# Proving physical proximity using symbolic models

Alexandre Debant
Univ Rennes, CNRS, IRISA, France
Email: Alexandre.Debant@irisa.fr

Stéphanie Delaune
Univ Rennes, CNRS, IRISA, France
Email: Stephanie.Delaune@irisa.fr

Cyrille Wiedling
Univ Rennes, CNRS, IRISA, France
Email: Cyrille.Wiedling@inria.fr

***Abstract*—For many modern applications like *e.g.* contactless payment, and keyless systems, ensuring physical proximity is a security goal of paramount importance. Formal methods have proved their usefulness when analysing standard security protocols. However, existing results and tools do not apply to *e.g.* distance bounding that aims to ensure physical proximity between two entities. This is due in particular to the fact that existing models do not represent in a faithful way the locations of the participants, and the fact that transmission of messages takes time.**

**In this paper, we propose several reduction results: when looking for an attack, it is actually sufficient to consider a simple scenario involving at most four participants located at some specific locations. An interesting consequence of our reduction results is that it allows one to reuse ProVerif, an automated tool developed for analysing standard security protocols. As an application, we analyse several distance bounding protocols, as well as a contactless payment protocol, and our experimental results confirm existing results on these protocols.**

***Index Terms*—Formal verification, security protocols, symbolic models, distance bounding protocols**

## I. Introduction

The shrinking size of microprocessors as well as the ubiquity of wireless communication have led to the proliferation of portable computing devices with novel security requirements. Whereas traditional security protocols achieve their security goals relying solely on cryptographic primitives like encryptions and hash functions, this is not the case anymore for many modern applications like *e.g.* contactless payment. Actually, a typical attack against these devices is the so-called relay attack, as demonstrated for EMV in [1]. Such an attack allows a malicious participant to relay communications between a victim's card (possibly inside a wallet) and a genuine terminal so that the victim's card, even if it is far away from the terminal, will pay the transaction. Due to the contactless nature of most of our communication, obtaining reliable information regarding physical proximity is of paramount importance and specific protocols, namely distance bounding protocols, were proposed to achieve this specific goal [2], [3]. They typically take into account the round trip time of messages and the transmission velocity to infer an upper bound of the distance between two participants.

This work has been partially supported by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement No 714955-POPSTAR).

In the context of standard security protocols, such as key establishment protocols, formal methods have proved their usefulness for providing security guarantees or detecting attacks. The purpose of formal verification is to provide rigorous frameworks and techniques to analyse protocols and find their flaws. For example, a flaw has been discovered in the Single-Sign-On protocol used *e.g.* by Google Apps [4]. This flaw has been found when analysing the protocol using formal symbolic methods, abstracting messages by a term algebra and using the Avantssar validation platform [5]. The techniques used in symbolic models have become mature and several verification tools are nowadays available [5]–[7].

However, protocols whose security relies on constraints from the physical world fall outside the scope of traditional symbolic models that are based on the omniscient attacker who controls the entire network, and who can for instance relay messages without introducing any delay. Due to the lack of formal symbolic models, distance bounding protocols are only analysed so far with respect to some specific attack types known as *e.g.* mafia fraud, and terrorist fraud. Recently, another type of attack, namely distance hijacking [8], has been discovered, and many protocols have been shown to be vulnerable to this new type of attacks. Following [9], [10], and more recently [11], our aim is to bridge the gap between informal approaches currently used to analyse these protocols and the formal approaches already used for analysing traditional security protocols.

*Our contributions.* To model timed protocols as well as the notion of physical proximity, we first propose a calculus in which communications are subject to physical restriction applying to honest agents and attackers. An attacker can only intercept messages at his location, and attackers can not instantaneously exchange their knowledge: transmitting messages takes time. This models reality, where the attackers' ability to observe and communicate messages depends on their locations. Then, our main contribution is to provide reduction results in the spirit of the one obtained in [12] for traditional protocols: if there is an attack, then there is one considering only few participants at some specific locations. The results slightly differ depending on the type of attacks we consider (mafia fraud or hijacking attack). Each result allows one to reduce the number of topologies to be considered from infinitely many to only one (involving at most 4 participants including the malicious ones). Our results hold in a rather general setting: we consider arbitrary cryptographic primitives that can be expressed using rewriting rules modulo an equational theory.

An interesting consequence of our reduction results is that it allows one to reuse techniques and tools developed for standard security protocols. Actually, we show how to encode these simple topologies, as well as the timing constraints, relying on the phase mechanism available in ProVerif. As an application, we analyse several distance bounding protocols, and a contactless payment protocol [1].

All files related to our case studies are available at [13], while omitted proofs are available in the full version of this paper [14].

*Related work.* Until recently, most distance bounding protocols have been analysed without a formal approach. Recent efforts have been made on proving security of distance bounding protocols. For instance, in 2001, Avoine *et al.* [15] proposed a framework in which many protocols have been analysed and compared in a unified manner [16]. A rather general model has been proposed by Boureanu *et al.* in [17]. This computational model captures all the classical types of attacks and generalises them enabling attackers to interact with many provers and verifiers. These models are very different from ours. Indeed, we consider here a *formal symbolic model* in which messages are no longer bitstrings but are abstracted away by terms. Some recent attempts have been made to design formal symbolic model suitable to analyse distance bounding protocols: *e.g.* a model based on multiset rewriting rules has been proposed in [9] and [11], another one based on strand spaces is available in [18]. Even if our model shares some similarities with those mentioned above, we design a new one based on the applied pi calculus [19] in order to connect our theoretical results with the ProVerif verification tool that we ultimately use to analyse protocols.

Our main reduction result follows the spirit of [20] where it is shown that it is sufficient to consider five specific topologies when analysing routing protocols. To our knowledge, the only work proposing a reduction result suitable for distance bounding protocols is [18]: the authors show that $n$ attackers are sufficient when analysing a configuration involving at most $n$ honest participants. However, we still need to consider an arbitrary number of participants when looking for an attack. Moreover, due to the way attackers are located (close to each honest participant), such a result can not be applied to analyse hijacking attacks that typically disallow the presence of an attacker in the neighbourhood of honest participants. In contrast, our result reduces to only one topology, even when considering an arbitrary number of honest participants, and it applies to the scenario mentioned above. A consequence of this result is that we can leverage the ProVerif tool to analyse such protocols. To do that we get some inspiration from [1]. Our contributions improve upon their work by providing a strong theoretical foundation to their idea. Moreover, in order to consider scenario in which attackers are far away, and thus unable to produce an answer within the delay, we slightly modify the tool to discard some attacker behaviours. This was mandatory to analyse distance hijacking scenarios relying on the ProVerif tool.

Recently, a methodology to analyse distance bounding protocols within the Tamarin verification tool has been proposed [11]. Their model does not allow one to consider the different class of attacks, *e.g.* they cannot prove that a protocol is mafia fraud resistant but vulnerable to a distance hijacking attack. All the case studies reported in [11] have been analysed in our framework, and a brief comparison is provided in Section VI.

## II. Messages

As usual in the symbolic setting, we model messages through a term algebra. We consider both equational theories and reduction relations to represent the properties of the cryptographic primitives.

### A. Term algebra

We consider two infinite and disjoint sets of *names*: $\mathcal{N}$ is the set of *basic names*, which are used to represent keys, nonces, whereas $\mathcal{A}$ is the set of *agent names*, *i.e.* names which represent the agents identities. We consider an infinite set $\Sigma_0$ of constant symbols that are used for instance to represent nonces drawn by the attacker. We also consider two infinite and disjoint sets of *variables*, denoted $\mathcal{X}$ and $\mathcal{W}$. Variables in $\mathcal{X}$ refer to unknown parts of messages expected by participants while variables in $\mathcal{W}$ are used to store messages learnt by the attacker.

We assume a signature $\Sigma$, *i.e.* a set of function symbols together with their arity. The elements of $\Sigma$ are split into *constructor* and *destructor* symbols, *i.e.* $\Sigma = \Sigma_c \uplus \Sigma_d$. We denote $\Sigma^+ = \Sigma \cup \Sigma_0$, and $\Sigma_c^+ = \Sigma_c \cup \Sigma_0$. Given a signature $\mathcal{F}$, and a set of atomic data $\mathsf{A}$, we denote by $\mathcal{T}(\mathcal{F}, \mathsf{A})$ the set of *terms* built from atomic data $\mathsf{A}$ by applying function symbols in $\mathcal{F}$. A *constructor term* is a term in $\mathcal{T}(\Sigma_c^+, \mathcal{N} \cup \mathcal{A} \cup \mathcal{X})$. We denote $vars(u)$ the set of variables that occur in a term $u$. A *message* is a constructor term $u$ that is *ground*, *i.e.* such that $vars(u) = \emptyset$. The application of a substitution $\sigma$ to a term $u$ is written $u\sigma$. We denote $dom(\sigma)$ its *domain*, and $img(\sigma)$ its *image*. The positions of a term are defined as usual.

**Example 1.** *We consider the following signature* $\Sigma_{\mathsf{ex}} = \Sigma_c \uplus \Sigma_d$*:*

- $\Sigma_c = \{\mathsf{commit}, \mathsf{sign}, \mathsf{sk}, \mathsf{vk}, \mathsf{ok}, \langle\rangle, \oplus, 0\}$
- $\Sigma_d = \{\mathsf{open}, \mathsf{getmsg}, \mathsf{check}, \mathsf{proj}_1, \mathsf{proj}_2, \mathsf{eq}\}$.

*The symbols* open *and* commit *(arity* 2*) represent a commitment scheme, whereas the symbols* sign*,* check *(arity* 2*),* getmsg*,* sk*, and* vk *(arity* 1*) are used to model signature. Pairing and projections are modelled using* $\langle\ \rangle$ *(arity* 2*), and* $\mathsf{proj}_i$ *with* $i \in \{1, 2\}$ *(arity* 1*). The symbols* $\oplus$ *(arity* 2*) and the constant* 0 *model the exclusive-or operator. We consider the symbol* eq *to model equality test.*

### B. Equational theory

Following the approach developed in [21], constructor terms are subject to an *equational theory*. This allows one to model the algebraic properties of the primitives. It consists of a finite set of equations of the form $u = v$ where $u, v \in \mathcal{T}(\Sigma_c, \mathcal{X})$,

and induces an equivalence relation $=_{\mathsf{E}}$ over constructor terms. Formally, $=_{\mathsf{E}}$ is the smallest congruence on constructor terms, which contains $u = v$ in $\mathsf{E}$, and that is closed under substitutions of terms for variables.

