A symbolic framework to analyse physical proximity in security protocols

Alexandre Debant
Univ Rennes, CNRS, IRISA, France

Stéphanie Delaune
Univ Rennes, CNRS, IRISA, France

Cyrille Wiedling
Univ Rennes, CNRS, IRISA, France

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 protocols 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. These reduction results allow one to use verification tools (e.g. ProVerif, Tamarin) developed for analysing more classical security properties. As an application, we analyse several distance bounding protocols, as well as a contactless payment protocol.

2012 ACM Subject Classification Theory of computation Program verification

Keywords and phrases cryptographic protocols, verification, formal methods, process algebra, Dolev-Yao model

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.29

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

1 Introduction

The shrinking size of microprocessors and 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 [14]. 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 communications, obtaining reliable information regarding physical proximity is of paramount importance and specific protocols, namely distance bounding protocols, were proposed to achieve this specific goal [11, 26]. 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.

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 [2]. 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 [3]. The techniques used in symbolic models have become mature and several verification tools are nowadays available, *e.g.* ProVerif [8], Tamarin [29].

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. Following [7, 24], and more recently [27], 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 restrictions. These constraints apply 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. Moreover each agent has clocks to be able to perform time measurements. Then, our main contribution is to provide reduction results in the spirit of the one obtained in [15] for traditional protocols: if there is an attack, then there is one considering only few participants at some specific locations. As it is usually done in distance bounding protocols, we consider different types of attacks : mafia fraud and hijacking attack. A mafia fraud is an attack in which an attacker tries to convince the verifier that an honest prover is close to him whereas he is far away. The notion of distance hijacking attack has been introduced more recently [17]. In such a scenario, a dishonest prover located far away succeeds in convincing a verifier that they are actually close, and he may only exploit the presence of honest participants in the neighboorhood to achieve his goal. Our results slightly differ depending on the type of attacks we consider and allow 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). They hold in a rather general setting: we consider arbitrary cryptographic primitives as soon as they can be expressed using rewriting rules modulo an equational theory.

An interesting consequence of our reduction results is that it allows one to use techniques and tools developed so far for traditional security protocols. As an application, we analyse several distance bounding protocols (relying on the automatic ProVerif tool), and a contactless payment protocol [14]. We confirmed some known vulnerabilities in a number of protocols, and discovered an unreported attack on the SPADE protocol [12]. All files related to our case studies as well as a full version of this paper are available [18, 19].

Related work. Recent efforts have been made on proving security of distance bounding protocols. For instance, in 2011, Avoine *et al.* [5] proposed a framework in which many protocols have been analysed and compared in a unified manner [4]. A rather general model has been proposed by Boureanu *et al.* in [10]. This computational model captures all the classical types of attacks and generalises them enabling attackers to interact with many

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:2–29:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 [7] and [27], another one based on strand spaces is available in [31]. Even if our model shares some similarities with those mentioned above, we design a new one based on the applied pi calculus [1] 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 [16] 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 [31]: they show that n attackers are sufficient when analysing a configuration involving at most n honest participants. However, an arbitrary number of participants is still needed 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 distance hijacking scenarios for which the presence of an attacker in the neighbourhood of the verifier is disallowed. 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. In particular, this allows us to leverage existing verification tools such as ProVerif and Tamarin. To do that we get some inspiration from [14]. Our contributions improve upon their work by providing a strong theoretical foundation to their idea.

Recently, a methodology to analyse distance bounding protocols within Tamarin has been proposed [27]. They do not try to define the different class of attacks as usually considered in distance bounding protocols. Instead, they provide a generic definition of secure distance bounding: when an honest verifier successfully ends a session with a prover P, then he has correctly computed an upper bound on his distance to either P (if P is honest) or to some other dishonest participant P' (if P is dishonest). The security analysis is then performed w.r.t. such a generic security property. This prevents them to draw meaningful conclusions on protocols such as Paysafe which is *not* supposed to resist to some particular classes of attacks.

2 Modelling timed security protocols

As usual in symbolic models, we rely on a term algebra for modelling messages exchanged by the participants, and on a process algebra to represent the protocols themselves.

2.1 Messages as terms

We consider two infinite and disjoint sets of *names*: a set $\mathcal{N}$ of *basic names* used to represent keys, nonces, and a set $\mathcal{A}$ of *agent names* used to represent agents identities. We consider an infinite set Σ_0 of constant symbols that are used *e.g.* 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 those in $\mathcal{W}$, namely *handles*, are used to store messages learnt by the attacker.

We assume a signature Σ, *i.e.* a set of function symbols together with their arity. The elements of Σ are split into *constructor* and *destructor* symbols, *i.e.* $\Sigma = \Sigma_c \uplus \Sigma_d$. We denote $\Sigma^+ = \Sigma \uplus \Sigma_0$, and $\Sigma_c^+ = \Sigma_c \uplus \Sigma_0$. Given a signature $\mathcal{F}$, and a set of atomic data A, we denote by $\mathcal{T}(\mathcal{F}, \mathsf{A})$ the set of *terms* built from atomic data A by applying function symbols in $\mathcal{F}$. A *constructor term* is a term in $\mathcal{T}(\Sigma_c^+, \mathcal{N} \uplus \mathcal{A} \uplus \mathcal{X})$. We denote $vars(u)$ the set of

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:3–29:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 σ 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 $\Sigma_{\mathsf{ex}} = \Sigma_c \uplus \Sigma_d$:

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

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 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 and ok a specific constant.

