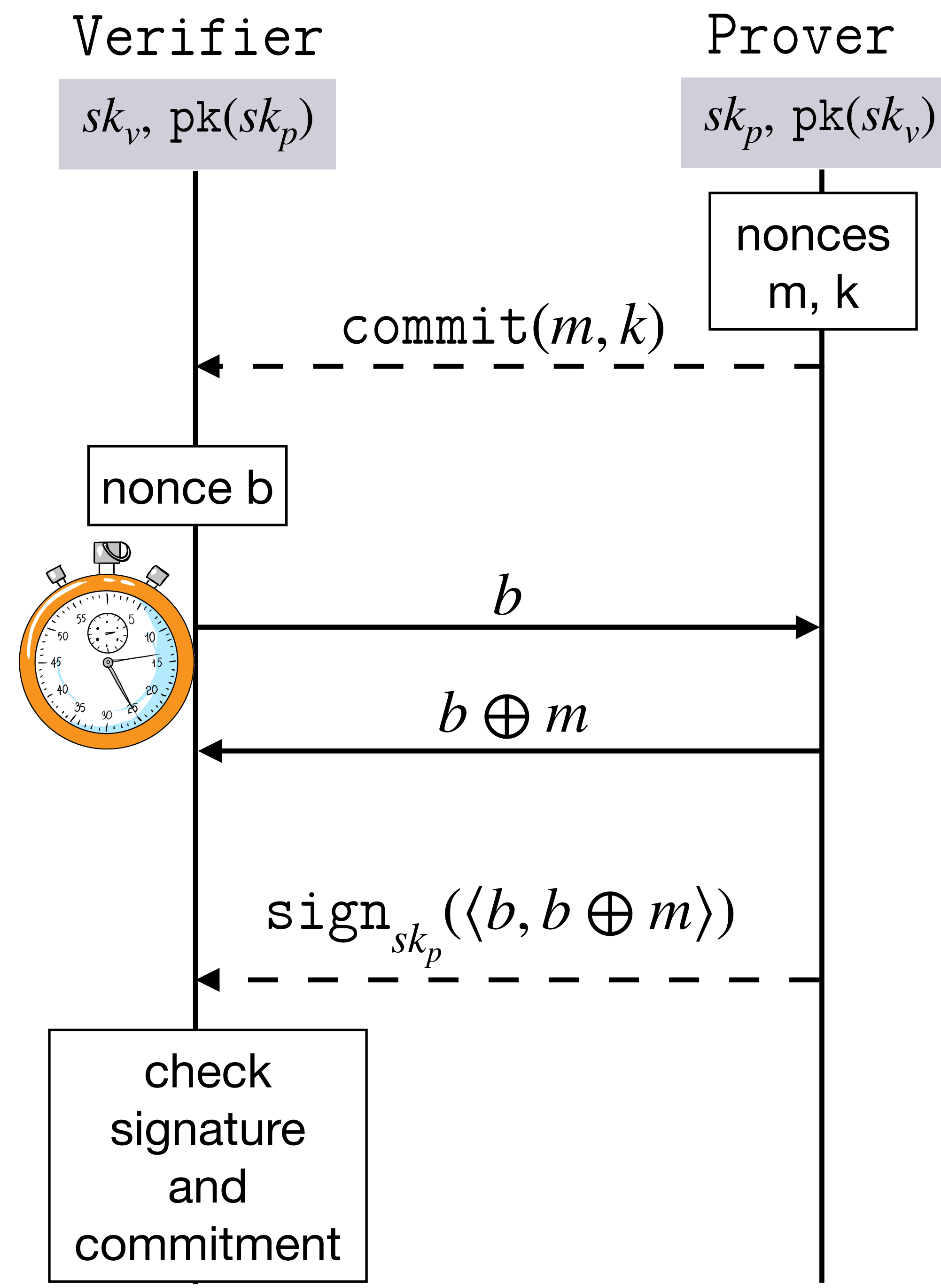
$\quad \mathtt{let}\ y_m = \mathtt{open}(y_c, y_k)\ \mathtt{in}$

$\quad \mathtt{let}\ y_{\mathtt{msg}} = \mathtt{getmsg}(y_{\mathtt{sign}})\ \mathtt{in}$