**Example 2.** *To reflect the algebraic properties of the exclusive-or operator, we consider the equational theory* $\mathsf{E}_{\mathsf{xor}}$ *generated by the following equations:*

$$\begin{array}{rclcrcl} (x \oplus y) \oplus z & = & x \oplus (y \oplus z) & \qquad & x \oplus y & = & y \oplus x \\ x \oplus 0 & = & x & & x \oplus x & = & 0 \end{array}$$

### C. Rewriting rules

As in [21], we also give a meaning to destructor symbols. This is done through a set of rewriting rules of the form $\mathsf{g}(t_1, \ldots, t_n) \to t$ where $\mathsf{g} \in \Sigma_d$, and $t, t_1, \ldots, t_n \in \mathcal{T}(\Sigma_c, \mathcal{X})$. A term $u$ can be *rewritten* in $v$ if there is a position $p$ in $u$, and a rewriting rule $\mathsf{g}(t_1, \ldots, t_n) \to t$ such that $u|_p = \mathsf{g}(t_1, \ldots, t_n)\theta$ for some substitution $\theta$. Moreover, we assume that $t_1\theta, \ldots, t_n\theta$ as well as $t\theta$ are messages. We only consider sets of rewriting rules that yield a *convergent* rewriting system, and we denote $u{\downarrow}$ the *normal form* of a term $u$.

For modelling purposes, we split the signature $\Sigma$ into two parts, $\Sigma_{\mathsf{pub}}$ and $\Sigma_{\mathsf{priv}}$, and we denote $\Sigma^+_{\mathsf{pub}} = \Sigma_{\mathsf{pub}} \cup \Sigma_0$. An attacker builds messages by applying public symbols to terms he knows and that are available through variables in $\mathcal{W}$. Formally, a computation done by the attacker is a *recipe*, *i.e.* a term in $\mathcal{T}(\Sigma^+_{\mathsf{pub}}, \mathcal{W})$.

**Example 3.** *Among symbols in* $\Sigma_{\mathsf{ex}}$*, only* $\mathsf{sk}$ *is in* $\Sigma_{\mathsf{priv}}$*. The properties of the symbols in* $\Sigma_d$ *are reflected through the following rewriting rules:*

$$\begin{array}{ll} \mathsf{check}(\mathsf{sign}(x, \mathsf{sk}(y)), \mathsf{vk}(y)) \to \mathsf{ok} & \mathsf{eq}(x, x) \to \mathsf{ok} \\ \mathsf{getmsg}(\mathsf{sign}(x, \mathsf{sk}(y))) \to x & \mathsf{proj}_1(\langle x_1, x_2 \rangle) \to x_1 \\ \mathsf{open}(\mathsf{commit}(x, y), y) \to x & \mathsf{proj}_2(\langle x_1, x_2 \rangle) \to x_2. \end{array}$$

## III. Timed security protocols

We now present our model which incorporates a notion of location, and time.

### A. Process algebra

Protocols are modelled through processes using the following grammar:

$$\begin{array}{rcl} P, Q & := & 0 \\ & | & \mathtt{new}\ n.P \\ & | & \mathtt{let}\ x = v\ \mathtt{in}\ P \\ & | & \mathtt{out}(u).P \\ & | & \mathtt{in}(x).P \\ & | & \mathtt{in}^{<t}(x).P \\ & | & \mathtt{reset}.P \end{array}$$

where $x \in \mathcal{X}$, $n \in \mathcal{N}$, $u \in \mathcal{T}(\Sigma^+_c, \mathcal{X} \uplus \mathcal{N} \uplus \mathcal{A})$, $v \in \mathcal{T}(\Sigma^+, \mathcal{X} \uplus \mathcal{N} \uplus \mathcal{A})$ and $t \in \mathbb{R}_+$.

Most of these constructions are rather standard. As usual, 0 denotes the empty process that does nothing, and the `new` instruction is used to model fresh name generation. Then, we have standard constructions to model inputs and outputs. We may note the special construction $\mathtt{in}^{<t}(x)$ that combines an input with a constraint on the local clock of the agent executing this action. This construction is in contrast with the approach proposed in e.g. [11] where input actions are not subject to any timing constraint, and are therefore always possible provided that enough time has elapsed. From this point of view, our model represents the reality more faithfully since an agent will not proceed an input arriving later than expected. The `reset` instruction will reset the local clock of the agent. Finally, the process $\mathtt{let}\ x = v\ \mathtt{in}\ P$ tries to evaluate $v$, the process $P$ is executed in case of success, and the process is blocked otherwise. Note that the usual conditional operator can be modelled as follows: $\mathtt{let}\ x = \mathsf{eq}(u, v)\ \mathtt{in}\ P$.

We write $fv(P)$ (resp. $fn(P)$) for the set of *free* variables (resp. names) occurring in $P$, *i.e.* the set of variables (resp. names) that are not in the scope of an `in` or a `let` (resp. a `new`). We consider *parametrised processes*, denoted $P(z_0, \ldots, z_n)$, where $z_0, \ldots, z_n$ are variables from a special set $\mathcal{Z}$ (disjoint from $\mathcal{X}$ and $\mathcal{W}$). Intuitively, these variables will be instantiated by agent names, and $z_0$ corresponds to the name of the agent that executes the process. A *role* $R = P(z_0, \ldots, z_n)$ is a parametrised process that does not contain any agent name, and such that $fv(R) \subseteq \{z_0, \ldots, z_n\}$. A *protocol* is a set of roles.

**Example 4.** *As a running example, we consider the signature-based Brands and Chaum distance bounding protocol [2] that is informally described below:*

$$\begin{array}{ll} 1.\ P \to V: & \mathsf{commit}(m, k) \\ 2.\ V \to P: & n \\ 3.\ P \to V: & n \oplus m \\ 4.\ P \to V: & k \\ 5.\ P \to V: & \mathsf{sign}(\langle n, n \oplus m \rangle, \mathsf{sk}(P)). \end{array}$$

*The prover $P$ generates a nonce $m$ and a key $k$, and sends a commitment to the verifier $V$. The verifier $V$ generates his own nonce $n$ and initiates the time measurement phase, also called the* rapid phase*. $P$ has to provide an answer as quickly as possible since $V$ will reject any answer arriving too late (a long response time does not give him any guarantee regarding its proximity with the prover). After this phase, $P$ sends a means to open the commitment, as well as a signature on the values exchanged during the rapid phase. When a verifier ends the protocol, the prover with whom he is communicating should be located in his neighbourhood. In our setting, we consider the following parametrised processes:*

$$\begin{array}{l} P(z_P) := \\ \quad \mathtt{new}\ m.\mathtt{new}\ k. \\ \quad \mathtt{out}(\mathsf{commit}(m, k)). \\ \quad \mathtt{in}(x_n). \\ \quad \mathtt{out}(x_n \oplus m). \\ \quad \mathtt{out}(k). \\ \quad \mathtt{out}(\mathsf{sign}(\langle x_n, x_n \oplus m \rangle, \mathsf{sk}(z_P))).0 \end{array}$$

$V(z'_V, z'_P) :=$
$\quad \mathtt{in}(y_c).\mathtt{new}\ n.$
$\quad \mathtt{reset}.\mathtt{out}(n).\mathtt{in}^{<2\times t_0}(y_0).$
$\quad \mathtt{in}(y_k).\mathtt{in}(y_{\mathsf{sign}}).$
$\quad \mathtt{let}\ y_m = \mathsf{open}(y_c, y_k)\ \mathtt{in}$
$\quad \mathtt{let}\ y_{\mathsf{check}} = \mathsf{check}(y_{\mathsf{sign}}, \mathsf{vk}(z'_P))\ \mathtt{in}$
$\quad \mathtt{let}\ y_{\mathsf{eq}} = \mathsf{eq}(\langle n, n \oplus y_m\rangle, \mathsf{getmsg}(y_{\mathsf{sign}}))\ \mathtt{in}\ 0.$

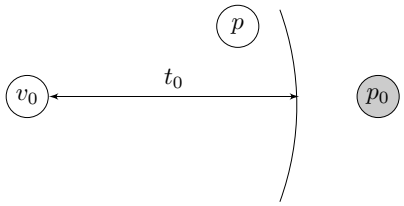


Fig. 1: Example of a topology

### B. Configuration and topology

Each process has a location. As in the classical Dolev-Yao model [22], the attackers control the entire network but interacting with agents who are far away takes time. To formalise this, our execution model is parametrised by a topology.

**Definition 1.** *A* topology *is a tuple* $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ *where:*

- $\mathcal{A}_0 \subseteq \mathcal{A}$ *is the finite set of agents composing the system;*
- $\mathcal{M}_0 \subseteq \mathcal{A}_0$ *is the subset of agents that are dishonest;*
- $\mathsf{Loc}_0 : \mathcal{A}_0 \to \mathbb{R}^3$ *is a mapping defining the position of each agent in space.*
- $p_0$ *and* $v_0$ *are two agents in* $\mathcal{A}_0$ *that represent respectively the prover and the verifier for which we analyse the security of the protocol.*

In our model, the distance between two agents is expressed by the time it takes for a message to travel from one to another. Therefore, we consider $\mathsf{Dist}_{\mathcal{T}_0} : \mathcal{A}_0 \times \mathcal{A}_0 \to \mathbb{R}$, based on $\mathsf{Loc}_0$ that will provide the time a message takes to travel between two agents. It is defined as follows:

$$\mathsf{Dist}_{\mathcal{T}_0}(a, b) = \frac{\|\mathsf{Loc}_0(a) - \mathsf{Loc}_0(b)\|}{c_0} \text{ for any } a, b \in \mathcal{A}_0$$

with $\|\cdot\| : \mathbb{R}^3 \to \mathbb{R}$ the euclidian norm and $c_0$ the transmission speed. We suppose, from now on, that $c_0$ is a constant for all agents, and thus an agent $a$ can recover, at time $t$, any message emitted by any other agent $b$ before $t - \mathsf{Dist}_{\mathcal{T}_0}(a, b)$.