Following the approach developed in [9], 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 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:

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

We also give a meaning to destructor symbols 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 θ. 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 Σ into two parts, Σ_{pub} and Σ_{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 handles 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 Σ_{ex}, only sk is in Σ_{priv}. The properties of the symbols in Σ_d are reflected through the following rewriting rules:

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

2.2 Protocols as processes

Protocols are modelled through processes using the following grammar:

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

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 process executing this action. This construction is in contrast with the approach proposed in [27] where input actions are not subject to any timing constraint, and are therefore

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:4–29:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 process. Finally, the process $\texttt{let}\ x = v\ \texttt{in}\ P$ 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: $\texttt{let}\ x = \mathsf{eq}(u, v)\ \texttt{in}\ P$.

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 [11] that is informally described on the right. 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, this 2-party protocol is modelled through the two following parametrised processes: $V(z_V, z_P)$ represents the role of the verifier played by agent z_V with agent z_P whereas $P(z'_P)$ represents the role of the prover played by agent z'_P.

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. $P \to V$: $\mathsf{sign}(\langle n, n \oplus m\rangle, \mathsf{sk}(P))$.

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

$P(z'_P) :=$
$\quad \texttt{new}\ m.\texttt{new}\ k.$
$\quad \texttt{out}(\mathsf{commit}(m, k)).$
$\quad \texttt{in}(x_n).$
$\quad \texttt{out}(x_n \oplus m).$
$\quad \texttt{out}(k).$
$\quad \texttt{out}(\mathsf{sign}(\langle x_n, x_n \oplus m\rangle, \mathsf{sk}(z'_P))).0$

2.3 Semantics

The operational semantics is defined using a relation over configurations, and is parametrised by a topology reflecting the fact that interactions between agents depend on their location.

▶ **Definition 5.** 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.

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:5–29:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In our model, the distance between two agents is expressed by the time it takes for a message to travel from one to another, $\mathsf{Dist}_{\mathcal{T}_0} : \mathcal{A}_0 \times \mathcal{A}_0 \to \mathbb{R}$, 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. This results in a distributed attacker with restricted, but more realistic, communication capabilities than those of the traditional omniscient Dolev-Yao attacker [21].

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. Messages that have been outputted so far are stored into a *frame* (introduced in [1]) extended to keep track of the time at which the message has been outputted and by whom.

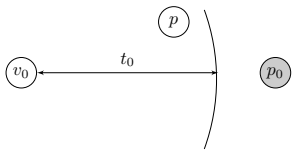
▶ **Definition 6.** Given a topology $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$, a *configuration* K over $\mathcal{T}_0$ is a tuple $(\mathcal{P}; \Phi; t)$, where:

- $\mathcal{P}$ 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 $1 \leq i \leq n$;
- $t \in \mathbb{R}_+$ is the global time of the system.

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

$$\lfloor \Phi \rfloor_a^t = \left\{ \mathsf{w}_i \xrightarrow{a_i,t_i} u_i \mid (\mathsf{w}_i \xrightarrow{a_i,t_i} u_i) \in \Phi \text{ and } a_i = a \text{ and } t_i \leq t \right\}.$$

▶ **Example 7.** Continuing Example 4, we may consider the topology $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ depicted on the right 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 $\mathsf{Dist}_{\mathcal{T}_0}(p, v_0) < t_0$ whereas $\mathsf{Dist}_{\mathcal{T}_0}(p_0, v_0) \geq t_0$.



A typical initial 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 prover's role, and v_0 the verifier's role with a dishonest agent p_0. The signature key of this dishonest participant is given to the attacker through w_1. A more realistic configuration would include other instances of these two roles and will give more knowledge to the attacker. This simple configuration is sufficient to present an attack.

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 1.

The TIM rule allows time to elapse, meaning that the global clock as well as the local clocks will be shifted by δ:

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

The RST rule allows an agent to reset the local clock of the process. 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.

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:6–29:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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

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

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

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

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

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

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 $t_a < t_g$.

Figure 1 Semantics of our calculus

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

▶ **Example 8.** Continuing Example 7, we may consider the following execution:

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

where $K_1 = (\lfloor P_1 \rfloor_p^{\delta_0} \uplus \lfloor V_1 \rfloor_{v_0}^{0};\{\mathsf{w}_1 \xrightarrow{p_0,0} \mathsf{sk}(p_0),\ \mathsf{w}_2 \xrightarrow{p,0} \mathsf{commit}(m',k')\};\delta_0)$. The two first arrows correspond to applications of the rule NEW to generate m' and k', the one without label is an instance of the TIM rule, and the two last arrows correspond respectively to the rule NEW and the rule RST. We have that:

- $P_1 = \mathtt{in}(x_n).\mathtt{out}(x_n \oplus m').\mathtt{out}(k').\mathtt{out}(\mathsf{sign}(\langle x_n, x_n \oplus m' \rangle, \mathsf{sk}(p)))$; and
- $V_1 = \mathtt{out}(n').\mathtt{in}^{<2\times t_0}(y_0).\mathtt{in}(y_k).\mathtt{in}(y_{\mathsf{sign}}).\mathtt{let}\ y_m = \mathsf{open}(\mathsf{commit}(m',k'),y_k)\ \mathtt{in}\ldots$

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.

