Proving physical proximity using symbolic models

Alexandre Debant, Stéphanie Delaune, Cyrille Wielding

Univ Rennes - IRISA - CNRS

2018-07-08

Introduction

Cryptographic protocols

Distributed programs that use cryptographic primitives to ensure security properties.

secrecy authentication

integrity

untraceability

Introduction

Cryptographic protocols

Distributed programs that use cryptographic primitives to ensure security properties.

secrecy authentication

integrity

untraceability

physical proximity

Example: Brands and Chaum - 1993

V_0 $(\mathsf{pk}(P))$ | P $(\mathsf{sk}(P))$

`new` m, k

$$\xleftarrow{\quad commit(m, k) \quad}$$

`new` b

$$\xrightarrow{\quad b \quad}$$

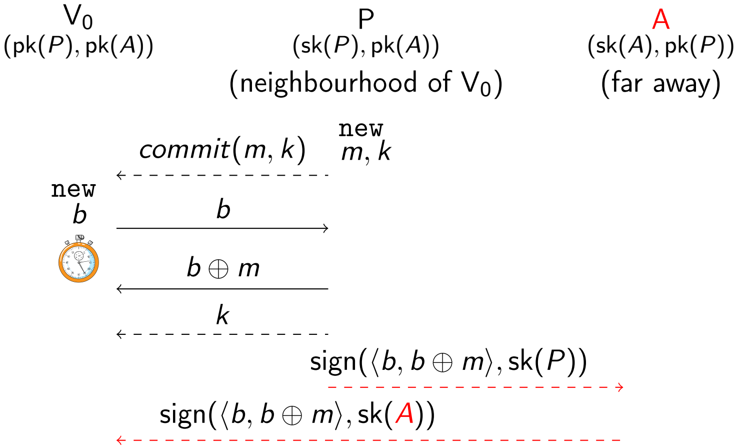
$$\xleftarrow{\quad b \oplus m \quad}$$

$$\xleftarrow{\quad k \quad}$$

$$\xleftarrow{\quad \mathsf{sign}(\langle b, b \oplus m \rangle, \mathsf{sk}(P)) \quad}$$

Brands and Chaum protocol

Example: Brands and Chaum - 1993

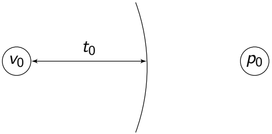


Attack against Brands and Chaum protocol

Classes of attacks

Mafia frauds - $\mathcal{C}_{\mathbf{MF}}$
(or Man-in-the-Middle)

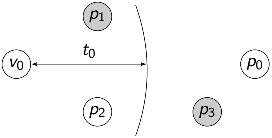
- V_0 is honest
- P_0 is honest



Classes of attacks

Mafia frauds - $\mathcal{C}_{\mathbf{MF}}$
(or Man-in-the-Middle)

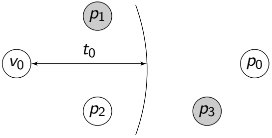
- V_0 is honest
- P_0 is honest



Classes of attacks

Mafia frauds - $\mathcal{C}_{\mathsf{MF}}$
(or Man-in-the-Middle)

- V_0 is honest
- P_0 is honest



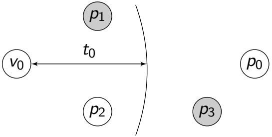
Distance hijacking - $\mathcal{C}_{\mathsf{DH}}$

- V_0 is honest
- P_0 is **dishonest**
- no dishonest agents close to V_0

Classes of attacks

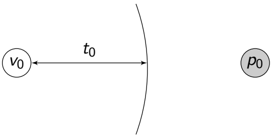
Mafia frauds - $\mathcal{C}_{\mathsf{MF}}$
(or Man-in-the-Middle)

- V_0 is honest
- P_0 is honest



Distance hijacking - $\mathcal{C}_{\mathsf{DH}}$

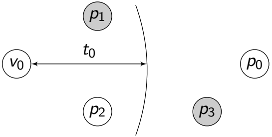
- V_0 is honest
- P_0 is **dishonest**
- no dishonest agents close to V_0