Note that our model is not restricted to a single dishonest node. In particular, our results apply to the case of several compromised nodes that communicate (and therefore share their knowledge). However, communication is subject to physical constraints: message transmission takes time determined by the distance between nodes. All agents, including attackers, are subject to these constraints. This results in a distributed attacker with restricted, but more realistic, communication capabilities than those of the traditional Dolev-Yao attacker.

Our semantics is given by a transition system over configurations that manipulates *extended processes*, i.e. expressions of the form $\lfloor P_a \rfloor_a^{t_a}$ with $a \in \mathcal{A}$, $P_a$ a process such that $fv(P_a) = \emptyset$, and $t_a \in \mathbb{R}_+$. Intuitively, $P_a$ describes the actions of agent $a$, and $t_a$ his local clock. In order to store the messages that have been outputted so far, we extend the notion of *frame* (introduced in [19]) to keep track of the time at which the message has been outputted and by whom.

**Definition 2.** *Given a topology* $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$, *a* configuration $K$ *over* $\mathcal{T}_0$ *is a tuple* $(\mathcal{P}; \Phi; t)$, *where:*

- $\mathcal{P}$ *is a multiset of* extended process $\lfloor P \rfloor_a^{t_a}$ *with* $a \in \mathcal{A}_0$;
- $\Phi = \{\mathsf{w}_1 \xrightarrow{a_1,t_1} u_1, \ldots, \mathsf{w}_n \xrightarrow{a_n,t_n} u_n\}$ *is an extended frame, i.e. a substitution such that* $\mathsf{w}_i \in \mathcal{W}$, $u_i \in \mathcal{T}(\Sigma_c^+, \mathcal{N} \uplus \mathcal{A})$, $a_i \in \mathcal{A}_0$ *and* $t_i \in \mathbb{R}_+$ *for* $1 \le i \le n$;
- $t \in \mathbb{R}_+$ *is the global time of the system.*

We write $\lfloor \Phi \rfloor_a^t$ for the restriction of $\Phi$ to the agent $a$ at time $t$, *i.e.* :

$$\lfloor \Phi \rfloor_a^t = \left\{ \mathsf{w}_i \xrightarrow{a_i,t_i} u_i \;\middle|\; \begin{array}{c} (\mathsf{w}_i \xrightarrow{a_i,t_i} u_i) \in \Phi \text{ and} \\ a_i = a \text{ and } t_i \le t \end{array} \right\}$$

**Example 5.** *Continuing Example 4, a topology* $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ *is depicted in Figure 1 where* $\mathcal{A}_0 = \{p_0, v_0, p\}$, *and* $\mathcal{M}_0 = \{p_0\}$. *The precise location of each agent is not relevant, only the distance between them matters. Here, we assume that* $\mathsf{Dist}_{\mathcal{T}_0}(p, v_0) < t_0$ *whereas* $\mathsf{Dist}_{\mathcal{T}_0}(p_0, v_0) \ge t_0$.

*A typical configuration is:*

$$K_0 = (\lfloor P(p) \rfloor_p^0 \uplus \lfloor V(v_0, p_0) \rfloor_{v_0}^0 ; \{\mathsf{w}_1 \xrightarrow{p_0,0} \mathsf{sk}(p_0)\}; 0)$$

*where* $p$ *is playing the role of the prover and* $v_0$ *the role of the verifier with a dishonest agent* $p_0$. *The extended frame only contains the signature key of the dishonest agent, i.e.* $\mathsf{sk}(p_0)$. *A more realistic configuration would include other instances of these two roles and will give more knowledge to the attacker, but we will see that this configuration is already sufficient to present an attack.*

### C. Semantics

Given a topology $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$, the semantics of processes is formally defined by the rules given in Figure 2. The TIM rule allows time to elapse, meaning that the global clock as well as the local clocks will be shifted by $\delta$:

$$\mathsf{Shift}(\mathcal{P}, \delta) = \biguplus_{P \in \mathcal{P}} \mathsf{Shift}(P, \delta) \text{ and}$$
$$\mathsf{Shift}(\lfloor P \rfloor_a^{t_a}, \delta) = \lfloor P \rfloor_a^{t_a + \delta}.$$

The RST rule allows an agent to reset his local clock. The other rules are rather standard. The IN rule allows an agent $a$ to evolve when receiving a message: the received message has necessarily been forged and sent at time $t_b$ by some agent $b$ who was in possession of all the necessary information at that time.

$$\text{TIM} \quad (\mathcal{P};\Phi;t) \longrightarrow_{\mathcal{T}_0} (\mathsf{Shift}(\mathcal{P},\delta);\Phi;t+\delta) \qquad \text{with } \delta \geq 0$$

$$\text{OUT} \quad (\lfloor \mathtt{out}(u).P \rfloor_a^{t_a} \uplus \mathcal{P};\Phi;t) \xrightarrow{a,\mathtt{out}(u)}_{\mathcal{T}_0} (\lfloor P \rfloor_a^{t_a} \uplus \mathcal{P};\Phi \uplus \{\mathsf{w} \xrightarrow{a,t} u\};t) \quad \text{with } \mathsf{w} \in \mathcal{W} \text{ fresh}$$

$$\text{LET} \quad (\lfloor \mathtt{let}\ x = u\ \mathtt{in}\ P \rfloor_a^{t_a} \uplus \mathcal{P};\Phi;t) \xrightarrow{a,\tau}_{\mathcal{T}_0} (\lfloor P\{x \mapsto u\downarrow\} \rfloor_a^{t_a} \uplus \mathcal{P};\Phi;t) \quad \text{when } u\downarrow \in \mathcal{T}(\Sigma_c^+, \mathcal{N} \uplus \mathcal{A})$$

$$\text{NEW} \quad (\lfloor \mathtt{new}\ n.P \rfloor_a^{t_a} \uplus \mathcal{P};\Phi;t) \xrightarrow{a,\tau}_{\mathcal{T}_0} (\lfloor P\{n \mapsto n'\} \rfloor_a^{t_a} \uplus \mathcal{P};\Phi;t) \text{ with } n' \in \mathcal{N} \text{ fresh}$$

$$\text{RST} \quad (\lfloor \mathtt{reset}.P \rfloor_a^{t_a} \uplus \mathcal{P};\Phi;t) \xrightarrow{a,\tau}_{\mathcal{T}_0} (\lfloor P \rfloor_a^{0} \uplus \mathcal{P};\Phi;t)$$

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

when there exist $b \in \mathcal{A}_0$ and $t_b \in \mathbb{R}_+$ such that $t_b \leq t - \mathsf{Dist}_{\mathcal{T}_0}(b,a)$ and:

- if $b \in \mathcal{A}_0 \setminus \mathcal{M}_0$ then $u \in img(\lfloor \Phi \rfloor_b^{t_b})$;
- if $b \in \mathcal{M}_0$ then $u = R\Phi\downarrow$ for some recipe $R$ such that for all $\mathsf{w} \in vars(R)$ there exists $c \in \mathcal{A}_0$ such that $\mathsf{w} \in dom(\lfloor \Phi \rfloor_c^{t_b - \mathsf{Dist}_{\mathcal{T}}(c,b)})$.

Moreover, in case $\star$ is $< t_g$ for some $t_g$, we assume in addition that $t_a < t_g$.

Fig. 2: Semantics of our calculus

We sometimes simply write $\rightarrow_{\mathcal{T}_0}$ instead of $\xrightarrow{a,\alpha}_{\mathcal{T}_0}$. The relation $\rightarrow_{\mathcal{T}_0}^*$ is the reflexive and transitive closure of $\rightarrow_{\mathcal{T}_0}$, and we often write $\xrightarrow{\mathsf{tr}}_{\mathcal{T}_0}$ to emphasise the sequence of labels $\mathsf{tr}$ that has been used during this execution.

**Example 6.** *Continuing Example 5, we may consider the following execution:*

$$K_0 \xrightarrow{p,\tau}_{\mathcal{T}_0} \xrightarrow{p,\tau}_{\mathcal{T}_0} \xrightarrow{p,\mathtt{out}(\mathsf{commit}(m',k'))}_{\mathcal{T}_0} \rightarrow_{\mathcal{T}_0} \xrightarrow{v_0,\mathtt{in}(\mathsf{commit}(m',k'))}_{\mathcal{T}_0} \xrightarrow{v_0,\tau}_{\mathcal{T}_0} \xrightarrow{v_0,\tau}_{\mathcal{T}_0} K_1$$

*where* $K_1 = (\mathcal{P}_1;\Phi_1;\delta_0)$ *where:*

- $\mathcal{P}_1 = \lfloor P_1 \rfloor_p^{\delta_0} \uplus \lfloor V_1 \rfloor_{v_0}^0$ *with* $P_1$ *and* $V_1$ *representing the evolution of the processes located in* $p$ *and* $v_0$.
  *More precisely we have:*
  - $P_1 = \mathtt{in}(x_n).\mathtt{out}(x_n \oplus m').\mathtt{out}(k').\mathtt{out}(\mathsf{sign}(\langle x_n, x_n \oplus m' \rangle, \mathsf{sk}(p)))$
  - $V_1 = \mathtt{out}(n').\mathtt{in}^{<2\times t_0}(y_0).\mathtt{in}(y_k).\mathtt{in}(y_{\mathsf{sign}}).\mathtt{let}\ y_m = \mathsf{open}(y_c, y_k)\ \mathtt{in} \ldots$
- $\Phi_1 = \{\mathsf{w}_1 \xrightarrow{p_0,0} \mathsf{sk}(p_0),\ \mathsf{w}_2 \xrightarrow{p,0} \mathsf{commit}(m',k')\}$

*This models the beginning of a normal execution between* $p$ *and* $v_0$. *The message outputted at location* $p$ *is received at location* $v_0$. *The instance of the rule TIM in between (here with* $\delta_0 = \mathsf{Dist}_{\mathcal{T}_0}(p, v_0)$*) allows the message to reach location* $v_0$.