3 Modelling physical proximity

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. Before to consider the type of frauds we are interested in, we first introduce the notion of valid initial configuration that aims to represent the scenarios that have to be analysed once the topology is fixed.

3.1 Valid initial configurations

We consider a topology $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$, a protocol $\mathcal{P}_{\mathsf{prox}}$ i.e. a set of roles), and we assume that the initial knowledge of dishonest participants is given through a template

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:7–29:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$\mathcal{I}_0$, 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:

$$\mathsf{Knows}(\mathcal{I}_0, a, \mathcal{A}_0) = \left\{ (u_0\{z_0 \mapsto a\})\sigma \text{ ground} \;\middle|\; \begin{array}{l} u_0 \in \mathcal{I}_0 \text{ and} \\ \sigma \text{ a substitution such that } img(\sigma) \subseteq \mathcal{A}_0 \end{array} \right\}$$

For the sake of simplicity, we extend our calculus with a special action of the form $\mathsf{end}(z_0, z_1)$ and we assume that configurations representing instances of distance bounding protocols contain a process (typically a session of the verifier) that ends with it. When analysing physical proximity, we consider any valid initial configuration as defined below:

▶ **Definition 9.** 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 $\lfloor P' \rfloor_{a'}^{t_{a'}} \in \mathcal{P}'$ 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, i.e. the agent a' who executes the role must be the first argument of the parametrised process. The second condition allows one to give some initial knowledge to each malicious node. We may note that we do not give any constraint regarding time. It is indeed important that all the possible initial configurations are analysed before declaring a protocol secure.

▶ **Example 10.** 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.

3.2 Mafia fraud and distance hijacking

A *mafia fraud* [20] is an attack in which generally three agents are involved: a verifier, an honest prover located outside the neighbourhood of the verifier, and an attacker. We consider here its general version which may involve an arbitrary number of participants. The aim of the attacker is to convince the verifier that the honest prover is actually close to it. We denote by $\mathcal{C}_{\mathsf{MF}}$ the set of all the mafia fraud topologies, i.e. any topology $\mathcal{T}$ such that $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ with $v_0, p_0 \in \mathcal{A}_0 \setminus \mathcal{M}_0$.

A *distance hijacking fraud* [17] is an attack in which a dishonest prover located far away succeeds in convincing a verifier that he is actually close to him. The dishonest prover may exploit honest entities located in the neighbourhood of the verifier. We denote by $\mathcal{C}_{\mathsf{DH}}$ the set of all the distance hijacking topologies, i.e. any topology $\mathcal{T}$ such that $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ with $p_0 \in \mathcal{M}_0$, $v_0 \in \mathcal{A}_0 \setminus \mathcal{M}_0$, and $\mathsf{Dist}_{\mathcal{T}_0}(v_0, a) \geq t_0$ for any $a \in \mathcal{M}_0$.

▶ **Definition 11.** 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:

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

We also say that K admits an attack w.r.t. t_0-proximity in $\mathcal{T}$.

In other words, there is an attack if starting from an initial valid configuration, the verifier a_1 successfully ends a session with an agent a_2 who is far away.

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:8–29:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

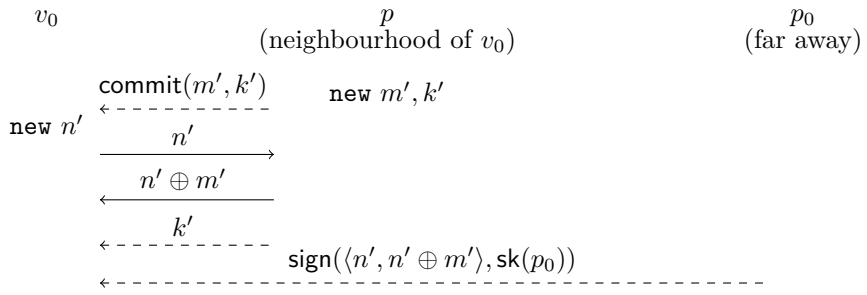


Figure 2 Distance hijacking attack on the Brands and Chaum's protocol

▶ **Example 12.** As reported in [17], the Brands and Chaum protocol is actually vulnerable to a *distance hijacking attack*. This attack is informally depicted in Figure 2. A honest prover who is in the neightbouhood of a legitimate verifier v_0 starts a session. At the end of the session, the dishonest prover p_0 who is far away hijacks the honest prover by sending a signature of the transcript of the rapid phase with his own signature key, namley $\mathsf{sk}(p_0)$. Upon reception of this signature, the verifier v_0 will believe that he played the session with p_0, and will wrongly conclude that p_0 is in his neighbourhood. Note that, the rapid phase (plain lines) can only be done by a prover who is in the neighbourhood of v_0 due to the guarded input that occurs in the verifer's role played by v_0.