Classes of attacks

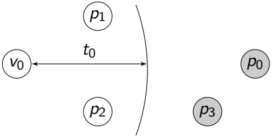
Mafia frauds - $\mathcal{C}_{\mathsf{MF}}$
(or Man-in-the-Middle)

- V_0 is honest
- P_0 is honest



Distance hijacking - $\mathcal{C}_{\mathsf{DH}}$

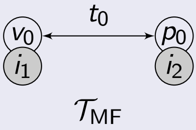
- V_0 is honest
- P_0 is **dishonest**
- no dishonest agents close to V_0

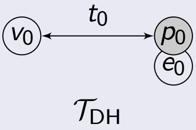


Contributions

Reduction results

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

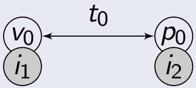




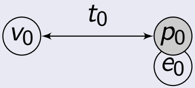
Contributions

Reduction results

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



$\mathcal{T}_{\mathsf{MF}}$



$\mathcal{T}_{\mathsf{DH}}$

Getting rid of topologies and time

- modelling in ProVerif using phases
- application to well-known DB protocols

Table of contents

Symbolic verification

Advantages:

- automated proofs
- efficient tools exist: **ProVerif**, Tamarin, Avispa...
- can express many security properties (authentication, secrecy, untraceability...)

But: cannot express physical proximity !
omniscient and ubiquitous attacker

How can we handle it?

Term algebra

Messages: terms built over a set of names $\mathcal{N}$ and a signature Σ given with either an equational theory E or a rewriting system

Example

- Names: $\mathcal{N} = \{a, n, k\}$
- Signature: $\Sigma = \{senc, sdec, pair, proj_1, proj_2, \oplus\}$

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

$$sdec(senc(x, y), y) \to x \qquad proj_1(pair(x, y)) \to x$$
$$proj_2(pair(x, y)) \to y$$

We have that: $sdec(senc(n \oplus 0), k), k)\!\downarrow\; =_{\mathsf{xor}} n$

Process algebra

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

$$\begin{array}{lcll} P & := & 0 & \text{null} \\ & \mid & \mathtt{new}\ n.P & \text{name restriction} \\ & \mid & \mathtt{let}\ x = u\ \mathtt{in}\ P & \text{conditional declaration} \\ & \mid & \mathtt{out}(u).P & \text{output} \\ & \mid & \mathtt{in}(x).P & \text{input} \end{array}$$

Process algebra

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

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

Process algebra

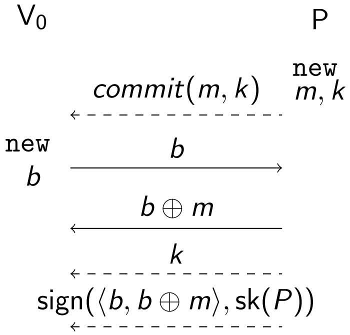
The role of an agent is described by a process following the grammar:

$$
\begin{array}{lcll}
P & := & 0 & \text{null} \\
 & | & \mathtt{new}\ n.P & \text{name restriction} \\
 & | & \mathtt{let}\ x = u\ \mathtt{in}\ P & \text{conditional declaration} \\
 & | & \mathtt{out}(u).P & \text{output} \\
 & | & \mathtt{in}(x).P & \text{input} \\
 & | & \mathtt{in}^{<t}(x).P & \text{guarded input} \\
 & | & \mathtt{reset}.P & \text{personal clock reset}
\end{array}
$$