$\quad \mathtt{let}\ y_{\mathtt{eq}} = \mathtt{eq}(\langle b, b \oplus y_m\rangle, y_{\mathtt{msg}})\ \mathtt{in}$

$\quad \mathtt{let}\ y_{\mathtt{eq}'} = \mathtt{eq}(b \oplus y_m, y_0)\ \mathtt{in}$

$\quad 0$



Translation into ProVerif $\mathit{Transf}(\mathcal{T}, \mathcal{P}_{\mathtt{prox}}, t_0)$

Given a process *P* we define:

- $P^{<}$: all the possible ways of splitting *P* in the phases 0, 1 and 2
- $P^{\geq}$: all the possible ways of splitting *P* in the phases 0, and 2

$\mathit{Transf}(\mathcal{T}, \mathcal{P}_{\mathtt{prox}}, t_0)$ is the multiset of processes derived from $\mathcal{P}$ when applying:

- $.^{<}$ for all instantiated roles of $\mathcal{P}$ executed by agents close to v_0
- $.^{\geq}$ for all instantiated roles of $\mathcal{P}$ executed by agents far from v_0

Proposition

If $(\mathcal{P}_{\mathtt{prox}} \cup V_0)$ admits an attack w.r.t. t_0– proximity in $\mathcal{T}$ then $(\mathit{Transf}(\mathcal{T}, \mathcal{P}, t_0) \uplus \overline{V_0}(v_0, p_0)\ ;\ \Phi_{\mathtt{init}})$ admits an attack in ProVerif.

Inspired by [Chothia et al. - FC'15]

Case analysis - DB protocols

Protocols	**MF**	**DH**
Brands and Chaum	✓	×
Meadows *et al.* $(n_V \oplus n_P, P)$	✓	✓
Meadows *et al.* $(n_V, n_P \oplus P)$	✓	×
TREAD-Asymmetric	×	×
TREAD-Symmetric	✓	×
MAD (One-Way)	✓	×
Swiss-Knife	✓	✓
Munilla *et al.*	✓	✓
CRCS	✓	×
Hancke and Kuhn	✓	✓

(× : attack found, ✓ : proved secure)

- Coherent with the recent analysis done in *[Mauw et al. S&P'18]* using Tamarin
- We never obtained false attacks

Conclusion

We have adapted an existing symbolic model to take time into account.

We obtained **two reductions results** that reduce the number of relevant topologies that need to be studied from infinitely many to only 2.

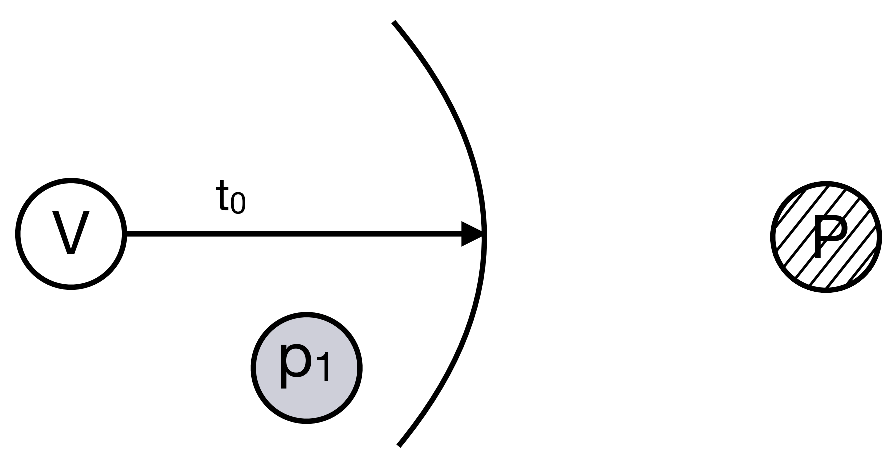
We provide a methodology to encode these reduced topologies into an **existing verification tool**, ProVerif, to be able to analyse well-known protocols w.r.t. authentication with physical proximity.

Future work

Goal: Establish reduction results to enable the verification for Terrorist frauds reusing existing tools.

Terrorist fraud

A remote dishonest prover cooperates with another dishonest agent, close to the verifier, to authenticates himself to the prover without giving any advantages for future attacks.



Challenge:

➡ Formally define the notion of semi-dishonest agents