We explain below how this attack is captured in our model. Continuing Example 7, 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 8:

$$K'_1 = (\lfloor P_1 \rfloor_p^{\delta_0} \uplus \lfloor V'_1 \rfloor_{v_0}^0 ; \{\mathsf{w}_1 \xrightarrow{p_0,0} \mathsf{sk}(p_0),\ \mathsf{w}_2 \xrightarrow{p,0} \mathsf{commit}(m', k')\}; \delta_0)$$

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 as follows:

$$\begin{aligned} &K'_1 \xrightarrow{v_0,\mathtt{out}(n')} \mathcal{T}_0 \to \mathcal{T}_0 \xrightarrow{p,\mathtt{in}(n')} \mathcal{T}_0 \\ &\xrightarrow{p,\mathtt{out}(n' \oplus m')} \mathcal{T}_0 \xrightarrow{p,\mathtt{out}(k')} \mathcal{T}_0 \to \mathcal{T}_0 \xrightarrow{v_0,\mathtt{in}^{<2\times t_0}(n' \oplus m')} \mathcal{T}_0 \xrightarrow{v_0,\mathtt{in}(k')} \mathcal{T}_0 \\ &\to \mathcal{T}_0 \xrightarrow{v_0,\mathtt{in}(\mathsf{sign}(\langle n', n' \oplus m' \rangle, \mathsf{sk}(p_0)))} \mathcal{T}_0\ (\lfloor P_2 \rfloor_p^{3\delta_0 + 2\delta'_0} \uplus \lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{2\delta_0 + 2\delta'_0} ; \Phi; 3\delta_0 + 2\delta'_0). \end{aligned}$$

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 $\delta_0 = \mathsf{Dist}_{\mathcal{T}_0}(v_0, p) = \mathsf{Dist}_{\mathcal{T}_0}(p, v_0)$ to allow the sent message to reach its destination and the guarded input passes because p is close to v_0. 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. $\delta'_0 = \mathsf{Dist}_{\mathcal{T}_0}(v_0, p_0)$), and some time is needed for the forged message to reach v_0 (actually $\delta'_0 = \mathsf{Dist}_{\mathcal{T}_0}(v_0, p_0)$). 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.

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:9–29:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

4 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 (see Figure 3) without missing any attacks.

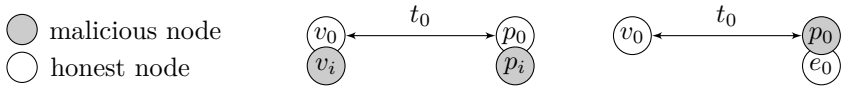


Figure 3 Topologies $\mathcal{T}_{\mathsf{MF}}^{t_0}$ (mafia fraud), and $\mathcal{T}_{\mathsf{DH}}^{t_0}$ (distance hijacking fraud)

4.1 Mafia fraud

A simple idea to reduce towards the topology $\mathcal{T}_{\mathsf{MF}}^{t_0}$ would 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 inspiration from [31], 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 [31], 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 lead to a feasible execution, we need an extra assumption. We require that each role of the protocol is executable. This is a reasonable assumption that only put weird protocols aside. This allows us to discard any role executed by a malicious participant. Intuitively, all these operations will be done directly by the attacker.

▶ **Definition 13.** 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 14.** Going back to our running example given in Example 4. We have that $V(z_0, z_1)$ is $\{z_1\}$-executable, and $P(z_0)$ is $\{\mathsf{sk}(z_0)\}$-executable. Thus, the protocol made of these two roles is $\{\mathsf{sk}(z_0), z_1\}$-executable.

We are now able to state our main reduction result regarding mafia fraud.

▶ **Theorem 15.** *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 $\mathcal{T}_{\mathsf{MF}}^{t_0}$.*

Proof. Since $\mathcal{T}_{\mathsf{MF}}^{t_0} \in \mathcal{C}_{\mathsf{MF}}$, we have that the existence of an attack in $\mathcal{T}_{\mathsf{MF}}^{t_0}$ is a mafia fraud. Regarding the other direction, we present the main steps of the proof below. A detailed proof is available in [19].

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:10–29:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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

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

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. In the spirit of [31], 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 p_i and all the dishonest agents that are located in v_0 on v_i.

◀

4.2 Distance hijacking attack

First, we may note that the reduction we did in case of mafia fraud is not possible anymore. 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).

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 16.** *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:*

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

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:

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

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:11–29:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 16, 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 1. 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 tr_0' only contains actions (a,α) 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 1) are such that either $2 \times \mathsf{Dist}_{\mathcal{T}}(v_0, b) < t$, or $vars(R) \subseteq dom(\Phi_{\mathtt{reset}})$.

Relying on Proposition 1, we are then able to provide a sketch of proof for Theorem 16 (the full and detailed proof is available in [19]).