Protocol

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

Example: Brands and Chaum - 1993



Brands and Chaum

Example: Brands and Chaum - 1993

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

V_0 | P

P: $\mathtt{new}$ m, k

$V_0 \xleftarrow{commit(m, k)} P$

V_0: $\mathtt{new}$ b

$V_0 \xrightarrow{b} P$

$V_0 \xleftarrow{b \oplus m} P$

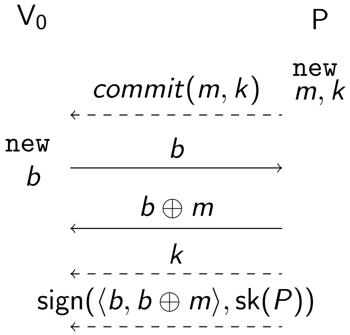
$V_0 \xleftarrow{k} P$

$V_0 \xleftarrow{\mathsf{sign}(\langle b, b \oplus m\rangle, \mathsf{sk}(P))} P$

Brands and Chaum

Example: Brands and Chaum - 1993

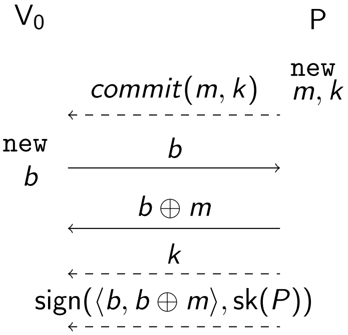
$$
\begin{aligned}
&V(z_V, z_P) := \\
&\quad \mathtt{in}(y_c).\mathtt{new}\ b. \\
&\quad \mathtt{reset.out}(b).\mathtt{in}^{<2\times t_0}(y_0). \\
&\quad \mathtt{in}(y_k).\mathtt{in}(y_{\text{sign}}). \\
&\quad \mathtt{let}\ y_m = \mathsf{open}(y_c, y_k)\ \mathtt{in} \\
&\quad \mathtt{let}\ y_{msg} =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



Brands and Chaum

Example: Brands and Chaum - 1993

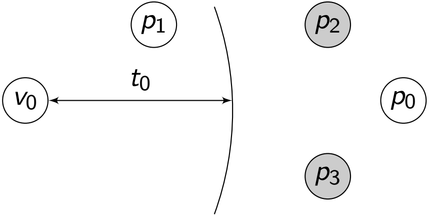
$$
\begin{array}{l}
V(z_V, z_P) := \\
\quad \mathtt{in}(y_c).\mathtt{new}\ b. \\
\quad \mathtt{reset}.\mathtt{out}(b).\mathtt{in}^{<2\times t_0}(y_0). \\
\quad \mathtt{in}(y_k).\mathtt{in}(y_{\mathsf{sign}}). \\
\quad \mathtt{let}\ y_m = \mathsf{open}(y_c, y_k)\ \mathtt{in} \\
\quad \mathtt{let}\ y_{msg}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



Brands and Chaum

Topology

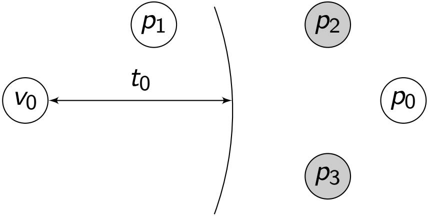
A **topology** is a tuple $\mathcal{T} = (\mathcal{A}, \mathrm{Loc}, \mathcal{M}, v_0, p_0)$.



Topology

A **topology** is a tuple $\mathcal{T} = (\mathcal{A}, \mathrm{Loc}, \mathcal{M}, v_0, p_0)$.

agents

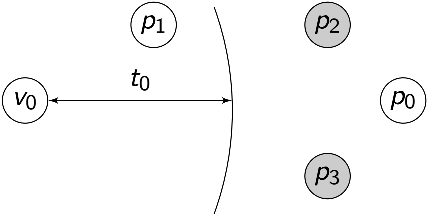


Topology

A **topology** is a tuple $\mathcal{T} = (\mathcal{A}, \mathrm{Loc}, \mathcal{M}, v_0, p_0)$.

agents

locations



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

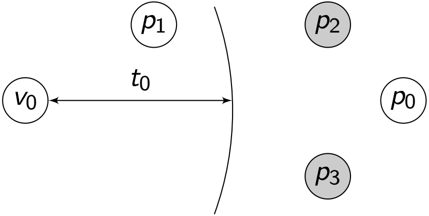
Topology

A **topology** is a tuple $\mathcal{T} = (\mathcal{A}, \mathrm{Loc}, \mathcal{M}, v_0, p_0)$.

agents

locations

dishonest agents

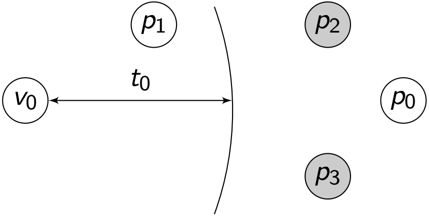


We define $\mathrm{Dist}_{\mathcal{T}}(a, b) = \frac{\|\mathrm{Loc}(a) - \mathrm{Loc}(b)\|}{c}$