## IV. Security properties

A distance bounding protocol is a protocol in which a party (the verifier) is assured of the identity of another party (the prover), as well as the fact that this prover is located in his neighbourhood. Several frauds are usually considered. We introduce in Section IV-B the ones that will be studied in this paper, and we explain how they will be formalised. Before to do that, we introduce the notion of $t_0$-proximity and the notion of valid initial configuration that aims to represent all the scenarios that have to be analysed once the topology has been fixed.

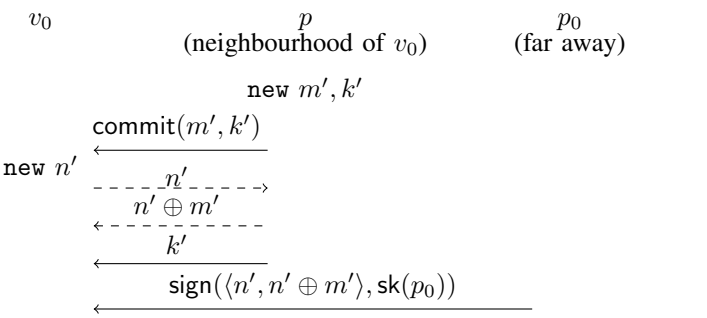



Fig. 3: Distance hijacking attack on the Brands and Chaum's protocol

### A. Physical proximity on a given topology

For the sake of simplicity, we assume that configurations representing instances of distance bounding protocols contain a process (typically a session of the verifier) that ends with a special action of the form $\mathsf{end}(v,p)$ claiming that agents $v$ and $p$ are close (typically $v$ is the verifier and $p$ the prover). Checking whether a protocol ensures physical proximity w.r.t. a given configuration $K_0$ is defined as follows.

**Definition 3.** *Let* $\mathcal{T}_0$ *be a topology and* $K_0$ *be a configuration over* $\mathcal{T}_0$. *We say that* $K_0$ *admits* an attack w.r.t. $t_0$-proximity *in* $\mathcal{T}_0$ *if*

$$K_0 \rightarrow_{\mathcal{T}_0}^* (\lfloor \mathsf{end}(a_1,a_2) \rfloor_a^{t_a} \uplus \mathcal{P};\Phi;t) \text{ with } \mathsf{Dist}_{\mathcal{T}_0}(a_1,a_2) \geq t_0.$$

**Example 7.** *Continuing Example 5, we consider the configuration* $K_0'$ *below:*

$$K_0' = (\lfloor P(p) \rfloor_p^0 \uplus \lfloor V'(v_0,p_0) \rfloor_{v_0}^0 ; \{\mathsf{w}_1 \xrightarrow{p_0,0} \mathsf{sk}(p_0)\}; 0)$$

*where* $V'(z_V', z_P')$ *is* $V(z_V', z_P')$ *in which the null process has been replaced by* $\mathsf{end}(z_V', z_P')$. *The configuration* $K_0'$ *can still follow the execution of Example 6:*

$$K_1' = (\lfloor P_1 \rfloor_p^{\delta_0} \uplus \lfloor V_1' \rfloor_{v_0}^0 ; \Phi_1; \delta_0)$$

*where $V_1'$ is $V_1$ in which the occurrence of the null process has been replaced by $\mathsf{end}(v_0, p_0)$. Now, we can pursue this execution in $\mathcal{T}_0$ as follows:*

$$K_1' \xrightarrow{v_0,\mathtt{out}(n')} \to \xrightarrow{p,\mathtt{in}(n')} \xrightarrow{p,\mathtt{out}(n'\oplus m')} \xrightarrow{p,\mathtt{out}(k')} \to \xrightarrow{v_0,\mathtt{in}^{<2\times t_0}(n'\oplus m')} \xrightarrow{v_0,\mathtt{in}(k')} \to \xrightarrow{v_0,\mathtt{in}(\mathsf{sign}(\langle n',n'\oplus m'\rangle,\mathsf{sk}(p_0)))} K_2'$$

*with:*
$K_2' = (\lfloor P_2 \rfloor_p^{3\delta_0+2\delta_0'} \uplus \lfloor \mathsf{end}(v_0,p_0)\rfloor_{v_0}^{2\delta_0+2\delta_0'}; \Phi; 3\delta_0+2\delta_0')$.

*The two first lines correspond to a normal execution of the protocol between $v_0$ and $p$. Note that, on each line, we need an instance of the TIM rule with $\delta_0 = \mathit{Dist}_{\mathcal{T}_0}(v_0,p) = \mathit{Dist}_{\mathcal{T}_0}(p,v_0)$ to allow the sent message to reach its destination. The last transition does not follow the normal execution of the protocol. Actually, the dishonest agent $p_0$ is responsible of this input. He built this message from the messages $n'$ and $n'\oplus m'$ that have been sent on the network, and the key $\mathsf{sk}(p_0)$ that is part of his initial knowledge. Note that he has to wait the necessary amount of time to allow these messages to reach him (e.g. $\delta_0' = \mathit{Dist}_{\mathcal{T}_0}(v_0,p_0)$), and some time is needed for the forged message to reach $v_0$ (actually $\delta_0' = \mathit{Dist}_{\mathcal{T}_0}(v_0,p_0)$). Therefore, the first rule of the last line is an instance of the TIM rule during which a delay of $2\delta_0'$ has elapsed.*

*Note that, according to Definition 3, this corresponds to an attack w.r.t. $t_0$-proximity on configuration $K_0'$. Actually, this is the hijacking attack that has been reported in [8] and described in Figure 3. Here, a dishonest prover $p_0$ exploits an honest prover $p$ (located in the neighbourhood of $v_0$) to provide the verifier $v_0$ with false information about the distance between $p_0$ and $v_0$.*

When analysing a distance bounding protocol, not all the configurations are interesting. Therefore, we first fix a topology $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$, and we consider any configuration that is valid w.r.t. this topology. Actually, we consider a protocol $\mathcal{P}_{\mathsf{prox}}$, and we assume that the initial knowledge of dishonest participants is given through a template $\mathcal{I}_0$, i.e. a set of terms in $\mathcal{T}(\Sigma_c^+, \mathcal{Z})$. Using this template $\mathcal{I}_0$, and considering a set of agents $\mathcal{A}_0$, we derive the initial knowledge of agent $a \in \mathcal{A}_0$ as follows:

$$\mathsf{Knows}(\mathcal{I}_0, a, \mathcal{A}_0) = \left\{ (u_0\{z_0 \mapsto a\})\sigma \;\middle|\; \begin{array}{l} u_0 \in \mathcal{I}_0 \\ vars(u_0\sigma) = \emptyset \\ img(\sigma) \subseteq \mathcal{A}_0 \end{array} \right\}$$

**Definition 4.** *Let $\mathcal{P}_{\mathsf{prox}}$ be a protocol, $V_0(z_0,z_1)$ be a parametrised role containing the special action $\mathsf{end}(z_0,z_1)$, $\mathcal{I}_0$ be a template, and $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ be a topology. A configuration $K = (\mathcal{P}; \Phi; t)$ is a* valid initial configuration *for the protocol $\mathcal{P}_{\mathsf{prox}}$ and $V_0$ w.r.t. $\mathcal{T}_0$ and $\mathcal{I}_0$ if:*

1) *$\mathcal{P} = \lfloor V_0(v_0,p_0)\rfloor_{v_0}^{t'} \uplus \mathcal{P}'$ for some $t'$ and for each $\lfloor P'\rfloor_{a'}^{t'} \in \mathcal{P}'$ there exists $P(z_0,\ldots,z_k) \in \mathcal{P}_{\mathsf{prox}}$, and $a_1,\ldots,a_k \in \mathcal{A}_0$ such that $P' = P(a',a_1,\ldots,a_k)$.*
2) *$img(\lfloor\Phi\rfloor_a^t) = \mathsf{Knows}(\mathcal{I}_0,a,\mathcal{A}_0)$ when $a \in \mathcal{M}_0$, and $img(\lfloor\Phi\rfloor_a^t) = \emptyset$ otherwise.*

The first condition says that we consider initial configurations made up of instances of the roles of the protocols, and we only consider roles executed by agents located at the right place. We may note that an agent can execute at the same time the role of a verifier and the role of a prover. The second condition allows one to give some initial knowledge to each malicious node. Finally we do not give so much constraints regarding time to be able to consider all the possible initial configurations before declaring a protocol secure.

**Example 8.** *Going back to Example 7 and considering the template $\mathcal{I}_0 = \{\mathsf{sk}(z_0)\}$, we have that $K_0'$ is a valid initial configuration w.r.t. $\mathcal{T}_0$ and $\mathcal{I}_0$.*

**Definition 5.** *Let $\mathcal{P}_{\mathsf{prox}}$ be a protocol, $V_0(z_0,z_1)$ be a parametrised role containing the special action $\mathsf{end}(z_0,z_1)$, $\mathcal{I}_0$ be a template, and $\mathcal{T}_0$ be a topology. We say that $\mathcal{P}_{\mathsf{prox}}$ admits an attack w.r.t. $t_0$-proximity in $\mathcal{T}_0$ if there exists a valid initial configuration $K$ for $\mathcal{P}_{\mathsf{prox}}$ and $V_0$ w.r.t. $\mathcal{T}_0$ and $\mathcal{I}_0$ such that $K$ admits an attack w.r.t. $t_0$-proximity in $\mathcal{T}_0$.*

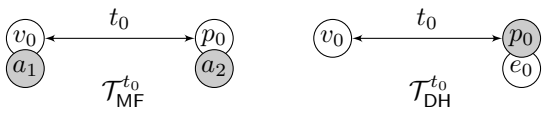
### *B. Classification of attacks*

We consider two types of attacks, namely mafia and distance hijacking frauds. We do not consider the notion of terrorist fraud. This does not fit in our model since here we assume colluding attackers who share their knowledge (provided that enough time has elapsed). Our notion of distance hijacking subsumes the usual notion of distance fraud, and therefore, due to space constraints, we choose to not detail this type of attack (results about distance fraud are given in [14]).