1. We start by removing `reset` instructions and by transforming any guarded input $\mathtt{in}^{<}$ (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 1 in order to obtain an attack trace in the relaxed semantics. We will exploit the extra conditions given by Proposition 1 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 $\mathcal{T}_{\mathsf{DH}}^{t_0}$) 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. ◀

▶ **Example 17.** The hijacking attack briefly described in Example 12 on the topology $\mathcal{T}_0 \in \mathcal{C}_{\mathsf{DH}}$ can be retrieved on the topology $\mathcal{T}_{\mathsf{DH}}^{t_0}$ starting with the valid initial configuration:

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

We may note that in the reduced topology the role of the prover has to be played by v_0 who is the only agent in the neighbourhood of himself. Such a configuration is indeed a valid initial configuration according to our definition. Actually, our reduction results still apply considering a model in which a same agent is *not* allowed to execute both roles. The resulting reduced topologies are depicted in Figure 4. Basically, we have to duplicate agents to ensure that both kinds of roles (verifier and prover) are available at each location.

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:12–29:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

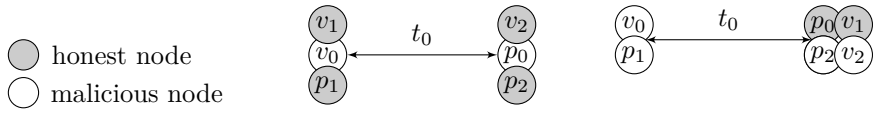


Figure 4 Reduced topologies for mafia fraud and distance hijacking when agents are *not* allowed to execute both roles.

5 Case studies using ProVerif

We have reduced the topology but we still have to take it into account when analysing the protocol preventing us from using automatic verification tools dedicated to traditional security protocols. In this section, we will explain how to get rid of the resulting topology and obtain interesting results on timed protocols relying on the ProVerif verification tool.

5.1 ProVerif in a nutshell

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

$$P := 0 \mid \mathtt{in}(x).P \mid \mathtt{let}\ x = v\ \mathtt{in}\ P \mid \mathtt{new}\ n.P \mid \mathtt{out}(u).P \mid i : P \mid !P$$

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 $i \in \mathbb{N}$.

The semantics is similar to the one introduced earlier, and formally defined through a relation, denoted $\Rightarrow$, over configurations (only partially described below) . A configuration is a tuple $(\mathcal{P}; \phi; i)$ where $\mathcal{P}$ is a multiset of processes (as given by the grammar), ϕ 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 $0 : P$.

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

5.2 Our transformation

Given a topology $\mathcal{T}$ (typically one in Figure 3), 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. In such a configuration, $\mathcal{P}$ is a multiset of (non-extended) processes with phases, and ϕ is a (non extended) frame. From now on, we assume that $V_0(v_0, p_0)$ only contains one block of the form $\mathtt{reset}.\mathtt{out}(n).\mathtt{in}^{<t}(x)$, i.e. it is of the form:

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

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 2.2 (with no `reset` instruction and no guarded input) into a process in the ProVerif calculus.

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:13–29:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- *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 ϕ is such that $img(\phi) = \bigcup_{a \in \mathcal{M}_0} \mathsf{Knows}(\mathcal{I}_0, a, \mathcal{A}_0)$, and $\mathcal{P}$ contains:
- block_1 . $1 : \mathtt{out}(m)$. $\mathtt{in}(x)$. $2 : \mathsf{block}_2$. $\mathsf{end}(v_0, p_0)$;
- $!R(a_0, .., a_n)$ 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 then establish the following result that justifies the transformation presented above.

▶ Proposition 2. 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:

$$\mathsf{block}_1 \text{ . } \mathtt{reset} \text{ . } \mathtt{out}(n) \text{ . } \mathtt{in}^{<t}(x) \text{ . } \mathsf{block}_2 \text{ . } \mathsf{end}(z_0, z_1) \quad \text{with } t \leq 2 \times t_0$$

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 tr during phase 1, the underlying recipe R is either of the form w, or only uses handles ouputted in phase 0.

This result allows us to turn any mafia attack attack or distance hijacking attack into an attack trace regarding the reachability of the event `end`.

5.3 Case studies

Regardless of the type of the considered attack, we only need to consider a single topology (depicted in Figure 3). Once down to this single topology, we can apply Proposition 2, and analyse the configuration $(\mathcal{P}; \phi; 0) = \mathcal{F}(\mathcal{T}^{t_0}_{\mathsf{XX}}, \overline{\mathcal{P}_{\mathsf{prox}}}, V_0, \mathcal{I}_0, t_0)$ with $\mathsf{XX} \in \{\mathsf{MF}, \mathsf{DH}\}$ in ProVerif. If the protocol is proved secure, then $\mathcal{P}_{\mathsf{prox}}$ is resistant to the class of attacks we have considered. Otherwise, we have to check whether the trace returned by ProVerif can be turned into a real attack in our timed model. In principle, it may happen that ProVerif returns an attack trace that is only suitable in the untimed model.

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 2. 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.

On all our case studies, ProVerif answered in less than one second (on a standard laptop) and all the traces returned by ProVerif are actually real attack traces (no false positive).