Topology

A **topology** is a tuple $\mathcal{T} = (\mathcal{A}, \text{Loc}, \mathcal{M}, v_0, p_0)$.

agents
locations
dishonest agents
specific agents



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

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

OUT $\quad (\lfloor \mathtt{out}(u).P \rfloor_a^{t_a}) \uplus \mathcal{P}; \Phi; t) \xrightarrow{a, \mathtt{out}(u)}_{\mathcal{T}_0} (\lfloor P \rfloor_a^{t_a} \uplus \mathcal{P}; \Phi'; t)$
with $\Phi' = \Phi \cup \{\mathrm{w} \xrightarrow{a,t} u\}$

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

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

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

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

if $\exists b \in \mathcal{A}, t_b \in \mathcal{R}_+$ such that $t_b \leq t - \mathsf{Dist}_{\mathcal{T}}(b, a)$ 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 $\bigcup_{c \in \mathcal{A}} \lfloor \Phi \rfloor_c^{t_b - \mathsf{Dist}_{\mathcal{T}}(c,b)}$

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

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

if $\exists b \in \mathcal{A}, t_b \in \mathcal{R}_+$ such that $t_b \leq t - \mathsf{Dist}_{\mathcal{T}}(b, a)$ 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 $\bigcup_{c \in \mathcal{A}} \lfloor \Phi \rfloor_c^{t_b - \mathsf{Dist}_{\mathcal{T}}(c,b)}$

Moreover if $\star =< t_g$ then $t_a < t_g$.

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

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

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

Configuration and semantics

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

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

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

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

NEW, LET, RST $\quad \ldots$

Security property: physical proximity

t_0-proximity

A protocol $\mathcal{P}_{prox}$ ensures t_0-proximity w.r.t. a topology $\mathcal{T} = (\mathcal{A}, \mathsf{Loc}, \mathcal{M}, v_0, p_0)$ and a configuration K if:

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

Security property: physical proximity

t_0-proximity

A protocol $\mathcal{P}_{prox}$ ensures t_0-proximity w.r.t. a topology $\mathcal{T} = (\mathcal{A}, \mathrm{Loc}, \mathcal{M}, v_0, p_0)$ and a configuration K if:

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

Mafia frauds (resp. Distance hijacking attacks)

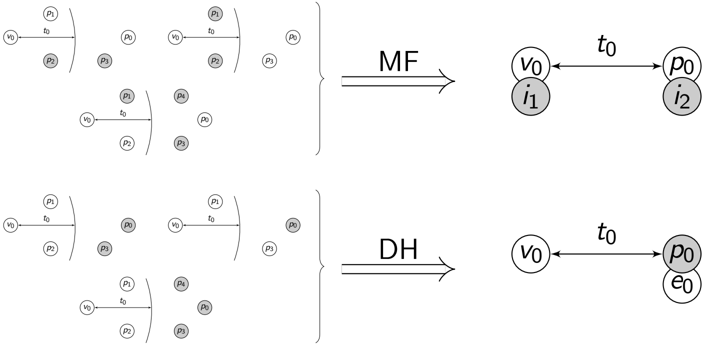
A protocol $\mathcal{P}_{prox}$ is resistant against Mafia frauds (resp. Distance hijacking attacks) if for all topologies $\mathcal{T} \in \mathcal{C}_{\mathrm{MF}}$ (resp. $\mathcal{C}_{\mathrm{DH}}$) and initial configurations K:

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

Table of contents

Reduction results

Only one topology is sufficient !

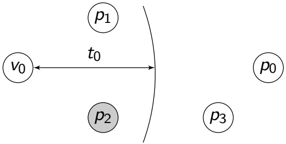


Mafia fraud attacks

Theorem

Let $\mathcal{P}_{prox}$ be an **executable** protocol.
$\mathcal{P}_{\mathsf{prox}}$ 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 $\mathcal{T}_{\mathsf{MF}}$.

Sketch of proof:



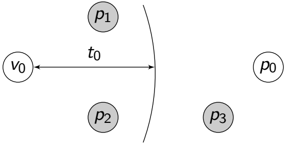
Mafia fraud attacks

Theorem

Let $\mathcal{P}_{prox}$ be an **executable** protocol.
$\mathcal{P}_{\mathsf{prox}}$ 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 $\mathcal{T}_{\mathsf{MF}}$.

Sketch of proof:

1. the honest agents become malicious –> no executed processes



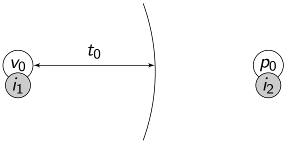
Mafia fraud attacks

Theorem

Let $\mathcal{P}_{prox}$ be an **executable** protocol.
$\mathcal{P}_{\mathsf{prox}}$ 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 $\mathcal{T}_{\mathsf{MF}}$.

Sketch of proof:

1. the honest agents become malicious –> no executed processes
2. we place them ideally [Nigam *et. al.*, 16]



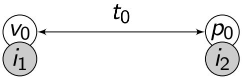
Mafia fraud attacks

Theorem

Let $\mathcal{P}_{prox}$ be an **executable** protocol.
$\mathcal{P}_{\mathrm{prox}}$ 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 $\mathcal{T}_{\mathrm{MF}}$.

Sketch of proof:

1. the honest agents become malicious –> no executed processes
2. we place them ideally [Nigam *et. al.*, 16]
3. we shorten the distance



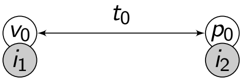
Mafia fraud attacks

Theorem

Let $\mathcal{P}_{prox}$ be an **executable** protocol.
$\mathcal{P}_{\mathsf{prox}}$ 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 $\mathcal{T}_{\mathsf{MF}}$.

Sketch of proof:

1. the honest agents become malicious –> no executed processes
2. we place them ideally [Nigam *et. al.*, 16]
3. we shorten the distance



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

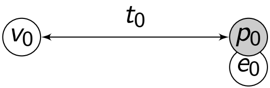
Distance hijacking attacks

Theorem

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

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

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



$\mathcal{T}_{\mathrm{DH}}$

In $\overline{\mathcal{P}_{\mathrm{prox}}}$ we only keep guards computed by v_0.

Table of contents

ProVerif [Blanchet, 01]

ProVerif is a verifier tool for cryptographic protocols.

http://proverif.inria.fr/

- fully automated proofs
- handles an unbounded number of sessions
- can model protocols defined by phases (*e.g.* e-voting)
  - $\rightarrow$ $(\texttt{phase}\ i).P$ represents a process P that can only be executed in phase i

ProVerif [Blanchet, 01]

ProVerif is a verifier tool for cryptographic protocols.

http://proverif.inria.fr/

- fully automated proofs
- handles an unbounded number of sessions
- can model protocols defined by phases (*e.g.* e-voting)
  - $\rightarrow$ $(\mathtt{phase}\ i).P$ represents a process P that can only be executed in phase i

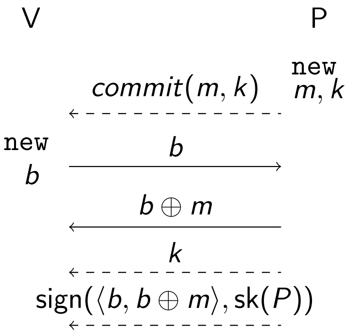
Phases in DB protocols:

- Phase 0 $\rightarrow$ slow initialisation phase
- Phase 1 $\rightarrow$ rapid phase
- Phase 2 $\rightarrow$ slow verification phase