**Mafia fraud:** A *mafia fraud* is an attack in which generally three agents are involved: a verifier, an honest prover located outside the neighbourhood of the verifier, and an external attacker. However, we consider here its general version in which an arbitrary number of participants may be involved in the attack. The aim of the attacker is to convince the verifier that the honest prover is actually close to it. Therefore, we consider any topology $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ such that both $v_0$ and $p_0$ are honest, i.e. $v_0, p_0 \in \mathcal{A}_0 \setminus \mathcal{M}_0$. We denote $\mathcal{C}_{\mathsf{MF}}$ the set of topologies that satisfy these requirements.

**Distance hijacking fraud:** A *distance hijacking fraud* is an attack in which a dishonest prover located far away succeeds in convincing an honest verifier that he is actually close to him. The dishonest prover may exploit honest entities located in the neighbourhood of the verifier. Therefore, we consider any topology $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ such that $p_0 \in \mathcal{M}_0$, $v_0 \in \mathcal{A}_0 \setminus \mathcal{M}_0$, and $\mathsf{Dist}_{\mathcal{T}_0}(v_0,a) \geq t_0$ for any $a \in \mathcal{M}_0$. We denote $\mathcal{C}_{\mathsf{DH}}$ the set of topologies that satisfy these requirements.

**Definition 6.** *Let $\mathcal{P}_{\mathsf{prox}}$ be a protocol, $V_0(z_0,z_1)$ be a parametrised role containing the special event $\mathsf{end}(z_0,z_1)$, and $\mathcal{I}_0$ be a template. We say that $\mathcal{P}_{\mathsf{prox}}$ admits a mafia fraud attack (resp. distance hijacking attack) w.r.t. $t_0$-proximity if there exist $\mathcal{T} \in \mathcal{C}_{\mathsf{MF}}$ (resp. $\mathcal{C}_{\mathsf{DH}}$), a valid initial configuration $K$ for $\mathcal{P}_{\mathsf{prox}}$ and $V_0$ w.r.t. $\mathcal{T}$ and $\mathcal{I}_0$ such that $K$ admits an attack w.r.t. $t_0$-proximity in $\mathcal{T}$.*




malicious node

honest node

Fig. 4: Topologies $\mathcal{T}^{t_0}_{\mathsf{MF}}$ (mafia fraud), and $\mathcal{T}^{t_0}_{\mathsf{DH}}$ (distance hijacking fraud)

Our main contribution is to provide reduction results that allow one to analyse the security of a protocol w.r.t. the frauds mentioned above considering only a specific topology. Then, we will show how to leverage an existing tool ProVerif to automatically analyse this notion of physical proximity.

## V. Reducing the topology

Our reduction results allow one to analyse the security of a protocol (w.r.t. $t_0$-proximity) considering only a specific and rather simple topology.

Before presenting them we want to highlight that these results can be easily adapted if we assume that an agent cannot execute at the same time the role of a verifier and a role of a prover. Indeed it is sufficient to duplicate each honest identities in the reduced topologies: one playing all the verifier roles and the other one playing all the prover roles.

### A. Mafia fraud

Regarding mafia fraud, the unique resulting topology is $\mathcal{T}^{t_0}_{\mathsf{MF}} = (\mathcal{A}_{\mathsf{MF}}, \mathcal{M}_{\mathsf{MF}}, \mathsf{Loc}_{\mathsf{MF}}, v_0, p_0)$ with $\mathcal{A}_{\mathsf{MF}} = \{p_0, v_0, a_1, a_2\}$, $\mathcal{M}_{\mathsf{MF}} = \{a_1, a_2\}$, $\mathsf{Loc}_{\mathsf{MF}}(v_0) = \mathsf{Loc}_{\mathsf{MF}}(a_1)$, $\mathsf{Loc}_{\mathsf{MF}}(p_0) = \mathsf{Loc}_{\mathsf{MF}}(a_2)$, and $\mathsf{Dist}_{\mathcal{T}^{t_0}_{\mathsf{MF}}}(p_0, v_0) = t_0$ as pictured in Figure 4.

A simple idea to reduce towards such a topology could be to move each node $n$ in the neighbourhood of $v_0$ at the same location as $v_0$, and to keep the other ones at distance (i.e. location of $p_0$). However, such a reduction will lengthen the distance between $n$ and $p_0$, and the resulting execution could not be feasible anymore. Since dishonest participants are allowed in the neighbourhood of $v_0$, getting some inspiration from [18], we consider a dishonest participant right next to each honest participant. Such a dishonest participant is ideally located to forge and send messages that will be received by honest agents close to him.

However, contrary to the result provided in [18], our goal is not only to reduce the number of dishonest agents but also the number of honest agents that are involved in an attack trace. In order to ensure that moving (and reducing) the honest agents will not cause any trouble, we need an extra assumption. We require that each role of the protocol is executable. This is a reasonable assumption that will allow one to discard any role executed by agents other than $v_0$ and $p_0$. Intuitively, these operations will be done directly by the attackers.

**Definition 7.** *Given a template $\mathcal{I}_0 = \{u_1, \ldots, u_k\}$, we say that a parametrised role $P(z_0, ..., z_n)$ is* $\mathcal{I}_0$-executable *if $fv(P) \subseteq \{z_0, ..., z_n\}$, $fn(P) = \emptyset$ and for any term $u$ (resp. $v$) occurring in an* `out` *or a* `let` *construction, there exists a recipe $R \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, \{\mathsf{w}_1, \ldots, \mathsf{w}_k\} \uplus \mathcal{N} \uplus \mathcal{X})$ such that $u = R\sigma\downarrow$ (resp. $v\downarrow = R\sigma\downarrow$) where $\sigma = \{\mathsf{w}_1 \mapsto u_1, \ldots, \mathsf{w}_k \mapsto u_k\}$.*

A protocol $\mathcal{P}$ is *$\mathcal{I}_0$-executable* if each role of $\mathcal{P}$ is $\mathcal{I}_0$-executable.

**Example 9.** *Going back to our running example given in Example 4. We have that $P(z_0)$ is $\{\mathsf{sk}(z_0)\}$-executable whereas $V(z_0, z_1)$ is $\{z_1\}$-executable. Therefore, we have that the protocol made of these two roles is $\{\mathsf{sk}(z_0), z_1\}$-executable.*

**Theorem 1.** *Let $\mathcal{I}_0$ be a template, $\mathcal{P}_{\mathsf{prox}}$ be a protocol $\mathcal{I}_0$-executable, and $V_0(z_0, z_1)$ be a parametrised role containing the special event $\mathsf{end}(z_0, z_1)$. We have that $\mathcal{P}_{\mathsf{prox}}$ 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 $\mathcal{T}^{t_0}_{\mathsf{MF}}$.*

### B. Distance hijacking attack

First, we may note that the reduction we did in case of mafia fraud is not possible anymore. Clearly, we need to consider honest participants in the neighbourhood of the verifier and moving them on the same location as $v_0$ will lengthen the distance with $p_0$. Moreover, there is no hope to reduce the number of attackers by placing them close to each honest participant since the addition of a malicious node in the neighbourhood of $v_0$ is not authorised when considering distance hijacking. Actually, adding such a dishonest node in the neighbourhood of $v_0$ will always introduce a false attack since in our model dishonest participants share their knowledge. Therefore, this dishonest participant would be able to impersonate the dishonest prover $p_0$ (who is actually far away).

Nevertheless, we will show that under reasonable conditions, we can reduce towards the topology $\mathcal{T}^{t_0}_{\mathsf{DH}} = (\mathcal{A}_{\mathsf{DH}}, \mathcal{M}_{\mathsf{DH}}, \mathsf{Loc}_{\mathsf{DH}}, v_0, p_0)$ with $\mathcal{A}_{\mathsf{DH}} = \{p_0, v_0, e_0\}$, $\mathcal{M}_{\mathsf{DH}} = \{p_0\}$, $\mathsf{Loc}_{\mathsf{DH}}(p_0) = \mathsf{Loc}_{\mathsf{DH}}(e_0)$, and $\mathsf{Dist}_{\mathcal{T}^{t_0}_{\mathsf{DH}}}(p_0, v_0) = t_0$ (see Figure 4).

The idea behind this reduced topology is the following: the agent $v_0$ plays the role $V_0$ and represents all the honest agents who would be in its proximity (provers and verifiers), $p_0$ represents all the possible malicious participants and $e_0$ represents all the honest participants who are far from $v_0$.

Given a process $P$, we denote $\overline{P}$ the process obtained from $P$ by removing `reset` instructions, and replacing all the occurrences of $\mathtt{in}^{<t}(x)$ by $\mathtt{in}(x)$. This transformation will be applied on the protocol but not on the role $V_0$ for which these instructions play a crucial role. Our reduction result ensures that no distance hijacking attack will be missed if we just analyse the transformed protocol in topology $\mathcal{T}^{t_0}_{\mathsf{DH}}$.

**Theorem 2.** *Let $\mathcal{I}_0$ be a template, $\mathcal{P}_{\mathsf{prox}}$ be a protocol, $t_0 \in \mathbb{R}_+$, and $V_0(z_0, z_1)$ be a parametrised role obtained using the following grammar:*

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

*where $x \in \mathcal{X}$, $n \in \mathcal{N}$, $u, u' \in \mathcal{T}(\Sigma_c^+, \mathcal{X} \cup \mathcal{N} \cup \{z_0, z_1\})$, $v \in \mathcal{T}(\Sigma^+, \mathcal{X} \cup \mathcal{N} \cup \{z_0, z_1\})$ and $t \leq 2 \times t_0$.*
*If $\mathcal{P}_{\mathsf{prox}}$ admits a distance hijacking attack w.r.t. $t_0$-proximity, then $\overline{\mathcal{P}_{\mathsf{prox}}}$ admits an attack against $t_0$-proximity in the topology $\mathcal{T}_{\mathsf{DH}}^{t_0}$.*

**Example 10.** *Let us explain how this reduced topology catches the distance hijacking attack presented in Example 7. As briefly explained above, the role the honest agent $p$ in $\mathcal{T}_0$ will be played by $v_0$. Hence, in $\mathcal{T}_{\mathsf{DH}}^{t_0}$, we consider the following initial configuration:*

$$K_0'' = (\lfloor \overline{P(v_0)} \rfloor_{v_0}^0 \uplus \lfloor V'(v_0, p_0) \rfloor_{v_0}^0 ; \{\mathsf{w}_1 \xrightarrow{p_0, 0} \mathsf{sk}(p_0)\}; 0)$$

*Starting with this configuration, we can follow the execution of Example 7 which is also a witness of the hijacking attack against $t_0$-proximity in the topology $\mathcal{T}_{\mathsf{DH}}^{t_0}$.*

## VI. Case studies using ProVerif

We have reduced the topology but we have still to take into account it when analysing the protocol preventing us to use automatic verification tool dedicated to traditional security protocol such as ProVerif [6]. In this section, we will explain how to get rid of the resulting topology and obtain interesting results on timed protocols relying on the notion of phases that is available in ProVerif.

### A. ProVerif in a nutshell

We consider a subset of the ProVerif calculus defined as follows:

$$\begin{array}{rcl} P := & 0 & | \ \mathtt{new}\ n.P \quad | \ \mathtt{let}\ x = v\ \mathtt{in}\ P \\ | & \mathtt{out}(u).P & | \ \mathtt{in}(x).P \\ | & i : P & | \ !P \end{array}$$

where $x \in \mathcal{X}$, $n \in \mathcal{N}$, $u \in \mathcal{T}(\Sigma_c^+, \mathcal{X} \cup \mathcal{N} \cup \mathcal{A})$, $v \in \mathcal{T}(\Sigma^+, \mathcal{X} \cup \mathcal{N} \cup \mathcal{A})$ and $i \in \mathbb{N}$.

The semantics is similar to the one introduced earlier, and formally defined through a relation, denoted $\Rightarrow$, over configurations (partially given in Figure 5). A configuration is a tuple $(\mathcal{P}; \phi; i)$ where $\mathcal{P}$ is a multiset of processes (as given by the grammar), $\phi$ is a frame as usual (with no decoration on the arrow), and $i \in \mathbb{N}$ is an integer that indicates the current phase. Intuitively, the process $!P$ executes $P$ an arbitrary number of times (in parallel), and only processes in the current phase are allowed to evolve. We often write $P$ instead of $0 : P$.

### B. Our transformation

Given a topology $\mathcal{T}$ (typically one in Figure 4), a protocol $\mathcal{P}_{\mathsf{prox}}$, a role $V_0$, and a template $\mathcal{I}_0$, we build a configuration $(\mathcal{P}; \phi; 0)$ on which the security analysis could be done using ProVerif. From now on, we assume that $V_0(v_0, p_0)$ only contains one block of the form $\mathtt{reset}.\mathtt{out}(m).\mathtt{in}^{<t}(x)$, i.e. it is of the form:

$$\mathsf{block}_1\ .\ \mathtt{reset}\ .\ \mathtt{out}(m)\ .\ \mathtt{in}^{<t}(x)\ .\ \mathsf{block}_2\ .\ \mathsf{end}(v_0, p_0)$$

where $\mathsf{block}_i$ is a sequence of actions (only simple inputs, outputs, let, and new instructions are allowed). The main idea is to use phase 1 to represent the rapid phase. Such a phase starts when $V_0$ performs its `reset` instruction, and ends when $V_0$ performs its $\mathtt{in}^{<t}(x)$ instruction. During this rapid phase, only participants that are close enough to $V_0$ can manipulate messages outputted in this rapid phase. The other ones are intuitively too far. Therefore, we mainly consider two transformations, namely $\mathcal{F}^<$ and $\mathcal{F}^{\geq}$, whose purposes are to transform a parametrised role of our process algebra given in Section III-A (with no `reset` instruction and no guarded input) into a process in the ProVerif calculus.

- *Transformation $\mathcal{F}^<$:* this transformation introduces the phase instructions with $i = 0$, 1 and 2 considering all the possible ways of splitting the role into three phases (0, 1, and 2). Each phase instruction is placed before an `in` instruction. Such a slicing is actually sufficient for our purposes.
- *Transformation $\mathcal{F}^{\geq}$:* this transformation does the same but we forbid the use of the instruction phase 1, jumping directly from phase 0 to phase 2.

The configuration, denoted $\mathcal{F}(\mathcal{T}, \overline{\mathcal{P}_{\mathsf{prox}}}, V_0, \mathcal{I}_0, t_0)$, is the tuple $(\mathcal{P}; \phi; 0)$ where $\phi$ is such that $img(\phi) = \bigcup_{a \in \mathcal{M}_0} \mathsf{Knows}(\mathcal{I}_0, a, \mathcal{A}_0)$, and $\mathcal{P}$ is the multiset that contains the following processes:

- $\mathsf{block}_1\ .\ 1 : \mathtt{out}(m)\ .\ \mathtt{in}^{<t}(x)\ .\ 2 : \mathsf{block}_2\ .\ \mathsf{end}(v_0, p_0)$;
- $!R(a_0, .., a_n)$ when $R(z_0, .., z_n) \in \mathcal{F}^<(\overline{\mathcal{P}_{\mathsf{prox}}})$, $a_0, .., a_n \in \mathcal{A}_0$, $\mathsf{Dist}_{\mathcal{T}}(v_0, a_0) < t_0$;
- $!R(a_0, .., a_n)$ when $R(z_0, .., z_n) \in \mathcal{F}^{\geq}(\overline{\mathcal{P}_{\mathsf{prox}}})$, $a_0, .., a_n \in \mathcal{A}_0$, $\mathsf{Dist}_{\mathcal{T}}(v_0, a_0) \geq t_0$.

We are then able to establish the following result that justifies the transformation presented above.

**Proposition 1.** *Let $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ be a topology, $\mathcal{P}_{\mathsf{prox}}$ a protocol, $t_0 \in \mathbb{R}_+$, $\mathcal{I}_0$ a template, and $V_0(z_0, z_1)$ a parametrised process of the form:*

$$\mathsf{block}_1\ .\ \mathtt{reset}\ .\ \mathtt{out}(m)\ .\ \mathtt{in}^{<t}(x)\ .\ \mathsf{block}_2\ .\ \mathsf{end}(z_0, z_1)$$

*with $t \leq 2 \times t_0$*

*Let $K_0$ be a valid initial configuration for the protocol $\mathcal{P}_{\mathsf{prox}}$ and $V_0$ w.r.t. $\mathcal{T}$ and $\mathcal{I}_0$. If $K_0$ admits an attack w.r.t. $t_0$-proximity in $\mathcal{T}$, then we have that:*

$$\mathcal{F}(\mathcal{T}_0, \overline{\mathcal{P}_{\mathsf{prox}}}, V_0, \mathcal{I}_0, t_0) \overset{\mathsf{tr}}{\Longrightarrow} (\{2 : \mathsf{end}(v_0, p_0)\} \uplus \mathcal{P}; \phi; 2).$$

*Moreover, in case there is no $a \in \mathcal{M}_0$ such that $\mathsf{Dist}_{\mathcal{T}_0}(a, v_0) < t_0$, we have that for any $\mathtt{in}(u)$ occurring in $\mathsf{tr}$ during phase 1, the underlying recipe $R$ is either of the form $\mathsf{w}$, or only uses handles ouputted in phase 0.*

$$
\begin{array}{rcl}
(i : \mathtt{in}(x).P \uplus \mathcal{P}; \phi; i) & \xrightarrow{\mathtt{in}(R\phi\downarrow)} & (i : P\{x \mapsto R\phi\downarrow\} \uplus \mathcal{P}; \phi; i) \text{ for some recipe } R \\
(i : !P \uplus \mathcal{P}; \phi; i) & \xRightarrow{\tau} & (i : P \uplus (i : !P) \uplus \mathcal{P}; \phi; i) \\
(\mathcal{P}; \phi; i) & \xrightarrow{\mathsf{phase}\ i'} & (\mathcal{P}; \phi; i') \quad \text{with } i' > i.
\end{array}
$$

Fig. 5: Semantics for processes with phases (some rules only)

This result allows us to turn any attack corresponding to a mafia fraud into a reachability property, namely the reachability of the event end. Regarding distance hijacking, in order to avoid false attacks, we will exploit the additional condition stated at the end of Proposition 1.

### C. Case studies

Regardless of the type of the considered attack, thanks to our reduction results (Section V), when analysing the protocol $\mathcal{P}_{\mathsf{prox}}$ w.r.t. $t_0$-proximity, we only need to consider a single topology (depicted in Figure 4). Once down to this single topology, we can apply Proposition 1, and analyse in ProVerif the reachability of the event $\mathsf{end}(v_0, p_0)$ starting with the configuration $(\mathcal{P}; \phi; 0) = \mathcal{F}(\mathcal{T}^{t_0}_{\mathsf{XX}}, \overline{\mathcal{P}_{\mathsf{prox}}}, V_0, \mathcal{I}_0, t_0)$ where $\mathsf{XX} \in \{\mathsf{MF}, \mathsf{DH}\}$. If the protocol is proved secure, then $\mathcal{P}_{\mathsf{prox}}$ is resistant to the class of attacks we have considered. Otherwise, the trace returned by ProVerif can be analysed to see if it is executable in our timed semantics, and thus corresponds to a real attack.

Actually, to obtain meaningful results regarding scenarios that only involved honest participants in the neighbourhood of $v_0$, we have to go one step further. Indeed, the attacker model behind ProVerif allows him to interact with any participant (even those that are far away) with no delay. To avoid these behaviours that are not possible in the rapid phase, we slightly modify the ProVerif code taking advantage of the extra condition stated in Proposition 1. During phase 1, we consider an attacker who is only able to forward messages previously sent, and forged new messages using his knowledge obtained in phase 0.

**Distance bounding protocols:** We apply our methodology to a number of well-known distance bounding protocols. In symbolic models, it is not possible to reason at the bit-level, and therefore we replace the bit-sized exchanges by a single challenge-response exchange using a fresh nonce (as done in Example 4). Sometimes, we also abstract the answer from the prover relying on an uninterpreted function symbol with relevant arguments. Finally, in order to rely on ProVerif, the xor operator has been abstracted (even if our theoretical development is generic enough to deal with such an operator). In constrast with [11] in which the xor operator is fully abstracted relying on a non interpreted function symbol, we model it as follows:

$$
\begin{array}{llll}
(x \oplus y) \oplus x \to y & & (x \oplus y) \oplus y \to x \\
x \oplus (x \oplus y) \to y & & y \oplus (x \oplus y) \to x.
\end{array}
$$

A brief description of some of the protocols is given in Appendices, and all the results are presented in Table I. For instance, we succeed in proving resistance against mafia fraud for the first version of the Meadows *et al.* protocol which could not be proved using the framework proposed in [23]. The results are consistent with the ones obtained in [8], [11]. Moreover, our method enables us to retrieve automatically the distance hijacking attacks already known on the Meadows *et al.* protocol.

| Protocols | MF | DH |
|---|---|---|
| Brands and Chaum [2] | ✓ | × |
| Meadows *et al.* $(n_V \oplus n_P, P)$ [23] | ✓ | ✓ |
| Meadows *et al.* $(n_V, n_P \oplus P)$ [23] | ✓ | × |
| TREAD-Asymmetric [24] | × | × |
| TREAD-Symmetric [24] | ✓ | × |
| MAD (One-Way) [25] | ✓ | × |
| Swiss-Knife [3] | ✓ | ✓ |
| Munilla *et al.* [26] | ✓ | ✓ |
| CRCS [27] | ✓ | × |
| Hancke and Kuhn ⋆ [28] | ✓ | ✓ |

⋆ the protocols Tree-based, Poulidor, and Uniform are actully equivalent to this one.

TABLE I: Results on our case studies and obtained in less than one second.
(×: attack found, ✓: proved secure)

**Paysafe protocol:** We studied the Paysafe payment protocol [1] designed to be resistant against mafia fraud attacks. More generally, contactless payment protocols need to prevent relay attacks where malicious agents would abuse from an honest agent to perform a payment, which corresponds to the mafia fraud scenario.

The Paysafe protocol is schematised in Figure 6 where plain arrows represents the rapid exchange phase. During the initialisation phase, the reader and the card exchange some identifiers, while during the authentication exchange, the reader ensures that the card is legitimate using signatures and certificates verifications. The main idea is to send nonces and constants during the rapid phase and to perform all the necessary checks later on. The aim is to increase the accuracy on the proximity property needed to ensure the security of the protocol. We also considered two other versions of PaySafe, also described in [1], where nonces from the Reader and the Card are removed.

Our results confirmed those presented in [1]. One would note that their methodology and ours, especially when it comes down to the use of ProVerif, are quite similar but we would like to emphasise the fact that our use of ProVerif is a consequence of our formal development. The authors of [11] reported a distance fraud attack which is not relevant in this context. Their methodology does not enable to restrict the analysis to mafia fraud scenarios. In contrast, our methodology is flexible enough and more suitable in this context.

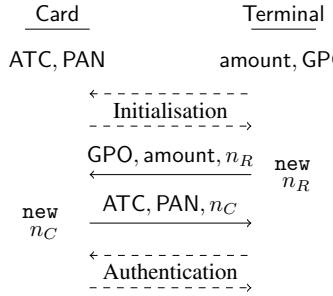



Fig. 6: PaySafe (simplified)

## VII. Conclusion

Regarding physical proximity, we have shown two main reduction results: if there is an attack on an arbitrary topology then there is an attack on a simple one having at most four nodes. Relying on these reduction results, we have shown how to use ProVerif to analyse several protocols provided they make use of primitives supported by the tool. Our methodology is flexible enough to draw meaningful conclusions on each class of attacks: hijacking attack, and mafia fraud.

As future work, we would like to extend our result to consider the notion of terrorist fraud. This would require to consider dishonest participants who only share a part of their knowledge. Another possible extension would be to take also into account the fact that computing messages takes time as it was done *e.g.* in [29], or to consider different channels speeds.

## References


[1] T. Chothia, F. D. Garcia, J. de Ruiter, J. van den Breekel, and M. Thompson, "Relay cost bounding for contactless EMV payments," in *Proc. 19th International Conference on Financial Cryptography and Data Security (FC'15)*, ser. Lecture Notes in Computer Science, vol. 8975. Springer, 2015, pp. 189–206.

[2] S. Brands and D. Chaum, "Distance-bounding protocols," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1993, pp. 344–359.

[3] C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira, "The swiss-knife RFID distance bounding protocol," in *International Conference on Information Security and Cryptology*. Springer, 2008, pp. 98–115.

[4] A. Armando, R. Carbone, L. Compagna, J. Cuéllar, and M. L. Tobarra, "Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for Google apps," in *Proc. 6th ACM Workshop on Formal Methods in Security Engineering (FMSE'08)*. ACM, 2008, pp. 1–10.

[5] A. Armando *et al.*, "The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures," in *Proc. 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, vol. 7214. Springer, 2012, pp. 267–282.

[6] B. Blanchet, "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules," in *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, 2001, pp. 82–96.

[7] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The Tamarin Prover for the Symbolic Analysis of Security Protocols," in *Proc. 25th International Conference on Computer Aided Verification (CAV'13)*, ser. LNCS, vol. 8044. Springer, 2013, pp. 696–701.

[8] C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun, "Distance hijacking attacks on distance bounding protocols," in *Proc. IEEE Symposium on Security and Privacy (S&P'12)*. IEEE, 2012, pp. 113–127.

[9] D. Basin, S. Capkun, P. Schaller, and B. Schmidt, "Formal reasoning about physical properties of security protocols," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 2, p. 16, 2011.

[10] M. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, and C. Talcott, "Towards timed models for cyber-physical security protocols," in *Joint Workshop on Foundations of Computer Security and Formal and Computational Cryptography*, 2014.

[11] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua, "Distance-bounding protocols: Verification without time and location," in *2018 IEEE Symposium on Security and Privacy (S&P)*, vol. 00, pp. 152–169. [Online]. Available: doi.ieeecomputersociety.org/10.1109/SP.2018.00001

[12] H. Comon-Lundh and V. Cortier, "Security properties: two agents are sufficient," *Programming Languages and Systems*, pp. 99–113, 2003.

[13] A. Debant, S. Delaune, and C. Wiedling, "http://people.irisa.fr/alexandre.debant/proving-physical-proximity-using-symbolic-methods.html."

[14] ——, "Proving physical proximity using symbolic models," Univ Rennes, CNRS, IRISA, France, Research Report, Feb. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01708336

[15] G. Avoine, M. A. Bingöl, S. Kardaş, C. Lauradoux, and B. Martin, "A framework for analyzing RFID distance bounding protocols," *Journal of Computer Security*, vol. 19, no. 2, pp. 289–317, 2011.

[16] G. Avoine, M. Bingol, I. Boureanu, S. Capkun, G. Hancke, S. Kardas, C. Kim, C. Lauradoux, B. Martin, J. Munilla *et al.*, "Security of distance-bounding: A survey," *ACM Computing Surveys*, 2017.

[17] I. Boureanu, A. Mitrokotsa, and S. Vaudenay, "Practical and provably secure distance-bounding," *Journal of Computer Security*, vol. 23, no. 2, pp. 229–257, 2015.

[18] V. Nigam, C. Talcott, and A. A. Urquiza, "Towards the automated verification of cyber-physical security protocols: Bounding the number of timed intruders," in *Proc. 21st European Symposium on Research in Computer Security (ESORICS'16)*. Springer, 2016, pp. 450–470.

[19] M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," in *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*. ACM Press, 2001, pp. 104–115.

[20] V. Cortier, J. Degrieck, and S. Delaune, "Analysing routing protocols: four nodes topologies are sufficient," in *International Conference on Principles of Security and Trust*. Springer, 2012, pp. 30–50.

[21] B. Blanchet, "Modeling and verifying security protocols with the applied pi calculus and proverif," *Foundations and Trends in Privacy and Security*, vol. 1, no. 1-2, pp. 1–135, 2016. [Online]. Available: https://doi.org/10.1561/3300000004

[22] D. Dolev and A. C. Yao, "On the security of public key protocols," in *Proc. 22nd Symposium on Foundations of Computer Science (FCS'81)*. IEEE Computer Society Press, 1981, pp. 350–357.

[23] C. Meadows, R. Poovendran, D. Pavlovic, L. Chang, and P. Syverson, "Distance bounding protocols: Authentication logic analysis and collusion attacks," in *Secure localization and time synchronization for wireless sensor and ad hoc networks*. Springer, 2007, pp. 279–298.

[24] G. Avoine, X. Bultel, S. Gambs, D. Gerault, P. Lafourcade, C. Onete, and J.-M. Robert, "A terrorist-fraud resistant and extractor-free anonymous distance-bounding protocol," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 800–814.

[25] S. Čapkun, L. Buttyán, and J.-P. Hubaux, "Sector: secure tracking of node encounters in multi-hop wireless networks," in *Proc. 1st ACM workshop on Security of ad hoc and sensor networks*. ACM, 2003, pp. 21–32.

[26] J. Munilla and A. Peinado, "Distance bounding protocols for rfid enhanced by using void-challenges and analysis in noisy channels," *Wireless communications and mobile computing*, vol. 8, no. 9, pp. 1227–1232, 2008.

[27] K. B. Rasmussen and S. Capkun, "Realization of rf distance bounding." in *USENIX Security Symposium*, 2010, pp. 389–402.

[28] G. P. Hancke and M. G. Kuhn, "An RFID distance bounding protocol," in *Proc. 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*. IEEE, 2005, pp. 67–73.

[29] V. Cheval and V. Cortier, "Timing attacks in security protocols: symbolic framework and proof techniques," in *Proc. 4th Conference on Principles of Security and Trust (POST'15)*, ser. LNCS, vol. 9036. Springer, Apr. 2015, pp. 280–299.

## APPENDIX A
## SKETCHES OF PROOF OF OUR MAIN RESULTS

**Theorem 1.** *Let $\mathcal{I}_0$ be a template, $\mathcal{P}_{\mathsf{prox}}$ be a protocol $\mathcal{I}_0$-executable, and $V_0(z_0, z_1)$ be a parametrised role containing the special event $\mathsf{end}(z_0, z_1)$. We have that $\mathcal{P}_{\mathsf{prox}}$ 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 $\mathcal{T}^{t_0}_{\mathsf{MF}}$.*

*Proof. (Sketch)* We consider an attack trace in $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0) \in \mathcal{C}_{\mathsf{MF}}$.

$K_0 \rightarrow^*_{\mathcal{T}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor^{t_v}_{v_0} \uplus \mathcal{P}; \Phi; t)$ with $\mathsf{Dist}_{\mathcal{T}}(v_0, p_0) \geq t_0$.

We proceed in three main steps:

1) We reduce the number of active agents (those that are actually executing a process) - we do this for honest and malicious agents. We transform honest agent (but $v_0$ and $p_0$) into malicious ones. This intuitively gives more power to the attacker, and malicious agents in the neighborhood of $v_0$ are allowed in a mafia fraud scenario. Then, relying on our executability condition, we discard processes executed by malicious agents. These actions can actually be mimicked by an attacker located at the same place.
2) We reduce the number of attackers by placing them ideally (one close to each honest agent). Since we have removed all honest agents but two, we obtain a topology with only two dishonest agents.
3) To conclude, we reduce the knowledge showing that we can project all the dishonest agents that are located in $p_0$ on $a_2$ and all the dishonest agents that are located in $v_0$ on $a_1$.

□

**Theorem 2.** *Let $\mathcal{I}_0$ be a template, $\mathcal{P}_{\mathsf{prox}}$ be a protocol, $t_0 \in \mathbb{R}_+$, and $V_0(z_0, z_1)$ be a parametrised role obtained using the following grammar:*

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

*where $x \in \mathcal{X}$, $n \in \mathcal{N}$, $u, u' \in \mathcal{T}(\Sigma^+_c, \mathcal{X} \cup \mathcal{N} \cup \{z_0, z_1\})$, $v \in \mathcal{T}(\Sigma^+, \mathcal{X} \cup \mathcal{N} \cup \{z_0, z_1\})$ and $t \leq 2 \times t_0$.*
*If $\mathcal{P}_{\mathsf{prox}}$ admits a distance hijacking attack w.r.t. $t_0$-proximity, then $\overline{\mathcal{P}_{\mathsf{prox}}}$ admits an attack against $t_0$-proximity in the topology $\mathcal{T}^{t_0}_{\mathsf{DH}}$.*

In order to establish this result, we will first transform the initial attack trace into an "attack" trace in an untimed model. This model (with no timing constraints to fullfill) is more suitable to reorder some actions in the trace. We will show in a second step how to come back in the original timed model. We consider the untimed configuration associated to a configuration $K = (\mathcal{P}; \Phi; t)$. Formally, we have $\mathsf{untimed}(K) = (\mathcal{P}'; \Phi')$ with:

$$\mathcal{P}' = \{ \lfloor P \rfloor_a \mid \lfloor P \rfloor^t_a \in \mathcal{P} \}, \text{ and}$$
$$\Phi' = \{ w \xrightarrow{a} u | \mathsf{w} \xrightarrow{a,t} u \in \Phi \}.$$

Then, we consider a *relaxed semantics* over untimed configurations: $K \overset{a,\alpha}{\rightsquigarrow}_{\mathcal{T}} K'$ if there exist $K_0$ and $K'_0$ such that $K_0 \xrightarrow{a,\alpha}_{\mathcal{T}} K'_0$ (for some rule other than the TIM rule), and for which $K = \mathsf{untimed}(K_0)$ (resp. $K' = \mathsf{untimed}(K'_0)$).

Under the same hypotheses as those stated in Theorem 2, we establish a result that allows one to "clean" an attack trace by pushing instructions (before or after) outside the rapid phase delimited by a `reset` and its following guarded input `in`. In the resulting trace, the only remaining actions in the rapid phase are those performed by agents who are close to $v_0$.

**Proposition 2.** *Let $K_0$ be a valid initial configuration for $\overline{\mathcal{P}_{\mathsf{prox}}}$ and $V_0$ w.r.t. a topology $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}, v_0, p_0)$ and $\mathcal{I}_0$. If $K_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}} K_1$ then there exists an execution $K'_0 \overset{\mathsf{tr}'}{\rightsquigarrow} K'_1$ such that $K'_i = \mathsf{untimed}(K_i)$ for $i \in \{0, 1\}$.*

*Moreover, for any sub-execution of $K'_0 \overset{\mathsf{tr}'}{\rightsquigarrow} K'_1$ of the form*

$$(\lfloor \mathtt{reset}.P \rfloor_{v_0} \uplus \mathcal{P}; \Phi_{\mathtt{reset}}) \overset{v_0,\tau}{\rightsquigarrow} (\lfloor P \rfloor_{v_0} \uplus \mathcal{P}; \Phi_{\mathtt{reset}}) \overset{\mathsf{tr}'_0}{\rightsquigarrow} K^-_{\mathtt{in}} \overset{v_0,\mathtt{in}^{<t}(u)}{\rightsquigarrow} K'_{\mathtt{in}}$$

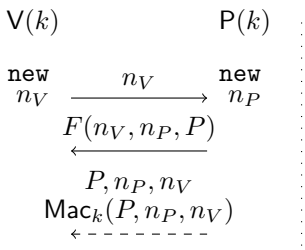
*where $\mathsf{tr}'_0$ only contains actions $(a, \alpha)$ with $\alpha \in \{\tau, \mathtt{out}(u), \mathtt{in}(u)\}$, we have that:*

- *$2 \times \mathsf{Dist}_{\mathcal{T}}(v_0, a) < t$ for any $(a, \alpha) \in \mathsf{tr}'_0$;*
- *for any $(a, \mathtt{in}^\star(v))$ occurring in $\mathsf{tr}'_0.(v_0, \mathtt{in}^{<t}(u))$, the agent $b$ responsible of the output and the recipe $R$ (as defined in Figure 2) are such that either $2\mathsf{Dist}_{\mathcal{T}}(v_0, b) < t$, or $vars(R) \subseteq dom(\Phi_{\mathtt{reset}})$.*

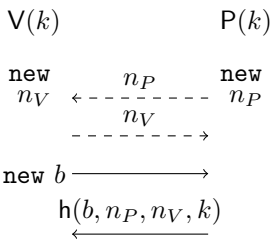
Relying on Proposition 2, we are then able to prove Theorem 2.

1) We start by removing `reset` instructions and by transforming any guarded input `in` (but those in $V_0$) into simple inputs. The resulting trace is still an attack trace w.r.t. $\overline{\mathcal{P}_{\mathsf{prox}}}$.
2) Then, we apply Proposition 2 in order to obtain an attack trace in the relaxed semantics. We will exploit the extra conditions given by Proposition 2 in order to lift the trace in the timed model at step 4.
3) We now consider another topology $\mathcal{T}'$ with two locations (as $\mathcal{T}^{t_0}_{\mathsf{DH}}$) and such that agents close to $v_0$ are now located with $v_0$, and those that are far away from $v_0$ in $\mathcal{T}$ are now located with $p_0$. This execution is still a valid trace in $\mathcal{T}'$ since we consider the relaxed semantics.
4) Then, to lift this execution trace into our timed model, the basic idea is to wait enough time before a `reset` instruction to allow messages to be received by all the participants before starting the rapid phase.
5) To conclude, as in the previous attack scenarios, we reduce the initial knowledge and the number of agents by applying a renaming on agent names. □

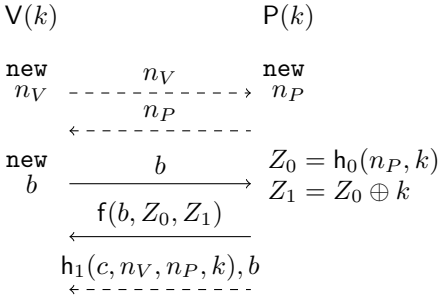
## APPENDIX B
## BRIEF DESCRIPTION OF OUR CASE STUDIES



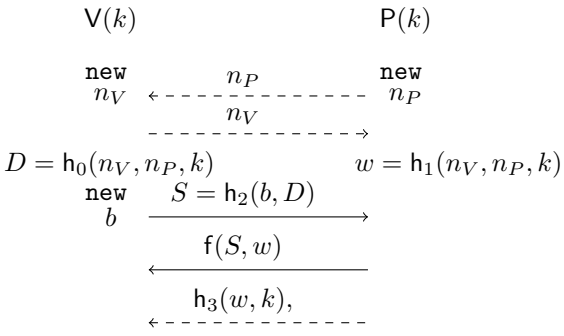

Meadows *et al.*



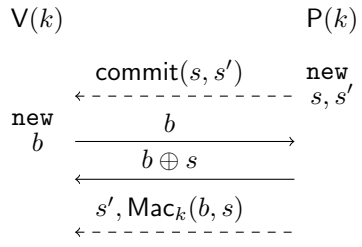

Hancke and Kuhn



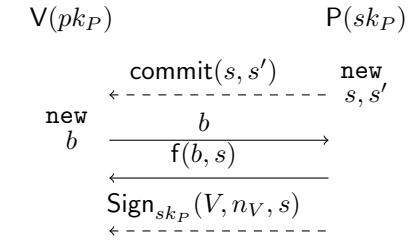

Swiss-Knife



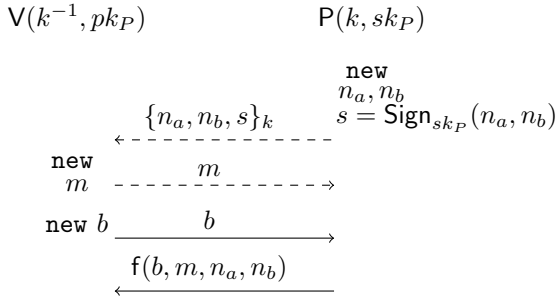

Munilla *et al.*




MAD (One-Way)




CRCS




TREAD