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

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:14–29:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Protocols	MF	DH	Protocols	MF	DH
Brands and Chaum [11]	✓	×	Munilla *et al.* [30]	✓	✓
Meadows *et al.* $(n_V \oplus n_P, P)$ [28]	✓	✓	CRCS [32]	✓	×
Meadows *et al.* $(n_V, n_P \oplus P)$ [28]	✓	×	Hancke and Kuhn ⋆ [23]	✓	✓
TREAD-Asymmetric [6]	×	×	Eff-pkDB [25]	✓	✓
TREAD-Symmetric [6]	✓	×			
SPADE [12]	×	×	PaySafe [14]	✓	*n.a.*
MAD (One-Way) [13]	✓	×	PaySafe-v2 [14]	×	*n.a.*
Swiss-Knife [26]	✓	✓	PaySafe-v3 [14]	×	*n.a.*

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

Table 1 Results on our case studies (×: attack found, ✓: proved secure, *n.a.*: not applicable)

the prover relying on an uninterpreted function symbol with relevant arguments. Finally, in order to rely on ProVerif, the xor operator has been abstracted as follows (even if our theoretical development is generic enough to deal with such an operator):

$$(x \oplus y) \oplus x \to y \quad (x \oplus y) \oplus y \to x \quad x \oplus (x \oplus y) \to y \quad y \oplus (x \oplus y) \to x.$$

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 [28]. The results are consistent with the ones obtained in [27, 17]. In addition, we discovered a new attack on the SPADE protocol [12] which has been designed to be mafia fraud resistant.

As described in Figure 5, the prover generates a nonce n_P, signs it and encrypts the resulting message with the public key of the verifier. Receiving such a message, the verifier answers by sending fresh nonces. Once these two messages are exchanged, the rapid phase begins: the prover has to answer to the challenge b relying on some hash functions applied on various nonces. Finally the protocol ends when the prover sends the final hash $\mathsf{h}_1(b, n_P, n_V, m, r)$. The attack we discovered (see the description given in Figure 6) is similar to the one obtained on TREAD-Asymmetric by [27]. Roughly, once a dishonest verifier a received a message $\{n_P, \sigma_P\}_{pk_a}$, he can use it to forge $\{n_P, \sigma_P\}_{pk_{p_0}}$ and acts as if the prover p_0 was in the neighboorhood of v_0. The attacker model presented in [12] does not allow them to capture this attack since they not consider dishonest verifers.

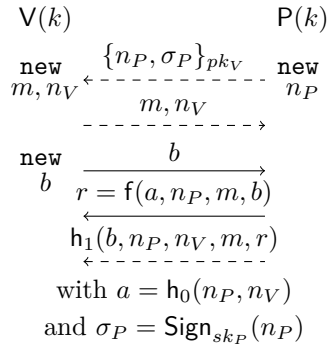


Figure 5 SPADE protocol

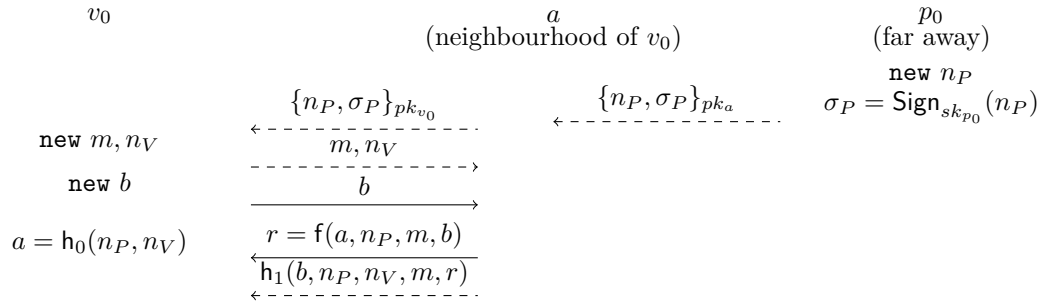


Figure 6 Mafia fraud on the SPADE protocol

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:15–29:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

PaySafe protocol. We studied the PaySafe protocol [14] 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 7 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 [14]: in PaySafe-v2 we just remove the reader nonce n_R and in PaySafe-v3 we remove n_R and n_C.

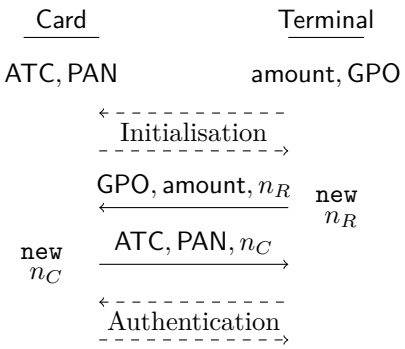


Figure 7 PaySafe (simplified)

Our results confirmed those presented in [14]. 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. Actually our ProVerif models differ from those given in [14]. We typically define richer scenario by giving additional knowledge to the attacker and allowing prover roles and verifier roles to be executed in phase 1. The authors of [27] reported a distance fraud attack which is not relevant in the context of EMV contactless payment protocols. Their result does not allow them to isolate each class of attacks, and therefore, they can not prove whether PaySafe is mafia fraud resistant or not.