Translation into ProVerif

$\mathit{Transf}(\mathcal{T}, \mathcal{P}, t_0)$

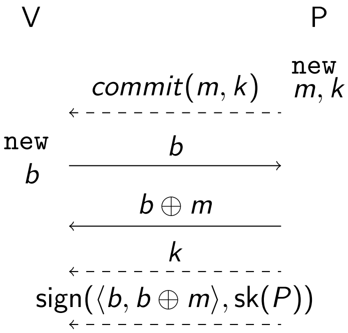
$$
\begin{array}{l}
V_0(v_0, p_0) := \\
\quad \texttt{in}(y_c).\texttt{new}\ b. \\
\quad \texttt{reset}.\texttt{out}(b).\texttt{in}^{<2\times t_0}(y_0). \\
\quad \texttt{in}(y_k).\texttt{in}(y_{\mathsf{sign}}). \\
\quad \texttt{let}\ y_m = \mathsf{open}(y_c, y_k)\ \texttt{in} \\
\quad \texttt{let}\ y_{msg} = \mathsf{getmsg}(y_{\mathsf{sign}})\ \texttt{in} \\
\quad \texttt{let}\ y_{\mathsf{check}} = \mathsf{check}(y_{\mathsf{sign}}, \mathsf{vk}(z_P))\ \texttt{in} \\
\quad \texttt{let}\ y_{\mathsf{eq}} = \mathsf{eq}(\langle b, b \oplus y_m\rangle, y_{msg})\ \texttt{in} \\
\quad \mathsf{end}(z_V, z_P). \\
\quad 0
\end{array}
$$



Brands and Chaum

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

$\overline{V_0}(v_0, p_0) :=$
$\quad \mathtt{in}(y_c).\mathtt{new}\ b.$
$\quad \mathtt{phase}\ 1.$
$\quad \mathtt{out}(b).\mathtt{in}(y_0).$
$\quad \mathtt{phase}\ 2.$
$\quad \mathtt{in}(y_k).\mathtt{in}(y_{\mathsf{sign}}).$
$\quad \mathtt{let}\ y_m = \mathsf{open}(y_c, y_k)\ \mathtt{in}$
$\quad \mathtt{let}\ y_{msg} = \mathsf{getmsg}(y_{\mathsf{sign}})\ \mathtt{in}$
$\quad \mathtt{let}\ y_{\mathsf{check}} = \mathsf{check}(y_{\mathsf{sign}}, \mathsf{vk}(z_P))\ \mathtt{in}$
$\quad \mathtt{let}\ y_{\mathsf{eq}} = \mathsf{eq}(\langle b, b \oplus y_m\rangle, y_{msg})\ \mathtt{in}$
$\quad \mathsf{end}(z_V, z_P).$
$\quad 0$



Brands and Chaum

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

Given a process P we define:

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

$Transf(\mathcal{T}, \mathcal{P}, t_{\text{prox}})$ is the multiset of processes derived from $\mathcal{P}$ when applying:

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

Proposition

If $(\mathcal{P}_{\text{prox}} \cup V_0)$ admits an attack w.r.t. t_0-proximity in $\mathcal{T}$ then $(Transf(\mathcal{T}, \mathcal{P}, t_0) \uplus \overline{V_0}(v_0, p_0); \Phi_{init}; 0)$ admits an attack in ProVerif.

Case analysis - DB protocols

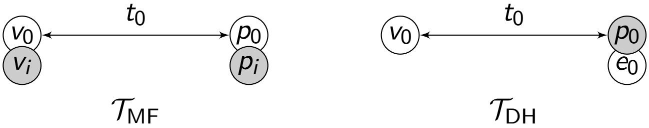
Protocols	MF	DH
Brands and Chaum	✓	×
Meadows *et al.* $(n_V \oplus n_P, P)$	✓	✓
Meadows *et al.* $(n_V, n_P \oplus P)$	✓	×
TREAD-Asymmetric	×	×
TREAD-Symmetric	✓	×
MAD (One-Way)	✓	×
Swiss-Knife	✓	✓
Munilla *et al.*	✓	✓
CRCS	✓	×
Hancke and Kuhn	✓	✓

(×: attack found, ✓: proved secure)
Coherent with the formal analysis recently done by Mauw *et. al.* using Tamarin

Conclusion

We have adapted the standard applied Pi-Calculus to take into account time and locations.

We obtained **two reduction results** that reduce the number of relevant topologies that need to be studied to only 2.



We provide a solution to prove t_0-proximity using a **usual verification tool**, ProVerif, and we applied it to analyse well-known protocols.

Future work

⇒ Define a more precise notion of time.

⇒ Take into account **Terrorist frauds**:

Terrorist frauds

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.