6 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. Our methodology is flexible enough to draw meaningful conclusions on each class of attacks: hijacking attack, and mafia fraud. The interested reader may also find some additional results regarding distance fraud in our companion report [19].

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. Our work should also benefit from the recent advances that have been made to integrate the exclusive-or operator in existing verification tool such as Tamarin [22]. Even if our formal development allows us to rely on the Tamarin prover (e.g. we obtain meaningful results on the Hancke and Kuhn protocol with Tamarin), it happens that Tamarin (automatic mode) behaves poorly on some of our case studies (e.g. Brands and Chaum) that uses the xor operator. This deserves further investigations.

References

1 M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:16–29:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 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)*, pages 1–10. ACM, 2008.

3 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)*, volume 7214, pages 267–282. Springer, 2012.

4 G Avoine, MA Bingol, Ioana Boureanu, S Capkun, G Hancke, S Kardas, CH Kim, C Lauradoux, B Martin, J Munilla, et al. Security of distance-bounding: A survey. *ACM Computing Surveys*, 2017.

5 Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardaş, Cédric Lauradoux, and Benjamin Martin. A framework for analyzing RFID distance bounding protocols. *Journal of Computer Security*, 19(2):289–317, 2011.

6 Gildas Avoine, Xavier Bultel, Sébastien Gambs, David Gerault, Pascal Lafourcade, Cristina Onete, and Jean-Marc 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*, pages 800–814. ACM, 2017.

7 David Basin, Srdjan Capkun, Patrick Schaller, and Benedikt Schmidt. Formal reasoning about physical properties of security protocols. *ACM Transactions on Information and System Security (TISSEC)*, 14(2):16, 2011.

8 B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Computer Society Press, 2001.

9 Bruno Blanchet. Modeling and verifying security protocols with the applied pi calculus and proverif. *Foundations and Trends in Privacy and Security*, 1(1-2):1–135, 2016. URL: `https://doi.org/10.1561/3300000004`, `doi:10.1561/3300000004`.

10 Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Practical and provably secure distance-bounding. *Journal of Computer Security*, 23(2):229–257, 2015.

11 Stefan Brands and David Chaum. Distance-bounding protocols. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 344–359. Springer, 1993.

12 Xavier Bultel, Sébastien Gambs, David Gerault, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WISEC 2016, Darmstadt, Germany, July 18-22, 2016*, pages 121–133. ACM, 2016.

13 Srdjan Čapkun, Levente Buttyán, and Jean-Pierre 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*, pages 21–32. ACM, 2003.

14 Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel, and Matthew Thompson. Relay cost bounding for contactless EMV payments. In *Proc. 19th International Conference on Financial Cryptography and Data Security (FC'15)*, volume 8975 of *Lecture Notes in Computer Science*, pages 189–206. Springer, 2015.

15 Hubert Comon-Lundh and Véronique Cortier. Security properties: two agents are sufficient. *Programming Languages and Systems*, pages 99–113, 2003.

16 Véronique Cortier, Jan Degrieck, and Stéphanie Delaune. Analysing routing protocols: four nodes topologies are sufficient. In *International Conference on Principles of Security and Trust*, pages 30–50. Springer, 2012.

17 Cas Cremers, Kasper B Rasmussen, Benedikt Schmidt, and Srdjan Capkun. Distance hijacking attacks on distance bounding protocols. In *Proc. IEEE Symposium on Security and Privacy (S&P'12)*, pages 113–127. IEEE, 2012.

18 Alexandre Debant, Stéphanie Delaune, and Cyrille Wiedling. `http://people.irisa.fr/Alexandre.Debant/proving-physical-proximity-using-symbolic-methods.html`.

19 Alexandre Debant, Stéphanie Delaune, and Cyrille Wiedling. Proving physical proximity using symbolic models. Research report, Univ Rennes, CNRS, IRISA, France, February 2018. URL: `https://hal.archives-ouvertes.fr/hal-01708336`.

20 Yvo Desmedt, Claude Goutier, and Samy Bengio. Special uses and abuses of the fiat-shamir passport protocol. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 21–39. Springer, 1987.

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

22 Jannik Dreier, Lucca Hirschi, Sasa Radomirovic, and Ralf Sasse. Automated Unbounded Verification of Stateful Cryptographic Protocols with Exclusive OR. In *CSF'2018 - 31st IEEE Computer Security Foundations Symposium*, Oxford, United Kingdom, July 2018. URL: `https://hal.archives-ouvertes.fr/hal-01780603`.

23 Gerhard P Hancke and Markus G Kuhn. An RFID distance bounding protocol. In *Proc. 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*, pages 67–73. IEEE, 2005.

24 Max Kanovich, Tajana Ban Kirigin, Vivek Nigam, Andre Scedrov, and Carolyn Talcott. Towards timed models for cyber-physical security protocols. In *Joint Workshop on Foundations of Computer Security and Formal and Computational Cryptography*, 2014.

25 Handan Kilinç and Serge Vaudenay. Efficient public-key distance bounding protocol. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 873–901, 2016.

26 Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. The swiss-knife RFID distance bounding protocol. In *International Conference on Information Security and Cryptology*, pages 98–115. Springer, 2008.

27 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)*, volume 00, pages 152–169. URL: `doi.ieeecomputersociety.org/10.1109/SP.2018.00001`, `doi:10.1109/SP.2018.00001`.

28 Catherine Meadows, Radha Poovendran, Dusko Pavlovic, LiWu Chang, and Paul Syverson. Distance bounding protocols: Authentication logic analysis and collusion attacks. In *Secure localization and time synchronization for wireless sensor and ad hoc networks*, pages 279–298. Springer, 2007.

29 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)*, volume 8044 of *LNCS*, pages 696–701. Springer, 2013.

30 Jorge Munilla and Alberto Peinado. Distance bounding protocols for rfid enhanced by using void-challenges and analysis in noisy channels. *Wireless communications and mobile computing*, 8(9):1227–1232, 2008.

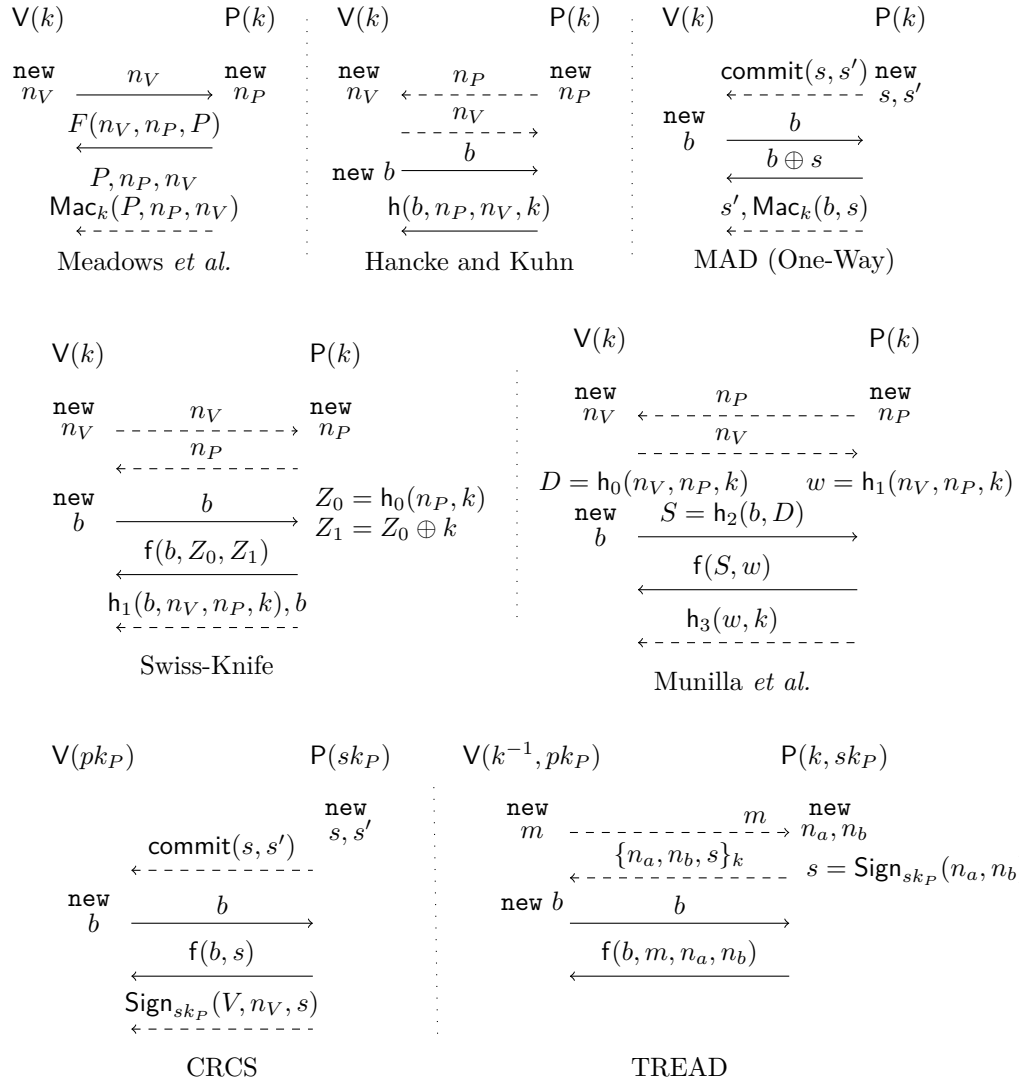
31 Vivek Nigam, Carolyn Talcott, and Abraão Aires Urquiza. Towards the automated verification of cyber-physical security protocols: Bounding the number of timed intruders. In *Proc. 21st European Symposium on Research in Computer Security (ESORICS'16)*, pages 450–470. Springer, 2016.

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:18–29:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

32 Kasper Bonne Rasmussen and Srdjan Capkun. Realization of RF distance bounding. In *USENIX Security Symposium*, pages 389–402, 2010.

A Brief description of our case studies



38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 29; pp. 29:19–29:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany