So near and yet so far - Symbolic verification of distance-bounding protocols

Alexandre Debant, Stéphanie Delaune, Cyrille Wiedling

So near and yet so far – Symbolic verification of distance-bounding protocols

ALEXANDRE DEBANT, Univ Rennes, CNRS, IRISA, Rennes, France
STÉPHANIE DELAUNE, Univ Rennes, CNRS, IRISA, Rennes, France
CYRILLE WIEDLING, DGA MI, Bruz, France

The continuous adoption of Near Field Communication (NFC) tags offers many new applications whose security is essential (e.g. contactless payments). In order to prevent flaws and attacks, we develop in this paper a framework allowing us to analyse the underlying security protocols, taking into account the location of the agents and the transmission delay when exchanging messages. We propose two reduction results to render automatic verification possible relying on the existing verification tool ProVerif. Our first result allows one to consider a unique topology to catch all the possible attacks, and the second one simplifies the security analysis when considering Terrorist fraud. Then, based on these results, we perform a comprehensive case studies analysis (27 protocols), where we obtain new proofs of security for some protocols and detect attacks on some others (cf. Table 1).

1 INTRODUCTION

Companies continuously look for new features that may improve the user experience for their customers. The development of contactless technologies like Wi-Fi, Radio Frequency IDentification (RFID) or Near Field Communication (NFC) has led to the birth of many innovations, e.g. keyless entry systems, in-store self-checkout, transport-ticketing, tap-to-pay transactions, etc. The quick adoption of these technologies by sellers and customers is explained by the convenience and efficiency they bring. For instance, regarding the tap-to-pay feature, transactions become quicker and, this allows one to replace the use of credit/debit cards and wallets by smart watches or smartphones (which include an NFC interface today) for everyday purchases.

However, all these advantages for contactless technologies should not conceal new threats that are coming up. An important security concern behind all these applications is to ensure the physical proximity of the two devices that are involved. Indeed, a malicious person should not be able to impersonate a remote and honest person to pay a bill on his behalf. While contact-based devices prevent such a scenario "by design", this scenario becomes possible with contactless devices. For instance, in [32, 36], relay attacks have been shown practical even considering the short communication range of the underlying technologies: the attacker uses a much faster communication channel, like WiFi, to relay messages between distant devices. All of these attacks are man-in-the-middle scenarios where the attacker may simply relay messages, or be more active, e.g. making some computations to forge new messages. Ideally, contactless applications should resist to these attacks.

In order to mitigate such attacks (and others), some security protocols, called *distance-bounding* protocols, have been designed. As usual, they rely on cryptographic primitives (e.g. encryption, signature, hash function) to ensure various security properties. For instance, they aim at enforcing physical proximity between two participants called respectively the *prover* and the *verifier*. Due to their critical applications, e.g. payment protocols, proving the security of distance-bounding protocols is of paramount importance. For a long time, these security analyses consisted in analysing the protocols against well-known attacks only, and prove their inapplicability. However, all these analyses provide a limited confidence since they focus on rather specific attacks only [3].

Authors' addresses: Alexandre Debant, Alexandre.Debant@irisa.fr, Univ Rennes, CNRS, IRISA, Rennes, France; Stéphanie Delaune, Stephanie.Delaune@irisa.fr, Univ Rennes, CNRS, IRISA, Rennes, France; Cyrille Wiedling, cyrille.wiedling@irisa.fr, DGA MI, Bruz, France.

In the early 2010s, computational models have been designed to formally verify the security of distance-bounding protocols in a more general setting [3, 33]. In these models, messages are represented by bitstrings, an attacker is any probabilistic polynomial time algorithm, and an attack applies if there exists an attacker with a non-negligible probability of success. These features are accurate enough to precisely analyse distance-bounding protocols and obtain strong security guarantees. Different frameworks have been proposed, and many protocols have been analysed within these frameworks [5]. Unfortunately, this required tedious hand-written proofs for each protocol. Automation appears to be a very difficult task in such a framework.

Symbolic verification of cryptographic protocols is a well-known approach suitable for automation. A common abstraction of such models is to abstract cryptographic primitives using function symbols, and to assume that these primitives work perfectly well. For instance, an encryption primitive will not leak any information regarding the plaintext to anyone who knows the ciphertext as soon as the decryption key remains unknown. Messages exchanged during a protocol are then represented by first-order terms built using these function symbols, as well as atomic data modelling nonces, keys, etc. Another common abstraction applies on the attacker model: symbolic models usually assume an omniscient and omnipresent attacker, the so-called Dolev-Yao attacker [30], which is able to intercept, forge and send messages at any time. Even considering these two abstractions, the automatic verification of cryptographic protocols remains a difficult problem. In most cases, proving security properties for an expressive enough class of protocols appears as an undecidable problem [49]. Nevertheless, procedures and tools, e.g. ProVerif [10, 11] or Tamarin [48], have been designed to tackle this problem, and have already proved their efficiency and usefulness to analyse real-world protocols. For instance, ProVerif has been used to analyse TLS 1.3 [9], the ARINC823 avionic protocol [12], as well as the Neuchâtel voting protocol [22]. Tamarin has been used to conduct a security analysis of complex protocols, e.g. 5G AKA with exclusive-or [8], group key agreement protocols [54], or the Noise framework [38] with Diffie-Hellman keys, to cite only a few. Unfortunately, these tools must be adapted in order to analyse protocols whose security relies on physical constraints. Indeed, by default, they model an omniscient attacker who controls the entire communication network and can therefore relay messages without introducing any delay. Applying these tools naïvely to analyse distance-bounding protocols will necessarily lead to false attacks. To overcome this limitation, some novel symbolic models have been proposed, e.g. [7, 44, 47]. This paper is part of this line of work and aims at allowing the formal symbolic verification of distance-bounding protocols leveraging some existing automatic verification tools.

Our Contributions

As already explained, we aim at proposing techniques and tools to allow formal symbolic verification of distance-bounding protocols. Based on a symbolic model we develop for that purpose, we propose some reduction results to allow automatic verification of these protocols in the existing tool ProVerif, and we conduct the security analysis of 27 protocols. Our main contributions are as follows:

(1) First, we establish that, for each class of attacks, considering a rather simple topology with at most four agents is enough to catch all the possible attacks (see Section 5). This reduces the number of topologies that need to be considered from infinitely many to only one (per class of attacks).
(2) Then, considering Terrorist fraud, a well-known type of attacks in the context of distance-bounding protocol, we prove that the dishonest prover has a best strategy for collusion on which the security analysis can focus on (see Section 6).

(3) Lastly, getting some inspiration from [20], we show how to encode the reduced topologies in the existing verification tool ProVerif, and we establish the correctness of this encoding (see Section 7). This allows us to propose a comprehensive case studies analysis (27 protocols) relying on our new framework. As detailed in Table 1, we obtain new proofs of security for some protocols and detect attacks on some others. The results on the case studies are presented in Section 8 .

These results have been obtained within a symbolic model that we developed to fit our specific needs. Based on the applied pi-calculus [2], this models allows to faithfully take into account the physical constraints by modelling agents' locations and time. About the physical constraints, we model that a message takes time to travel from one location to another. We also propose formal definitions for each class of attacks (Distance fraud, Mafia fraud, Distance Hijacking attack, and Terrorist fraud) considering an arbitrary number of agents, each located at arbitrary locations. The model is detailed in Section 3 regarding the protocols, and in Section 4 regarding the security properties.

We extend here results that have been published at FSTTCS'18 [26] and ESORICS'19 [27] by providing detailed proofs of our reduction results, more examples to illustrate the different concepts, a detailed comparison with existing works, and some additional case studies. The detailed proofs as well as the full material used to conduct the case studies are available in the supplementary material provided with this document and in our GitLab repository: https://gitlab.inria.fr/adebant/db-verif.

2 BACKGROUND

In this section, we recall relevant background related to distance-bounding protocols, and their formal verification in the symbolic setting. The SPADE protocol described in Figure 1 will be used as a running example throughout this paper.

2.1 Distance bounding protocols

In 1993, Brands and Chaum proposed the first distance-bounding protocol [15]. Since then, more than 40 distance-bounding protocols have been designed (see e.g. [5]), and they all mostly follow the same structure. Typically, a distance-bounding protocol starts by an initialisation phase in which the prover and the verifier exchange data to prepare the fast phase that follows. This fast phase allows the verifier to estimate his distance with the prover. For this, the verifier measures the time that elapses during multiple challenge/response exchanges. Finally, distance-bounding protocols may require a final phase during which the two parties exchange some extra information.

An example of such a protocol is the SPADE protocol introduced in [16], and presented in Figure 1. It relies on the use of public-key encryption, signature, pseudo-random functions, and the exclusive-or operator. The protocol starts with the prover sending his nonce n_P and a signature σ of this nonce using his signing secret key $\mathsf{ssk}(P)$. This pair of information is encrypted using the public-key of the verifier. Once, the verifier has decrypted and verified the signature, he sends back two nonces, m_V and n_V, which will be used by the prover during the fast phase. To make this phase as fast as possible, the prover pre-computes two session values H^0 and H^1. After a moment, the verifier initiates the fast phase during which the time measurement is performed. The verifier generates and sends a random bit c_i, and the prover has to reply immediately with the i-th bit of H^0, denoted H^0_i, if $c_i = 0$, or the i-th bit of H^1, denoted H^1_i, otherwise. This fast exchange is repeated a fixed number of times. After the fast phase, the prover is expected to provide $\mathsf{prf}(\langle n_P, n_V, m_V, c_1 \dots c_n, r_1 \dots r_n \rangle)$ to the verifier as a transcript of the session for an additional check. If all the checks pass and if enough correct answers are received within a sufficiently short

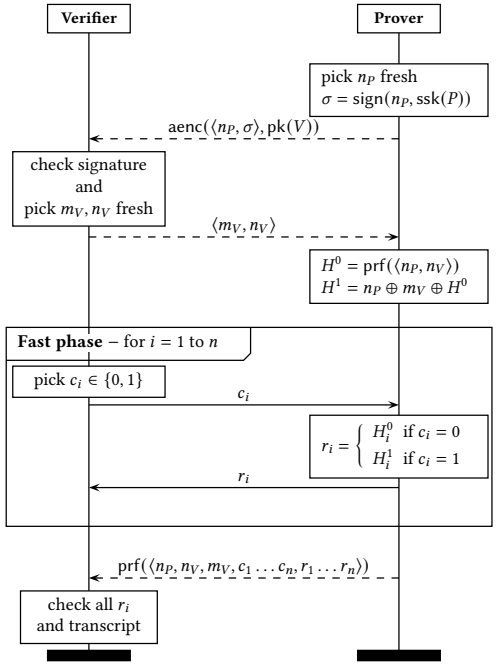


Fig. 1. SPADE protocol

time after the corresponding challenge c_i has been sent out, then the verifier is convinced that the prover is located in his vicinity and accepts the session.

2.2 Different types of fraud

Regarding the literature, the different attack scenarios that may apply on distance-bounding protocols have been gathered into three main classes: *Mafia fraud* [29], *Terrorist fraud* [28], and *Distance fraud* [28]. Recently, Cremers *et al.* have discovered a new class of attacks called *Distance Hijacking attack* [23]. Nowadays, the security analyses are thus performed w.r.t. to these four classes of attacks which differ mainly by the honesty and the locations of the main protagonists:

- *Mafia fraud:* an attacker located in-between an honest verifier and an honest distant prover tries to make the verifier authenticate the prover.
- *Distance Hijacking attack:* a distant dishonest prover abuses honest parties to be authenticated by an honest verifier.
- *Distance fraud:* a distant dishonest prover tries to be authenticated by an honest verifier.
- *Terrorist fraud:* a distant dishonest prover accepts to collude with an attacker to be authenticated once by an honest verifier. However the collusion should not give any advantage to the attacker for future attacks, meaning that the attacker should not be able to authenticate again on behalf of the dishonest prover who colluded with him.

The Mafia fraud is the class that encompasses the aforementioned relay attacks that applies to tap-to-pay transactions or keyless entry systems embedded in cars. The Terrorist fraud is an advanced scenario in which a prover accepts to collude with the attacker once. However, this collusion should not enable the attacker to mount future attacks. By consequence, in such a scenario,

the dishonest prover will not accept, for instance, to reveal his long-term keys since this will render future attacks possible. The two remaining classes of attacks, Distance fraud and Distance Hijacking attack, are very similar. In case of Distance fraud, the attacker is alone, whereas he can abuse honest parties in a Distance Hijacking attack. Hence, Distance Hijacking attacks encompass Distance frauds.

2.3 Related work

In 2007, Meadows *et al.* [47] proposed the first symbolic model to analyse distance-bounding protocols. This framework relies on an authentication logic that faithfully features locations and time, and all the proofs are carried out manually. Since then, some progress has been done towards automation.

In 2011, Basin *et al.* [7] proposed another symbolic framework to analyse distance-bounding protocols. It extends an existing model based on multiset rewriting rules with time and locations in a general setting. The protocol descriptions and the deduction capabilities of the attackers remain unchanged; only the network rule has been adapted: a message can be received at time t_R by an agent B if, and only if, it has been sent by an agent A soon enough to let the message travel from A's location to B's, i.e., at time t_S such that $t_R \geq t_S + Dist(A, B)$. This property faithfully models physical restrictions and has been used to analyse distance-bounding protocols. Instead of relying on hand-written proofs, the authors encoded their model into the theorem-proving assistant Isabelle/HOL [52] in order to provide a partially automated framework. Indeed, many lemmas are generic enough to be protocol-independent and thus re-usable across different analyses. The approach followed by the authors demonstrates a noteworthy effort to provide a rigorous framework to analyse distance-bounding protocols. Unfortunately, conducting security proofs within their framework requires a significant effort as protocol-dependent lemmas must be manually defined by the end-user (typically between 8 and 20 for each case study).

In 2015, Chothia *et al.* [20] aimed at verifying a new payment protocol, named PaySafe, designed to ensure physical proximity in contactless transactions. To this aim, they developed an approach based on the ProVerif tool. They managed to encode rather simple topologies that involve only two locations: the verifier's location and a remote location (typically for the honest prover when considering relay attacks). This encoding relies on the notion of phases provided by ProVerif: three phases are defined, one for each phase of the protocol, following the general structure of distance-bounding protocols previously presented. Agents at the verifier's location are allowed to act during any phase, but remote agents cannot act during the fast phase (intuitively they are too far to respond to the challenge). The main limitations of this approach are two-fold: they focus on a unique topology without providing any justification; and they do not formally justify its encoding in ProVerif. Despite that, this is an appealing approach and we actually get inspiration from this work to set up our framework.

Meanwhile we established the results presented here, Mauw *et al.* [44, 45] extended Basin *et al.*'s framework [7] to cope with the automation issue, and Chothia *et al.* [19] presented an extension of the applied pi-calculus to model distance-bounding protocols. These works deserve detailed attention, and therefore a precise comparison with our framework will be done later on (see Section 4.4 regarding modelling, and Section 8.5 regarding automation).

3 MODEL

We model security protocols using a process algebra inspired from the applied pi-calculus [2] used by, e.g., the ProVerif verification tool [10, 11]. In this setting, participants are modelled as processes, and the communication between them is modelled by means of the exchange of messages that are represented by a term algebra. However, modelling distance-bounding protocols requires several

features. For instance, the location of a participant is a relevant piece of information to know the time needed by a message to travel from one point to another. Timing information is also of paramount importance since a participant may simply decide to discard a message which arrives too late. Therefore, in this section, we extend the applied pi-calculus to model these features.

3.1 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. This provides a lot of flexibility and allows one to model various cryptographic primitives.

3.1.1 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 and nonces, whereas $\mathcal{A}$ is the set of *agent names*, *i.e.* names which represent the agent identities. We consider an infinite set Σ_0 of constant symbols that are used to represent values known by the attacker, as well as 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* 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 $\Sigma^+ = \Sigma \cup \Sigma_0$, and $\Sigma_c^+ = \Sigma_c \cup \Sigma_0$.

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 $\mathcal{T}(\Sigma_c^+, \mathcal{N} \cup \mathcal{A} \cup \mathcal{X})$. 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 σ to a term u is written $u\sigma$. We denote $dom(\sigma)$ its *domain*, and $img(\sigma)$ its *image*. We have that $dom(\sigma) = \{x \mid x\sigma \neq x\}$ and $img(\sigma) = \{x\sigma \mid x \in dom(\sigma)\}$. The *positions* of a term are sequences of integers, and are defined as usual. We use ϵ to denote the root position of a term. We use $t|_p$ to represent the term at position p in t. More formally, we have that $t|_\epsilon = t$, and $t|_{i.p} = t_i|_p$ when $t = \mathsf{f}(t_1, \ldots, t_n)$ and $1 \leq i \leq n$.

Example 1. *We consider the signature* $\Sigma^{\mathsf{ex}} = \Sigma_c^{\mathsf{ex}} \cup \Sigma_d^{\mathsf{ex}}$ *with:*

- $\Sigma_c^{\mathsf{ex}} = \{\mathsf{prf}/1, \mathsf{pk}/1, \mathsf{sk}/1, \mathsf{spk}/1, \mathsf{ssk}/1, \mathsf{aenc}/2, \mathsf{sign}/2, \langle\ ,\ \rangle/2, \mathsf{xor}/2, \mathbb{0}/0, \mathsf{ok}/0, \mathsf{answer}/3\}$;
- $\Sigma_d^{\mathsf{ex}} = \{\mathsf{adec}/2, \mathsf{check}/2, \mathsf{proj}_1/1, \mathsf{proj}_2/1, \mathsf{eq}/2\}$.

The symbol prf *models a hash function, and* pk, sk, spk, ssk *are used to model the public/private keys, respectively for encryption and signature, of an agent. We model asymmetric encryption and decryption using* aenc *and* adec*, and a signature mechanism through the* sign *and* check *symbols. The* $\langle\ ,\ \rangle$, proj_1 *and* proj_2 *symbols are for concatenation and, respectively, left and right projections, while* xor *models the exclusive-or operator together with the symbol* $\mathbb{0}$ *as the identity element. We provide an equality test using* eq *and the constant* ok*. Finally,* answer *is a function symbol, a cryptographic hash, used to model the answer provided by the prover.*

Remark 1. *For readability purposes, we may use, in the rest of this document the following notations:* $\langle t_1, \ldots, t_n \rangle = \langle t_1, \langle t_2, \ldots, \langle t_{n-1}, t_n \rangle \rangle \rangle$ *for the concatenation of n elements, and* $\pi_j(\langle t_1, \ldots, t_n \rangle) = \mathsf{proj}_1 \circ \mathsf{proj}_2^{j-1}(\langle t_1, \ldots, t_n \rangle)$ *the j-th projection on a n-tuple, with* $j \in \{1, \ldots, n-1\}$*, while, for* $j = n$*, we define* $\pi_j = \mathsf{proj}_2^{n-1}(\langle t_1, \ldots, t_n \rangle)$.

3.1.2 Equational theory. Following the approach developed in [11], constructor terms are subject to an *equational theory* allowing us to model the algebraic properties of the primitives. An equational theory 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. We assume that it is not degenerate, i.e. there exist u, v such that $u \neq_{\mathsf{E}} v$.

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

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

3.1.3 Rewriting rules. As in [11], we give a meaning to destructor symbols. This is done through a set of rewrite rules of the form $\mathrm{g}(t_1, \ldots, t_n) \to t$ where $\mathrm{g} \in \Sigma_d$, 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 rewrite rule $\mathrm{g}(t_1, \ldots, t_n) \to t$ such that $u|_p =_{\mathsf{E}} \mathrm{g}(t_1, \ldots, t_n)\theta$ for some substitution θ such that $t_1\theta, \ldots, t_n\theta$ are constructor terms. In such a case, we have that $v = u[t\theta]_p$ i.e. u in which the term at position p has been replaced by $t\theta$. In the following, we only consider sets of rewrite rules that yield a *convergent* rewriting system (modulo E), 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 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* Σ^{ex}, *only* sk *and* ssk *are in* Σ_{priv}. *The property of the destructor symbols are reflected by the following rewriting rules:*

$$\begin{array}{rcl@{\qquad}rcl@{\qquad\qquad}rcl} \mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), \mathsf{sk}(y)) & \to & x & \mathsf{proj}_1(\langle x, y\rangle) & \to & x & \mathsf{eq}(x, x) & \to & \mathsf{ok}. \\ \mathsf{check}(\mathsf{sign}(x, \mathsf{ssk}(y)), \mathsf{spk}(y)) & \to & x & \mathsf{proj}_2(\langle x, y\rangle) & \to & y & & & \end{array}$$

Note that $\mathsf{eq}(u, v)$ *reduces to a message if, and only if,* $u =_{\mathsf{E}} v$.

3.2 Protocols

As usual, protocols are modelled by a set of roles, i.e. processes describing the behaviours of each participant.

3.2.1 Process algebra. Our process algebra is inspired from the applied pi-calculus [2]. We do not consider else branches. Actually, we do not have conditional. Instead, equality tests are performed through our `let` construction relying on the destructor symbol eq. Our grammar below does not feature parallel composition and replication. These operators are directly taken into account in our notion of configuration given in Section 3.3.

$$\begin{array}{rcl@{\quad}l@{\quad}l@{\quad}l@{\quad}l} P, Q := & 0 & | & \mathtt{in}(x).P & | & \mathtt{in}^{<t}(x).P & | & \mathtt{let}\ x = v\ \mathtt{in}\ P \\ & | & \mathtt{new}\ n.P & | & \mathtt{out}(u).P & | & \mathtt{reset}.P & | & \mathtt{end}(u_0, u_1) \end{array}$$

where $x \in \mathcal{X}$, $n \in \mathcal{N}$, $u, u_0, u_1 \in \mathcal{T}(\Sigma^+_c, \mathcal{X} \uplus \mathcal{N} \uplus \mathcal{A})$, $v \in \mathcal{T}(\Sigma^+, \mathcal{X} \uplus \mathcal{N} \uplus \mathcal{A})$, and $t \in \mathbb{R}_+$.

The four instructions on the left are all standard to the applied pi-calculus. As usual, the null process 0 does nothing. The restriction `new` $n.P$ generates a fresh name and then executes P. We have constructions to model input and output actions. The `let` $x = v$ `in` P construction tries to evaluate v to get a constructor term u, then x is bound to u and P is executed; if the evaluation of v fails, then the process is blocked. The last three instructions are more specific to distance-bounding protocols. Agents can perform time measurements using their local clock. The `reset` instruction allows them to reset their local clock, whereas a guarded input $\mathtt{in}^{<t}(x).P$ will be used to model an agent waiting for a message which is supposed to arrive soon enough, i.e. before t units of time since his last reset instruction. Finally, we consider a specific command $\mathtt{end}(u_0, u_1)$ that will be used to express security properties.

As usual, we denote by $fv(P)$ (resp. $fn(P)$) the set of free variables (resp. names) occurring in the process P, i.e. the set of variables (resp. names) which are not bound by an input or a let construction (resp. a new). Conversely, we denote $bv(P)$ (resp. $bn(P)$) the set of bound variables

(resp. bound names) in the process. Throughout the paper, we consider 2-party protocols only. A *2-party protocol* is a pair (V, P) of processes, called respectively the *verifier role* and the *prover role*, such that $fv(\mathsf{V}) = \{z_V^0, z_V^1\}$, $fv(\mathsf{P}) = \{z_P^0, z_P^1\}$, and $fn(\mathsf{V}) = fn(\mathsf{P}) = \emptyset$. Moreover, we assume that any guarded input is preceded by a reset instruction, and we also assume that the command $\mathsf{end}(u_0, u_1)$ occurs neither in V nor in P. When needed, we write $\mathsf{V}(z_V^0, z_V^1)$ (resp, $\mathsf{P}(z_P^0, z_P^1)$) to make these variables explicit. Intuitively, these variables will be instantiated by agent names: z_X^0 corresponds to the name of the agent executing the process, whereas z_X^1 is the name of its interlocutor.

Example 4. *As a running example, we consider the SPADE distance-bounding protocol [16] presented in Section 2 (see Figure 1). Symbolic analysis does not allow one to reason at the bit level, and thus, as done in e.g. [19, 24, 26, 44], the fast phase is abstracted by a single challenge/response exchange, and operations performed at the bit level are abstracted too. We thus abstract the answer by the uninterpreted symbol of function* answer *taking the challenge and the two pre-computed values stored in x_3 and x_4 as argument. We give below the verifier role written in our formalism.*

$$\begin{aligned}\mathsf{V}(z_V^0, z_V^1) := \quad & \mathsf{in}(x^1_{\mathsf{rep}}).\ \mathsf{let}\ \langle x_1, x_2\rangle = \mathsf{adec}(x^1_{\mathsf{rep}}, \mathsf{sk}(z_V^0))\ \mathsf{in}\\ & \mathsf{let}\ x^1_{ok} = \mathsf{eq}(x_1, \mathsf{check}(x_2, \mathsf{spk}(z_V^1)))\ \mathsf{in}\\ & \mathsf{new}\ n_V.\ \mathsf{new}\ m_V.\ \mathsf{out}(\langle m_V, n_V\rangle).\\ & \mathsf{new}\ c_V.\ \mathsf{reset}.\ \mathsf{out}(c_V).\ \mathsf{in}^{<2\cdot t_0}(x^2_{\mathsf{rep}}).\\ & \mathsf{let}\ x_3 = \mathsf{prf}(\langle x_1, n_V\rangle)\ \mathsf{in}\ \ \mathsf{let}\ x_4 = x_1 \oplus m_V \oplus x_3\ \mathsf{in}\\ & \mathsf{let}\ x^2_{ok} = \mathsf{eq}(x^2_{\mathsf{rep}}, \mathsf{answer}(c_V, x_3, x_4))\ \mathsf{in}\\ & \mathsf{in}(x^3_{\mathsf{rep}}).\ \mathsf{let}\ x^3_{ok} = \mathsf{eq}(x^3_{\mathsf{rep}}, \mathsf{prf}(\langle x_1, n_V, m_V, c_V, \mathsf{answer}(c_V, x_3, x_4)\rangle))\ \mathsf{in}\ 0\end{aligned}$$

3.2.2 Executability. We restrict ourselves to consider protocols modelling realistic behaviours. We assume that all the messages occurring in $\mathsf{V}(z_V^0, z_V^1)$ (resp. $\mathsf{P}(z_P^0, z_P^1)$) are deducible by the agent z_V^0 (resp. z_P^0) executing the role using the previous inputs he already received, and his initial knowledge, formalised by a finite set of terms $\mathcal{I}_0$.

Definition 1. *Given a set $\mathcal{I}_0 = \{u_1, \ldots, u_k\}$ of terms, a role $\mathsf{R}(z^0, z^1)$ is $\mathcal{I}_0$-executable if, for any term u (resp. v) occurring in an* out *(resp. a* let*) construction, there exists a term $C \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, \{w_1, \ldots, w_k\} \cup bn(\mathsf{R}) \cup bv(\mathsf{R}))$ such that $u =_{\mathsf{E}} C\sigma\downarrow$ (resp. $v\downarrow =_{\mathsf{E}} C\sigma\downarrow$) where $\sigma = \{w_1 \mapsto u_1, \ldots, w_k \mapsto u_k\}$.*

A 2-party protocol (V, P) is $(\mathcal{I}_0^V, \mathcal{I}_0^P)$-executable if V is $\mathcal{I}_0^V$-executable, and P is $\mathcal{I}_0^P$-executable.

In the following, we denote by $\mathsf{V}_{\mathsf{end}}$ the role corresponding to V (with variables z_V^0, z_V^1) in which the null process has been replaced by the command $\mathsf{end}(z_V^0, z_V^1)$.

Example 5. *Going back to Example 4, each agent is provided with an identity, two secret keys, and the corresponding public keys. In our model, public keys are derived from the identities (using a public function symbol), it is therefore not mandatory to put them into the initial knowledge of an agent. Thus, the sets $\mathcal{I}_0^X = \{z_X^1, \mathsf{sk}(z_X^0), \mathsf{ssk}(z_X^0)\}$ with $X \in \{V, P\}$ can be used to model the initial knowledge of an agent playing the role X.*

The process $\mathsf{V}(z_V^0, z_V^1)$ (resp. $\mathsf{P}(z_P^0, z_P^1)$) as given in Example 4 is $\mathcal{I}_0^V$-executable (resp. $\mathcal{I}_0^P$-executable). For instance, we have that the first output of $\mathsf{P}(z_P^0, z_P^1)$, i.e. $u = \mathsf{aenc}(\langle n_P, \mathsf{sign}(n_P, \mathsf{ssk}(z_0))\rangle, \mathsf{pk}(z_P^1))$, is such that $u = C\sigma$ with $C = \mathsf{aenc}(\langle n_P, \mathsf{sign}(n_P, w_3)\rangle, \mathsf{pk}(w_1))$.

3.3 Semantics

The semantics of our calculus 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 2. *A* topology *is a tuple $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0)$ 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 malicious;*
- $Loc_0 : \mathcal{A}_0 \to \mathbb{R}^3$ *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 w.r.t. whom the analysis is performed.*

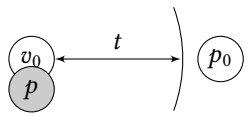
In our model, the distance between two agents is expressed by the time it takes for a message to travel from one to another. Therefore, we consider $\mathsf{Dist}_{\mathcal{T}_0} : \mathcal{A}_0 \times \mathcal{A}_0 \to \mathbb{R}$, based on Loc_0 that will provide the time a message takes to travel 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 Euclidean norm and c_0 the transmission speed. We suppose, from now on, that c_0 is a constant for all agents. Using this equation, 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 unlike in the classical Dolev-Yao model [30], our model does not consider a unique attacker. We may consider several compromised nodes, which communicate and share their knowledge but these communications are also subject to physical constraints: no message can travel faster than the transmission speed.

Example 6. *Let us consider the topology* $\mathcal{T}$ *(depicted below) composed with three agents,* $\mathcal{A}_0 = \{v_0, p_0, p\}$*, with one of them dishonest,* $\mathcal{M}_0 = \{p\}$*.*



Since it is not really the physical locations of agents that matter but the distance, in time, between them, we just consider, in this topology, that the location of agents satisfy the following properties:

$$Dist_{\mathcal{T}}(v_0, p) = 0 \text{ and } Dist_{\mathcal{T}}(v_0, p_0) \geq t,$$

where t *is an arbitrary positive value, used to define* $\mathcal{T}$*. It means that* v_0 *and* p *share the same location and the transmission of messages takes no time at all, while it takes more than* t *to reach* p_0*.*

Our notion of configuration manipulates *extended processes*, i.e. expressions of the form $\lfloor P \rfloor_a^{t_a}$. Intuitively, P describes the actions of agent a, and t_a is the value of the local clock of this process. We also have to store messages that have been output so far. They are stored into a frame (as done e.g. in [2]), extended to keep track of the time at which the message has been output and by whom. More formally, given a topology $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$, a *configuration* $\mathcal{K}$ (built on top of $\mathcal{T}_0$) is a triplet $(\mathcal{P}; \Phi; t)$ where:

- $\mathcal{P}$ is a finite multiset of *extended process* $\lfloor P \rfloor_a^{t_a}$ with $fv(P) = \emptyset$, $a \in \mathcal{A}_0$ and $t_a \in \mathbb{R}_+$;
- $\Phi = \{w_1 \xrightarrow{a_1,t_1} u_1, \ldots, w_n \xrightarrow{a_n,t_n} u_n\}$ is an extended frame, i.e. a substitution such that $w_i \in \mathcal{W}$, $u_i \in \mathcal{T}(\Sigma_c^+, \mathcal{N} \uplus \mathcal{A})$, $a_i \in \mathcal{A}_0$ and $t_i \in \mathbb{R}_+$ for $1 \leq i \leq n$;
- $t \in \mathbb{R}_+$ is the global time of the system.

An *initial frame* is a frame such that $t_i = 0$ $(1 \leq i \leq n)$. We write $\lfloor \Phi \rfloor_a^t$ for the restriction of Φ to the agent a at time t, i.e.:

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

Example 7. *Continuing our running example, we consider the topology* $\mathcal{T}$ *as described in Example 6 with* $t = t_0$*. This value corresponds to half the time of the guarded input in the role of the verifier in SPADE as described in Example 4. A configuration could be* $\mathcal{K}_0 = (\mathcal{P}_0; \Phi_0; 0)$ *with:*

- $\mathcal{P}_0 = \lfloor \mathsf{V}_{\mathsf{end}}(v_0, p_0) \rfloor^0_{v_0} \uplus \lfloor \mathsf{P}(p_0, p) \rfloor^0_{p_0}$;
- $\Phi_0 = \left\{ w_1 \xrightarrow{p,0} v_0, w_2 \xrightarrow{p,0} p_0, w_3 \xrightarrow{p,0} p, w_4 \xrightarrow{p,0} \mathsf{sk}(p), w_5 \xrightarrow{p,0} \mathsf{ssk}(p) \right\}$.

In this configuration, v_0 plays the verifier's role with p_0 and p_0 plays the prover's role with the dishonest agent p. We provide the identities of the agents involved in the topology and the knowledge of the secret keys of the dishonest agent p to the attacker through the frame. Intuitively, this initial frame contains the initial knowledge of agent p as indicated in the sets $(\mathcal{I}_0^V, \mathcal{I}_0^P)$ given in Example 5. Of course, this is just an example, and in general security analyses are conducted considering configurations that include more agents, other instances of the two roles, and eventually additional terms in the knowledge of the attacker.

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

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

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

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

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

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

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

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

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 presented in Figure 2. The first rule (TIM) allows time to elapse, meaning that the global clock as well as the local clocks will be shifted by δ:

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

Note that this rule assumes that all the clocks synchronously evolve. Our model abstracts away clock drift. The second rule (OUT) corresponds to the output of a term by some process: the corresponding term is added to the frame of the current configuration, which means that the attacker can now have access to the sent term. The third rule (LET) corresponds to the evaluation of the term u, and in case $u\downarrow$ is a message, then P in which x is replaced by $u\downarrow$ is executed. The rule (NEW) is used to generate fresh random values. The (RST) rule allows an agent to reset the local clock of the process. 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. Moreover, in case of a guarded input with guard t_g, the local clock of the process has to satisfy the guard, i.e. $t < t_g$. Note that the specific command $\mathsf{end}(a_1, a_2)$ which can only occurs at the end of a process cannot be executed.

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 write $\xrightarrow{\mathsf{tr}}_{\mathcal{T}_0}$ to emphasise the sequence of labels tr that has been used during an execution.

Example 8. *In order to illustrate the semantics presented in Figure 2, the configuration of Example 7 can be executed as follows:*

$$\mathcal{K}_0 \xrightarrow{p_0,\tau}_{\mathcal{T}} \xrightarrow{p_0,\mathsf{out}(\mathsf{aenc}(\langle n'_P,\mathsf{sign}(n'_P,\mathsf{ssk}(p_0))\rangle,\mathsf{pk}(p)))}_{\mathcal{T}} \rightarrow_{\mathcal{T}} \xrightarrow{v_0,\mathsf{in}(\mathsf{aenc}(\langle n'_P,\mathsf{sign}(n'_P,\mathsf{ssk}(p_0))\rangle,\mathsf{pk}(v_0)))}_{\mathcal{T}} \mathcal{K}_1$$

where the configuration $\mathcal{K}_1$ is described as follows:

$$\mathcal{K}_1 = (\lfloor V_1(v_0,p_0)\rfloor^{t_0}_{v_0} \uplus \lfloor P_1(p_0,p)\rfloor^{t_0}_{p_0}; \Phi_0 \cup \left\{ w_6 \xrightarrow{p_0,0} \mathsf{aenc}(\langle n'_P,\mathsf{sign}(n'_P,\mathsf{ssk}(p_0))\rangle,\mathsf{pk}(p)) \right\}; t_0)$$

where $V_1(z_0,z_1)$ is the process $\mathsf{V}_{\mathsf{end}}(z_0,z_1)$ without the first input, and $P_1(z_0,z_1)$ is the process $\mathsf{P}(z_0,z_1)$ without the first nonce generation and the first output. More precisely, in the execution above, the first transition is the nonce n'_P generation executed by p_0. After p_0's output, the next transition is a TIM rule in order to let enough time for the message to reach the attacker p. Then, the final transition models the input, by v_0, of a message constructed by p based on the previous output sent by p_0. The recipe used to build the message is $R_1 = \mathsf{aenc}(\mathsf{adec}(w_6,w_4),\mathsf{pk}(w_1))$.

4 SECURITY PROPERTIES

As stated in the introduction, three different classes of attacks are traditionally considered in the analysis of distance-bounding protocols: Distance fraud, Mafia fraud and Terrorist fraud. In [23], a fourth type of attacks has been identified: the so-called Distance Hijacking attack. All these attacks have a similar goal. They aim to make a verifier V believes that a prover P is physically closer to the verifier V than it really is. However, they differ by the scenarios under study when performing the security analysis. In the following, we propose a formal definition for each class of attacks. Before presenting them, we have to define the set of configurations that need to be studied when analysing a given protocol (V,P). We call them *valid initial configurations*.

Definition 3. *Let (V,P) be a protocol, $\mathcal{T}_0 = (\mathcal{A}_0,\mathcal{M}_0,Loc_0,v_0,p_0)$ be a topology, and Φ_0 be an initial frame. A configuration $\mathcal{K}$ is a* valid initial configuration *for (V,P) w.r.t. $\mathcal{T}_0$ and Φ_0 if:*

- $\mathcal{K} = (\lfloor \mathsf{V}_{\mathsf{end}}(v_0,p_0)\rfloor^0_{v_0} \uplus \mathcal{P}'; \Phi_0; 0)$, *and*
- *for each $\lfloor P'\rfloor^{t'}_{a'} \in \mathcal{P}'$, we have $t' = 0$, $a' \in \mathcal{A}_0$, and either $P' = \mathsf{V}(a',b')$ or $P' = \mathsf{P}(a',b')$ for some $b' \in \mathcal{A}_0$.*

Typically, a valid initial configuration contains instances of the roles of the protocol under study. We simply assume that the specific role $\mathsf{V}_{\mathsf{end}}$ that will be used to encode the security properties occurs only once in the configuration, and that it is instantiated by the two identities appearing in the topology. Depending on the type of frauds we consider, set of topologies under study may vary.

4.1 Mafia fraud

The distance-bounding protocols have originally been designed to resist Mafia frauds. They consist in scenarios where an attacker tries to convince a verifier that an honest prover is close to him even if the prover is actually distant.

Definition 4. *Given $t_0 \in \mathbb{R}_+$, we denote $C^{t_0}_{\mathsf{MF}}$ the set of topologies $\mathcal{T} = (\mathcal{A}_0,\mathcal{M}_0,Loc_0,v_0,p_0)$ corresponding to Mafia fraud scenarios, i.e. such that $v_0,p_0 \in \mathcal{A}_0 \setminus \mathcal{M}_0$ and $Dist_{\mathcal{T}}(v_0,p_0) \geq t_0$.*

In such scenarios, the initial frame only depends on the underlying topology, and more precisely only on the status (honest/dishonest) of each agent and not on their precise location. It contains the initial knowledge of each dishonest agent that is defined through a *template* $\mathcal{I}_0 = (\mathcal{I}_0^V, \mathcal{I}_0^P)$ such that $\mathcal{I}_0^V$ is a set of terms in $\mathcal{T}(\Sigma_c^+, \{z_V^0, z_V^1\})$ and $\mathcal{I}_0^P$ is a set of terms in $\mathcal{T}(\Sigma_c^+, \{z_P^0, z_P^1\})$. Given a set of agents $\mathcal{A}_0$ and a template $\mathcal{I}_0 = (\mathcal{I}_0^V, \mathcal{I}_0^P)$, the initial knowledge of an agent $a \in \mathcal{A}_0$ is defined as follows:

$$\mathsf{Knows}(\mathcal{I}_0, a, \mathcal{A}_0) = \begin{array}{l} \{(u_0\{z_V^0 \mapsto a, z_V^1 \mapsto b\}) \text{ ground} \mid u_0 \in \mathcal{I}_0^V \text{ and } b \in \mathcal{A}_0\} \\ \cup \quad \{(u_0\{z_P^0 \mapsto a, z_P^1 \mapsto b\}) \text{ ground} \mid u_0 \in \mathcal{I}_0^P \text{ and } b \in \mathcal{A}_0\} \end{array}.$$

The initial frame associated to a topology $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ and a template $\mathcal{I}_0$, denoted $\Phi_{\mathcal{I}_0}^{\mathcal{T}}$, is then defined by:

$$\lfloor img(\Phi_{\mathcal{I}_0}^{\mathcal{T}}) \rfloor_a^0 = \mathsf{Knows}(\mathcal{I}_0, a, \mathcal{A}_0) \text{ when } a \in \mathcal{M}_0, \text{ and } \lfloor img(\Phi_{\mathcal{I}_0}^{\mathcal{T}}) \rfloor_a^0 = \emptyset \text{ otherwise.}$$

Up to a renaming of the frame variables and some duplicates, $\Phi_{\mathcal{I}_0}^{\mathcal{T}}$ is uniquely defined.

Example 9. *One may note that the initial frame Φ_0 presented in Example 7, is such that $\Phi_0 = \Phi_{\mathcal{I}_0}^{\mathcal{T}}$ where $\mathcal{I}_0$ is the template made of the sets $(\mathcal{I}_0^V, \mathcal{I}_0^P)$ introduced in Example 5 and $\mathcal{T}$ is the topology presented in Example 6.*

Definition 5. *Let $\mathcal{I}_0$ be a template and (V, P) be a protocol. We say that (V, P)* admits a Mafia fraud *w.r.t. t_0-proximity if there exist $\mathcal{T} \in C_{\mathsf{MF}}^{t_0}$, and a valid initial configuration $\mathcal{K}$ for (V, P) w.r.t. $\mathcal{T}$ and $\Phi_{\mathcal{I}_0}^{\mathcal{T}}$ such that:*

$$\mathcal{K} \to_{\mathcal{T}}^* (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathcal{P}; \Phi; t) \text{ with } \mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0).$$

The reachability of the $\mathsf{end}(v_0, p_0)$ event indicates that honest prover p_0 has been authenticated by the honest verifier v_0. We only consider configurations built from the topology $\mathcal{T} \in C_{\mathsf{MF}}^{t_0}$, and thus, by definition of $C_{\mathsf{MF}}^{t_0}$, we have that p_0 is distant from the verifier v_0. Note however that the definition of $C_{\mathsf{MF}}^{t_0}$ does not impose any other restriction: some other agents may be involved in a Mafia fraud.

Example 10. *The SPADE protocol has been proved secure w.r.t. Mafia fraud [16]. However, their model was preventing the attacker to act as a verifier. By relaxing this constraint, we found (using the approach developed in this paper) that the protocol becomes vulnerable to Mafia fraud as presented in Figure 3. The attacker (here named p) first acts as a verifier and we assume that the honest prover p_0 initiates a session with her by sending $\mathsf{aenc}(\langle n_p, \sigma \rangle, \mathsf{pk}(p))$. He obtains a valid signature $\sigma = \mathsf{sign}(n_P, \mathsf{ssk}(p_0))$ that he can use to forge a valid message that he will send to v_0, i.e. $\mathsf{aenc}(\langle n_p, \sigma \rangle, \mathsf{pk}(v_0))$. Then, the protocol executes almost normally but the attacker who is close to the verifier and who knows n_p participates in the fast phase. The verifier v_0 thinks that he executed the protocol with p_0 whereas he is talking to the attacker p. Therefore, a Mafia fraud exists: an attacker can make an honest verifier accepts a distant honest prover.*

To show how this attack is captured in our framework, we consider the template $\mathcal{I}_0$ depicted in Example 5, the topology $\mathcal{T} \in C_{\mathsf{MF}}^{t_0}$ used in Example 6 and the configuration $\mathcal{K}_0$ defined in Example 7. According to Definition 3, $\mathcal{K}_0$ is a valid initial configuration for the protocol SPADE w.r.t. $\mathcal{T}$ and Φ_0. We have that $\lfloor img(\Phi_0) \rfloor_p^0 = \mathsf{Knows}(\mathcal{I}_0, p, \mathcal{A}_0)$, $\lfloor img(\Phi_0) \rfloor_{v_0}^0 = \emptyset$, and $\lfloor img(\Phi_0) \rfloor_{p_0}^0 = \emptyset$. The protocol

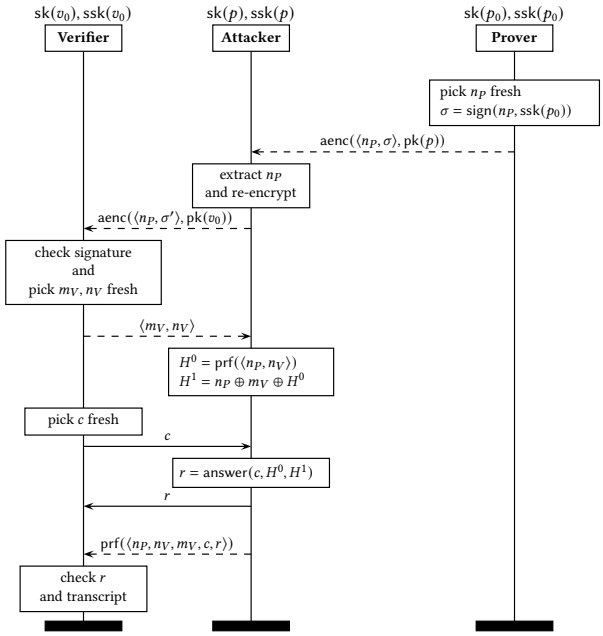


Fig. 3. Mafia fraud against the SPADE protocol

SPADE is vulnerable to a Mafia fraud as witnessed by the following execution:

$$\mathcal{K}_0 \xrightarrow{p_0,\tau}_{\mathcal{T}} \xrightarrow{p_0,\mathsf{out}(\mathsf{aenc}(\langle n'_P,\mathsf{sign}(n'_P,\mathsf{ssk}(p_0))\rangle,\mathsf{pk}(p)))}_{\mathcal{T}} \xrightarrow{}_{\mathcal{T}} \xrightarrow{v_0,\mathsf{in}(\mathsf{aenc}(\langle n'_P,\mathsf{sign}(n'_P,\mathsf{ssk}(p_0))\rangle,\mathsf{pk}(v_0)))}_{\mathcal{T}}$$

$$\xrightarrow{v_0,\tau}_{\mathcal{T}} \xrightarrow{v_0,\tau}_{\mathcal{T}} \xrightarrow{v_0,\tau}_{\mathcal{T}} \xrightarrow{v_0,\tau}_{\mathcal{T}} \xrightarrow{v_0,\mathsf{out}(\langle m'_V,n'_V\rangle)}_{\mathcal{T}} \xrightarrow{v_0,\tau}_{\mathcal{T}} \xrightarrow{v_0,\mathsf{out}(c'_V)}_{\mathcal{T}}$$

$$\xrightarrow{v_0,\mathsf{in}^{<2\cdot t_0}(\mathsf{answer}(c'_V,\mathsf{prf}(\langle n'_P,n'_V\rangle),n'_P\oplus m'_V\oplus\mathsf{prf}(\langle n'_P,n'_V\rangle)))}_{\mathcal{T}} \xrightarrow{v_0,\tau}_{\mathcal{T}} \xrightarrow{v_0,\tau}_{\mathcal{T}} \xrightarrow{v_0,\tau}_{\mathcal{T}}$$

$$\xrightarrow{v_0,\mathsf{in}(\mathsf{prf}(\langle n'_P,n'_V,m'_V,c'_V,\mathsf{answer}(c'_V,\mathsf{prf}(\langle n'_P,n'_V\rangle),n'_P\oplus m'_V\oplus\mathsf{prf}(\langle n'_P,n'_V\rangle)))))}_{\mathcal{T}} \xrightarrow{v_0,\tau}_{\mathcal{T}} \mathcal{K}_f$$

where the configuration $\mathcal{K}_f$ is equal to $\mathcal{K}_f = (\lfloor \mathsf{end}(v_0,p_0)\rfloor_{v_0}^0 \uplus \lfloor P_1(p_0,p)\rfloor_{p_0}^{t_0}; \Phi_f; t_0)$, with $P_1(z_0,z_1)$ is the process $\mathsf{P}(z_0,z_1)$ without the generation of the first nonce and without the first output, and Φ_f is as follows:

$$\Phi_f = \Phi_0 \cup \left\{ w_6 \xrightarrow{p_0,0} \mathsf{aenc}(\langle n'_P, \mathsf{sign}(n'_P, \mathsf{ssk}(p_0))\rangle, \mathsf{pk}(p)), w_7 \xrightarrow{v_0,t_0} \langle m'_V, n'_V\rangle, w_8 \xrightarrow{v_0,t_0} c'_V \right\}.$$

The first line of the execution is the one described in Example 8. Then, v_0 checks the input using two LET transitions, and generates two nonces m'_V and n'_V before sending them. After that, v_0 initiates a RST transition before it inputs a message constructed by p using the following recipe:

$$R_2 = \mathsf{answer}(w_8, \mathsf{prf}(\langle R', \mathsf{proj}_2(w_7)\rangle), R' \oplus \mathsf{proj}_1(w_7) \oplus \mathsf{prf}(\langle R', \mathsf{proj}_2(w_7)\rangle))$$

with $R' = \mathsf{proj}_1(\mathsf{adec}(w_6, w_4))$. The verifier v_0 checks the input through three LET transitions and inputs a final message constructed by the attacker p using the following recipe:

$$R_3 = \mathsf{prf}(\langle R', \mathsf{proj}_2(w_7), \mathsf{proj}_1(w_7), w_8, R_2\rangle).$$

The verifier performs one last check with a LET transition and reaches finally $\mathsf{end}(v_0, p_0)$.

Example 11. *In his thesis [37], Gerault proposed a fix to prevent the Mafia fraud presented above. This fix consists in adding the identity of the verifier inside the first signature, i.e. the first message becomes* $\mathsf{aenc}(\langle n_P, \sigma\rangle, pk(v_0))$ *with* $\sigma = \mathsf{sign}(\langle n_P, v_0\rangle, \mathsf{sk}(p_0))$. *This indeed prevents the aforementioned Mafia fraud attack, and actually the approach developed here will allow us to show in Section 8 (Table 1) that this fixed protocol is secure w.r.t. Mafia frauds.*

4.2 Terrorist fraud

In some aspects, a Terrorist fraud may be considered as a strong version of a Mafia fraud. In this scenario, a remote prover accepts to collude with an attacker located in the vicinity of a verifier to be authenticated once. However, this help must not give an advantage to the attacker to mount future attacks, like impersonations.

Modelling Terrorist fraud is trickier than modelling Mafia fraud. Indeed, the distant prover is neither honest (he may not follow the protocol rules) nor fully dishonest (he does not accept to reveal his long-term keys which would eventually lead to an impersonation attack). To model these scenarios, first, we will consider all the possible behaviours for this semi-dishonest prover that allow the attacker to authenticate at least once, considering the simple topology defined below.

Definition 6. *Given* $t_0 \in \mathbb{R}_+$, *we denote* $\mathcal{T}^{t_0}_{\mathsf{simple}}$ *the topology* $\mathcal{T} = (\{v_0, p_0, p\}, \{p\}, Loc_0, v_0, p_0)$ *such that* $Dist_{\mathcal{T}}(v_0, p) = 0$, *and* $Dist_{\mathcal{T}}(v_0, p_0) = t_0$.

We may note that this topology is not uniquely defined (e.g. the precise location of each agent is not fixed) but this does not have any impact on our development.

Then, to be Terrorist fraud resistant, we have to check that any of these behaviours will allow the attacker to re-authenticate later on. This shows that the prover cannot reveal enough information to let the attacker authenticate once without running the risk of being impersonated after.

Definition 7. *Let* $t_0 \in \mathbb{R}_+$, *and* (V, P) *be a protocol. A* semi-dishonest prover *for* (V, P) *w.r.t.* t_0 *is a process* P_{sd} *together with an initial frame* Φ_{sd} *such that:*

$$(\{\lfloor \mathsf{V}_{\mathsf{end}}(v_0, p_0)\rfloor^0_{v_0}\,;\, \lfloor P_{\mathsf{sd}}\rfloor^0_{p_0}\}; \emptyset; 0) \to^*_{\mathcal{T}^{t_0}_{\mathsf{simple}}} (\{\lfloor \mathsf{end}(v_0, p_0)\rfloor^{t_v}_{v_0}\,;\, \lfloor 0\rfloor^{t_p}_{p_0}\}; \Phi; t)$$

for some t, t_v, t_p, *and* Φ *such that* Φ *and* Φ_{sd} *coincide up to their timestamps.*

Note that a semi-dishonest prover can be completely dishonest in the sense that he may leak all his credentials. This will indeed allow the attacker to play the role of the prover and to be authenticated by the verifier. However, such a semi-dishonest prover cannot be equal to the role of the prover. Indeed, such an honest prover role will not allow a distant prover to authenticate with a verifier; unless for a badly designed protocol. The semi-dishonest prover will need to leak some information to the attacker who is close to the verifier to be authenticated.

In our definition of semi-dishonest prover, we do not try to precisely characterize its knowledge, and we therefore consider some unrealistic behaviours. For instance, it may seem unrealistic to assume that such a semi-dishonest prover knows the secret key of the verifier. This leads to a fairly strong notion of Terrorist fraud resistance. Later on, we will show that, for the class of well-formed protocols, it is actually sufficient to consider a particular semi-dishonest prover (the most general one) to perform the security analysis. This most general one is derived from the prover role, and does not have unexpected behaviours.

Example 12. *Throughout Section 4.2 and Section 6 devoted to Terrorist frauds, we consider the updated version of the SPADE protocol as given in Example 11, fixing the Mafia fraud described above. This will allow us to better illustrate the different notions introduced in these sections. As we*

have seen, this update adds the identity of the verifier in the signature within the first message, i.e. $\mathsf{aenc}(\langle n_P, \mathsf{sign}(\langle n_P, z_P^1\rangle, \mathsf{ssk}(z_P^0))\rangle, \mathsf{pk}(z_P^1))$ *and modifies the corresponding consistency checks on the Verifier side. Considering this new protocol, a semi-dishonest prover could be as follows:*

$$\begin{aligned} P_{\mathsf{sd}}^1 := \quad & \mathsf{new}\ n_P.\ \mathsf{out}(\mathsf{aenc}(\langle n_P, \mathsf{sign}(\langle n_P, v_0\rangle, \mathsf{ssk}(p_0))\rangle, \mathsf{pk}(v_0))).\\ & \mathtt{in}(x).\ \mathtt{let}\ x_1 = \mathsf{proj}_1(x)\ \mathtt{in}\ \ \mathtt{let}\ x_2 = \mathsf{proj}_2(x)\ \mathtt{in}\\ & \mathtt{let}\ x_3 = \mathsf{prf}(\langle n_P, x_2\rangle)\ \mathtt{in}\ \ \mathtt{let}\ x_4 = n_P \oplus x_1 \oplus x_3\ \mathtt{in}\\ & \mathsf{out}(x_3).\ \mathsf{out}(x_4).\ \mathtt{in}(y_c).\\ & \mathsf{out}(\mathsf{answer}(y_c, x_3, x_4)).\ \mathsf{out}(\mathsf{prf}(\langle n_P, x_2, x_1, y_c, \mathsf{answer}(y_c, x_3, x_4)\rangle)).0. \end{aligned}$$

Up to some renaming of fresh names, the corresponding frame Φ_{sd}^1 *is:*

$$\begin{aligned} \Phi_{\mathsf{sd}}^1 = \Big\{ & w_6 \xrightarrow{p_0,0} \mathsf{aenc}(\langle n_P, \mathsf{sign}(\langle n_P, v_0\rangle, \mathsf{ssk}(p_0))\rangle, \mathsf{pk}(v_0)), w_7 \xrightarrow{v_0,t_0} \langle m_V, n_V\rangle,\\ & w_8 \xrightarrow{p_0,2\cdot t_0} \mathsf{prf}(\langle n_P, n_V\rangle), w_9 \xrightarrow{p_0,2\cdot t_0} n_P \oplus m_V \oplus \mathsf{prf}(\langle n_P, n_V\rangle), w_{10} \xrightarrow{v_0,3\cdot t_0} c_V,\\ & w_{11} \xrightarrow{p_0,4\cdot t_0} \mathsf{answer}(c_V, \mathsf{prf}(\langle n_P, n_V\rangle, n_P \oplus m_V \oplus \mathsf{prf}(\langle n_P, n_V\rangle),\\ & w_{12} \xrightarrow{p_0,4\cdot t_0} \mathsf{prf}(\langle n_P, n_V, m_V, c_V, \mathsf{answer}(c_V, \mathsf{prf}(\langle n_P, n_V\rangle, n_P \oplus m_V \oplus \mathsf{prf}(\langle n_P, n_V\rangle)))\rangle)\Big\}. \end{aligned}$$

This corresponds to a semi-dishonest prover who performs the computations of both H^0 *and* H^1 *and reveals these two messages (before the fast phase starts) to his accomplice who is close to the verifier. Therefore, the accomplice will be able to answer to the challenges provided by the verifier using the recipe* $R = \mathsf{answer}(w_{10}, w_8, w_9)$ *where* w_{10} *is the handle binding the challenge sent by the verifier, and* w_8*,* w_9 *are the two handles binding the messages* x_3 *and* x_4 *leaked by the semi-dishonest prover and that his accomplice can receive before the fast phase starts.*

We are now able to define our notion of Terrorist fraud resistance. Intuitively, if the distant semi-dishonest prover p_0 gives to his accomplice p enough information to pass authentication once, then this accomplice will be able to impersonate p_0 in the future, i.e. p will be able to make the verifier authenticate p_0, even if p_0 no longer colludes.

Definition 8. *Let* $\mathcal{I}_0$ *be a template. We say that a protocol* (V, P) *is* Terrorist fraud resistant *w.r.t.* t_0*-proximity if, for all semi-dishonest provers* P_{sd} *(w.r.t.* t_0*) with frame* Φ_{sd}*, there exist a topology* $\mathcal{T} \in \mathcal{C}_{\mathsf{MF}}^{t_0}$ *and a valid initial configuration* $\mathcal{K}$ *w.r.t.* $\mathcal{T}$ *and* $\Phi_{\mathcal{I}_0}^{\mathcal{T}} \cup \Phi_{\mathsf{sd}}$ *such that*

$$\mathcal{K} \to_{\mathcal{T}}^* (\ \lfloor \mathsf{end}(v_0, p_0)\rfloor_{v_0}^{t'} \uplus \mathcal{P}; \Phi; t) \text{ where } \mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0).$$

Again, the reachability of the $\mathsf{end}(v_0, p_0)$ event indicates that the honest prover p_0 has been authenticated by the honest verifier v_0. Contrary to the Mafia frauds, we may note that the initial configuration $\mathcal{K}$ has been built relying on the frame $\Phi_{\mathcal{I}_0}^{\mathcal{T}} \cup \Phi_{\mathsf{sd}}$ (and not $\Phi_{\mathcal{I}_0}^{\mathcal{T}}$ only). This reflects the fact that the prover has colluded with the attacker to be authenticated once, and the attacker has gained some knowledge. We now want to be sure that this additional knowledge will not allow the attacker to mount further attacks, i.e. to authenticate again.

We can note that unlike standard reachability properties, exhibiting a trace of execution is not sufficient to prove the existence of a Terrorist fraud. Indeed, it requires to exhibit a semi-dishonest prover and check that for all the possible executions, none of them leads to a re-authentication. Unfortunately, exhibiting a trace is not sufficient to prove Terrorist fraud resistance neither: the re-authentication must be possible for all the semi-dishonest provers. Each direction requires us to explore an infinite set: either the set of traces or the set of semi-dishonest provers. A situation in which Terrorist fraud analysis is simple is when the protocol already suffers from a Mafia fraud. The following proposition applies.

Proposition 1. *Let $\mathcal{I}_0$ be a template, and (V,P) be a protocol. If (V,P) admits a Mafia fraud, then it is Terrorist fraud resistant (w.r.t. t_0-proximity).*

Indeed, an execution that witnesses the existence of a Mafia fraud can be leveraged as a witness of re-authentication for any semi-dishonest prover. By definition of a Mafia fraud, the execution starts with a configuration $\mathcal{K}$ that is a valid initial configuration for (V,P) w.r.t. $\mathcal{T}$ and $\Phi^{\mathcal{T}}_{\mathcal{I}_0}$. It can be extended as a valid initial configuration for (V,P) w.r.t. $\mathcal{T}$ and $\Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi_{\mathsf{sd}}$ by adding elements in the initial frame. This does not alter the trace since it simply increases the initial knowledge of the attackers.

Regarding the literature, this implication is debatable. Indeed, a contradictory implication is sometimes stated [3, 19]: if a protocol admits a Mafia fraud then it admits a Terrorist fraud. To explain such a gap, we start by recalling the informal definition of Terrorist fraud: a Terrorist fraud is a scenario in which a distant dishonest prover accepts to collude with an attacker to be authenticated once by an honest verifier, but without giving any advantage to the attacker in the future. Therefore, there are two ways to formally define it:

(1) consider all the dishonest provers that do not give any advantage to the attacker for future (i.e. do not enable a re-authentication); and check whether they can be authenticated once;
(2) consider all the dishonest provers that can be authenticated once; and check whether they give an advantage to the attacker for future attacks.

The first definition is appealing since it expresses Terrorist fraud as a unique reachability issue: is authentication possible? The implication "Mafia fraud implies Terrorist fraud" is quite immediate: if there is a Mafia fraud then authentication is possible. However, it remains to define the set of "dishonest provers that do not give any advantage to the attacker" and this appears to be a difficult task, especially in symbolic models. The second definition is more in line with ours. It provides a more complex security property but it does not elude the difficulty of defining the "advantage" by pushing it inside the definition of admissible collusion. The two definitions seem to match when analysing protocols that are Mafia fraud resistant.

Example 13. *The original SPADE protocol is vulnerable to a Mafia fraud. Following the previous remark we can thus immediately conclude that it is Terrorist fraud resistant. Now, considering the fixed version presented in Example 11 and which is resistant to Mafia frauds, we can show that it is also Terrorist fraud resistant, i.e. for all the semi-dishonest provers there exists a trace of re-authentication. For illustration purposes, we consider the semi-dishonest prover given in Example 12, and we show that re-authentication is indeed possible. To do this, the attacker replays the first message sent by p_0 and stored in w_6 (first element in the frame Φ^1_{sd}). Then, he is able to answer all the messages requested by v_0 since he can deduce n_P from the frame:*

$$n_P = R\Phi^1_{\mathsf{sd}}\downarrow \text{ with } R = w_8 \oplus w_9 \oplus \mathsf{proj}_1(w_7).$$

Indeed, all the messages (except the first one that the attacker replays) of a protocol session are built applying public symbols of function (e.g. prf, $\oplus$, answer) to the atomic data n_P (known by the attacker), n'_V and m'_V two fresh values generated by the verifier and sent to the attacker during the execution of the session.

The analysis done in Example 13 is not sufficient to establish that the protocol is Terrorist fraud resistant. Indeed, a single dishonest prover has been considered so far. Moreover, the analysis has been done considering a rather simple topology. The reduction results developed in Section 5 and Section 6, will allow us to formally establish Terrorist fraud resistance. In particular, in Section 6, we show that the semi-dishonest prover given in Example 12 is the most general one, and thus the only one that has to be considered when performing the security analysis.

4.3 Distance Hijacking attacks

Another class of attacks a distance-bounding protocol should resist is the Distance Hijacking attack in which a dishonest prover tries to be authenticated by a remote verifier. To succeed the dishonest prover may hijack honest agents located (or not) in the vicinity of the verifier.

Definition 9. *Given $t_0 \in \mathbb{R}_+$, we denote $C_{\mathsf{DH}}^{t_0}$ the set of topologies $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0)$ corresponding to Distance Hijacking scenarios, i.e. such that $p_0 \in \mathcal{M}_0$, and $Dist_{\mathcal{T}}(a, v_0) \geq t_0$ for any $a \in \mathcal{M}_0$.*

This definition encompasses the particular case in which nobody is located in the vicinity of the verifier, known as Distance fraud.

Definition 10. *Let $\mathcal{I}_0$ be a template. We say that a protocol* (V, P) admits a Distance Hijacking attack *w.r.t. t_0-proximity if there exist $\mathcal{T} \in C_{\mathsf{DH}}^{t_0}$, and a valid initial configuration $\mathcal{K}$ w.r.t. $\mathcal{T}$ and $\Phi_{\mathcal{I}_0}^{\mathcal{T}}$ such that*

$$\mathcal{K} \rightarrow_{\mathcal{T}}^* (\, \lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathcal{P}; \Phi; t) \text{ with } \mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0).$$

This definition is similar to the one for Mafia fraud. However, we do not consider the same set of topologies. Note that a topology in $C_{\mathsf{DH}}^{t_0}$ does not allow a dishonest agent to be present in the vicinity of v_0. Moreover, contrary to the case of a Mafia fraud, p_0 is assumed to be dishonest.

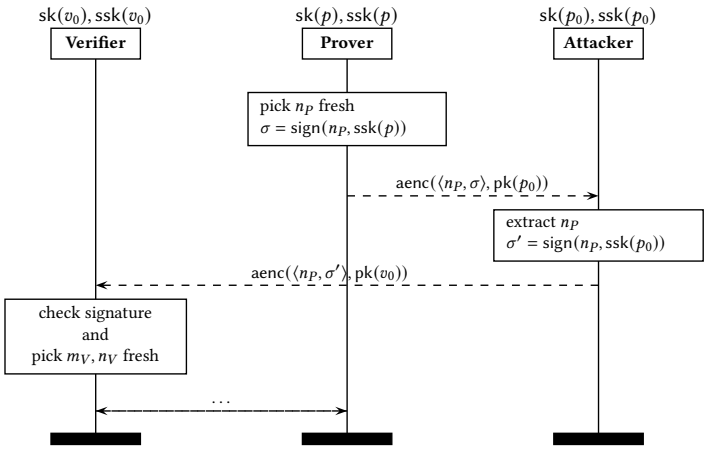


Fig. 4. Distance hijacking attack against the SPADE protocol

Example 14. *The SPADE protocol has been proved secure w.r.t. Distance Hijacking attacks [16]. However, as already said, their model was preventing the attacker to act as a verifier. By relaxing this constraint, the protocol becomes vulnerable to a Distance Hijacking attack, briefly presented in Figure 4. The attacker acts as a verifier and we assume that an honest prover initiates a session with him. He is then able to reuse the nonce n_P sent by the prover to initiate a session with the verifier v_0. The rest of the protocol will then be executed between the verifier v_0 and the honest prover p (who thinks he is talking with the verifier p_0). Formally, this attack is caught by our definition considering:*

- *$\mathcal{T} = (\{v_0, p_0, p\}, \{p_0\}, Loc_0, v_0, p_0)$ with $Dist_{\mathcal{T}}(v_0, p_0) = t_0$ and $Dist_{\mathcal{T}}(v_0, p) = 0$, and*

- $\mathcal{K}_0 = (\lfloor \mathsf{V}_{\mathsf{end}}(v_0, p_0) \rfloor^0_{v_0} \uplus \lfloor \mathsf{P}(p, p_0) \rfloor^0_p ; \Phi_0; 0)$ *where*
$$\Phi_0 = \{w_1 \xrightarrow{p_0,0} v_0,\ w_2 \xrightarrow{p_0,0} p_0,\ w_3 \xrightarrow{p_0,0} p,\ w_4 \xrightarrow{p_0,0} \mathsf{sk}(p_0),\ w_5 \xrightarrow{p_0,0} \mathsf{ssk}(p_0)\}.$$

4.4 Comparison with existing approaches regarding the modelling aspect

We elaborate now on two recent symbolic models, developed for analysing distance-bounding protocols by Chothia *et al.* [19, 20] and Mauw *et al.* [44, 45]. In particular, we will compare the definitions of the security properties.

4.4.1 Chothia et al.*'s model.* This model, published in 2015, is the basis of our work. Hence, our definitions of Mafia fraud and Distance Hijacking attack are in line with those proposed in [20]. The main difference lies in the quantification over the topologies: Chothia *et al.* define the security properties w.r.t. to rather simple topologies made of, at most, 4 agents, without providing any justification. The reduction results described in Section 5 provide such a justification and also make clear the requirements.

Now, regarding Terrorist fraud, the definition proposed in [19] differs from ours. While we are following the second definition of Terrorist fraud mentioned in Section 4.2, they are following the first one. Their security property involves what is called a *terrorist prover* which performs any operation on behalf of the attacker. It can for example encrypt, decrypt, or sign any value the attacker wishes, but never reveals its secrets. Even if this notion of terrorist prover is appealing, because suitable for automation, they do not explain how to write such a process and we do think that it may be a difficult task. In the following, we present two examples that illustrate such difficulties.

Example 15. *Consider a protocol such that the prover role relies on a secret k and a hash function* h. *Given an input u sent by his accomplice, a legitimate behaviour of the terrorist prover may be to reveal the hash value of the input data together with his secret key, i.e.* $\mathsf{h}(\langle k, u\rangle)$. *Indeed such a message might help his accomplice and does not reveal any information about k. Formally, the terrorist prover should contain the oracle:* $\mathsf{P}_1 = \mathsf{in}(x).\mathsf{out}(\mathsf{h}(\langle k, x\rangle))$. *In the same spirit, we could argue that the oracle* $\mathsf{P}_2 = \mathsf{in}(x).\mathsf{out}(\mathsf{h}(\langle x, k\rangle))$ *is also useful, and perhaps also the oracle* $\mathsf{P}_3 = \mathsf{in}(x_1).\mathsf{in}(x_2).\mathsf{out}(\mathsf{h}(\langle x_1, \langle k, x_2\rangle\rangle))$, *etc. Iterating such a reasoning, it is unclear how to write a finite terrorist prover that will provide all the valuable help his accomplice may need.*

Another issue comes up when considering equational theories modelling operators with algebraic properties, like the exclusive-or. It seems difficult (perhaps even impossible) to be sure that the terrorist prover we consider will not reveal some secrets (possibly indirectly).

Example 16. *Consider an equational theory made of three public symbols of function* g, f_1 *and* f_2 *such that* $\mathsf{g}(\mathsf{f}_1(x, y), \mathsf{f}_2(x, y)) = y$. *Following the idea developed in [19], the terrorist prover should contain the two oracles* $\mathsf{P}_1 = \mathsf{in}(x).\mathsf{out}(\mathsf{f}_1(x, k))$ *and* $\mathsf{P}_2 = \mathsf{in}(x).\mathsf{out}(\mathsf{f}_2(\langle x, k\rangle))$. *Even though these two oracles are individually legitimate, together, they will allow an attacker to get* $\mathsf{f}_1(m, k)$ *and* $\mathsf{f}_2(m, k)$ *for some message m and thus retrieve the secret key k. This example clearly shows that it is not obvious to describe in a syntactic way the help the terrorist prover is willing to provide (even for rather simple equational theories).*

4.4.2 Mauw et al.*'s model.* Meanwhile we were developing our framework [26, 27], Mauw *et al.* proposed another model based on multiset rewriting. Regarding Terrorist fraud, their security property is completely in line with ours. Indeed, they seem equivalent up to a different convention naming (i.e. *valid extension* instead of semi-dishonest prover), and small differences that might arise due to the specificities of each formalism (applied pi-calculus versus multiset rewriting rules). They also consider two-step scenarios in which a semi-dishonest prover tries to authenticate once and then check whether it can be re-authenticated later on. However, while we are considering a

unique and rather simple topology $\mathcal{T}^{t_0}_{\text{simple}}$ for the first authentication, they consider any topology $\mathcal{T} \in \mathcal{C}^{t_0}_{\text{MF}}$. Their definition is thus slightly more general than ours.

Regarding Mafia fraud and Distance Hijacking attack, we can note a difference. Indeed, instead of modelling these two classes of attacks separately, they define a unique property that gathers both, named *secure distance-bounding*. They claim that:

> A protocol is *distance-bounding secure* if for all trace of execution tr that contains the event $\text{claim}(V, P, x, y)$ for an honest agent V, then there are two actions (t_x, α_x) and (t_y, α_y) in tr which correspond to x and y and such that $(t_y - t_x) \leq 2 \cdot \text{Dist}(V, P')$ with $P \approx P'$.

The notation $P \approx P'$ allows to replace agent P by any dishonest agent when P is dishonest. Indeed, in this case, it can share all its credentials with an accomplice who can then impersonate him. The verifier can thus only estimate its distance to the closest malicious agent. It is important to note that this security property gathers both Mafia fraud and Distance Hijacking attack since it applies to scenarios in which the prover P may be honest or dishonest.

According to the authors, this property can be constrained to focus on Mafia fraud only by assuming the prover P to be honest. Therefore, we have that:

- a protocol admits a Mafia fraud if *secure distance-bounding* does not hold when considering P to be honest; and
- a protocol admits a Distance Hijacking attack if *secure distance-bounding* holds when considering P to be honest, and does not hold otherwise.

This means that their framework does not allow one to analyse these security properties independently whereas we think it may be interesting in practice. For instance, if the infrastructure in which the distance-bounding protocol is implemented enforces, by design, that a malicious agent cannot enter in the verifier's proximity, e.g. relying on CCTV (Closed-Circuit Television).

5 REDUCING TOPOLOGIES

When analysing a protocol w.r.t. Distance Hijacking, Mafia or Terrorist fraud, an infinite number of topologies must be considered. Indeed, an infinite number of them belong to $\mathcal{C}^{t_0}_{\text{DH}}$ and $\mathcal{C}^{t_0}_{\text{MF}}$. In this section, we establish three reduction results to get rid of this source of unboundedness. More precisely we prove that, for each class of attacks, we can focus the analysis on a rather simple topology (depicted in Figure 5) involving at most four agents.

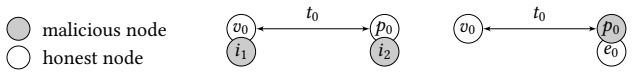


Fig. 5. Topologies $\mathcal{T}^{t_0}_{\text{MF}}$ and $\mathcal{T}^{t_0}_{\text{DH}}$

5.1 Mafia and Terrorist frauds

The same set of topologies, $\mathcal{C}^{t_0}_{\text{MF}}$, must be considered when verifying Mafia or Terrorist fraud resistance. In the reminder of this section, we prove that if a trace exists considering an arbitrary topology $\mathcal{T} \in \mathcal{C}^{t_0}_{\text{MF}}$ then, up to time shifts, the same trace can be executed in $\mathcal{T}^{t_0}_{\text{MF}}$. This proof proceeds in four stages:

(1) we transform all the honest agents but v_0 and p_0 in dishonest ones (Lemma 1);
(2) relying on the executability hypothesis, we simplify the initial configuration (Lemma 2);
(3) we reduce the number of attackers by placing them ideally (Lemma 3);
(4) we rename agents preserving their locations to reach the reduced topology $\mathcal{T}^{t_0}_{\text{MF}}$ (Lemma 4).

The first stage consists in proving that we can corrupt honest participants without altering the set of possible traces. In other words, a dishonest agent is able to behave as an honest one.

Lemma 1. *Let $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0)$ be a topology, and $\mathcal{K}_0$ be a configuration built on $\mathcal{T}_0$. Let $\mathcal{H}_0 \subseteq \mathcal{A}_0 \setminus \mathcal{M}_0$. Let $\mathcal{K}$ be a configuration such that $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} \mathcal{K}$. We have that $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}'} \mathcal{K}$ where $\mathcal{T}' = (\mathcal{A}_0, \mathcal{M}_0 \cup \mathcal{H}_0, Loc_0, v_0, p_0)$.*

In the second stage, we prove that, relying on the executability hypothesis, it is not necessary to consider processes executed by dishonest agents. Indeed, the initial knowledge contained in the frame is actually enough to mimic these processes.

Lemma 2. *Let $\mathcal{T}_0$ be a topology, $\mathcal{D}_0$ be a subset of malicious agents, and $\mathcal{K}_0 = (\mathcal{P}_0; \Phi_0; t_0)$ be a configuration built on $\mathcal{T}_0$ such that P_a is executable w.r.t. $img(\lfloor \Phi_0 \rfloor_a^{t_0})$ for any $\lfloor P_a \rfloor_a^{t_a} \in \mathcal{P}_0$ with $a \in \mathcal{D}_0$. Let $\mathcal{K} = (\mathcal{P}; \Phi_0 \uplus \Phi^+; t)$ be a configuration such that $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} \mathcal{K}$. We have that*

$$(\overline{\mathcal{P}_0}; \Phi_0; t_0) \xrightarrow{\overline{\mathsf{tr}}\sigma}_{\mathcal{T}_0} (\overline{\mathcal{P}}\sigma; \Phi_0 \uplus \overline{\Phi^+}\sigma; t)$$

where $\overline{\mathcal{P}_0}$ (resp. $\overline{\mathcal{P}}$, $\overline{\Phi^+}$, $\overline{\mathsf{tr}}$) is obtained from $\mathcal{P}_0$ (resp. $\mathcal{P}$, Φ^+, tr) by removing processes (resp. frame or trace elements) located in $a \in \mathcal{D}_0$ and $\sigma(n) = \mathsf{c}_0 \in \Sigma_0$ for any name n freshly generated to trigger the rules NEW executed by agent $a \in \mathcal{D}_0$ in tr.

The two first stages simplify the initial configuration but many dishonest agents located at many different locations still belong to the topology. To reduce the number of these dishonest agents, we got some inspiration from the work done by Nigam *et al.* in [51]. In [51], the authors prove that considering topologies in which each honest agent is accompanied by a dishonest one located at the same place is enough. These canonical topologies are defined as follows: given a set $H = \{a_1, \ldots, a_p\}$ of honest agents together with a location function $\mathsf{Loc}_H : H \mapsto \mathbb{R}^3$, we define the canonical topology $\mathcal{T}_{\mathsf{Loc}_H}$ associated to Loc_H as $\mathcal{T}_{\mathsf{Loc}_H} = (H \uplus \mathcal{M}_0, \mathcal{M}_0, \mathsf{Loc}_H \uplus \mathsf{Loc}_0, v_0, p_0)$ where:

$$\mathcal{M}_0 = \{i_1, \ldots, i_p\} \text{ with } i_1 \ldots, i_p \in \mathcal{A} \setminus H; \text{ and } \mathsf{Loc}_0(i_j) = \mathsf{Loc}_H(a_j) \text{ for } j \in \{1, \ldots, p\}.$$

Lemma 3. *Let $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc, v_0, p_0)$ be a topology, $\mathcal{K}_0 = (\mathcal{P}_0; \Phi_0; t_0)$ and $\mathcal{K}$ be two configurations built on $\mathcal{T}_0$ such that $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}} \mathcal{K}$, and H be a set of agents such that*

$$\{a \mid \lfloor P \rfloor_a^t \in \mathcal{P}_0 \text{ or } \lfloor \Phi_0 \rfloor_a^t \neq \emptyset\} \subseteq H \subseteq \mathcal{A}_0 \setminus \mathcal{M}_0.$$

We have that $(\mathcal{P}_0; \Phi_0; t_0) \xrightarrow{\mathsf{tr}}_{\mathcal{T}'} \mathcal{K}$ where $\mathcal{T}'$ is the canonical topology associated to H and $Loc_{|H}$.

Note that H must at least contain the names of the agents executing processes in $\mathcal{P}_0$ or occurring in the frame Φ_0 to maintain that $\mathcal{K}_0$ is a configuration when considering the topology $\mathcal{T}_{\mathsf{Loc}_H}$.

Finally, the last step consists in reducing the frame. Indeed, this frame may still contain messages involving names of agents who are no longer present in the topology. To avoid such a situation, we prove in Lemma 4 that a renaming of agents can be applied without affecting the execution.

Lemma 4. *Let $\mathcal{K}, \mathcal{K}'$ be two configurations built on $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc, v_0, p_0)$ such that $\mathcal{K} \xrightarrow{\mathsf{tr}}_{\mathcal{T}} \mathcal{K}'$, and $\rho : \mathcal{A} \to \mathcal{A}_0$ be a renaming such that $Loc(\rho(a)) = Loc(a)$ for any $a \in \mathcal{A}_0$, and $\rho(a) \in \mathcal{M}_0$ for any $a \in \mathcal{M}_0$. We have that $\mathcal{K}\rho \xrightarrow{\mathsf{tr}\rho}_{\mathcal{T}} \mathcal{K}'\rho$.*

Combining these four lemmas we are now able to state and prove (see Appendix A) the main reduction that allows us to get rid of the quantification over all the topologies by simply analysing a protocol w.r.t. to the topology $\mathcal{T}_{\mathsf{MF}}^{t_0}$. Given a frame Φ_0, we note $names(\Phi_0)$ the set of agent names occurring in it.

Theorem 1. *Let $\mathcal{I}_0 = (\mathcal{I}_0^V, \mathcal{I}_0^P)$ be a template, (V,P) a protocol, $t_0 \in \mathbb{R}_+$ a threshold, and Φ_0 an initial frame such that $names(\Phi_0) \subseteq \{v_0, p_0\}$. There exists a topology $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0) \in C_{\mathsf{MF}}^{t_0}$ and a valid initial configuration $\mathcal{K}$ for (V,P) w.r.t. $\mathcal{T}_0$ and $\Phi_{\mathcal{I}_0}^{\mathcal{T}_0} \cup \Phi_0$ such that*

$$\mathcal{K} \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathcal{P}; \Phi; t)$$

if, and only if, there exists a valid initial configuration $\mathcal{K}'$ for (V,P) w.r.t. $\Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}} \cup \Phi_0$ such that

$$\mathcal{K}' \xrightarrow{\mathsf{tr}'}_{\mathcal{T}_{\mathsf{MF}}^{t_0}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathcal{P}'; \Phi'; t').$$

As a direct corollary of the theorem above (applied with $\Phi_0 = \emptyset$), we have that we can restrict ourselves to consider the topology $\mathcal{T}_{\mathsf{MF}}^{t_0}$ when looking for a Mafia fraud.

Corollary 1. *Let $\mathcal{I}_0$ be a template, (V,P) a protocol, and $t_0 \in \mathbb{R}_+$ a threshold. We have that (V,P) admits a Mafia fraud w.r.t. t_0-proximity if, and only if, there is an attack against t_0-proximity in $\mathcal{T}_{\mathsf{MF}}^{t_0}$.*

Theorem 1 also allows us to focus on a single topology when considering Terrorist fraud resistance. However, this requires us to show that we can restrict ourselves to consider initial frames satisfying the hypothesis of Theorem 1, i.e., such that $names(\Phi_0) \subseteq \{v_0, p_0\}$.

Corollary 2. *Let $\mathcal{I}_0$ be a template, (V,P) a protocol, and $t_0 \in \mathbb{R}_+$ a threshold. We have that (V,P) is Terrorist fraud resistant w.r.t. t_0-proximity if, and only if, for all semi-dishonest prover P_{sd} w.r.t. t_0 with frame Φ_{sd} such that $names(\Phi_{\mathsf{sd}}) \subseteq \{v_0, p_0\}$, there exists a valid initial configuration $\mathcal{K}_0$ w.r.t. $\mathcal{T}_{\mathsf{MF}}^{t_0}$ and $\Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}} \cup \Phi_{\mathsf{sd}}$ such that $\mathcal{K}_0 \to^*_{\mathcal{T}_{\mathsf{MF}}^{t_0}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t'} \uplus \mathcal{P}; \Phi; t)$.*

5.2 Distance Hijacking attacks

Unfortunately, the reduction proposed in case of Mafia and Terrorist fraud does not apply for Distance Hijacking attack. Indeed, the third step of the reduction consists in placing a dishonest agent close to each honest one. This step introduces a dishonest agent in the vicinity of the verifier which is prohibited when considering Distance Hijacking scenarios.

Nevertheless, we show that under reasonable conditions, we can reduce towards the topology depicted in Figure 5 defined as follows: $\mathcal{T}_{\mathsf{DH}}^{t_0} = (\mathcal{A}_{\mathsf{DH}}, \mathcal{M}_{\mathsf{DH}}, \mathsf{Loc}_{\mathsf{DH}}, v_0, p_0)$ 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 $\mathsf{Dist}_{\mathcal{T}_{\mathsf{DH}}^{t_0}}(p_0, v_0) = t_0$.

Given a process P, we denote $\overline{P}$ the process obtained from P by removing `reset` instructions, and replacing each occurrence of $\mathsf{in}^{<t}(x)$ by $\mathsf{in}(x)$. This notation is extended to a multiset of (extended) processes by applying the transformation on each (extended) process.

Theorem 2. *Let $\mathcal{I}_0$ be a template, (V,P) a protocol, and $t_0 \in \mathbb{R}_+$ a threshold. Moreover, we assume that $\mathsf{V}(v_0, p_0)$ is derived from the following grammar:*

$$\begin{array}{rcl} P & := & 0 \mid \mathsf{in}(x).P \mid \mathsf{let}\ x = v\ \mathsf{in}\ P \\ & \mid & \mathsf{new}\ n.P \mid \mathsf{out}(u).P \mid \mathsf{reset}.\mathsf{out}(u').\mathsf{in}^{<t}(x).P \end{array}$$

where $x \in \mathcal{X}, n \in \mathcal{N}, u, u' \in \mathcal{T}(\Sigma_c^+, \mathcal{X} \cup \mathcal{Z} \cup \mathcal{N})$, $v \in \mathcal{T}(\Sigma^+, \mathcal{X} \cup \mathcal{Z} \cup \mathcal{N})$ and $t \leq 2 \cdot t_0$. If (V,P) admits a Distance Hijacking attack w.r.t. t_0-proximity, then there exists a valid initial configuration $\mathcal{K}_0$ for (V,P) w.r.t. $\mathcal{T}_{\mathsf{DH}}^{t_0}$ and $\Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{DH}}^{t_0}}$ such that $\mathcal{K}_0 = (\mathcal{P}_0 \cup \{\lfloor \mathsf{V}_{\mathsf{end}}(v_0, p_0) \rfloor_{v_0}^0\}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{DH}}^{t_0}}; 0)$ and

$$(\overline{\mathcal{P}_0} \cup \{\lfloor \mathsf{V}_{\mathsf{end}}(v_0, p_0) \rfloor_{v_0}^0\}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{DH}}^{t_0}}; 0) \to^*_{\mathcal{T}_{\mathsf{DH}}^{t_0}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathcal{P}'; \Phi; t).$$

The assumption on the shape of the role V means that the verifier is made of (possibly many) timed challenge/response steps. Such a step starts by setting the local clock of the verifier to 0 using the `reset` command and finishes by a guarded input to prevent remote agents from answering.

Moreover, the resulting trace in the reduced topology $\mathcal{T}_{\mathsf{DH}}^{t_0}$ does no strictly appear as a Distance Hijacking attack w.r.t. t_0-proximity. Indeed, the initial configuration must be weakened by keeping only the reset actions and the guarded inputs of the tested verifier role, i.e. the one that contains the $\mathsf{end}(v_0, p_0)$ command. As we will see in Section 8, this change is tight enough to not introduce false attacks and enable the verification of many distance-bounding protocols.

The proof of this theorem is more complex than the one presented in the previous section for Mafia and Terrorist frauds. The idea of this reduction is to move all the agents in the vicinity of v_0 at the same location as v_0. However, this will lengthen the distance between such an agent and those who are distant from v_0. This may invalidate the witness trace of attack if a message travelling between these two agents was needed to pass a guarded input. To ease the manipulations of the actions in a trace, we will first define more expressive labels, i.e. annotations, and then an untimed semantics, which will give us more flexibility to reorder actions in the trace such that no remote agent execute actions during a challenge/response step. Finally, we will show how to lift a trace in this untimed semantics into a trace in our original timed semantics.

Annotations. We extend the existing labels with more informative annotations. More precisely, the first annotation will identify which process in the multiset has performed the action (session identifier). This will allow us to identify which specific agent performed some action. We also annotate the label with the global time at which the action has been done. In case of an output, we indicate the name of the handle w that has been used to store the output in the frame too. Finally, in case of an input, we indicate by a triplet (b, t_b, R) the name b of the agent responsible of the corresponding output, the time t_b at which this output has been performed, as well as the recipe R used to build this output.

Formally, a label is either empty (for the TIM rule) or of the form a, α with $\alpha \in \{\tau, \mathsf{out}(u), \mathsf{in}^\star(u)\}$, and thus an annotated label is:

- empty for the TIM rule;
- (a, α, s, t, w) when the underlying action a, α is of the form $a, \mathsf{out}(u)$. In such a case, s is the session identifier of the agent responsible of this action, t is the global time at which this output has been done, and w is the handle added in the frame;
- $(a, \alpha, s, t, (b, t_b, R))$ when the underlying action a, α is of the form $a, \mathsf{in}^\star(u)$. In such a case, s is the session identifier of the agent responsible of this action, t is the global time at which this input has been done, b is the agent responsible of the corresponding output, t_b the time at which this output has been done ($t_b \leq t$), and R the recipe that has been used to forge this output;
- $(a, \alpha, s, t, \emptyset)$ otherwise.

Thanks to these annotations, we are able to formally define dependencies between actions. This will be useful when reordering actions.

Definition 11. *Given an annotated execution $\mathcal{K}_0 \xrightarrow{L_1}_{\mathcal{T}} \cdots \xrightarrow{L_n}_{\mathcal{T}} \mathcal{K}_n$ with $L_i = (a_i, \alpha_i, s_i, t_i, r_i)$, $L_j = (a_j, \alpha_j, s_j, t_j, r_j)$, we say that L_j is* dependent *of L_i, denoted $L_j \hookrightarrow L_i$, if $i < j$, and:*

- *either $s_i = s_j$ (and thus $a_i = a_j$), and in that case L_j is* sequentially-dependent *of L_i, denoted $L_j \hookrightarrow_s L_i$;*
- *or $\alpha_i = \mathsf{out}(v)$, $\alpha_j = \mathsf{in}^\star(u)$, and $r_i \in vars(R_j)$ with $r_j = (b_j, t_{b_j}, R_j)$. In that case L_j is* data-dependent *of L_i, denoted $L_j \hookrightarrow_d L_i$.*

We note $\hookrightarrow^*$ the transitive closure of $\hookrightarrow$ and $L_j \not\hookrightarrow L_i$ when L_j is not dependent of L_i, i.e. $L_j \not\hookrightarrow^* L_i$.

Note that if two actions are dependent, i.e. $(a_j, \alpha_j, s_j, t_j, r_j) \hookrightarrow^* (a_i, \alpha_i, s_i, t_i, r_i)$, then either they have been executed by the same agent or a message must have travelled from the location of agent a_i to the location of agent a_j. This is formally stated in the following lemma.

Lemma 5. *Let $\mathcal{T}$ be a topology, $\mathcal{K}_0 \xrightarrow{L_1.....L_n}_{\mathcal{T}} \mathcal{K}_1$ be an execution, and $i, j \in \{1, \dots, n\}$ be such that $L_j \hookrightarrow^* L_i$. We have that $t_j \geq t_i + Dist_{\mathcal{T}}(a_i, a_j)$ where $L_i = (a_i, \alpha_i, s_i, t_i, r_i)$ and $L_j = (a_j, \alpha_j, s_j, t_j, r_j)$.*

5.2.1 Untimed semantics. To have more flexibility when reordering actions, we define an untimed semantics. Given a configuration $\mathcal{K} = (\mathcal{P}; \Phi; t)$, we note $\mathsf{untimed}(\mathcal{K})$ the configuration associated to $\mathcal{K}$, i.e. $\mathsf{untimed}(\mathcal{K}) = (\mathcal{P}'; \Phi')$ with:

- $\mathcal{P}' = \{ \lfloor P \rfloor_a \mid \lfloor P \rfloor_a^t \in \mathcal{P} \text{ for some } t\}$;
- $\Phi' = \{w \xrightarrow{a} u \mid (w \xrightarrow{a,t} u) \in \Phi \text{ for some } t\}$.

The *untimed semantics* is then defined as follows: $\mathcal{K} \overset{a,\alpha,s,r}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}'$ if there exist $\mathcal{K}_0$ and $\mathcal{K}_0'$ such that $\mathcal{K}_0 \xrightarrow{a,\alpha,s,t,r'}_{\mathcal{T}} \mathcal{K}_0'$ (for some rule other than TIM) with $\mathcal{K} = \mathsf{untimed}(\mathcal{K}_0)$, $\mathcal{K}' = \mathsf{untimed}(\mathcal{K}_0')$, and r is equal to r' up to the time annotation t_b in case of an input.

We may note that Definition 11 can be immediately adapted for untimed actions. As mentioned above, this untimed semantics provides more flexibility to reorder actions in a trace. Indeed, if two actions are independent, i.e. do not belong to the same process nor are an output and an input causally dependent, then we can swap the actions in the trace.

Lemma 6. *Let $\mathcal{T}$ be a topology, and $\mathcal{K}_0 \overset{L_1}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K} \overset{L_2}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_2$ be an execution such that $L_2 \not\hookrightarrow L_1$. We have that $\mathcal{K}_0 \overset{L_2}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}' \overset{L_1}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_2$ for some configuration $\mathcal{K}'$.*

5.2.2 Towards the proof of Theorem 2. This proof proceeds in five steps:

(1) we prove that an attack against t_0-proximity for (V, P) in topology $\mathcal{T}$ remains an attack considering the configuration in which the reset instructions and the guards have been removed;
(2) we consider the trace witnessing the attack w.r.t. the untimed semantics and we reorder the actions: only agents close to v_0 will act between a reset and the following guarded input;
(3) we move the agents close to v_0 at v_0's location and the agents far from v_0 to p_0's location;
(4) we transform this untimed trace into a trace w.r.t. the timed semantics;
(5) we reduce the size of the initial frame by applying Lemma 4.

Among these five steps, we may note that the first, the third and the fifth ones are almost immediate. Indeed, in the first step, we simply remove reset actions and replace guarded inputs by standard inputs. These transformations enable more behaviours. In the third step, the main point is that the topology is no longer relevant in the untimed semantics. Finally, the fifth step is the same as the last step of the proof of Theorem 1 presented before.

Let us focus on the steps (2) and (4). The step (2) is a consequence of the conjunction of Proposition 2 and Lemma 5. Even if the full proof is presented in Appendix B, the main idea is to move all the actions that do not depend on the reset or the guarded input outside the fast phase (thanks to Lemma 6). Then, applying Lemma 5 on the remaining actions in the fast phase, we deduce that they must have been executed by agents close to the verifier v_0.

Proposition 2. *Let $\mathcal{T}$ be a topology, and $\mathcal{K}_0 \overset{\mathsf{tr}_1 \dots \mathsf{tr}_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$ be an execution with $n \geq 2$. We have that there exists a bijection $\varphi : \{1, \dots, n\} \to \{1, \dots, n\}$ such that:*

- $\mathcal{K}_0 \overset{\mathsf{tr}'_1 \dots \mathsf{tr}'_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$ *with* $\mathsf{tr}_i = \mathsf{tr}'_{\varphi(i)}$ *for all* $i \in \{1, \dots, n\}$*; and*
- *for all j such that $\varphi(1) < j < \varphi(n)$, we have that* $\mathsf{tr}'_{\varphi(n)} \hookrightarrow^* \mathsf{tr}'_j \hookrightarrow^* \mathsf{tr}'_{\varphi(1)}$.

The step (4) consists in transforming a trace in the untimed semantics into a trace in the original one. To do so, we can first remark that, after having cleaned the trace, all the agents acting between a reset and the following guarded input are located at the same place. Thanks to this observation, it then become easy to lift the trace into the timed semantics. This allows us to prove Theorem 2 whose full proof is presented in Appendix B.

5.3 About restricted agents

The reduction results presented in the two previous sections assume that an agent is able to act as a verifier and as a prover at the same time. This assumption is evident when looking at Lemma 4: the projection function ρ does not make any difference between agent identities that execute prover roles or verifier roles.

Depending on the application, it can happen that some scenarios are not possible. For example, some protocols, like EMV-payment protocols, have been designed assuming that identities are correctly distributed by an authority, i.e. an agent cannot simultaneously be a prover and a verifier. Hopefully, our reduction results can easily be adapted to handle this restriction. We obtain similar reduced topologies in which almost all the agents must be duplicated in order to get one honest (resp. dishonest) prover representative and one honest (resp. dishonest) verifier representative at each location. These two topologies are presented in Figure 6.

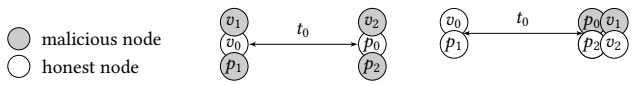


Fig. 6. Topologies $\mathcal{T}^{t_0}_{\mathsf{MF}_{new}}$, and $\mathcal{T}^{t_0}_{\mathsf{DH}_{new}}$

All the results presented so far in this chapter apply modulo small changes. As expected, Definition 3 has to be adapted in order to carefully fill the initial configuration depending on each identity (either a prover of a verifier). Finally, amongst all the technical lemmas, only Lemma 4 must be slightly modified in order to respect the status (prover/verifier) of the agents when applying the projection function. The proof can be immediately adapted keeping in mind that a prover (resp. verifier) identity must only be projected on another prover (resp. verifier) identity.

Note that it is important to decide whether an agent is able to act as a prover and a verifier at the same time or not. Indeed, depending on this choice of modelling, protocols may be proved secure or not. An example of such a protocol is the SPADE protocol [16] which can be proved Mafia fraud resistant when distinguishing both status, but suffers from a Mafia fraud if an agent can act as a verifier and a prover at the same time.

6 REDUCING ORACLES

When considering Terrorist frauds, another reduction result is needed to get rid of the infinite number of semi-dishonest provers that should be considered. This result only holds for a restricted class of protocols, named *well-formed distance-bounding protocols*, that match few additional assumptions. These assumptions rely on what we call a *unifier* for a set $\mathcal{U}$ of equations and a *quasi-free* symbol of function. These two notions are introduced in a preliminaries section before developing our reduction result in the following one. The detailed proofs of lemmas, propositions, and theorems introduced in this section are available in Appendix C.

6.1 Preliminaries

As usual, given a set $\mathcal{U}$ of equations between terms, σ is a *unifier* for $\mathcal{U}$ if $u_1\sigma\downarrow =_{\mathsf{E}} u_2\sigma\downarrow$ and both $u_1\sigma\downarrow$ and $u_2\sigma\downarrow$ are constructor terms for any $u_1 = u_2 \in \mathcal{U}$. We denote by $\mathsf{csu}(\mathcal{U})$ a minimal set of unifiers for $\mathcal{U}$ which is also *complete*, i.e. such that for any σ unifier of $\mathcal{U}$, there exists $\theta \in \mathsf{csu}(\mathcal{U})$ such that $\sigma =_{\mathsf{E}} \tau \circ \theta$ for some τ.

From now on, we assume that for all set of equations $\mathcal{U}$, if $\mathsf{csu}(\mathcal{U})$ exists then it is reduced to a singleton. We note $\theta_{\mathcal{U}}$ this element. Even if this assumption seems very restrictive, it is satisfied as soon as the rewriting system contains only one rule per destructor symbol which is often verified.

Example 17. *Let us consider:*

$$\mathcal{U} = \Big\{ x_1 = \mathsf{proj}_1(\mathsf{adec}(x^1_{\mathsf{rep}}, \mathsf{sk}(v))), x_2 = \mathsf{proj}_2(\mathsf{adec}(x^1_{\mathsf{rep}}, \mathsf{sk}(v))), x^1_{\mathsf{ok}} = \mathsf{eq}(x_1, \mathsf{check}(x_2, \mathsf{spk}(p))),$$
$$x_3 = \mathsf{prf}(\langle x_1, n_V\rangle), x_4 = x_1 \oplus m_V \oplus x_3, x^2_{\mathsf{ok}} = \mathsf{eq}(x^2_{\mathsf{rep}}, \mathsf{answer}(c_V, x_3, x_4)),$$
$$x^3_{\mathsf{ok}} = \mathsf{eq}(x^3_{\mathsf{rep}}, \mathsf{prf}(\langle x_2, n_V, m_V, c_V, \mathsf{answer}(c_V, x_3, x_4)\rangle)) \Big\}.$$

We have that $\mathsf{csu}(\mathcal{U}) = \{\theta_{\mathcal{U}}\}$*, where* $\theta_{\mathcal{U}}$ *is the following substitution:*

$$\Big\{ x^1_{\mathsf{rep}} \mapsto \mathsf{aenc}(\langle x_1, \mathsf{sign}(x_1, \mathsf{ssk}(p))\rangle, \mathsf{pk}(v)), x_2 \mapsto \mathsf{sign}(x_1, \mathsf{ssk}(P)), x^1_{\mathsf{ok}} \mapsto \mathsf{ok},$$
$$x_3 \mapsto \mathsf{prf}(\langle x_1, n_V\rangle), x_4 \mapsto x_1 \oplus m_V \oplus \mathsf{prf}(\langle x_1, n_V\rangle),$$
$$x^2_{\mathsf{rep}} \mapsto \mathsf{answer}(c_V, \mathsf{prf}(\langle x_1, n_V\rangle)), x_1 \oplus m_V \oplus \mathsf{prf}(\langle x_1, n_V\rangle)), x^2_{\mathsf{ok}} \mapsto \mathsf{ok},$$
$$x^3_{\mathsf{rep}} \mapsto \mathsf{prf}(\langle x_1, n_V, m_V, c_V, \mathsf{answer}(c_V, \mathsf{prf}(\langle x_1, n_V\rangle), x_1 \oplus m_V \oplus \mathsf{prf}(\langle x_1, n_V\rangle)))), x^3_{\mathsf{ok}} \mapsto \mathsf{ok} \Big\}.$$

A symbol of function $\mathsf{f} \in \Sigma^+_c$ is *quasi-free* if it occurs neither in the equations used to generate the relation $=_{\mathsf{E}}$ nor in the right-hand side of a rewriting rule. This implies that such a symbol cannot be introduced during a normalisation.

Since such symbols does not interact with the rewriting system and the equational theory, we will be able to reduce the number semi-dishonest provers as soon as the challenge only appears under quasi-free symbols of function in the answer. The following lemma states that if a term u reduces to a term $\mathsf{f}(u_1, \ldots, u_k)$ with f a quasi-free symbol of function then it must contain $\mathsf{f}(u_1, \ldots, u_k)$ as a subterm (up to some normalisation steps).

Lemma 7. *Let* t_0 *be a term such that* $t_0\downarrow =_{\mathsf{E}} \mathsf{f}(u_1, \ldots, u_k)$ *with* f *a quasi-free function symbol. We have that there exist* $u'_1, \ldots, u'_k$ *such that* $\mathsf{f}(u'_1, \ldots, u'_k) \in st(t_0)$ *and* $u'_i\downarrow =_{\mathsf{E}} u_i$ *for any* $i \in \{1, \ldots, k\}$.

Example 18. *In order to illustrate Lemma 7, let us consider* f *a quasi-free symbol of function and the term* $u = \mathsf{adec}(\mathsf{aenc}(\mathsf{f}(\mathsf{proj}_1(\langle x, y\rangle)), \mathsf{pk}(id)), \mathsf{sk}(id))$*. We have that* u *reduces to* $\mathsf{f}(x)$*, i.e.* $u\downarrow = \mathsf{f}(x)$*, which already occurs in* $st(u)$ *up to a normalisation step. Indeed,* $\mathsf{f}(\mathsf{proj}_1(\langle x, y\rangle)) \in st(u)$ *and* $\mathsf{f}(\mathsf{proj}_1(\langle x, y\rangle)) \to \mathsf{f}(x)$.

Given an execution, we know that each input message can be forged by some recipes. Assuming that u is the answer to some challenge c, and that a recipe R has been used to compute u. The following lemma allows us to decompose this recipe R in order to obtain sub-recipes that deduce the maximal subterms of u which do not contain c.

Lemma 8. *Let* Φ *be a frame and* $c \in \mathcal{N}$ *such that* $c \notin st(img(\Phi))$*, and* $\Phi^+ = \Phi \cup \{w_c \xrightarrow{v_0, t_0} c\}$*. Let* R *be a recipe such that* $R\Phi^+\downarrow =_{\mathsf{E}} u$*. Let* C *be a context of minimal size made of quasi-free public function symbols such that* $u = C[c, u_1, \ldots, u_p]$ *for some* $u_1, \ldots, u_p$ *and* c *does not occur in* $st(\{u_1, \ldots, u_p\})$*. For any* $i \in \{1, \ldots, p\}$*, we have that there exists* R_i *such that* $R_i\Phi^+\downarrow =_{\mathsf{E}} u_i$.

Example 19. *In order to illustrate Lemma 8, we consider the frame* $\Phi = \{w_0 \xrightarrow{a,t_a} u_1, w_1 \xrightarrow{a,t_a} u_2\}$, *the term* $u = \mathsf{f}(c, \langle u_1, u_2\rangle)$, *and the recipe* $R = \mathsf{f}(w_c, \langle w_1, w_2\rangle)$. *We have that* $R(\Phi \uplus \{w_c \xrightarrow{v_0,t_v} c\})\downarrow = u$. *Thanks to Lemma 8, we have that R contains a subterm which deduces* $\langle u_1, u_2\rangle$. *Indeed, such a recipe is for example* $R' = \langle w_1, w_2\rangle$. *This reasoning may be applied to more complex terms whenever all the symbols on top of c in u are quasi-free.*

6.2 Most general semi-dishonest prover

To get rid of all the semi-dishonest provers when verifying Terrorist fraud resistance, we have to restrict the class of protocols we consider.

Definition 12. *A* well-formed distance-bounding *protocol is a protocol* (V, P) *such that:*

(i) *The two roles have the following form:*
- $\mathsf{V}(z_V^0, z_V^1) = \mathtt{block}_V.\mathtt{new}\ c_V.\mathtt{reset}.\mathtt{out}(c_V).\mathtt{in}^{<2\cdot t_0}(x).\mathtt{block}'_V$; *and*
- $\mathsf{P}(z_P^0, z_P^1) = \mathtt{block}_P.\mathtt{in}(y_c).\mathtt{out}(u).\mathtt{block}'_P$

where $\mathtt{block}_X$ *and* $\mathtt{block}'_X$ *with* $X \in \{V, P\}$ *is a sequence of actions without reset and guarded input instructions. Moreover, we assume that* $\mathtt{out}(c)$ *(resp.* $\mathtt{in}(y_c)$*) corresponds to the* $i_0{}^{th}$ *communication action of* $\mathsf{V}(z_V^0, z_V^1)$ *(resp.* $\mathsf{P}(z_P^0, z_P^1)$*) for some* i_0.

(ii) $(\lfloor\widetilde{\mathsf{V}}\rfloor_{v_0}^0 \uplus \lfloor\widetilde{\mathsf{P}}\rfloor_{p_0}^0; \emptyset; 0) \xrightarrow{\mathrm{tr}}_{\mathcal{T}^0_{\mathrm{basic}}} (\lfloor 0\rfloor_{v_0}^0 \uplus \lfloor 0\rfloor_{p_0}^0; \Phi; 0)$ *with*

$$\mathrm{tr} = \begin{cases} (a_1, \mathtt{out}(m_1)).(b_1, \mathtt{in}(m_1)) \dots (a_{i_0-1}, \mathtt{out}(m_{i_0-1})).(b_{i_0-1}, \mathtt{in}(m_{i_0-1})) \\ (v_0, \mathtt{out}(m_{i_0})).(p_0, \mathtt{in}(m_{i_0})).(p_0, \mathtt{out}(m_{i_0+1})).(v_0, \mathtt{in}^{<2\cdot t_0}(m_{i_0+1})) \\ (a_{i_0+2}), \mathtt{out}(m_{i_0+2}).(b_{i_0+2}, \mathtt{in}(m_{i_0+2})) \dots (a_n, \mathtt{out}(m_n)).(b_n, \mathtt{in}(m_n)) \end{cases}$$

up to τ *actions, and* $\{a_i, b_i\} = \{v_0, p_0\}$ *for any* $i \in \{1, \dots, n\} \setminus \{i_0, i_0+1\}$. *The processes* $\widetilde{\mathsf{V}}$ *(resp.* $\widetilde{\mathsf{P}}$*) corresponds to* $\mathsf{V}(v_0, p_0)$ *(resp.* $\mathsf{P}(p_0, v_0)$*) in which* new *commands are removed, and* $\mathcal{T}^0_{\mathrm{basic}}$ *is the topology only composed of agents* v_0, p_0 *honest and located at the same place.*

(iii) *Let* $\mathcal{U} = \{x = u \mid$ "let $x = u$ in " *occurs in* $\mathsf{V}(v_0, p_0)\}$ *and* $\{\theta_{\mathcal{U}}\}$ *its complete set of unifiers. We assume that* $(x_1, \dots, x_k)\theta_{\mathcal{U}}\downarrow\sigma =_{\mathsf{E}} m_{i_1}, \dots, m_{i_k}$ *where* $x_1, \dots, x_k$ *are the variables occurring in input in the role* $\mathsf{V}_0(v_0, p_0)$, $i_1, \dots, i_k$ *are the indices among* $1, \dots, n$ *corresponding to input performed by* v_0, *and* σ *is a bijective renaming from variables to* $bn(\mathsf{P}(p_0, v_0))$.

(iv) *We assume the existence of a context C made of quasi-free public function symbols such that* $u = C[y_c, u_1, \dots, u_l]$, *and* y_c *does not occur in* $u_1, \dots, u_l$.

The first condition put some restrictions on the syntactic definition of the protocol. Indeed, we assume that a well-formed distance-bounding protocol is a two-party protocol with rather simple roles: the fast phase of the verifier role consists in a single challenge/response exchange. The second condition assumes that if no attacker interferes, these two roles together will execute until the end. For sake of simplicity we consider instances of the roles in which bound names are not freshened. This is possible because we consider only one instance of each role (and they do not share bound names). The third condition gives us a constraint about the messages that are exchanged. Even if this condition may seem restrictive, it is always verified. Otherwise, it would mean that some terms that are exchanged are never checked, i.e. are useless. Finally, the fourth condition is used to ensure that there exists at least one semi-dishonest prover. This semi-dishonest prover will output the terms $u_1, \dots, u_l$ in advance to let his accomplice compute (as indicated by C) the answer to the challenge from $u_1, \dots, u_l$ and the challenge c_V he will receive from the verifier. Actually, the best strategy for the semi-dishonest prover will consist in considering $C_{\mathsf{V},\mathsf{P}}$ the smallest context (in terms of number of symbols) satisfying the requirements.

Example 20. *We can demonstrate that the SPADE protocol described in Example 4 is a well-formed distance-bounding protocol. The item (i) is straightforward with the descriptions of* $\mathsf{V}(z_V^0, z_V^1)$ *and*

$\mathsf{P}(z_P^0, z_P^1)$, *which also give* $i_0 = 3$. *For item (ii), we can exhibit the corresponding trace (up to* τ *actions and name freshening):*

$$\mathsf{tr} = (p_0, \mathsf{out}(m_1)) \,.\, (v_0, \mathsf{in}(m_1)) \,.\, (v_0, \mathsf{out}(m_2)) \,.\, (p_0, \mathsf{in}(m_2)) \,.\, (v_0, \mathsf{out}(c_V)) \,.\, (p_0, \mathsf{in}(c_V))$$
$$(p_0, \mathsf{out}(m_4)) \,.\, (v_0, \mathsf{in}^{<2\cdot t_0}(m_4)) \,.\, (p_0, \mathsf{out}(m_5)) \,.\, (v_0, \mathsf{in}(m_5))$$

with the following abbreviations:

$$m_1 = \mathsf{aenc}(\langle n_P, \mathsf{sign}(n_P, \mathsf{ssk}(p_0))\rangle, \mathsf{pk}(p)),$$
$$m_2 = \langle m_V, n_V\rangle,$$
$$m_4 = \mathsf{answer}(c_V, \mathsf{prf}(\langle n_P, n_V\rangle), n_P \oplus m_V \oplus \mathsf{prf}(\langle n_P, n_V\rangle)), \text{ and}$$
$$m_5 = \mathsf{prf}(\langle n_P, n_V, m_V, c_V, \mathsf{answer}(c_V, \mathsf{prf}(\langle n_P, n_V\rangle), n_P \oplus m_V \oplus \mathsf{prf}(\langle n_P, n_V\rangle))\rangle).$$

The set $\mathcal{U}$ *described in item (iii) is exactly the same as the one presented in Example 17, which also provides the corresponding* $\theta_{\mathcal{U}}$. *The substitution* σ *is reduced to* $\{x_1 \mapsto n_P\}$. *Finally, for item (iv), we have that* $C = \mathsf{answer}(\_, \_, \_)$ *satisfies the requirement since* $u = \mathsf{answer}(y_c, x_3, x_4)$.

Up to now, according to Definition 8 (and Corollary 2) we had to consider all the possible semi-dishonest provers. Restricting our analysis to well-formed distance-bounding protocols, we are now able to define the *most general semi-dishonest prover* and prove that it is sufficient to focus on it during our security analysis.

Definition 13. *Let* (V, P) *be a well-formed distance-bounding protocol. Following the notations of Definition 12, we note* P^* *the following process:*

$$\big(\mathsf{block}_P.\mathsf{out}(u_1) \ldots \mathsf{out}(u_l).\mathsf{in}(y_c).\mathsf{out}(u).\mathsf{block}'_P\big)\{z_P^0 \mapsto p_0, z_P^1 \mapsto v_0\}$$

where $u_1, \ldots, u_l$ *are the terms such that* $u = C_{\mathsf{V},\mathsf{P}}[y_c, u_1, \ldots, u_l]$.

Example 21. *If we consider the semi-dishonest prover presented in Example 12, one can see that it corresponds to the process* P^* *defined above.*

The behaviour of the most general semi-dishonest prover consists in providing to his accomplice all the material needed to pass the fast phase just before it starts. For this purpose, he computes and sends the maximal sub-terms of u that do not contain the challenge y_c. His accomplice is then able to re-construct the response to the challenge using the context $C_{\mathsf{V},\mathsf{P}}$.

As assumed in the speech up to now, the process P^* is a semi-dishonest prover, i.e. put together with $\lfloor \mathsf{V}(v_0, p_0) \rfloor_{v_0}^0$ the two processes can be fully executed. This is immediate from the definition of a well-formed distance-bounding protocol and the close correspondence between P^* and $\mathsf{P}(p_0, v_0)$. Lemma 9 formally states this property.

Lemma 9. *Let* (V, P) *be a well-formed distance-bounding protocol. The process* P^* *(as defined in Definition 13) is a semi-dishonest prover, called* the most general semi-dishonest prover. *Moreover, we can assume that the trace* tr^* *witnessing this fact is such that :*

$$\mathsf{tr}^* = \begin{cases} (a_1, \mathsf{out}(m_1)).(b_1, \mathsf{in}(m_1)) \ldots (a_{i_0-1}, \mathsf{out}(m_{i_0-1})).(b_{i_0-1}, \mathsf{in}(m_{i_0-1})). \\ (p_0, \mathsf{out}(m^1_{i_0+1})). \ldots . (p_0, \mathsf{out}(m^l_{i_0+1})). \\ (v_0, \mathsf{out}(m_{i_0})).(v_0, \mathsf{in}^{<2\cdot t_0}(m_{i_0+1})) \\ (p_0, \mathsf{in}(m_{i_0})).(p_0, \mathsf{out}(m_{i_0+1})). \\ (a_{i_0+2}, \mathsf{out}(m_{i_0+2})).(b_{i_0+2}, \mathsf{in}(m_{i_0+2})) \ldots (a_n, \mathsf{out}(m_n)).(b_n, \mathsf{in}(m_n)) \end{cases}$$

up to τ *actions where:*

- $\{a_i, b_i\} = \{v_0, p_0\}$ *for any* $i \in \{1, \ldots, n\} \setminus \{i_0, i_0 + 1\}$;
- $m_{i_0+1} =_{\mathsf{E}} C_{\mathsf{V},\mathsf{P}}[m_{i_0}, m^1_{i_0+1}, \ldots, m^l_{i_0+1}]$

- $(x_1, \ldots, x_k)\theta_{\mathcal{U}}\downarrow\sigma =_{\mathsf{E}} m_{i_1}, \ldots, m_{i_k}$ *where* $x_1, \ldots, x_k$ *are the variables occurring in input in the role* $\mathsf{V}(v_0, p_0)$ *and* $i_1, \ldots, i_k$ *are the indices among* $1, \ldots, n$ *corresponding to input performed by* v_0, $\mathcal{U} = \{x = u \mid \text{"}\mathtt{let}\ x = u\ \mathtt{in}\text{"}$ *occurs in* $\mathsf{V}(v_0, p_0)\}$, *and* σ *is a bijective renaming from variables to* $bn(\mathsf{P}^*)$. *This equality holds up to a bijective renaming of names freshly generated along the execution.*

We note Φ^* the initial frame associated to the most general semi-dishonest prover, i.e. the frame resulting from the execution of tr^* in which all the annotated times are set to 0.

A noteworthy point is the strong correspondence that exists between Φ^* and any frame Φ_{sd} associated to an arbitrary semi-dishonest prover P_{sd}. We can prove that any semi-dishonest prover discloses, at least, as much information as P^*, up to some substitution σ.

Proposition 3. *Let* (V, P) *be a well-formed distance-bounding protocol, and* P^* *be its most general semi-dishonest prover with* Φ^* *its associated frame. Let* $exec^*$ *be an execution witnessing the fact that* P^* *together with* Φ^* *is a semi-dishonest prover (as given in Lemma 9). We have that* $exec^*$ *is as follows:*

$$exec^* : (\{\lfloor \mathsf{V}(v_0, p_0)\rfloor^0_{v_0}, \lfloor \mathsf{P}^*\rfloor^0_{p_0}\}; \emptyset; 0) \xrightarrow{\mathsf{tr}^*}_{\mathcal{T}^{t_0}_{\text{simple}}} (\{\lfloor 0\rfloor^{t^*_v}_{v_0}, \lfloor 0\rfloor^{t^*_p}_{p_0}\}; \Phi^*; t^*).$$

Let P_{sd} *be a semi-dishonest prover for* (V, P) *together with its associated frame* Φ_{sd}, *and exec be the execution witnessing this fact, i.e.*

$$exec : (\{\lfloor \mathsf{V}(v_0, p_0)\rfloor^0_{v_0}, \lfloor \mathsf{P}_{\mathsf{sd}}\rfloor^0_{p_0}\}; \emptyset; 0) \xrightarrow{\mathsf{tr}}_{\mathcal{T}^{t_0}_{\text{simple}}} (\{\lfloor 0\rfloor^{t_v}_{v_0}, \lfloor 0\rfloor^{t_p}_{p_0}\}; \Phi_{\mathsf{sd}}; t).$$

We have that there exists a substitution $\sigma : \mathcal{N} \to \mathcal{T}(\Sigma_c^+, \mathcal{N} \cup \mathcal{A})$ *from names freshly generated by* P^* *to constructor terms such that:*

(i) if $(v_0, \mathtt{in}(u)) \in \mathsf{tr}^*$ *(resp.* $(v_0, \mathtt{in}^{<t}(u)) \in \mathsf{tr}^*$*), then* $(v_0, \mathtt{in}(u\sigma)) \in \mathsf{tr}$ *(resp.* $(v_0, \mathtt{in}^{<t}(u\sigma)) \in \mathsf{tr}$*);*
(ii) if $(a, \mathtt{out}(u)) \in \mathsf{tr}^*$ *for some* $a \in \{v_0, p_0\}$, *then* $R\Phi_{\mathsf{sd}}\downarrow =_{\mathsf{E}} u\sigma$ *for some recipe* R.

6.3 Main result

We are now able to state and prove the main reduction result which allows us to get rid of the universal quantification over the semi-dishonest prover by focusing on the most general one.

Theorem 3. *Let* $\mathcal{I}_0$ *be a template,* (V, P) *be a well-formed distance-bounding protocol. Let* Φ^* *be the frame associated to the most general semi-dishonest prover of* (V, P). *We have that* (V, P) *is Terrorist fraud resistant w.r.t.* t_0*-proximity if, and only if, there exist a topology* $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0) \in \mathcal{C}^{t_0}_{\mathsf{MF}}$ *and a valid initial configuration* $\mathcal{K}_0$ *for* (V, P) *w.r.t.* $\mathcal{T}$ *and* $\Phi^* \cup \Phi^{\mathcal{T}}_{\mathcal{I}_0}$ *such that* $\mathcal{K}_0 \to^*_{\mathcal{T}} (\lfloor \mathsf{end}(v_0, p_0)\rfloor^{t'}_{v_0} \uplus \mathcal{P}; \Phi; t)$.

The following corollary is immediate putting together Theorem 3 and Theorem 1. It shows that when checking for Terrorist fraud resistance, it is sufficient to focus on a particular semi-dishonest prover and a particular topology.

Corollary 3. *Let* $\mathcal{I}_0$ *be a template,* (V, P) *be a well-formed distance-bounding protocol. Let* Φ^* *be the frame associated to the most general semi-dishonest prover of* (V, P). *We have that* (V, P) *is Terrorist fraud resistant w.r.t.* t_0*-proximity if, and only if, there exists a valid initial configuration* $\mathcal{K}_0$ *for* (V, P) *w.r.t.* $\mathcal{T}^{t_0}_{\mathsf{MF}}$ *and* $\Phi^* \cup \Phi^{\mathcal{T}^{t_0}_{\mathsf{MF}}}_{\mathcal{I}_0}$ *such that* $\mathcal{K}_0 \to^*_{\mathcal{T}^{t_0}_{\mathsf{MF}}} (\lfloor \mathsf{end}(v_0, p_0)\rfloor^{t'}_{v_0} \uplus \mathcal{P}; \Phi; t)$.

In	$(i : \mathtt{in}(x).P \uplus \mathcal{P}; \phi; i) \xrightarrow{\mathtt{in}(u)} (i : P\{x \mapsto u\} \uplus \mathcal{P}; \phi; i)$	when $R\phi\downarrow = u$ for some recipe R and u is a message
Out	$(i : \mathtt{out}(u).P \uplus \mathcal{P}; \phi; i) \xrightarrow{\mathtt{out}(u)} (i : P \uplus \mathcal{P}; \phi \uplus \{w \mapsto u\}; i)$	with $w \in \mathcal{W}$ fresh.
Let	$(i : \mathtt{let}\ x = v\ \mathtt{in}\ P \uplus \mathcal{P}; \phi; i) \xRightarrow{\tau} (i : P\{x \mapsto v\downarrow\} \uplus \mathcal{P}; \phi; i)$	when $v\downarrow$ is a message.
New	$(i : \mathtt{new}\ n.P \uplus \mathcal{P}; \phi; i) \xRightarrow{\tau} (i : P\{n \mapsto n'\} \uplus \mathcal{P}; \phi; i)$	with $n' \in \mathcal{N}$ fresh.
Rep	$(i : !P \uplus \mathcal{P}; \phi; i) \xRightarrow{\tau} (i : P \uplus (i : !P) \uplus \mathcal{P}; \phi; i)$	
Move	$(\mathcal{P}; \phi; i) \xrightarrow{\mathtt{phase}\ i'} (\mathcal{P}; \phi; i')$	with $i' > i$.
Phase	$(i : i' : P \uplus \mathcal{P}; \phi; i) \xRightarrow{\tau} (i' : P \uplus \mathcal{P}; \phi; i)$	

Fig. 7. Semantics for the ProVerif calculus

7 ENCODING IN PROVERIF

In Section 5, we proved that we can focus on two rather simple topologies when analysing a protocol. Unfortunately, existing tools are not suitable to model them. In the following, we present a framework based on the existing verification tool ProVerif, to get rid of these reduced topologies.

7.1 ProVerif in a nutshell

ProVerif [10] is an automated verification tool developed to analyse standard protocols, i.e. without time and location considerations. It has been successfully applied to secure messaging protocols [43], e-voting schemes [21, 39], or avionic protocols [12]. ProVerif allows to model a wide class of cryptographic primitives like symmetric/asymmetric encryption, signatures, hash functions... and describes protocols through a process algebra close to the one presented in Section 3.2.1. Our methodology applies considering the following subset of the ProVerif calculus:

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

Almost all the commands are similar to those defined in our timed semantics. The main difference is the notion of phase, denoted $i : P$. Informally, phases model synchronisation points in the execution of a protocol. An execution always starts at phase 0. It can arbitrarily increase during the execution but during phase i only processes that are actually at this stage can be executed. Another difference is the presence of the replication $!P$ command which was deliberately omitted in our timed model since an arbitrary number of sessions may be added in a valid initial configuration. This syntactic sugar will be useful to define the configuration under study in ProVerif.

This new semantics is formally described by a relation $\Rightarrow$ over configurations. A configuration is a tuple $(\mathcal{P}; \phi; i)$ where $\mathcal{P}$ is a multiset of processes, ϕ is a usual frame (without time annotations) and $i \in \mathbb{N}$ is the current phase. All the rules are given in Figure 7.

7.2 Our transformation

Given a protocol (V, P), we propose a transformation that encodes the reduced topologies $\mathcal{T}_{\mathsf{MF}}^{t_0}$ and $\mathcal{T}_{\mathsf{DH}}^{t_0}$ into the ProVerif tool. However it requires that the role $\mathsf{V}_{\mathsf{end}}(z_0, z_1)$ containing the special command $\mathsf{end}(z_0, z_1)$ is made of a unique challenge/response exchange, i.e. is of the form:

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

where block_i is a sequence of actions (only simple inputs, outputs, let, and new instructions are allowed).

The main idea for the transformation is to use the notion of phases provided by ProVerif to encode the fast phase, i.e. the challenge/response exchange. To do so we consider three phases: phase 0 before the fast phase, phase 1 that starts when executing the `reset` action and stops just after the input of the response, i.e. the $\mathtt{in}^{<t}(x)$ action, and finally phase 2 for the remaining of the protocol. The two locations of the reduced topologies are then modelled as follows: agents close the the verifier are allowed to execute actions during phase 1 while distant agents are not. Given a parametrised role as defined in Section 3.2.1 (with no `reset` and $\mathtt{in}^{<t}(x)$ commands), the two corresponding transformations, denoted $\mathcal{F}^{<}$ and $\mathcal{F}^{\geq}$, are thus defined as follows:

- *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 $\mathcal{F}(\mathcal{T}, (\mathsf{V}, \mathsf{P}), \Phi_0, t_0)$, is the tuple $(\mathcal{P}; \phi; 0)$ where ϕ is such that $img(\phi) = img(\Phi_0)$, and $\mathcal{P}$ is the multiset that contains the following processes:

- $\mathsf{V}'_{\mathsf{end}}(v_0, p_0) = \mathsf{block}_1 \,.\, 1 : \mathtt{out}(u) \,.\, \mathtt{in}(x) \,.\, 2 : \mathsf{block}_2 \,.\, \mathsf{end}(v_0, p_0)$;
- $!R(a_0, .., a_n)$ when $R(z_0, .., z_n) \in \{\mathcal{F}^{<}(\overline{\mathsf{V}}), \mathcal{F}^{<}(\overline{\mathsf{P}})\}$, $a_0, .., a_n \in \mathcal{A}_0$, $\mathsf{Dist}_{\mathcal{T}}(v_0, a_0) < t_0$;
- $!R(a_0, .., a_n)$ when $R(z_0, .., z_n) \in \{\mathcal{F}^{\geq}(\overline{\mathsf{V}}), \mathcal{F}^{\geq}(\overline{\mathsf{P}})\}$, $a_0, .., a_n \in \mathcal{A}_0$, $\mathsf{Dist}_{\mathcal{T}}(v_0, a_0) \geq t_0$;

Relying on Proposition 2 and Lemma 5 we are able to establish the following result that formally justifies the correctness of our transformation. The detailed proof is available in Appendix D.

Proposition 4. *Let $t_0 \in \mathbb{R}_+$, $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0)$ be a topology, and (V, P) be a protocol such that $\mathsf{V}_{\mathsf{end}}(v_0, p_0)$ has the following form:*

$$\mathsf{block}_1 \,.\, \mathtt{reset} \,.\, \mathtt{out}(u) \,.\, \mathtt{in}^{<t}(x) \,.\, \mathsf{block}_2 \,.\, \mathsf{end}(v_0, p_0) \quad \text{with } t \leq 2 \cdot t_0$$

Let $\mathcal{K}_0$ be a valid initial configuration for (V, P) w.r.t. $\mathcal{T}$ and Φ_0. If $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathcal{P}; \Phi; t)$ with $Dist_{\mathcal{T}_0}(v_0, p_0) \geq 2 \cdot t_0$ then we have that:

$$\mathcal{F}(\mathcal{T}_0, (\mathsf{V}, \mathsf{P}), \Phi_0, t_0) \overset{\mathsf{tr}'}{\Longrightarrow} (\{2 : \mathsf{end}(v_0, p_0)\} \uplus \mathcal{P}'; \phi; 2).$$

Moreover, in case there is no $a \in \mathcal{M}_0$ such that $Dist_{\mathcal{T}}(a, v_0) < t_0$, 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 output in phase 0.

8 CASE STUDIES

All the results presented above enabled us to perform a comprehensive case studies analysis considering more than 25 protocols including new EMV (payment) protocols. This analysis has been conducted on a standard laptop and ProVerif always returned in less than few seconds except on two examples for which few minutes are necessary. All the material is available in the supplementary material provided together with this document and in our GitLab repository [1].

8.1 Methodology

All our results allow us to limit the security analysis (w.r.t. Distance Hijacking attack, Mafia fraud, or Terrorist fraud) of a given protocol to a single ProVerif file. Given the hypothesis that the protocol is an executable 2-party protocol (Definition 1), our methodology works as described below.

While it would be necessary to check all the possible topologies and configurations as stated in Definitions 5, 8, or 10, depending on the security property (resp. Mafia fraud, Terrorist fraud or Distance Hijacking attack) analysed, our reduction results (resp. Theorem 1 for Mafia and Terrorist

fraud, or Theorem 2 for Distance Hijacking attack) limit the number of topologies to consider to only one (resp. $\mathcal{T}_{\mathsf{MF}}^{t_0}$ for Mafia and Terrorist fraud, or $\mathcal{T}_{\mathsf{DH}}^{t_0}$ for Distance Hijacking attack).

In the case of Mafia fraud and Distance Hijacking attack, if the role V is made of a unique challenge/response exchange and therefore is of the form stated in Section 7.2, Proposition 4 allows us to encode, with respect to the topology studied, the two roles V and P into ProVerif according to the transformation also described in that section. While this transformation may seem complicated on paper, it is quite straightforward to apply and could be automatised, if needed, for complex protocols. Nevertheless, depending on the number of inputs in the processes, this transformation may lead to some heavy ProVerif files in terms of lines of code.

If ProVerif can *not* reach the end event, Proposition 4 gives us that the distance-bounding protocol considered is secure against Mafia fraud or Distance Hijacking attack. If an attack is discovered, it should be analysed to see whether it is executable in our timed semantics, and thus corresponds to a real attack. This whole methodology is summarized by the following corollaries:

Corollary 4. *Let $\mathcal{I}_0$ be a template, $t_0 \in \mathbb{R}_+$ a threshold, and (V,P) be an $\mathcal{I}_0$-executable 2-party distance-bounding protocol, where V is made of a unique challenge/response exchange. If (V,P) admits a Mafia fraud w.r.t. t_0-proximity, then we have :*

$$\mathcal{F}(\mathcal{T}_{\mathsf{MF}}^{t_0}, (\mathsf{V},\mathsf{P}), \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}}, t_0) \overset{\mathsf{tr}}{\Longrightarrow} (\{2 : \mathsf{end}(v_0, p_0)\} \uplus \mathcal{P}'; \phi; 2).$$

Corollary 5. *Let $\mathcal{I}_0$ be a template, $t_0 \in \mathbb{R}_+$ a threshold, and (V,P) be an $\mathcal{I}_0$-executable 2-party distance-bounding protocol, where V is made of a unique challenge/response exchange. If (V,P) admits a Distance Hijacking attack w.r.t. t_0-proximity, then we have :*

$$\mathcal{F}(\mathcal{T}_{\mathsf{DH}}^{t_0}, (\mathsf{V},\mathsf{P}), \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{DH}}^{t_0}}, t_0) \overset{\mathsf{tr}}{\Longrightarrow} (\{2 : \mathsf{end}(v_0, p_0)\} \uplus \mathcal{P}'; \phi; 2).$$

In the case of Terrorist fraud, Definition 8 suggests to consider all possible semi-dishonest provers and we still need to do so after use of Theorem 1. By adding the hypothesis that our protocol is also a well-formed distance-bounding protocol, we can use Lemma 9 to define the most general semi-dishonest prover, and Theorem 3, to encode this process into a frame of initial knowledge. Then, again by assuming that V is made of a unique challenge/response exchange, we make use of Proposition 4 to encode into ProVerif, with respect to the topology $\mathcal{T}_{\mathsf{MF}}^{t_0}$, the computed initial frame, representing the knowledge learn from the semi-dishonest prover, and the roles V and P following our transformation.

If ProVerif can reach the end event, and if we are able to execute the trace in our timed semantics, then the protocol is Terrorist fraud resistant, otherwise, it is vulnerable to a Terrorist fraud. This whole methodology is illustrated by the following corollary:

Corollary 6. *Let $\mathcal{I}_0$ be a template, $t_0 \in \mathbb{R}_+$ a threshold, and (V,P) be an $\mathcal{I}_0$-executable 2-party well-formed distance-bounding protocol, where V is made of a unique challenge/response exchange. If (V,P) is a Terrorist fraud resistant w.r.t. t_0-proximity, then we have :*

$$\mathcal{F}(\mathcal{T}_{\mathsf{MF}}^{t_0}, (\mathsf{V},\mathsf{P}), \Phi^* \cup \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}}, t_0) \overset{\mathsf{tr}}{\Longrightarrow} (\{2 : \mathsf{end}(v_0, p_0)\} \uplus \mathcal{P}'; \phi; 2).$$

It is worth noting that ProVerif never returned false attacks across our case studies. The over-approximations related to Theorem 2, Proposition 4 and the core procedure of ProVerif appear tight enough to analyse distance-bounding protocols.

8.2 Limitations

Unfortunately, this methodology has some limitations due to either our theoretical development or the existing tools and in particular, the ProVerif tool we decided to use. To overcome them, some abstractions have been done when analysing protocols.

First, as mentioned in Example 4, our theoretical model does not allow an accurate modelling of bit-level operations. The first abstraction is to collapse the fast phase, often made of several round-trip bits, into a simple challenge/response exchange using a fresh nonce as a challenge. In addition, when the response is too bit-level dependent then it is abstracted using an uninterpreted symbol of function with relevant arguments. Following this abstraction, the protocols Tree-based [6], Poulidor [55], and Uniform [46] have the same modelling as the Hancke and Kuhn protocol.

Then, even if our theoretical model is generic enough to model associative and commutative operators, ProVerif, like almost all the existing automated verification tools, does not support them. Since distance-bounding protocols often rely on the exclusive-or operator, we decided to model a weaker operator through the following rules:

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

Similarly, few protocols rely on an algebraic properties of the exponentiation in finite cyclic fields. Denoting by g a generator, we decided to model such an operator by the following equation:

$$\mathsf{exp}(\mathsf{exp}(g, x), y) = \mathsf{exp}(\mathsf{exp}(g, y), x).$$

Finally, ProVerif was always returning false attacks when applying our methodology to Distance Hijacking scenarios, i.e. scenarios in which there is no dishonest agent in the neighbourhood of the verifier. Indeed, even if we do not consider dishonest agents executing roles of the protocols during phase 1, the built-in ProVerif attacker is still able to interact with participants. To prevent such behaviours, we slightly modified the ProVerif tool: during phase 1 the attacker can only forward messages already sent, or forge new messages using knowledge obtained in phase 0. Based on the extra property stated in Proposition 4, this modification will not miss attacks. Indeed, according to Proposition 4, a recipe involved in an input in phase 1 is either of the form w (a forward message), or only uses handles from phase 0 (i.e. the message has been forged using knowledge obtained in phase 0). Due to some optimisations in the ProVerif code, we noticed that we could not prevent the attacker from using native tuples in phase 1. Therefore, we model tuples using our own function symbols.

8.3 Application to distance-bounding protocols

First, one may note that almost all the protocols meet the requirements of our theoretical development. The Brands and Chaum with signature and MAD protocols are the two unique protocols that do not match the requirements for Terrorist fraud analysis. Indeed, the former does not satisfy item **(iv)** of Definition 12 since the response to a challenge c is $c \oplus m$ with $\oplus$ a non quasi-free symbol of function. The last does not satisfies item **(i)** due to a lack of freshness of the challenge.

Our results, summarised in Table 1, are in line with those presented by Mauw *et. al.* in [44, 45]. However, some differences with the ones presented by Chothia *et. al.* in [19] exist. First, due to a slightly different modelling of the protocol, the Hancke and Kuhn protocol is claimed to be Terrorist fraud resistant in [19], whereas an attack exists. Second, in [19], it is assumed that a message cannot be received by both a local and a distant agent. This modelling choice is more restrictive than ours and prevent them from detecting some Distance Hijacking attacks, e.g. the one on the MAD protocol, as well as the one on the Meadows protocol (variant $f := \langle n_V, n_P \oplus \mathsf{id_P} \rangle$).

The SPADE protocol, presented as a running example in the previous sections, has been analysed and ProVerif correctly retrieved the Mafia fraud depicted in Example 10. This attack, which also

applies on the TREAD protocol, has been reported to the authors who proposed a fix. It consists in adding the identity of the verifier V inside the signature in the first message of the protocol. This variant has been proved Mafia fraud resistant. Coming back again on the running example, the reader can see that SPADE is proved Terrorist fraud resistant as explained in Examples 13 and 21.

ProVerif always returned in less than few second except for the analysis of the Distance Hijacking scenario for the protocol SKI for which the analysis takes about 1 minute.

8.4 Application to payment protocols

In addition to standard distance-bounding protocols, we also applied our technique to new EMV protocols, PaySafe, NXP and MasterCard RRP, designed to avoid relay attacks in the context of payment protocols. A noteworthy difference between these three protocols lies in the threshold used to define the proximity of the participants. In PaySafe, the threshold is a constant known by the reader at the beginning of the protocol. In contrast, in NXP and MasterCard RRP, the threshold is sent by the card to the reader during the initialisation step. Unfortunately, our model does not allow to consider messages including time values (a message is a term built over names and agent identities). To overcome this limitation, similarly to [19], we decided to first prove authenticity and integrity (two standard properties) of the threshold using ProVerif and then verify Mafia fraud, Distance Hijacking attack and Terrorist fraud assuming a public constant threshold.

All the results are presented in Table 2. ProVerif returns in few seconds except for the Distance Hijacking scenario for the MasterCard RRP protocol. On this example, ProVerif was unable to reach a conclusion within few hours, and thus we decided to consider ProVerif's native tuples outside the fast phase. We actually only replace external pairs. This encoding allows us to get a result within 2 minutes.

As expected, the three protocols are resistant to relay attacks, i.e. Mafia fraud, but not to Distance Hijacking attack and Terrorist fraud. These weaknesses are not surprising since these protocols have not been designed to resist to these classes of attacks. However, it is legitimate to wonder if these scenarios of attacks are relevant when considering payments protocols. In case of Distance Hijacking scenarios, one may note that an attacker may have incentives to abuse an honest agent. Indeed, a purchase might be considered as a proof of location since readers are assumed to only authenticate close participants. Distance Hijacking resistance may thus be desirable. In contrast, allowing Terrorist fraud could be a feature of the card. It would permit its user to agree for a one-time payment to a third-party while not being physically next to it, without risking any non-expected following payment, similarly to the current virtual credit card system. For this reason Terrorist fraud resistance does seem to be relevant.

8.5 Comparison with existing approaches regarding the verification aspect

We pursue the comparison between our approach and the ones developed in [19, 44, 45] focusing on how automation has been achieved.

As already mentioned, in the approach developed in [19], the security analysis is performed focusing on the rather simple topologies similar to $\mathcal{T}_{\mathsf{MF}}^{t_0}$ and $\mathcal{T}_{\mathsf{DH}}^{t_0}$. Here, we provide a strong formal foundation showing that this can indeed be done without missing any attack. This corresponds to the reduction results presented in Section 5. Another difference can be found in the way of modelling scenarios without malicious agents in the vicinity of the verifier (e.g., Distance Hijacking attacks). Whereas Chothia *et al.* resort to private channels during the challenge/response exchange to prevent the attacker from interacting with participants, relying on the theoretical result established in Proposition 4, we decided to slightly modify the ProVerif tool to forbid the ProVerif's attacker to manipulate messages during phase 1. This has been achieved by discarding Horn clauses

Protocols	MF	DH	TF	Hyp.
Basin's toy example [7]	✓	✓	✓	–
Brands and Chaum [15]				
• Signature	✓	✗	*o.o.s.*	xor
• Fiat-Shamir	✓	✗	✗	exp.
CRCS No-revealing sign [53]				
• No-revealing sign	✓	✓	✗	–
• Revealing sign	✓	✗	✗	–
Eff-PKDB [41]				
• No protection *(new)*	✓	✓	✓	xor, exp.
• Protected *(new)*	✓	✓	✓	xor, exp.
Hancke and Kuhn	✓	✓	✗	–
MAD (One-Way) [17]	✓	✗	*o.o.s.*	xor
Meadows *et al.* [47]				
• $f := \langle n_V \oplus n_P, \mathrm{id_P} \rangle$ *(new)*	✓	✓	✗	xor
• $f := \langle n_V, n_P \oplus \mathrm{id_P} \rangle$	✓	✗	✗	xor
• $f := n_V \oplus \mathrm{h}(n_P, \mathrm{id_P})$	✓	✓	✗	xor
• $f := \langle n_V, \mathrm{id_P}, n_P \rangle$	✓	✓	✗	–
Munilla *et al.* [50]	✓	✓	✗	–
SKI [14] *(new)*	✓	✓	✓	xor
SPADE				
• Original [16] *(new)*	✗	✗	✓	xor
• Fixed [37] *(new)*	✓	✗	✓	xor
Swiss-Knife				
• Original [42]	✓	✓	✓	xor
• Modified version [35] *(new)*	✓	✓	✗	xor
TREAD asymmetric [4, 37]				
• Original (using $\mathrm{id_{priv}}$)	✗	✗	✓	xor
• Fixed (using $\mathrm{id_{priv}}$) *(new)*	✓	✗	✓	xor
• Original (using $\mathrm{id_{pub}}$)	✗	✗	✓	xor
• Fixed (using $\mathrm{id_{pub}}$) *(new)*	✓	✗	✓	xor
TREAD symmetric [4]	✓	✗	✓	xor

Table 1. Results on our case studies (✗: attack found, ✓: proved secure, *o.o.s.*: out of scope).
MF = Mafia fraud, DH = Distance Hijacking attack, TF = Terrorist fraud.
All the results are obtained within few seconds on a standard laptop.
(new) means that no symbolic analysis reported before.

Protocols	MF	DH	TF
MasterCard RRP [34]	✓	✗	✗
NXP [40]	✓	✗	✗
PaySafe [20]	✓	✗	✗

Table 2. Results of the analysis of EMV protocols (✗: attack found, ✓: proved secure).
MF = Mafia fraud, DH = Distance Hijacking attack, TF = Terrorist fraud
All the results (but DH for MasterCard RRP) are obtained within few seconds on a standard laptop.

allowing the attacker to perform deduction during phase 1. To be precise, this corresponds to few lines of codes that have been added in the file pitrans.ml.

Regarding the security analysis conducted in [44, 45] and relying on the Tamarin tool one may note that the results have been obtained with the autoprove mode of the tool, and do not require any interaction with the user. Moreover, as already mentioned, the security definitions proposed in [44, 45] are in line with ours. In particular, when considering Terrorist fraud, their notion of "valid extensions" of a protocol corresponds to our notion of semi-dishonest provers. However, as expected, one must consider all the possible "valid initial extensions" when analysing a protocol. This source of unboundedness makes the automation of the analysis difficult: no existing tools are able to handle it. In [45], authors illustrate their methodology on a toy example for which they provide a hand-written proof justifying they can focus on a unique extension. One may note that this last corresponds to what we call the most-general semi-dishonest prover. Instead of requiring a hand-written proof for each protocol, we decided to go further in terms of automation and, under reasonable assumptions, we formally defined this most-general semi-dishonest prover and proved its completeness. This corresponds to the reduction result presented in Section 6. In case the conditions needed to apply this reduction result are not satisfied, the corresponding protocol is declared out of scope (o.o.s.) in Table 1. Note that since these two protocols are vulnerable to a Terrorist fraud, we might have decided to describe the well-known semi-dishonest provers that lead to this attack, and prove using ProVerif that no re-authentication is possible. We preferred to precise (o.o.s) in order to highlight the limitation of our reduction result.

Finally, to make the analysis possible using the existing tool Tamarin, Mauw *et al.* prove the equivalence between the security properties expressed in a timed model and purely causality-based properties, i.e. properties solely based on the order of the actions in the trace of execution. Unlike us, in case of Distance Hijacking scenarios (i.e. without malicious agents close to the verifier), they have not modified the tool to prevent undesirable behaviours of the built-in attacker. Instead, they decided to tag all the actions of malicious agents and make the security property trivially satisfied whenever such a tag occurs during the challenge/response exchange. Unfortunately, similarly to ProVerif, the built-in attacker model of Tamarin allows the attacker to act at anytime and bypass their tag mechanism. This may lead to false attacks when analysing a protocol w.r.t. Distance Hijacking attacks in their framework. Therefore, it is important to check the attack returned by their tool to make sure that it corresponds to an attack scenario. This issue has been acknowledged by the authors.

9 CONCLUSION

This paper has been dedicated to the symbolic verification of distance-bounding protocols with the introduction of a new model allowing one to take into consideration the location of the agents and the fact that transmitting a message takes time. Then, formal definitions of the three main

classes of attacks against which distance-bounding protocols are usually analysed have been proposed. A specific interest has been paid on the definition of Terrorist frauds for which modelling the collusion behaviours was a challenging problem. Regarding automatic verification, we managed to analyse many distance-bounding protocols relying on two reduction results and the phase mechanism available in the ProVerif tool.

Despite these positive results, we would like to highlight some limitations. First, neither our framework nor those proposed in [19, 44, 45] take mobility into account. We think that this is achievable following the approach recently proposed in [13]. This model allows agents to perform arbitrary movements as soon as they do not move faster than messages. Nonetheless this will require significant changes to our model to adapt e.g. our security definitions to make precise when the agents are required to be close (they are now able to move!).

Second, none of the existing frameworks (including ours) allow to faithfully model the exclusive-or operator which is an operator used by most of the distance-bounding protocols. We may note that our theoretical development (i.e. reduction results developed in Sections 5 and 6) does not suffer from this limitation, this is only a limitation of the existing automated tools. The Tamarin tool is probably the most advanced one regarding this aspect thanks to its recent extension provided in [31]. However, this does not seem to be the panacea. For example, we tried to perform our case studies relying on this tool but we faced up many non-termination issues. The same behaviours seem to have appeared in [44, 45] since several protocols have been modelled relying on a weak form of exclusive-or (as in our ProVerif encodings).

Another interesting direction would be to improve the modelling of the fast phase which relies on bit-level operations, and to switch from a qualitative to a quantitative security analysis. The recent probabilistic model proposed by Chadha *et al.* [18] could serve at a starting point. However, even if this model is promising, it also opens challenging problems. The fast phase is not isolated to the rest of the protocol, and thus the interactions between messages modelled at the bit level, and the remaining ones modelled through a standard term algebra remain unclear.

ACKNOWLEDGEMENTS

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).

REFERENCES

[1] https://gitlab.inria.fr/adebant/db-verif.

[2] 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.

[3] G. Avoine, M. Ali Bingöl, S. Kardas, C. Lauradoux, and B. Martin. A framework for analyzing RFID distance bounding protocols. *Journal of Computer Security*, 19(2):289–317, 2011.

[4] G. Avoine, X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J.-M. Robert. A terrorist-fraud resistant and extractor-free anonymous distance-bounding protocol. In *Proc. 12th ACM Asia Conference on Computer and Communications Security (AsiaCCS'17)*, pages 800–814. ACM Press, 2017.

[5] G. Avoine et al. Security of distance-bounding: A survey. *ACM Computing Surveys*, 51(5):94:1–94:33, 2019.

[6] G. Avoine and A. Tchamkerten. An efficient distance bounding rfid authentication protocol: balancing false-acceptance rate and memory requirement. In *Proc. 12th International Conference on Information Security (ISC'09)*, pages 250–261. Springer, 2009.

[7] 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)*, 14(2):16, 2011.

[8] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler. A formal analysis of 5G authentication. In *Proc. 25th ACM Conference on Computer and Communications Security (CCS'18)*, pages 1383–1396, 2018.

[9] K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *Proc. 38th IEEE Symposium on Security and Privacy (S&P'17)*, pages 483–503, 2017.

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

[11] B. 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.

[12] B. Blanchet. Symbolic and computational mechanized verification of the ARINC823 avionic protocols. In *Proc. 30th IEEE Computer Security Foundations Symposium (CSF'17)*, pages 68–82, 2017.

[13] I. Boureanu, T. Chothia, A. Debant, and S. Delaune. Security Analysis and Implementation of Relay-Resistant Contactless Payments. In *Proc. 27th ACM Conference on Computer and Communications Security (CCS'20)*, pages 879–898, 2020.

[14] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Secure and lightweight distance-bounding. In *Proc. 2nd International Workshop on Lightweight Cryptography for Security and Privacy (LightSec'13)*, volume 8162 of *LNCS*, pages 97–113. Springer, 2013.

[15] S. Brands and D. Chaum. Distance-bounding protocols. In *Proc. Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'93)*, pages 344–359. Springer, 1993.

[16] X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J.-M. Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *Proc. 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, (WISEC'16)*, pages 121–133. ACM Press, 2016.

[17] 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*, pages 21–32. ACM, 2003.

[18] R. Chadha, A. Prasad Sistla, and M. Viswanathan. Verification of randomized security protocols. In *Proc. 32nd Annual IEEE Symposium on Logic in Computer Science, (LICS'17)*, pages 1–12. IEEE Computer Society, 2017.

[19] T. Chothia, J. de Ruiter, and B. Smyth. Modelling and analysis of a hierarchy of distance bounding attacks. In *Proc. 27th USENIX Security Symposium (USENIX'18)*, 2018.

[20] 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)*, LNCS, 2015.

[21] V. Cortier, A. Filipiak, and J. Lallemand. BeleniosVS: Secrecy and verifiability against a corrupted voting device. In *Proc. 32nd Computer Security Foundations Symposium (CSF'19)*, pages 367–381, 2019.

[22] V. Cortier, D. Galindo, and M. Turuani. A formal analysis of the neuchâtel e-voting protocol. In *Proc. 3rd IEEE European Symposium on Security and Privacy (EuroS&P'18)*, pages 430–442, 2018.

[23] C. Cremers, K. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *Proc. 33rd IEEE Symposium on Security and Privacy (S&P'12)*, pages 113–127, 2012.

[24] A. Debant and S. Delaune. Symbolic verification of distance bounding protocols. In *Proc. 8th International Conference on Principles of Security and Trust (POST'19)*, LNCS, pages 149–174. Springer, 2019.

[25] A. Debant, S. Delaune, and Wiedling C. So near and yet so far - symbolic verification of distance-bounding protocols. Research report, Univ Rennes, CNRS, IRISA, France, October 2020.

[26] A. Debant, S. Delaune, and C. Wiedling. A symbolic framework to analyse physical proximity in security protocols. In *Proc. 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, (FSTTCS'18)*, volume 122 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[27] A. Debant, S. Delaune, and C. Wiedling. Symbolic analysis of terrorist fraud resistance. In *Proc. 24th European Symposium on Research in Computer Security (ESORICS'19)*, volume 11735 of *LNCS*, pages 383–403. Springer, 2019.

[28] Y. Desmedt. Major security problems with the 'unforgeable' (Feige)-Fiat-Shamir proofs of identity and how to overcome them. In *Proc. SECURICOM'88*.

[29] Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. In *Proc. 7th Conference on the Theory and Applications of Cryptographic Techniques (CRYPTO'87)*, pages 21–39. Springer, 1987.

[30] 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, 1981.

[31] J. Dreier, L. Hirschi, S. Radomirovic, and R. Sasse. Automated Unbounded Verification of Stateful Cryptographic Protocols with Exclusive OR. In *Proc. 31st IEEE Computer Security Foundations Symposium (CSF'18)*, 2018.

[32] S. Drimer, S. J Murdoch, et al. Keep your enemies close: Distance bounding against smartcard relay attacks. In *Proc. 16th USENIX Security Symposium, (USENIX'07)*, volume 312, 2007.

[33] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. A formal approach to distance-bounding RFID protocols. In *Proc. 14th International Conference on Information Security (ISC'11)*, volume 7001 of *LNCS*. Springer, 2011.

[34] EMVCo. EMV contactless specifications for payment systems, version 2.6, 2016.

[35] M. Fischlin and C. Onete. Subtle kinks in distance-bounding: an analysis of prominent protocols. In *Proc. 6th ACM conference on Security and privacy in wireless and mobile networks*, pages 195–206, 2013.

[36] A. Francillon, B. Danev, and S. Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *Proc. Network and Distributed System Security Symposium (NDSS'11)*. The Internet Society, 2011.

[37] D. Gérault. *Security Analysis of Contactless Communication Protocols*. PhD thesis, Université Clermont Auvergne, 2018.

[38] G. Girol, L. Hirschi, R. Sasse, D. Jackson, C. Cremers, and D. Basin. A spectral analysis of Noise: A comprehensive, automated, formal analysis of Diffie-Hellman protocols. In *Proc. 29th USENIX Security Symposium, (USENIX'20)*.

[39] L. Hirschi and C. Cremers. Improving automated symbolic analysis of ballot secrecy for e-voting protocols: A method based on sufficient conditions. In *Proc. 4th IEEE European Symposium on Security and Privacy (EuroS&P'19)*, pages 635–650, 2019.

[40] P. Janssens. Proximity check for communication devices, October 31 2017. US Patent 9,805,228.

[41] H. Kilinç and S. Vaudenay. Efficient public-key distance bounding protocol. In *Proc. 22nd International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'16)*, volume 10032 of *LNCS*, pages 873–901, 2016.

[42] Chong Hee Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira. The Swiss-Knife RFID distance bounding protocol. In *Proc. 11th International Conference on Information Security and Cryptology (ICISC'08)*, LNCS. Springer, 2008.

[43] N. Kobeissi, K. Bhargavan, and B. Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *Proc. 2nd IEEE European Symposium on Security and Privacy (EuroS&P'17)*, pages 435–450, 2017.

[44] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua. Distance-bounding protocols: Verification without time and location. In *Proc. 39th IEEE Symposium on Security and Privacy (S&P'18)*, pages 152–169, 2018.

[45] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua. Post-Collusion Security and Distance Bounding. In *Proc. 26th ACM Conference on Computer and Communications Security (CCS'19)*, pages 941–958. ACM, 2019.

[46] S. Mauw, J. Toro-Pozo, and R. Trujillo-Rasua. A class of precomputation-based distance-bounding protocols. In *Proc. 1st IEEE European Symposium on Security and Privacy (EuroS&P'16)*. IEEE, 2016.

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

[48] 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)*, LNCS. Springer, 2013.

[49] J. Mitchell, A. Scedrov, N. Durgin, and P. Lincoln. Undecidability of bounded security protocols. In *Proc. Workshop on Formal Methods and Security Protocols*, 1999.

[50] 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*, 8(9):1227–1232, 2008.

[51] 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)*, pages 450–470. Springer, 2016.

[52] T. Nipkow, L. C Paulson, and M. Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283. Springer Science & Business Media, 2002.

[53] K. B. Rasmussen and S. Capkun. Realization of rf distance bounding. In *Proc. 19th USENIX Security Symposium, (USENIX'10)*, pages 389–402, 2010.

[54] B. Schmidt, R. Sasse, C. Cremers, and D. Basin. Automated verification of group key agreement protocols. In *Proc. 35th IEEE Symposium on Security and Privacy (S&P'14)*, pages 179–194, 2014.

[55] R. Trujillo-Rasua, B. Martin, and G. Avoine. The Poulidor distance-bounding protocol. In *Proc. International Workshop on Radio Frequency Identification: Security and Privacy Issues*, pages 239–257. Springer, 2010.

A PROOFS FOR SECTION 5.1

Lemma 1. *Let $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0)$ be a topology, and $\mathcal{K}_0$ be a configuration built on $\mathcal{T}_0$. Let $\mathcal{H}_0 \subseteq \mathcal{A}_0 \setminus \mathcal{M}_0$. Let $\mathcal{K}$ be a configuration such that $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} \mathcal{K}$. We have that $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}'} \mathcal{K}$ where $\mathcal{T}' = (\mathcal{A}_0, \mathcal{M}_0 \cup \mathcal{H}_0, Loc_0, v_0, p_0)$.*

Proof. We show this result by induction on the length of the derivation $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} \mathcal{K}$. The base case, i.e. tr is the empty trace, is trivial. To conclude, it is sufficient to show that:

$$\mathcal{K}_1 \xrightarrow{a,\alpha}_{\mathcal{T}_0} \mathcal{K}_2 \text{ implies } \mathcal{K}_1 \xrightarrow{a,\alpha}_{\mathcal{T}'} \mathcal{K}_2.$$

Let $\mathcal{K}_1 = (\mathcal{P}_1; \Phi_1; t_1)$. We consider each rule of the semantics one by one. Actually, the only rule that depends on the status (honest/dishonest) of the agents of the underlying topology is the rule IN. In such a case, we have that $\alpha = \mathsf{in}^\star(u)$ for some u. Moreover, following the notation introduced in Figure 2, we know that there exist $b \in \mathcal{A}_0$ (the agent responsible of the corresponding output) and $t_b \in \mathbb{R}_+$ (the time at which the output has been triggered). The only interesting case is when $b \in \mathcal{H}_0$, and therefore b is now a malicious agent in the topology $\mathcal{T}'$ whereas b was an honest one in the topology $\mathcal{T}_0$. Since, the IN rule was triggered in $\mathcal{T}_0$, we know that $u \in img(\lfloor \Phi_1 \rfloor_b^{t_b})$, and therefore there exists $w \in dom(\lfloor \Phi_1 \rfloor_b^{t_b})$ such that $w\Phi_1\downarrow = u$. Actually, it is easy to see that choosing the recipe $R = w$ (and $c = b$) allows us to conclude. Indeed, we have that $t_b - \mathsf{Dist}_{\mathcal{T}'}(b, b) = t_b$, and therefore we conclude since we have already shown that $w \in dom(\lfloor \Phi_1 \rfloor_b^{t_b})$. □

Lemma 2. *Let $\mathcal{T}_0$ be a topology, $\mathcal{D}_0$ be a subset of malicious agents, and $\mathcal{K}_0 = (\mathcal{P}_0; \Phi_0; t_0)$ be a configuration built on $\mathcal{T}_0$ such that P_a is executable w.r.t. $img(\lfloor \Phi_0 \rfloor_a^{t_0})$ for any $\lfloor P_a \rfloor_a^{t_a} \in \mathcal{P}_0$ with $a \in \mathcal{D}_0$. Let $\mathcal{K} = (\mathcal{P}; \Phi_0 \uplus \Phi^+; t)$ be a configuration such that $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} \mathcal{K}$. We have that*

$$(\overline{\mathcal{P}_0}; \Phi_0; t_0) \xrightarrow{\overline{\mathsf{tr}}\sigma}_{\mathcal{T}_0} (\overline{\mathcal{P}}\sigma; \Phi_0 \uplus \overline{\Phi^+}\sigma; t)$$

where $\overline{\mathcal{P}_0}$ (resp. $\overline{\mathcal{P}}$, $\overline{\Phi^+}$, $\overline{\mathsf{tr}}$) is obtained from $\mathcal{P}_0$ (resp. $\mathcal{P}$, Φ^+, tr) by removing processes (resp. frame or trace elements) located in $a \in \mathcal{D}_0$ and $\sigma(n) = \mathsf{c}_0 \in \Sigma_0$ for any name n freshly generated to trigger the rules NEW executed by agent $a \in \mathcal{D}_0$ in tr.

Proof. We show this result assuming that $\mathcal{D}_0$ contains a unique element a_0. The general result can then easily be obtained by a simple induction on the size of $\mathcal{D}_0$. More precisely, we show the following results by induction on the length of $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} \mathcal{K} = (\mathcal{P}; \Phi_0 \uplus \Phi^+; t)$:

(1) for all $\lfloor P \rfloor_{a_0}^{t_a} \in \mathcal{P}$, $P\sigma$ is executable w.r.t. $\Psi(t)$;
(2) for all $w \xrightarrow{a_0, t_a} u \in \Phi^+$, there exists $R \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, dom(\Psi(t_a)))$ such that $R\Psi(t_a)\downarrow =_{\mathsf{E}} u\sigma$;
(3) $(\overline{\mathcal{P}_0}; \Phi_0, t_0) \xrightarrow{\overline{\mathsf{tr}}\sigma}_{\mathcal{T}} (\overline{\mathcal{P}}\sigma; \Phi_0 \uplus \overline{\Phi^+}\sigma; t)$;

where $\Psi(t) = \Phi_0 \uplus \{ \lfloor \Phi^+\sigma \rfloor_b^{t - \mathsf{Dist}_{\mathcal{T}_0}(b, a_0)} \mid b \neq a_0 \}$.

The base case, i.e. when tr is empty, is trivial. Now, we assume that

$$\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} \mathcal{K} = (\mathcal{P}; \Phi_0 \uplus \Phi^+; t) \xrightarrow{a,\alpha}_{\mathcal{T}_0} \mathcal{K}' = (\mathcal{P}'; \Phi_0 \uplus \Phi'; t')$$

and thanks to our induction hypothesis, we know that the three properties above hold on $\mathcal{K}$. We note σ' the substitution regarding the trace $\mathsf{tr}.(a, \alpha)$. We do a case analysis on the rule involved in the last step.

Rule TIM. In such a case, we have that $\Phi' = \Phi^+$ and $\mathcal{P}' = \mathcal{P}$. Since $t' \geq t$, we have that $\Psi(t) \subseteq \Psi(t')$, and this allows us to conclude.

Rule OUT. Items (1) and (3) are quite obvious. Regarding item (2), the only non trivial case is when $a = a_0$. We have to show that the element added to the frame, namely $w \xrightarrow{a_0,t} u$ satisfies the expected property. Let $\mathcal{P} = \{ \lfloor \mathtt{out}(u).P \rfloor^{t_a}_{a_0} \} \cup \mathcal{P}_1$ and $\mathcal{P}' = \{processfloorPa_0t_a\} \cup \mathcal{P}_1$. By item (1), we know that the process $\mathtt{out}(u\sigma).P\sigma$ responsible of this output is executable w.r.t. $img(\Psi(t))$, and thus there exists $R \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, dom(\Psi(t)) \cup bn(\mathtt{out}(u\sigma).P\sigma) \cup bv(\mathtt{out}(u\sigma).P\sigma))$ such that $R\Psi(t)\downarrow = u\sigma$. More precisely $R \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, dom(\Psi(t)))$ because $u\sigma$ does not contain any bound names/variables.

Rule LET. Items (2) and (3) are quite immediate. Indeed, regarding item (2) we conclude by remarking that $\sigma' = \sigma$ and $\Phi' = \Phi^+$. Regarding item (3) we simply note that if a term $u\downarrow$ is a constructor term then $u\sigma\downarrow$ is a constructor term too. Finally, the only non trivial point to establish is item (1) when $a = a_0$. Let $\mathcal{P} = \lfloor \mathtt{let}\ x = v\ \mathtt{in}\ P' \rfloor^{t_a}_{a_0} \uplus \mathcal{P}_1$, $\mathcal{P}' = \lfloor P'\{x \mapsto v\downarrow\} \rfloor^{t_a}_{a_0} \uplus \mathcal{P}_1$, $\Phi' = \Phi^+$, $t' = t$, and $\sigma' = \sigma$. By hypothesis, we know that the process $(\mathtt{let}\ x = v\sigma\ \mathtt{in}\ P'\sigma)$ is executable w.r.t. $\Psi(t)$, thus there exists $R \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, dom(\Psi(t)))$ such that $R\Psi(t)\downarrow = v\sigma\downarrow$ (not that v does not contain any bound names/variables). To prove item (1), we have to show that $(P'\{x \mapsto v\downarrow\})\sigma$ is executable w.r.t. $\Psi(t') = \Psi(t)$. Let u be a term occurring in an output (resp. a let) in $(P'\{x \mapsto v\downarrow\})\sigma = P'\sigma\{x \mapsto v\sigma\downarrow\}$. We have that there exists u_0 that occurs in an output (resp. a let) in $P'\sigma$ such that $u_0\{x \mapsto v\sigma\downarrow\} = u$, and by hypothesis, we know that $(\mathtt{let}\ x = v\sigma\ \mathtt{in}\ P'\sigma)$ is executable w.r.t. $\Psi(t)$, i.e. there exists $R_0 \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, dom(\Psi(t)) \cup bn(P'\sigma) \cup bv(P'\sigma) \cup \{x\})$ such that $R_0\Psi(t)\downarrow = u_0\downarrow$ (actually there is no need of normalisation in case of an output). Let $R' = R_0\{x \mapsto R\}$. We have that $R' \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, dom(\Psi(t)) \cup bn(P'\sigma) \cup bv(P'\sigma))$ and

$$R'\Psi(t')\downarrow = (R_0\{x \mapsto R\})\Psi(t)\downarrow = (R_0\Psi(t)\{x \mapsto v\sigma\downarrow\})\downarrow = (u_0\downarrow\{x \mapsto v\sigma\downarrow\})\downarrow = u\downarrow.$$

Note that $u = u_0\{x \mapsto v\sigma\downarrow\}$ is a constructor term when u_0 is is a constructor term. Thus, no normalisation is needed in case u is a term occurring in an output, and we have that $R'\Psi(t')\downarrow = u$.

Rule NEW. In that case we have that $\mathcal{P} = \lfloor \mathtt{new}\ n.P' \rfloor^{t_0}_{a} \uplus \mathcal{P}_1$, $\mathcal{P}' = \lfloor P'\{n \mapsto n'\} \rfloor^{t_a}_{a} \uplus \mathcal{P}_1$, $\Phi' = \Phi^+$ and $t' = t$. The only interesting case is when $a = a_0$ and thus $\sigma' = \sigma \cup \{n' \mapsto \mathsf{c}_0\}$. Note that items (2) and (3) are immediate: we have that $\overline{\mathsf{tr}.(a_0, \tau)} = \overline{\mathsf{tr}}$ and n' is fresh and thus neither occurs in $\mathcal{P}_1$ nor in Φ^+.

Regarding item (1) we have to prove that $P'\{n \mapsto n'\}\sigma'$ is executable w.r.t. $\Psi(t)$. Let u be a term occurring in an output (resp. a let) in $P'\{n \mapsto n'\}$. We have that there exists u_0 that occurs in an output (resp. a let) in P' such that $u_0\{n \mapsto n'\} = u$. By hypothesis, $(\mathtt{new}\ n.P')\sigma$ is executable w.r.t. $\Psi(t)$, and thus there exists $R \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, dom(\Psi(t)) \cup bn(P'\sigma) \cup \{n\} \cup bv(P'\sigma))$ such that $R\Psi(t)\downarrow = u_0\sigma$ in case of an output (resp. $u_0\sigma\downarrow$ in case of a let). We note $R' = R\{n \mapsto \mathsf{c}_0\}$. We have that $P'\sigma = P'\sigma'$, and thus we obtain that $R' \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, dom(\Psi(t)) \cup bn(P'\sigma') \cup bv(P'\sigma'))$ and:

$$R'\Psi(t)\downarrow = R\{n \mapsto \mathsf{c}_0\}\Psi(t)\downarrow = R\Psi(t)\downarrow\{n \mapsto \mathsf{c}_0\}\downarrow = u_0\sigma\downarrow\{n \mapsto \mathsf{c}_0\}\downarrow = u\sigma'\downarrow.$$

Note that $u\sigma' = u_0\sigma\{n \mapsto \mathsf{c}_0\}$ is a constructor term when u_0 is a constructor term. Thus, no normalisation is needed in case u is a term occurring in an output.

Rule RST. In such a case, the result trivially holds.

Rule IN. In case $a = a_0$, the only non trivial point is to establish item (1), and this can be done in a similar way as it was done in Rule LET. Otherwise, i.e. when $a \neq a_0$, the non trivial point is to establish item (3). Let $\alpha = \mathtt{in}^\star(u)$. We have to establish that the IN rule can still be fired with the value $u\sigma\downarrow$ despite the fact that some elements have been removed from Φ^+.

Following the notations introduced in Section 3.3, we know that there exist $b \in \mathcal{A}_0$ and $t_b \in \mathbb{R}_+$ such that $t_b \leq t - \mathrm{Dist}_{\mathcal{T}_0}(b, a)$ and a recipe R such that $R(\Phi_0 \uplus \Phi^+)\downarrow = u$ and for all $w \in vars(R)$ there exists $c \in \mathcal{A}_0$ such that $w \in dom(\lfloor \Phi_0 \uplus \Phi^+ \rfloor^{t_b - \mathrm{Dist}_{\mathcal{T}_0}(c,b)}_{c})$.

(1) If $b \notin \mathcal{M}_0$ then we know that $R = w_0$ for some w_0 and we have that $\lfloor \Phi_0 \uplus \Phi^+ \rfloor_b^{t_b} = \lfloor \Phi_0 \uplus \overline{\Phi^+} \rfloor_b^{t_b}$ because $b \neq a_0$. Thus, the rule can be applied considering the frame $\lfloor \Phi_0 \uplus \overline{\Phi^+}\sigma \rfloor_b^{t_b}$.
(2) If $b \in \mathcal{M}_0$, then in case $c \neq a_0$, or $c = a_0$ with $w \in \Phi_0$, then it is straightforward (σ does not applies on it). The interesting case is when $w \in dom(\lfloor \Phi^+ \rfloor_{a_0}^{t_b - \mathsf{Dist}_{\mathcal{T}_0}(a_0,b)})$, and we have to reconstruct $(w\Phi^+)\sigma\downarrow$ with elements available in $\Phi_0 \uplus \overline{\Phi^+}\sigma$.
Let w be a variable such that $w \in vars(R)$ and $w \in dom(\lfloor \Phi^+ \rfloor_{a_0}^{t_b - \mathsf{Dist}_{\mathcal{T}_0}(a_0,b)})$. Thanks to item (2), we know that there exists $t_w \leq t_b - \mathsf{Dist}_{\mathcal{T}_0}(a_0, b)$ and a recipe $R_w \in \mathcal{T}(\Sigma_{\mathsf{pub}}, dom(\Psi(t_w)))$ such that $R_w\Psi(t_w)\downarrow =_{\mathsf{E}} (w\Phi^+)\sigma$. By definition of $\Psi(\_)$, we have that

$$\Psi(t_w) = \Phi_0 \uplus \{ \lfloor \Phi^+\sigma \rfloor_c^{t_w - \mathsf{Dist}_{\mathcal{T}_0}(c,a_0)} \mid c \neq a_0 \}.$$

Moreover, we know that

$$t_w - \mathsf{Dist}_{\mathcal{T}_0}(c, a_0) \leq t_b - (\mathsf{Dist}_{\mathcal{T}_0}(c, a_0) + \mathsf{Dist}_{\mathcal{T}_0}(a_0, b)) \leq t_b - \mathsf{Dist}_{\mathcal{T}_0}(c, b).$$

Thus, we have that $vars(R_w) \subseteq dom(\Phi_0) \uplus dom(\{ \lfloor \Phi^+\sigma \rfloor_c^{t_b - \mathsf{Dist}_{\mathcal{T}_0}(c,b)} \mid c \neq a_0 \})$.
Let θ be the substitution with domain $vars(R) \cap dom(\lfloor \Phi^+\sigma \rfloor_{a_0}^{t_b - \mathsf{Dist}_{\mathcal{T}_0}(a_0,b)})$ and such that $\theta(w) = R_w$ as defined above. We have that $R\theta \in \mathcal{T}(\Sigma^+_{\mathsf{pub}}, dom(\Phi_0 \uplus \overline{\Phi^+}\sigma))$ and

$$\begin{array}{rcl} R\theta(\Phi_0 \uplus \overline{\Phi^+}\sigma)\downarrow & = & R(\Phi_0 \uplus \{w \mapsto R_w(\Phi_0 \uplus \overline{\Phi^+}\sigma) \mid w \in dom(\theta)\})\downarrow \\ & = & R(\Phi_0 \uplus \{w \mapsto (w\Phi^+)\sigma \mid w \in dom(\theta)\})\downarrow \\ & = & R(\Phi_0 \uplus \Phi^+\sigma)\downarrow \\ & = & (R(\Phi_0 \uplus \Phi^+))\sigma\downarrow \\ & = & (R(\Phi_0 \uplus \Phi^+)\downarrow)\sigma\downarrow \\ & = & u\sigma\downarrow \end{array}$$

Note that for all $w \in vars(R\theta)$, by definition of $\Psi(t_w)$, we have that $w \xrightarrow{c,t_c} v \in \Phi_0 \uplus \overline{\Phi^+}\sigma$ (for some c, t_c, v) such that $t_c \leq t_b - \mathsf{Dist}_{\mathcal{T}_0}(c, b)$, and thus the IN rule can be triggered at the same time and relying on the same agent b as in the original execution trace.

□

Lemma 3. *Let $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc, v_0, p_0)$ be a topology, $\mathcal{K}_0 = (\mathcal{P}_0; \Phi_0; t_0)$ and $\mathcal{K}$ be two configurations built on $\mathcal{T}_0$ such that $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}} \mathcal{K}$, and H be a set of agents such that*

$$\{a \mid \lfloor P \rfloor_a^t \in \mathcal{P}_0 \text{ or } \lfloor \Phi_0 \rfloor_a^t \neq \emptyset\} \subseteq H \subseteq \mathcal{A}_0 \setminus \mathcal{M}_0.$$

We have that $(\mathcal{P}_0; \Phi_0; t_0) \xrightarrow{\mathsf{tr}}_{\mathcal{T}'} \mathcal{K}$ where $\mathcal{T}'$ is the canonical topology associated to H and $Loc_{|H}$.

Proof. We show this result by induction on the length of the derivation $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}'} \mathcal{K}$. The base case, i.e. tr is the empty trace, is trivial. To conclude, it is sufficient to show that:

$$\mathcal{K}_1 \xrightarrow{a,\alpha}_{\mathcal{T}} \mathcal{K}_2 \text{ implies } \mathcal{K}_1 \xrightarrow{a,\alpha}_{\mathcal{T}'} \mathcal{K}_2.$$

In the following, we note $\mathcal{T}' = (\mathcal{A}', \mathcal{M}', \mathsf{Loc}', v_0, p_0)$. We do the proof considering each rule of the semantics one by one but, actually, the only rule that depends on the underlying topology is the rule IN. In such a case, we have that $\alpha = \mathtt{in}^\star(u)$ for some message u. Moreover, we denote $\mathcal{K}_1 = (\mathcal{P}_1; \Phi_1; t_1)$ and following the notations introduced in Figure 2, we know that there exist $b \in \mathcal{A}_0$ (the agent responsible of the corresponding output) and $t_b \leq t_1 - \mathsf{Dist}_{\mathcal{T}}(b, a)$ (the time at which the output has been triggered) that satisfy the conditions of the rule. We distinguish two cases:

(1) In case $b \in \mathcal{A}_0 \setminus \mathcal{M}_0$, then we know that there exists $w \in img(\lfloor \Phi_1 \rfloor_b^{t_b})$ such that $w\Phi_1\downarrow = u$. By definition of H, we have that $b \in H$, and therefore $b \in \mathcal{A}' \setminus \mathcal{M}'$. The same rule applies for the same reason.

(2) In case $b \in \mathcal{M}_0$, then we know that there exists a recipe R such that $R\Phi_1\downarrow = u$, and for all $w \in vars(R)$ there exists $c \in \mathcal{A}_0$ such that $w \in dom(\lfloor \Phi_1 \rfloor_c^{t_b - \mathsf{Dist}_{\mathcal{T}}(c,b)})$. We show that the same rule applies using the same recipe R. However, the agent responsible of the output will be the agent i_a such that $\mathsf{Loc}'(i_a) = \mathsf{Loc}'(a) = \mathsf{Loc}(a)$ (note that $a \in H$), and this output will be performed at time t_1 (instead of t_b). We have that $R\Phi_1\downarrow = u$. Now, let $w \in vars(R)$. Let $c \in \mathcal{A}_0$ such that $w \in dom(\lfloor \Phi_1 \rfloor_c^{t_b - \mathsf{Dist}_{\mathcal{T}}(c,b)})$. We have that $c \in H$. It remains to show that $w \in dom(\lfloor \Phi_1 \rfloor_c^{t_1 - \mathsf{Dist}_{\mathcal{T}'}(c,i_a)})$. For this, it is actually sufficient to establish that $t_1 - \mathsf{Dist}_{\mathcal{T}'}(c, i_a) \geq t_b - \mathsf{Dist}_{\mathcal{T}}(c, b)$. We know that:

$$\begin{array}{rrcl} & t_1 - \mathsf{Dist}_{\mathcal{T}}(b,a) & \geq & t_b \\ \Rightarrow & t_1 - \mathsf{Dist}_{\mathcal{T}}(b,a) - \mathsf{Dist}_{\mathcal{T}}(c,b) & \geq & t_b - \mathsf{Dist}_{\mathcal{T}}(c,b) \\ \Rightarrow & t_1 - (\mathsf{Dist}_{\mathcal{T}}(b,a) + \mathsf{Dist}_{\mathcal{T}}(c,b)) & \geq & t_b - \mathsf{Dist}_{\mathcal{T}}(c,b) \\ \Rightarrow & t_1 - \mathsf{Dist}_{\mathcal{T}}(c,a) & \geq & t_b - \mathsf{Dist}_{\mathcal{T}}(c,b) \end{array}$$

The last implication comes from the triangle inequality for the distance. Then, we obtain the expected result since $\mathsf{Loc}'(i_a) = \mathsf{Loc}'(a)$, and thus $\mathsf{Dist}_{\mathcal{T}}(c,a) = \mathsf{Dist}_{\mathcal{T}'}(c,a)$ since $a, c \in H$.

This concludes the proof. □

Lemma 4. *Let $\mathcal{K}, \mathcal{K}'$ be two configurations built on $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc, v_0, p_0)$ such that $\mathcal{K} \xrightarrow{\mathsf{tr}}_{\mathcal{T}} \mathcal{K}'$, and $\rho : \mathcal{A} \to \mathcal{A}_0$ be a renaming such that $Loc(\rho(a)) = Loc(a)$ for any $a \in \mathcal{A}_0$, and $\rho(a) \in \mathcal{M}_0$ for any $a \in \mathcal{M}_0$. We have that $\mathcal{K}\rho \xrightarrow{\mathsf{tr}\rho}_{\mathcal{T}} \mathcal{K}'\rho$.*

Proof. We show this result by induction on the length of $\mathcal{K} \xrightarrow{\mathsf{tr}}_{\mathcal{T}} \mathcal{K}'$. The base case, i.e. tr is the empty trace, is trivial. To conclude, it is sufficient to show that $\mathcal{K}_1 \xrightarrow{a,\alpha}_{\mathcal{T}} \mathcal{K}_2$ with $\mathcal{K}_1$ a configuration built on $\mathcal{T}$ implies that $\mathcal{K}_1\rho \xrightarrow{\rho(a),\rho(\alpha)}_{\mathcal{T}} \mathcal{K}_2\rho$. First, note that $\mathcal{K}_1$ only involves processes located at $a \in \mathcal{A}_0$, and therefore actions along the derivation are only executed by agents in $\mathcal{A}_0$ whose locations remain unchanged by ρ. We consider each rule of the semantics one by one. The only rules that are not trivial are the rules LET and IN.

Case of the rule LET. In such a case, we have that $\mathcal{K}_1 = (\mathcal{P}_1; \Phi_1; t_1)$ and $\mathcal{K}_2 = (\mathcal{P}_2; \Phi_2; t_2)$ with $\mathcal{P}_1 = \lfloor \mathtt{let}\ x = u\ \mathtt{in}\ P \rfloor_a^{t_a} \uplus \mathcal{P}$, $\mathcal{P}_2 = \lfloor P\{x \mapsto u\downarrow\} \rfloor_a^{t_a} \uplus \mathcal{P}$, $\Phi_2 = \Phi_1$, and $t_2 = t_1$. We know that $u\downarrow \in \mathcal{T}(\Sigma_c^+, \mathcal{N} \uplus \mathcal{A})$, and therefore we have that $(u\rho)\downarrow \in \mathcal{T}(\Sigma_c^+, \mathcal{N} \uplus \mathcal{A})$ by applying the same rewriting rule at each step. This allows us to apply the rule LET and to obtain the expected result.

Case of the rule IN. In such a case, we have that $\mathcal{K}_1 = (\mathcal{P}_1; \Phi_1; t_1)$ and $\mathcal{K}_2 = (\mathcal{P}_2; \Phi_2; t_2)$ with $\mathcal{P}_1 = \lfloor \mathtt{in}^\star(x).P \rfloor_a^{t_a} \uplus \mathcal{P}$, $\mathcal{P}_2 = \lfloor P\{x \mapsto u\} \rfloor_a^{t_a} \uplus \mathcal{P}$, $\Phi_2 = \Phi_1$, and $t_2 = t_1$. We know that there exists $b \in \mathcal{A}_0$, $t_b \in \mathbb{R}_+$ such that $t_b \leq t_1 - \mathsf{Dist}_{\mathcal{T}}(b,a)$, and a recipe R such that $u = R\Phi_1\downarrow$, and for all $w \in vars(R)$ there exists $c \in \mathcal{A}_0$ such that $w \in dom(\lfloor \Phi_1 \rfloor_c^{t_b - \mathsf{Dist}_{\mathcal{T}}(c,b)})$. Moreover, when $b \in \mathcal{A}_0 \setminus \mathcal{M}_0$, we know that $R \in \mathcal{W}$. To conclude that the rule IN can be applied, we simply have to show that $R(\Phi\rho)\downarrow = u\rho$. Note that the renaming ρ keeps the locations of the agents in $\mathcal{A}_0$ unchanged, and therefore the conditions about distance are still satisfied. We have that $R(\Phi\rho)\downarrow = (R\Phi)\rho\downarrow$ since $R \in \mathcal{T}(\Sigma_{\mathsf{pub}}^+, \mathcal{W})$. We know that $(R\Phi)\downarrow = u$, and therefore we have that $(R\Phi)\rho\downarrow = u\rho$ by applying the same rewriting rule at each step. Note that $u\rho$ does not contain any destructor symbol and is thus in normal form. To conclude, it remains to ensure that when $\rho(b) \in \mathcal{A}_0 \setminus \mathcal{M}_0$, then $R \in \mathcal{W}$. Actually, we know that if $\rho(b) \notin \mathcal{M}_0$ then $b \notin \mathcal{M}_0$ by hypothesis, and this allows us to conclude. □

The following proposition shows that, assuming that each guarded input is preceded by a reset in the configuration under study, then it is possible to follow the same execution starting with an initial configuration, i.e. with global time set to 0.

Proposition 5. *Let $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc, v_0, p_0)$ be a topology and $\mathcal{K} = (\mathcal{P}; \Phi; t)$ be a configuration such that any guarded input in $\mathcal{P}$ is preceded by a reset. Let $\mathcal{K}' = (\mathcal{P}'; \Phi'; t')$ be a configuration such that $\mathcal{K} \xrightarrow{\mathrm{tr}}_{\mathcal{T}} \mathcal{K}'$. We have that $\mathcal{K}_0 = (\mathcal{P}; \Phi; 0) \xrightarrow{\mathrm{tr}}_{\mathcal{T}} \mathcal{K}'_0$ for some $\mathcal{K}'_0 = (\mathcal{P}'_0; \Phi'_0; t'_0)$ such that:*

(1) $\{(P, a) \mid \lfloor P \rfloor^{-}_{a} \in \mathcal{P}'\} = \{(P, a) \mid \lfloor P \rfloor^{-}_{a} \in \mathcal{P}'_0\}$;
(2) $\{(w, a, u) \mid w \xrightarrow{a,\_} u \in \Phi'\} = \{(w, a, u) \mid w \xrightarrow{a,\_} u \in \Phi'_0\}$.

Proof. We prove the result together with the three properties below by induction on the length of the derivation $\mathcal{K} \xrightarrow{\mathrm{tr}}_{\mathcal{T}} \mathcal{K}' = (\mathcal{P}'; \Phi \uplus \Psi'; t')$ where $\delta = \max(\{\mathsf{Dist}_{\mathcal{T}}(a, b) \mid a, b \in \mathcal{A}_0\} \cup \{t\})$.

(i) if $\lfloor P \rfloor^{t_a}_{a} \in \mathcal{P}'$ and P contains a guarded input that is not preceded by a reset then $\lfloor P \rfloor^{t_a}_{a} \in \mathcal{P}'_0$ (with the same value for t_a);
(ii) $t'_0 = t' - t + \delta$;
(iii) $\Phi'_0 = \Phi \uplus \mathsf{Shift}(\Psi', \delta - t)$ with $\mathsf{Shift}(\Psi', \delta - t) = \{w \xrightarrow{a, t_a + \delta - t} u \mid w \xrightarrow{a, t_a} u \in \Psi'\}$.

Base case: In such a case, we have that $\mathcal{K}' = \mathcal{K}$, and thus $t' = t$, $\mathcal{P}' = \mathcal{P}$, $\Phi' = \Phi$, and $\Psi' = \emptyset$. Let $\mathcal{K}'_0 = (\mathsf{Shift}(\mathcal{P}, \delta); \Phi; \delta)$. We have that $\mathcal{K}_0 \rightarrow_{\mathcal{T}} \mathcal{K}'_0$ using the TIM rule. Note that item (i) is satisfied since by hypothesis in $\mathcal{P}'$ (= $\mathcal{P}$), any guarded input is preceded by a reset. Regarding (ii), we have that $t'_0 = \delta = t' - t + \delta$ since $t' = t$. Since $\Phi'_0 = \Phi$, and $\Psi' = \emptyset$, item (iii) is also satisfied. Then items (1) and (2) are trivially satisfied.

Induction step: In such a case, we have that $\mathcal{K} \xrightarrow{\mathrm{tr}}_{\mathcal{T}} \mathcal{K}'' \xrightarrow{a,\alpha}_{\mathcal{T}} \mathcal{K}'$ with $\mathcal{K}'' = (\mathcal{P}''; \Phi''; t'')$ and $\Phi'' = \Phi \uplus \Psi''$. By induction hypothesis, we know that there exists $\mathcal{K}''_0 = (\mathcal{P}''_0; \Phi''_0; t''_0)$ such that $\mathcal{K}_0 \xrightarrow{\mathrm{tr}}_{\mathcal{T}} \mathcal{K}''_0$ with:

(1) $\{(P, a) \mid \lfloor P \rfloor^{-}_{a} \in \mathcal{P}''\} = \{(P, a) \mid \lfloor P \rfloor^{-}_{a} \in \mathcal{P}''_0\}$;
(2) $\{(w, a, u) \mid w \xrightarrow{a,\_} u \in \Phi''\} = \{(w, a, u) \mid w \xrightarrow{a,\_} u \in \Phi''_0\}$.

We have also that:

(i) if $\lfloor P \rfloor^{t_a}_{a} \in \mathcal{P}''$ and P contains a guarded input that is not preceded by a reset then $\lfloor P \rfloor^{t_a}_{a} \in \mathcal{P}''_0$;
(ii) $t''_0 = t'' - t + \delta$;
(iii) $\Phi''_0 = \Phi \uplus \mathsf{Shift}(\Psi'', \delta - t)$.

We consider the rule involved in $\mathcal{K}'' \xrightarrow{a,\alpha}_{\mathcal{T}} \mathcal{K}'$ and we show that the same rule can be applied on $\mathcal{K}''_0$, and allows one to get $\mathcal{K}'_0$ with the five properties stated above.

Case TIM rule. In such a case, we have that $\mathcal{K}' = (\mathcal{P}''; \Phi''; t'' + \delta_0)$ for some δ_0, and we apply the same rule with the delay δ_0 on $\mathcal{K}''_0$. We obtain $\mathcal{K}'_0 = (\mathcal{P}''_0; \Phi''_0; t''_0 + \delta_0)$, and we easily check that all the properties are satisfied.

Case OUT rule. In such a case, $\mathcal{K}' = (\mathcal{P}'; \Phi'' \uplus \{w \xrightarrow{a,t''} u\}; t'')$ (for some $\mathcal{P}'$, w, a, and u). Relying on item 1, we apply the same rule on $\mathcal{K}''_0$, and obtain $\mathcal{K}'_0 = (\mathcal{P}'_0; \Phi''_0 \uplus \{w \xrightarrow{a,t''_0} u\}; t''_0)$ (for some $\mathcal{P}'_0$). We have that items 1 and 2 are clearly satisfied as well as item (i). Now, regarding item (ii), we have that $t'_0 = t''_0$ and $t' = t''$. Therefore, we conclude that $t'_0 = t' - t + \delta$ thanks to our induction hypothesis. Now, to establish that $\Phi'_0 = \Phi \uplus \mathsf{Shift}(\Psi', \delta - t)$, relying on our induction hypothesis, it only remains to show that $t''_0 = t'' + (\delta - t)$ (this is item (ii)).

Case LET and NEW rules. In such a case, $\mathcal{K}' = (\mathcal{P}'; \Phi''; t'')$ (for some $\mathcal{P}'$), and we apply that same rule on $\mathcal{K}_0''$, and obtain $\mathcal{K}_0' = (\mathcal{P}_0'; \Phi_0''; t_0'')$. We conclude easily (for each property) relying on our induction hypothesis.

Case RST rule. In such a case, $\mathcal{K}' = (\mathcal{P}'; \Phi''; t'')$ with $\mathcal{P}' = \{ \lfloor P \rfloor_a^0 \} \cup \mathcal{Q}'$ for some P, a, t^a and $\mathcal{Q}'$, and $\mathcal{P}'' = \{ \lfloor \mathsf{reset}.P \rfloor_a^{t^a} \} \cup \mathcal{Q}'$. We apply the same rule on $\mathcal{K}_0''$, and obtain $\mathcal{K}_0' = (\mathcal{P}_0'; \Phi_0''; t_0'')$. Relying on our induction hypothesis, we easily obtain the fact that items 1 and 2 are satisfied. Regarding item (i), we conclude using our induction hypothesis for processes in $\mathcal{Q}'$, and the property is satisfied for $\lfloor P \rfloor_a^0$. Regarding items (ii) and (iii), since the global time and the frame have not evolved, we conclude relying on our induction hypothesis.

Case IN rule. In such a case, $\mathcal{K}' = (\mathcal{P}'; \Phi''; t'')$ (for some $\mathcal{P}'$), and we apply the same rule on $\mathcal{K}_0''$ to get $\mathcal{K}_0' = (\mathcal{P}_0'; \Phi_0''; t_0'')$. The difficult part is to show that the rule can indeed be applied on $\mathcal{K}_0''$. By hypothesis, we know that $\mathcal{K}'' \xrightarrow{a, \mathtt{in}^\star(u)}_{\mathcal{T}} \mathcal{K}'$, and thus $\mathcal{P}'' = \lfloor \mathtt{in}^\star(x).P \rfloor_a^{t^a} \cup \mathcal{Q}$ for some x, a, t^a and $\mathcal{Q}$, and we know that there exists $b \in \mathcal{A}_0$ and $t^b \in \mathbb{R}_+$ such that $t^b < t'' - \mathsf{Dist}_{\mathcal{T}}(b, a)$ 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 $w \in vars(R)$ there exists $c \in \mathcal{A}_0$ such that $w \in dom(\lfloor \Phi'' \rfloor_c^{t^b - \mathsf{Dist}_{\mathcal{T}}(c,b)})$.

Moreover, in case $\star$ is $< t_g$ for some t_g, we know in addition that $t^a < t_g$.

We first assume that $\mathtt{in}^\star(u)$ is a simple input (not a guarded one). We show that we can apply on $\mathcal{K}_0''$ the same rule using the same recipe R. The message will be sent by the same agent $b \in \mathcal{A}_0$. The time t_0^b at which the message is sent is $t_0^b = t^b + (\delta - t)$. Note that $t_0^b \geq t^b$ since $\delta - t \geq 0$, and $img(\lfloor \Psi'' \rfloor_b^{t^b}) = img(\lfloor \Psi_0'' \rfloor_b^{t_0^b})$ (and $dom(\lfloor \Psi'' \rfloor_b^{t^b}) = dom(\lfloor \Psi_0'' \rfloor_b^{t_0^b})$ as well) since $\Psi_0'' = \mathsf{Shift}(\Psi'', \delta - t)$, and $t_0^b = t^b + (\delta - t)$. We distinguish two cases:

- if $b \in \mathcal{A}_0 \setminus \mathcal{M}_0$ then we know that

$$\begin{array}{rcl} u & \in & img(\lfloor \Phi'' \rfloor_b^{t^b}) \\ & = & img(\lfloor \Phi \rfloor_b^{t^b}) \cup img(\lfloor \Psi'' \rfloor_b^{t^b}) \\ & \subseteq & img(\lfloor \Phi \rfloor_b^{t_0^b}) \cup img(\lfloor \Psi_0'' \rfloor_b^{t_0^b}) \\ & = & img(\lfloor \Phi_0'' \rfloor_b^{t_0^b}) \end{array}$$

- if $b \in \mathcal{M}_0$ then we consider $w \in vars(R)$. We have that:

$$\begin{array}{rcl} w & \in & dom(\lfloor \Phi'' \rfloor_c^{t^b - \mathsf{Dist}_{\mathcal{T}}(c,b)}) \\ & = & dom(\lfloor \Phi \rfloor_c^{t^b - \mathsf{Dist}_{\mathcal{T}}(c,b)}) \cup dom(\lfloor \Psi'' \rfloor_c^{t^b - \mathsf{Dist}_{\mathcal{T}}(c,b)}) \\ & \subseteq & dom(\lfloor \Phi \rfloor_c^{t_0^b - \mathsf{Dist}_{\mathcal{T}}(c,b)}) \cup dom(\lfloor \Psi_0'' \rfloor_c^{t_0^b - \mathsf{Dist}_{\mathcal{T}}(c,b)}) \\ & = & dom(\lfloor \Phi'' \rfloor_c^{t_0^b - \mathsf{Dist}_{\mathcal{T}}(c,b)}) \end{array}$$

Thus, in both cases, this allows us to conclude. Now, in case $\star$ is $< t_g$, by hypothesis we know that $t^a < t_g$, and thanks to item (i), we know that $\mathcal{P}_0'' = \lfloor \mathtt{in}^\star(x).P \rfloor_a^{t^a} \cup \mathcal{Q}_0$. This allows us to conclude. □

Theorem 1. *Let $\mathcal{I}_0 = (\mathcal{I}_0^V, \mathcal{I}_0^P)$ be a template, (V, P) a protocol, $t_0 \in \mathbb{R}_+$ a threshold, and Φ_0 an initial frame such that $names(\Phi_0) \subseteq \{v_0, p_0\}$. There exists a topology $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0) \in \mathcal{C}_{\mathsf{MF}}^{t_0}$ and a valid initial configuration $\mathcal{K}$ for (V, P) w.r.t. $\mathcal{T}_0$ and $\Phi_{\mathcal{I}_0}^{\mathcal{T}_0} \cup \Phi_0$ such that*

$$\mathcal{K} \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathcal{P}; \Phi; t)$$

if, and only if, there exists a valid initial configuration $\mathcal{K}'$ for (V, P) w.r.t. $\Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}} \cup \Phi_0$ such that

$$\mathcal{K}' \xrightarrow{\mathsf{tr}'}_{\mathcal{T}_{\mathsf{MF}}^{t_0}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_v} \uplus \mathcal{P}'; \Phi'; t').$$

Proof. Since $\mathcal{T}_{\mathsf{MF}}^{t_0} \in \mathcal{C}_{\mathsf{MF}}^{t_0}$, the implication from right to left is easy to prove. Thus, we consider the other one ($\Rightarrow$). Let $\mathcal{T}_0 = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0) \in \mathcal{C}_{\mathsf{MF}}^{t_0}$ and $\mathcal{K}_0 = (\mathcal{P}_0; \Phi_{\mathcal{I}_0}^{\mathcal{T}_0} \cup \Phi_0; 0)$ be a valid initial configuration for (V, P) w.r.t. $\mathcal{T}_0$ and $\Phi_{\mathcal{I}_0}^{\mathcal{T}_0} \cup \Phi_0$ such that

$$\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} (\lfloor \mathsf{end}(v_0, p_0).P \rfloor_{v_0}^{t_a} \uplus \mathcal{P}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_0} \cup \Phi_0 \cup \Phi; t).$$

To establish this result, we combine the previous lemmas to show that there exists a corresponding trace of execution in $\mathcal{T}_{\mathsf{MF}}^{t_0}$. As already announced, the proof is performed in four steps.

Step 1. Applying Lemma 1 with $\mathcal{H} = \mathcal{A}_0 \setminus (\mathcal{M}_0 \cup \{v_0, p_0\})$, we obtain a topology $\mathcal{T}_1 = (\mathcal{A}_1, \mathcal{M}_1, \mathsf{Loc}_1, v_0, p_0)$ where $\mathcal{A}_1 = \mathcal{A}_0$, $\mathcal{M}_1 = \mathcal{A}_0 \setminus \{v_0, p_0\}$, and $\mathsf{Loc}_1 = \mathsf{Loc}_0$. We have that:

$$\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_1} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_a} \uplus \mathcal{P}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_0} \cup \Phi_0 \cup \Phi; t).$$

We now consider the configuration $\mathcal{K}_0^+ = (\mathcal{P}_0; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1} \cup \Phi_0; 0)$. Since $\Phi_{\mathcal{I}_0}^{\mathcal{T}_0} \subseteq \Phi_{\mathcal{I}_0}^{\mathcal{T}_1}$ (indeed by adding malicious agents, we have only increased the knowledge of the attacker) and thus we have that:

$$\mathcal{K}_0^+ \xrightarrow{\mathsf{tr}}_{\mathcal{T}_1} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_a} \uplus \mathcal{P}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1} \cup \Phi_0 \cup \Phi; t).$$

Step 2. Since the configuration $\mathcal{K}_0$ is valid, we know that for all $\lfloor Q \rfloor_{a_0}^{t_a} \in \mathcal{P}_0$ (but $\mathsf{V}_{\mathsf{end}}(v_0, p_0)$), either $Q = \mathsf{V}(a_0, a_1)$ or $Q = \mathsf{P}(a_0, a_1)$ for some $a_1 \in \mathcal{A}_0 = \mathcal{A}_1$. We assume w.l.o.g. that we are in the first case. Let $\tau = \{z_V^0 \mapsto a_0, z_V^1 \mapsto a_1\}$. We know that for any term u occurring in an output or a let construction in Q, there exists a corresponding term u' occurring in an output or a let construction in V such that $u = u'\tau$. Let $\mathcal{I}_V^0 = \{v_1, \ldots, v_k\}$, and $\sigma = \{w_1 \mapsto v_1, \ldots, w_k \mapsto v_k\}$. Since V is $\mathcal{I}_V^0$-executable by definition of a protocol, there exists a term $R \in \mathcal{T}(\Sigma_{\mathsf{pub}}^+, \{w_1, \ldots, w_k\} \cup bn(\mathsf{V}) \cup bv(\mathsf{V}))$ such that $u' = R\sigma\!\downarrow$. Thus, we know that $u\!\downarrow = u'\tau\!\downarrow = R\sigma\!\downarrow\tau\!\downarrow = R\sigma\tau\!\downarrow$. Actually, we have that $\{v_1\tau, \ldots, v_k\tau\} \subseteq \mathsf{Knows}(\mathcal{I}_0, a_0, \mathcal{A}_1)$. Therefore if $a_0 \in \mathcal{M}_1$, since $\mathsf{Knows}(\mathcal{I}_0, a_0, \mathcal{A}_1) \subseteq img(\lfloor \Phi_{\mathcal{I}_0}^{\mathcal{T}_1} \rfloor_{a_0}^0)$, we have that Q is executable w.r.t. $img(\Phi_{\mathcal{I}_0}^{\mathcal{T}_1} \cup \Phi_0)$.

Therefore, we can apply Lemma 2 with $\mathcal{K}_0^+ = (\mathcal{P}_0; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1} \cup \Phi_0; 0)$ and $\mathcal{D}_0 = \mathcal{M}_1$ and conclude that:

$$(\overline{\mathcal{P}_0}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1} \cup \Phi_0; 0) \xrightarrow{\overline{\mathsf{tr}}\sigma_2}_{\mathcal{T}_1} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_a} \uplus \overline{\mathcal{P}}\sigma_2; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1} \cup \Phi_0 \cup \overline{\Phi}\sigma_2; t)$$

where $\sigma_2(n) = \mathsf{c}_0 \in \Sigma_0$ for any name n freshly generated by an agent in $\mathcal{M}_1$ and $\overline{\mathcal{P}}$ (resp. $\overline{\Phi}$, and $\overline{\mathsf{tr}}$) is obtained from $\mathcal{P}$ (resp. Φ, tr) by removing processes (frame elements, actions) located in $a \in \mathcal{D}_0 = \mathcal{M}_1$.

Step 3. We now consider $\Phi_{\mathcal{I}_0}^{\mathcal{T}_1+}$ the same as $\Phi_{\mathcal{I}_0}^{\mathcal{T}_1}$ but frame elements located at $a \in \mathcal{M}_1 \setminus \{v_0, p_0, i_2\}$ are moved to v_0 and those located at i_2 are moved to p_0. Let $\delta_0 = \max(\mathsf{Dist}_{\mathcal{T}_1}(a, b))$ for any $a, b \in \mathcal{A}_1$. Clearly, we have that:

$$(\overline{\mathcal{P}_0}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1+} \cup \Phi_0; \delta_0) \xrightarrow{\overline{\mathsf{tr}}\sigma_2}_{\mathcal{T}_1} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_a} \uplus \overline{\mathcal{P}}\sigma_2; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1+} \cup \Phi_0 \cup \mathsf{Shift}(\overline{\Phi}\sigma_2, \delta_0); t + \delta_0).$$

Indeed, adding a delay δ_0 in the initial configuration, all the messages moved from an agent $a \in \mathcal{M}_1$ to v_0 or p_0 can be used by a at the time δ_0 and thus the initial execution can be followed shifted by δ_0. Applying Proposition 5, we can turn the first configuration into an initial one, i.e. with a global time set to 0. Therefore, we have that:

$$(\overline{\mathcal{P}_0}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1+} \cup \Phi_0; 0) \xrightarrow{\overline{\mathsf{tr}}\sigma_2}_{\mathcal{T}_1} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_a} \uplus \mathcal{P}'; \Phi'; t')$$

where $\mathcal{P}'$ (resp. Φ') coincides with $\overline{\mathcal{P}}\sigma_2$ (resp. $\Phi_{\mathcal{I}_0}^{\mathcal{T}_1+} \cup \Phi_0 \cup \mathsf{Shift}(\overline{\Phi}\sigma_2, \delta_0)$) up to local clocks (resp. timestamps). Then, we apply Lemma 3 on $\mathcal{T}_1$ and $(\overline{\mathcal{P}_0}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1+} \cup \Phi_0; 0)$ with $H = \{v_0, p_0\}$. We deduce that

$$(\overline{\mathcal{P}_0}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1+} \cup \Phi_0; 0) \xrightarrow{\overline{\mathsf{tr}}\sigma_2}_{\mathcal{T}_2} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_a} \uplus \mathcal{P}'; \Phi'; t').$$

where $\mathcal{T}_2 = (\{v_0, p_0, i_1, i_2\}, \{i_1, i_2\}, \mathsf{Loc}_H, v_0, p_0)$ is the canonical topology associated to H and $\mathsf{Loc}|_H$.

Step 4. To reduce the size of the initial frame, now we apply Lemma 4 on the configuration $\mathcal{K}_0' = (\overline{\mathcal{P}_0}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1+} \cup \Phi_0; 0)$ using $\rho : \mathcal{A} \to \mathcal{A}_1' = \{v_0, p_0, i_1, i_2\}$ such that $\rho(a) = i_1$ for any $a \notin \mathcal{A}_1'$, and $\rho(a) = a$ otherwise. We have that:

$$(\overline{\mathcal{P}_0}\rho; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1+}\rho \cup \Phi_0\rho; 0) \xrightarrow{\overline{\mathsf{tr}}\sigma_2\rho}_{\mathcal{T}_2} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t_a} \uplus \mathcal{P}'\rho; \Phi'\rho; t').$$

One may note that we can shorten the distance between v_0, i_1 and p_0, i_2 and keep the trace executable. The resulting topology, i.e. $(\mathcal{A}_1', \{i_1, i_2\}, \mathsf{Loc}, v_0, p_0)$ with $\mathsf{Loc}(v_0) = \mathsf{Loc}(i_1)$, $\mathsf{Loc}(p_0) = \mathsf{Loc}(i_2)$ and $\mathsf{Dist}_{\mathcal{T}}(v_0, p_0) = t_0$, corresponds to $\mathcal{T}_{\mathsf{MF}}^{t_0}$.

To end this proof, it remains to turn $(\overline{\mathcal{P}_0}\rho; \Phi_{\mathcal{I}_0}^{\mathcal{T}_1+}\rho \cup \Phi_0\rho; 0)$ into a valid initial configuration. To do that, we simply have to move frame elements (those that we have added during Step 1 and moved during Step 3) located in v_0 to i_1 and those located in p_0 to i_2. This will not change the underlying execution since i_1 (resp. i_2) is located at the same place as v_0 (resp. p_0). In case there is no frame elements in i_1 or i_2 we add some elements, more precisely we add $\mathsf{Knows}(\mathcal{I}_0, i_1, \mathcal{A}_1')$ or $\mathsf{Knows}(\mathcal{I}_0, i_2, \mathcal{A}_1')$. These additional elements will not alter the underlying execution and the resulting frame corresponds to $\Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}}$ up to possibly many replicates of the initial knowledge of agent i_1.

We may note that the process part $\overline{\mathcal{P}_0}\rho$ of the configuration satisfies Definition 3. Indeed, since $\mathcal{K}_0$ is valid, we have that $\lfloor \mathsf{V}_{\mathsf{end}}(v_0, p_0) \rfloor_{v_0}^{0} \in \mathcal{P}_0$ and by construction $\overline{\mathcal{P}_0}\rho$ still contains $\lfloor \mathsf{V}_{\mathsf{end}}(v_0, p_0) \rfloor_{v_0}^{0}$. Moreover, for all $\lfloor P' \rfloor_{a'}^{t'} \in \overline{\mathcal{P}_0}\rho$, by construction, we know that $a' \in \{v_0, p_0\} = \mathcal{A}_1' \setminus \mathcal{M}_1'$. Finally, we have that there exists $\lfloor P'' \rfloor_{a'}^{t'} \in \mathcal{P}_0$ such that $P'' = \mathsf{V}(a', a_1)$ (resp. $P'' = \mathsf{P}(a', a_1)$) and $P' = P''\rho = \mathsf{V}(a', \rho(a_1))$ (resp. $P' = P''\rho = \mathsf{P}(a', \rho(a_1))$). with $\rho(a_1) \in \mathcal{A}_1'$. Therefore, we have that $(\overline{\mathcal{P}_0}\rho; \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}} \cup \Phi_0; 0) = (\overline{\mathcal{P}_0}\rho; \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}} \cup \Phi_0\rho; 0)$ is a valid initial configuration for (V, P) w.r.t. $\mathcal{T}_{\mathsf{MF}}^{t_0}$.

This concludes the proof. □

Corollary 2. *Let $\mathcal{I}_0$ be a template, (V, P) a protocol, and $t_0 \in \mathbb{R}_+$ a threshold. We have that (V, P) is Terrorist fraud resistant w.r.t. t_0-proximity if, and only if, for all semi-dishonest prover P_{sd} w.r.t. t_0 with frame Φ_{sd} such that $names(\Phi_{\mathsf{sd}}) \subseteq \{v_0, p_0\}$, there exists a valid initial configuration $\mathcal{K}_0$ w.r.t. $\mathcal{T}_{\mathsf{MF}}^{t_0}$ and $\Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}} \cup \Phi_{\mathsf{sd}}$ such that $\mathcal{K}_0 \to^*_{\mathcal{T}_{\mathsf{MF}}^{t_0}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{t'} \uplus \mathcal{P}; \Phi; t)$.*

Proof. The implication from right to left is almost trivial. We consider a semi-dishonest prover P_{sd} with frame Φ_{sd}. If $names(\Phi_{\mathsf{sd}}) \subseteq \{v_0, p_0\}$, then we choose the topology $\mathcal{T}_{\mathsf{MF}}^{t_0} \in \mathcal{C}_{\mathsf{MF}}^{t_0}$ and we conclude relying on our hypothesis. Otherwise, we consider a bijective renaming σ from agent names outside $\{v_0, p_p\}$ to fresh constants in Σ_0 (i.e. never used in (V, P)) and it is easy to see that $P_{\mathsf{sd}}\sigma$ with associated frame $\Phi_{\mathsf{sd}}\sigma$ is also a semi-dishonest prover. Now, we can rely on our hypothesis to obtain an execution trace in $\mathcal{T}_{\mathsf{MF}}^{t_0}$ (starting with $\Phi_{\mathsf{sd}}\sigma$) and applying σ^{-1} on it allows us to conclude.

Regarding the other implication, we consider a semi-dishonest prover P_{sd} with frame Φ_{sd} such that $names(\Phi_{\mathsf{sd}}) \subseteq \{v_0, p_0\}$. By hypothesis, there exists a valid initial configuration $\mathcal{K}_0$ for (V, P)

w.r.t. $\mathcal{T} \in \mathcal{C}_{\mathsf{MF}}$ and $\Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi_{\mathsf{sd}}$ such that:

$$\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor^{t_v}_{v_0} \uplus \mathcal{P}; \Phi; t) \text{ with } \mathsf{Dist}_{\mathcal{T}}(v_0, p_0) \geq t_0.$$

Theorem 1 applies since $names(\Phi_{\mathsf{sd}}) \subseteq \{v_0, p_0\}$, and thus we easily conclude. □

B PROOFS FOR SECTION 5.2

Lemma 5. *Let $\mathcal{T}$ be a topology, $\mathcal{K}_0 \xrightarrow{L_1 \ldots . L_n}_{\mathcal{T}} \mathcal{K}_1$ be an execution, and $i, j \in \{1, \ldots, n\}$ be such that $L_j \hookrightarrow^* L_i$. We have that $t_j \geq t_i + Dist_{\mathcal{T}}(a_i, a_j)$ where $L_i = (a_i, \alpha_i, s_i, t_i, r_i)$ and $L_j = (a_j, \alpha_j, s_j, t_j, r_j)$.*

Proof. By definition of $\hookrightarrow^*$, there exists $k \geq 1$ and $i_1, \ldots, i_k \in \{1, \ldots, n\}$ such that:

$$L_j = L_{i_1} \hookrightarrow L_{i_2} \hookrightarrow \ldots \hookrightarrow L_{i_k} = L_i.$$

We do the proof by induction on the length of this sequence. If $k = 1$ then $L_i = L_j$, and thus $t_i = t_j$ and $a_i = a_j$. The result trivially holds. Otherwise, relying on the induction hypothesis, we deduce that $t_{i_2} \geq t_i + \mathsf{Dist}_{\mathcal{T}}(a_i, a_{i_2})$. We distinguish two cases depending on the nature of the dependency $L_j \hookrightarrow L_{i_2}$.

Case $L_j \hookrightarrow_s L_{i_2}$. In such a case, we have that $a_j = a_{i_2}$ and $j \geq i_2$, and thus $t_j \geq t_{i_2}$. Therefore, we have:

$$t_j \geq t_{i_2} \geq t_i + \mathsf{Dist}_{\mathcal{T}}(a_i, a_{i_2}) = t_i + \mathsf{Dist}_{\mathcal{T}}(a_i, a_j).$$

Case $L_j \hookrightarrow_d L_{i_2}$. In such a case, we have that $r_j = (b, t_b, R)$, $r_{i_2} = w$ and $w \in vars(R)$. By definition of the IN rule we have that $t_j \geq t_b + \mathsf{Dist}_{\mathcal{T}}(b, a_j)$, and $t_b \geq t_{i_2} + \mathsf{Dist}_{\mathcal{T}}(a_{i_2}, b)$. Combining these inequalities with the one given by our induction hypothesis, we get:

$$t_{i_2} + t_j + t_b \geq t_i + t_b + t_{i_2} + \mathsf{Dist}_{\mathcal{T}}(a_i, a_{i_2}) + \mathsf{Dist}_{\mathcal{T}}(b, a_j) + \mathsf{Dist}_{\mathcal{T}}(a_{i_2}, b).$$

Relying on the triangle inequality, we get $t_j \geq t_i + \mathsf{Dist}_{\mathcal{T}}(a_i, a_j)$, and we conclude the proof. □

Lemma 6. *Let $\mathcal{T}$ be a topology, and $\mathcal{K}_0 \overset{L_1}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K} \overset{L_2}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_2$ be an execution such that $L_2 \not\hookrightarrow L_1$. We have that $\mathcal{K}_0 \overset{L_2}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}' \overset{L_1}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_2$ for some configuration $\mathcal{K}'$.*

Proof. Let $\mathcal{K}_0 = (\mathcal{P}_0; \Phi_0)$, $\mathcal{K} = (\mathcal{P}; \Phi)$, and $\mathcal{K}_2 = (\mathcal{P}_2; \Phi_2)$ be such that $\mathcal{K}_0 \overset{L_1}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K} \overset{L_2}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_2$ with $L_2 \not\hookrightarrow L_1$. Let $L_1 = (a_1, \alpha_1, s_1, r_1)$ and $L_2 = (a_2, \alpha_2, s_2, r_2)$. Since $L_2 \not\hookrightarrow L_1$ we have that $s_1 \neq s_2$. Therefore, we have that $\mathcal{P}_0 = \lfloor \mathsf{act}_1.P_1 \rfloor_{a_1} \cup \lfloor \mathsf{act}_2.P_2 \rfloor_{a_2} \uplus \mathcal{Q}$, $\mathcal{P} = \lfloor P'_1 \rfloor_{a_1} \cup \lfloor \mathsf{act}_2.P_2 \rfloor_{\mathsf{act}_2} \uplus \mathcal{Q}$, and $\mathcal{P}_2 = \lfloor P'_1 \rfloor_{a_1} \cup \lfloor P'_2 \rfloor_{a_2} \uplus \mathcal{Q}$ for some actions act_1 and act_2.

In case $\alpha_1 = \mathtt{out}(v)$ and $\alpha_2 = \mathtt{in}^\star(u)$, we have that $\Phi_2 = \Phi = \Phi_0 \uplus \{w \xrightarrow{a_1} v\}$. Since $L_2 \not\hookrightarrow_d L_1$, we know that $w \notin vars(r_2)$, and thus $vars(r_2) \subseteq dom(\Phi_0)$. Now, let $\mathcal{K}' = (\lfloor \mathsf{act}_1.P_1 \rfloor_{a_1} \uplus \lfloor P'_2 \rfloor_{a_2} \uplus \mathcal{Q}; \Phi_0)$. Relying on the fact that $w \notin vars(r_2)$ in case $\alpha_1 = \mathtt{out}(v)$ and $\alpha_2 = \mathtt{in}^\star(u)$, it is easy to see that $\mathcal{K}_0 \overset{L_2}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}' \overset{L_1}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_2$. The other cases can be done in a rather similar way. □

Proposition 2. *Let $\mathcal{T}$ be a topology, and $\mathcal{K}_0 \overset{\mathsf{tr}_1 \ldots \mathsf{tr}_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$ be an execution with $n \geq 2$. We have that there exists a bijection $\varphi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that:*

- *$\mathcal{K}_0 \overset{\mathsf{tr}'_1 \ldots \mathsf{tr}'_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$ with $\mathsf{tr}_i = \mathsf{tr}'_{\varphi(i)}$ for all $i \in \{1, \ldots, n\}$; and*
- *for all j such that $\varphi(1) < j < \varphi(n)$, we have that $\mathsf{tr}'_{\varphi(n)} \hookrightarrow^* \mathsf{tr}'_j \hookrightarrow^* \mathsf{tr}'_{\varphi(1)}$.*

Proof. We split the proof in two parts: first we prove that there exists a bijective function φ_1 cleaning the trace between tr_1 and tr_n moving actions independent from tr_1 before it. Then we prove that there exists a bijective function φ_2 cleaning the trace moving actions from which tr_n does not depend on after it. Considering $\varphi = \varphi_2 \circ \varphi_1$ we will be able to conclude.

Claim 1. *Let $\mathcal{K}_0 \overset{\mathsf{tr}_1 \ldots \mathsf{tr}_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$ be an execution with $n \geq 1$. There exists a bijective function $\varphi_1 : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that:*

- *$\mathcal{K}_0 \overset{\mathsf{tr}'_1 \ldots \mathsf{tr}'_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$ with $\mathsf{tr}_i = \mathsf{tr}'_{\varphi_1(i)}$ for all $i \in \{1, \ldots, n\}$; and*
- *for all $j > \varphi_1(1)$, $\mathsf{tr}'_j \hookrightarrow^* \mathsf{tr}'_{\varphi_1(1)}$.*

We show this claim by induction on the length of the execution. If $n = 1$ then the results holds considering $\varphi_1 = id$ and $\mathsf{tr}'_1 = \mathsf{tr}_1$. Otherwise, we have that $\mathcal{K}_0 \overset{\mathsf{tr}_1 \ldots \mathsf{tr}_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n \overset{\mathsf{tr}_{n+1}}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_{n+1}$, and by induction hypothesis we have that there exists a bijective function $\varphi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that:

- $\mathcal{K}_0 \overset{\mathsf{tr}'_1 \ldots \mathsf{tr}'_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$ with $\mathsf{tr}_i = \mathsf{tr}'_{\varphi(i)}$ for all $i \in \{1, \ldots, n\}$; and
- for all j such that $\varphi(1) < j \leq n$, we have $\mathsf{tr}'_j \hookrightarrow^* \mathsf{tr}'_{\varphi(1)}$.

If $\mathsf{tr}_{n+1} \hookrightarrow^* \mathsf{tr}'_{\varphi(1)} (= \mathsf{tr}_1)$ then we consider the bijective function $\varphi_1 = \varphi \cup \{n+1 \mapsto n+1\}$ and this allows us to conclude. Otherwise, we have that $\mathsf{tr}_{n+1} \not\hookrightarrow^* \mathsf{tr}'_{\varphi(1)} (= \mathsf{tr}_1)$ and, by induction hypothesis, we have $\mathsf{tr}_{n+1} \not\hookrightarrow^* \mathsf{tr}'_j$ for any $\varphi(1) < j \leq n$. Repeatedly applying Lemma 6 we obtain that

$$\mathcal{K}_0 \overset{\mathsf{tr}'_1 \ldots \mathsf{tr}_{n+1} \mathsf{tr}_{\varphi(1)} \ldots \mathsf{tr}'_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_{n+1}.$$

Considering the bijective function φ_1 defined as follows:

$$\varphi_1(i) = \begin{cases} \varphi(i) & \text{if } \varphi(i) < \varphi(1) \\ \varphi(i) + 1 & \text{if } \varphi(i) \geq \varphi(1) \\ \varphi(1) & \text{if } i = n+1 \end{cases}$$

we prove the claim.

Claim 2. *Let $\mathcal{K}_0 \overset{\mathsf{tr}_1 \ldots \mathsf{tr}_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$ be an execution with $n \geq 1$. There exists a bijective function $\varphi_2 : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that:*

- *$\mathcal{K}_0 \overset{\mathsf{tr}'_1 \ldots \mathsf{tr}'_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$ with $\mathsf{tr}_i = \mathsf{tr}'_{\varphi_2(i)}$ for all $i \in \{1, \ldots, n\}$; and*
- *for all $j < \varphi_2(n)$, we have that $\mathsf{tr}'_{\varphi_2(n)} \hookrightarrow^* \mathsf{tr}'_j$.*

Similarly to the proof done for the previous claim, we first apply the induction hypothesis to $\mathcal{K}_1 \overset{\mathsf{tr}_2 \ldots \mathsf{tr}_{n+1}}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_{n+1}$ and we obtain $\varphi : \{2, \ldots, n\} \to \{2, \ldots, n\}$ (a shift of 1 has been applied to ease the reasoning). If $\mathsf{tr}_1 \hookrightarrow^* \mathsf{tr}'_{\varphi_2(n)} (= \mathsf{tr}_n)$ then we conclude considering $\varphi_2 = \varphi \cup \{1 \mapsto 1\}$. Otherwise, by repeatedly applying Lemma 6, we move tr_1 at the right place in the trace, i.e. just after $\mathsf{tr}'_{\varphi_2(n)}$.

We are now able to prove the corollary combining these two claims. First we apply Claim 1 considering the trace $\mathcal{K}_0 \overset{\mathsf{tr}_1 \ldots \mathsf{tr}_{n-1}}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_{n-1}$. We obtain the existence of $\varphi_1 : \{1, \ldots, n-1\} \to \{1, \ldots, n-1\}$ and a new execution

$$\mathcal{K}_0 \overset{\mathsf{tr}'_1 \ldots \mathsf{tr}'_{n-1}}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_{n-1} \overset{\mathsf{tr}_n}{\rightsquigarrow}_{\mathcal{T}} \mathcal{K}_n$$

such that:

- $\mathsf{tr}_i = \mathsf{tr}'_{\varphi_1(i)}$ for all $i \in \{1, \ldots, n-1\}$; and
- for all j such that $\varphi_1(1) < j < n$, we have that $\mathsf{tr}'_j \hookrightarrow^* \mathsf{tr}'_{\varphi_1(1)} (= \mathsf{tr}_1)$.

For sake of uniformity, let $\mathsf{tr}'_n = \mathsf{tr}_n$, and we extend φ_1 on $\{1,\dots,n\}$ as follows: $\varphi_1(n) = n$.

Then we apply Claim 2 on the resulting execution starting at $\mathsf{tr}'_{\varphi_1(1)+1}$ to obtain that there exists a bijective function $\varphi_2 : \{\varphi_1(1)+1,\dots,n\} \to \{\varphi_1(1)+1,\dots,n\}$ and an execution

$$\mathcal{K}_0 \overset{\mathsf{tr}'_1 \dots \mathsf{tr}'_{\varphi_1(1)} \cdot \mathsf{tr}''_{\varphi_1(1)+1} \cdots \mathsf{tr}''_n}{\rightsquigarrow_{\mathcal{T}}} \mathcal{K}_n$$

such that:

- $\mathsf{tr}'_i = \mathsf{tr}''_{\varphi_2(i)}$ for all $i \in \{\varphi_1(1)+1,\dots,n\}$; and
- for all j such that $\varphi_1(1)+1 \le j < \varphi_2(n)$, we have that $\mathsf{tr}_n = \mathsf{tr}'_n = \mathsf{tr}''_{\varphi_2(n)} \hookrightarrow^* \mathsf{tr}''_j$.

For sake of uniformity, let $\mathsf{tr}''_1 \dots \mathsf{tr}''_{\varphi(1)} = \mathsf{tr}'_1 \dots \mathsf{tr}'_{\varphi(1)}$, and we extend φ_2 on $\{1,\dots,n\}$ as follows: $\varphi_2(i) = i$ for all $i \in \{1,\dots,\varphi(1)\}$.

We now show that the bijective function $\varphi = \varphi_2 \circ \varphi_1$ satisfies the requirements, i.e.:

(1) $\mathsf{tr}_i = \mathsf{tr}''_{\varphi(i)}$ for all $i \in \{1,\dots,n\}$;
(2) for all j such that $\varphi(1) < j < \varphi(n)$, we have that $\mathsf{tr}''_{\varphi(n)} \hookrightarrow^* \mathsf{tr}''_j \hookrightarrow^* \mathsf{tr}''_{\varphi(1)}$.

First, we note that:

- $\varphi(n) = \varphi_2(n)$; and
- $\varphi(i) = \varphi_1(i)$ when $\varphi_1(i) \le \varphi_1(1)$.

Now, we establish that the 2 requirements are satisfied:

(1) First, we have that $\mathsf{tr}_n = \mathsf{tr}'_n = \mathsf{tr}''_{\varphi_2(n)} = \mathsf{tr}''_{\varphi(n)}$. Otherwise, considering $i \in \{1,\dots,n-1\}$, we have that $\mathsf{tr}_i = \mathsf{tr}'_{\varphi_1(i)}$. Now, we distinguish two cases:
   - $\varphi_1(i) \le \varphi_1(1)$: we have that $\mathsf{tr}''_{\varphi_1(i)} = \mathsf{tr}'_{\varphi_1(i)}$, and thus $\mathsf{tr}_i = \mathsf{tr}''_{\varphi_1(i)} = \mathsf{tr}''_{\varphi(i)}$.
   - $\varphi_1(i) > \varphi_1(1)$: we have that $\mathsf{tr}'_{\varphi_1(i)} = \mathsf{tr}''_{\varphi_2(\varphi_1(i))} = \mathsf{tr}''_{\varphi(i)}$, and thus $\mathsf{tr}_i = \mathsf{tr}''_{\varphi(i)}$.
(2) We have shown that:
   - for all j such that $\varphi_1(1) < j < n$, we have that $\mathsf{tr}'_j \hookrightarrow^* \mathsf{tr}'_{\varphi_1(1)} = \mathsf{tr}''_{\varphi_1(1)}$, and thus for all j such that $\varphi_1(1) < j < \varphi_2(n)$, we have that $\mathsf{tr}''_j \hookrightarrow^* \mathsf{tr}'_{\varphi_1(1)} = \mathsf{tr}''_{\varphi(1)}$ since $\varphi_1(1) = \varphi(1)$ and
   $$\{\mathsf{tr}'_{\varphi_1(1)+1},\dots,\mathsf{tr}'_{n-1}\} \supseteq \{\mathsf{tr}''_{\varphi_1(1)+1},\dots,\mathsf{tr}''_{\varphi_2(n)-1}\}$$
   - for all j such that $\varphi_1(1)+1 \le j < \varphi_2(n)(=\varphi(n))$, we have that
   $$\mathsf{tr}_n = \mathsf{tr}'_n = \mathsf{tr}''_{\varphi_2(n)} \hookrightarrow^* \mathsf{tr}''_j.$$

This concludes the proof. □

Given a configuration $\mathcal{K} = (\mathcal{P};\Phi;t)$ (resp. $\mathcal{K} = (\mathcal{P};\Phi)$), we note $\phi(\mathcal{K})$ its associated frame, i.e. $\phi(\mathcal{K}) = \Phi$.

Theorem 2. *Let $\mathcal{I}_0$ be a template, (V,P) a protocol, and $t_0 \in \mathbb{R}_+$ a threshold. Moreover, we assume that $\mathsf{V}(v_0,p_0)$ is derived from the following grammar:*

$$\begin{array}{rcllllll} P & := & 0 & | & \mathtt{in}(x).P & | & \mathtt{let}\ x = v\ \mathtt{in}\ P \\ & | & \mathtt{new}\ n.P & | & \mathtt{out}(u).P & | & \mathtt{reset.out}(u').\mathtt{in}^{<t}(x).P \end{array}$$

where $x \in \mathcal{X}$, $n \in \mathcal{N}$, $u,u' \in \mathcal{T}(\Sigma_c^+, \mathcal{X} \cup \mathcal{Z} \cup \mathcal{N})$, $v \in \mathcal{T}(\Sigma^+, \mathcal{X} \cup \mathcal{Z} \cup \mathcal{N})$ and $t \le 2 \cdot t_0$. If (V,P) admits a Distance Hijacking attack w.r.t. t_0-proximity, then there exists a valid initial configuration $\mathcal{K}_0$ for (V,P) w.r.t. $\mathcal{T}_{\mathsf{DH}}^{t_0}$ and $\Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{DH}}^{t_0}}$ such that $\mathcal{K}_0 = (\mathcal{P}_0 \cup \{\lfloor \mathsf{V}_{\mathsf{end}}(v_0,p_0)\rfloor_{v_0}^0\}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{DH}}^{t_0}}; 0)$ and

$$(\overline{\mathcal{P}_0} \cup \{\lfloor \mathsf{V}_{\mathsf{end}}(v_0,p_0)\rfloor_{v_0}^0\}; \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{DH}}^{t_0}}; 0) \to^*_{\mathcal{T}_{\mathsf{DH}}^{t_0}} (\lfloor \mathsf{end}(v_0,p_0)\rfloor_{v_0}^{t_v} \uplus \mathcal{P}'; \Phi; t).$$

Proof. Let $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}, v_0, p_0) \in C_{\mathsf{DH}}$ and $\mathcal{K}_0 = (\mathcal{P}_0; \Phi^{\mathcal{T}}_{I_0}; 0)$ be a valid initial configuration for (V, P) w.r.t. $\mathcal{T}$ and $\Phi^{\mathcal{T}}_{I_0}$ such that

$$\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor^{t_v}_{v_0} \cup \mathcal{P}; \Phi; t) = \mathcal{K}_{\mathsf{end}}$$

where tr is a sequence of annotated labels.

Step 1: We first remove `reset` commands and replace guarded inputs occurring in processes other than $\mathsf{V}_{\mathsf{end}}(v_0, p_0)$ by simple inputs. Denoting $\mathcal{K}^\star_0$ (resp. $\mathcal{K}^\star_{\mathsf{end}}$), the counterpart of $\mathcal{K}_0$ (resp. $\mathcal{K}_{\mathsf{end}}$) in which `reset` commands have been removed and guarded inputs have been replaced by simple inputs but the occurrences occurring in $\mathsf{V}_{\mathsf{end}}(v_0, p_0)$, following the same trace as before, we have that:

$$\mathcal{K}^\star_0 \xrightarrow{\mathsf{tr}_1 \ldots . \mathsf{tr}_n}_{\mathcal{T}} \mathcal{K}^\star_{\mathsf{end}}.$$

Indeed, all the required conditions to trigger a simple input will be satisfied since a guarded input is like a simple input with a constraint regarding time. We denote $\mathcal{K}^\star_i$ with $i \in \{0, \ldots, n\}$ the intermediate configurations, and thus we have that $\mathcal{K}^\star_{\mathsf{end}} = \mathcal{K}^\star_n$.

Step 2: By definition of the untimed semantics we have that:

$$\widetilde{\mathcal{K}}^\star_0 \overset{\widetilde{\mathsf{tr}_1}}{\rightsquigarrow}_{\mathcal{T}} \widetilde{\mathcal{K}}^\star_1 \overset{\widetilde{\mathsf{tr}_2}}{\rightsquigarrow}_{\mathcal{T}} \ldots \overset{\widetilde{\mathsf{tr}_n}}{\rightsquigarrow}_{\mathcal{T}} \widetilde{\mathcal{K}}^\star_{\mathsf{end}} = \widetilde{\mathcal{K}}^\star_n$$

where $\widetilde{\mathcal{K}}^\star_{\mathsf{end}}$, $\widetilde{\mathcal{K}}^\star_i$, and $\widetilde{\mathsf{tr}}_i$ are the untimed counterparts of $\mathcal{K}^\star_{\mathsf{end}}$, $\mathcal{K}^\star_i$ and tr_i.

Due to the specific shape of the process $\mathsf{V}_{\mathsf{end}}(v_0, p_0)$, we know that $\mathsf{tr}_1 . \ldots . \mathsf{tr}_n$ contains subsequences $\mathsf{tr}_{i_0} . \ldots . \mathsf{tr}_{j_0}$ such that tr_{i_0} corresponds to a reset action and tr_{j_0} to a guarded input. Note that these two actions tr_{i_0} and tr_{j_0} are performed by v_0. Moreover we know that for all index i in between i_0 and j_0 we have that tr_i is not a guarded input. Applying Proposition 2 to such a subsequence we obtain that there exists a bijective function $\varphi : \{i_0, \ldots, j_0\} \to \{i_0, \ldots, j_0\}$ such that:

- $\widetilde{\mathcal{K}}^\star_{i_0 - 1} \overset{\mathsf{tr}'_{i_0} . \ldots . \mathsf{tr}'_{j_0}}{\rightsquigarrow}_{\mathcal{T}} \widetilde{\mathcal{K}}^\star_{j_0}$ with $\widetilde{\mathsf{tr}}_i = \mathsf{tr}'_{\varphi(i)}$ for all $i \in \{i_0, \ldots, j_0\}$; and
- for all j such that $\varphi(i_0) < j < \varphi(j_0)$, we have that $\mathsf{tr}'_{\varphi(j_0)} \hookrightarrow^* \mathsf{tr}'_j \hookrightarrow^* \mathsf{tr}'_{\varphi(i_0)}$.

We are now showing that any agent involved in an action between $\mathsf{tr}'_{\varphi(i_0)}$ and $\mathsf{tr}'_{\varphi(j_0)}$ in the new resulting trace is actually in the vicinity of v_0, i.e. an agent in the set:

$$\mathsf{Close}(v_0) \overset{\mathsf{def}}{=} \{a \in \mathcal{A}_0 \mid \mathsf{Dist}_{\mathcal{T}}(v_0, a) < t_0\}.$$

Let j be such that $\varphi(i_0) < j < \varphi(j_0)$. Since we have that $\mathsf{tr}'_{\varphi(j_0)} \hookrightarrow^* \mathsf{tr}'_j \hookrightarrow^* \mathsf{tr}'_{\varphi(i_0)}$, we deduce that $\widetilde{\mathsf{tr}}_{j_0} \hookrightarrow^* \widetilde{\mathsf{tr}}_{\varphi^{-1}(j)} \hookrightarrow^* \widetilde{\mathsf{tr}}_{i_0}$ and thus $\mathsf{tr}_{j_0} \hookrightarrow^* \mathsf{tr}_{\varphi^{-1}(j)} \hookrightarrow^* \mathsf{tr}_{i_0}$. Denoting $\mathsf{tr}_i = (a_i, \alpha_i, s_i, t_i, r_i)$ for all $i \in \{1, \ldots, n\}$, and applying Lemma 5 twice, we obtain that:

$$t_{j_0} \geq t_{\varphi^{-1}(j)} + \mathsf{Dist}_{\mathcal{T}}(a_{\varphi^{-1}(j)}, v_0) \text{ and } t_{\varphi^{-1}(j)} \geq t_{i_0} + \mathsf{Dist}_{\mathcal{T}}(v_0, a_{\varphi^{-1}(j)}).$$

Therefore, we have that $t_{j_0} - t_{i_0} \geq 2\mathsf{Dist}(v_0, a_{\varphi^{-1}(j)})$, and exploiting the shape of $\mathsf{V}_{\mathsf{end}}(v_0, p_0)$ we deduce that $2 \times t_0 > t_{j_0} - t_{i_0} \geq 2\mathsf{Dist}(v_0, a_{\varphi^{-1}(j)})$. In summary we have that all the actions executed between $\mathsf{tr}'_{\varphi(i_0)}$ and $\mathsf{tr}'_{\varphi(j_0)}$ are executed by agents $a \in \mathsf{Close}(v_0)$.

Similarly, for any index j such that $\varphi(i_0) < j \leq \varphi(j_0)$ and $\alpha_{\varphi^{-1}(j)} = \mathtt{in}(u)$ we have that either $b \in \mathsf{Close}(v_0)$ or $vars(R) \subseteq \phi(\widetilde{\mathcal{K}}^\star_{i_0 - 1})$ where $r_{\varphi^{-1}(j)} = (b, t_b, R)$. Indeed, assume that there exists $w \in vars(R) \setminus dom(\phi(\widetilde{\mathcal{K}}^\star_{i_0 - 1}))$. We note i the index corresponding to this output and we have that $\varphi(i_0) < i < \varphi(j_0)$. Thus, we have that:

$$\mathsf{tr}'_{\varphi(j_0)} \hookrightarrow^* \mathsf{tr}'_j \hookrightarrow_d \mathsf{tr}'_i \hookrightarrow^* \mathsf{tr}'_{\varphi(i_0)}.$$

Lemma 5 and the definition of $\hookrightarrow_d$ give us the following equations:

- $t_{j_0} - t_{\varphi^{-1}(j)} \geq \mathsf{Dist}_{\mathcal{T}}(a_{\varphi^{-1}(j)}, v_0)$,
- $t_{\varphi^{-1}(j)} - t_b \geq \mathsf{Dist}_{\mathcal{T}}(b, a_{\varphi^{-1}(j)})$,
- $t_b - t_{\varphi^{-1}(i)} \geq \mathsf{Dist}_{\mathcal{T}}(a_{\varphi^{-1}(i)}, b)$, and
- $t_{\varphi^{-1}(i)} - t_{i_0} \geq \mathsf{Dist}_{\mathcal{T}}(v_0, a_{\varphi^{-1}(i)})$.

Relying on the triangle inequality, this leads to $2 \times t_0 > t_{j_0} - t_{i_0} \geq 2\mathsf{Dist}_{\mathcal{T}}(v_0, b)$, and this allows us to deduce that $b \in \mathsf{Close}(v_0)$.

Step 3: We consider the topology $\mathcal{T}' = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}', v_0, p_0)$ such that $\mathsf{Loc}'(v_0) = \mathsf{Loc}(v_0)$, and $\mathsf{Loc}'(p_0)$ is such that $\mathsf{Dist}_{\mathcal{T}'}(v_0, p_0) = t_0$ and:

$$\mathsf{Loc}'(a) = \begin{cases} \mathsf{Loc}'(v_0) & \text{if } a \in \mathsf{Close}(v_0) \\ \mathsf{Loc}'(p_0) & \text{otherwise.} \end{cases}$$

In this topology, the agents distant from v_0 are moved to p_0, and agents in the neighbourhood of v_0 are moved to v_0. We denote $\widetilde{\mathsf{tr}_i} = \mathsf{tr}'_i$ for $i \in \{1, \ldots, i_0 - 1, j_0 + 1, \ldots, n\}$, i.e. those not affected by Step 2. In this topology $\mathcal{T}'$, we only have two locations, and we have that $\widetilde{\mathcal{K}_0^\star} \overset{\mathsf{tr}'_1 \ldots \mathsf{tr}'_n}{\rightsquigarrow}_{\mathcal{T}'} \widetilde{\mathcal{K}_{\mathsf{end}}^\star}$ since the locations of the agents are no longer relevant in the untimed semantics.

Step 4: We now show that we can come back to a timed execution, i.e. one executable in the timed semantics by induction on the number of guarded inputs in the trace. Given a configuration $\widehat{\mathcal{K}_0}$ such that $\mathsf{untimed}(\widehat{\mathcal{K}_0}) = \widetilde{\mathcal{K}_0^\star}$, we show that there exists a configuration $\widehat{\mathcal{K}_n}$ such that $\mathsf{untimed}(\widehat{\mathcal{K}_n}) = \widetilde{\mathcal{K}_n^\star} = \widetilde{\mathcal{K}_{\mathsf{end}}^\star}$. To show this result, we split our execution trace $\widetilde{\mathcal{K}_0^\star} \overset{\mathsf{tr}'_1 \ldots \mathsf{tr}'_n}{\rightsquigarrow}_{\mathcal{T}'} \widetilde{\mathcal{K}_n^\star}$ on several blocks of actions: a block is either a trace with no guarded input, or a sequence of actions starting with a `reset` and ending at the next occurrence of a guarded input. Note that the block of the first kind can easily be lifted to the timed semantics. Regarding the block of the second kind, we show that the lifting is possible thanks to the properties established at Step 2.

To conclude this step, we now show how to exploit properties established at Step 2 to lift a block starting with a reset instruction and ending with a guarded input.

Let $\widetilde{\mathcal{K}_{\mathtt{reset}}^\star} \overset{\mathsf{tr}'_{i_0} \ldots \mathsf{tr}'_{j_0}}{\rightsquigarrow}_{\mathcal{T}'} \widetilde{\mathcal{K}_{\mathtt{in}}^\star}$ be such a block, and let $\widehat{\mathcal{K}}$ be such that $\mathsf{untimed}(\widehat{\mathcal{K}}) = \widetilde{\mathcal{K}_{\mathtt{reset}}^\star}$, and let $\widehat{t}$ the global time of configuration $\widehat{\mathcal{K}}$. We have to show that there exists $\widehat{\mathcal{K}_{\mathtt{in}}}$ such that $\widehat{\mathcal{K}} \xrightarrow{\widehat{\mathsf{tr}_{i_0}} \ldots \widehat{\mathsf{tr}_{j_0}}}_{\mathcal{T}'} \widehat{\mathcal{K}_{\mathtt{in}}}$, with tr'_i the untimed counterpart of $\widehat{\mathsf{tr}_i}$ and $\mathsf{untimed}(\widehat{\mathcal{K}_{\mathtt{in}}}) = \widetilde{\mathcal{K}_{\mathtt{in}}^\star}$. We start by applying the rule TIM with the delay δ equals to $2 \times t_0$. Let $\widehat{\mathcal{K}_+}$ be the resulting configuration. Then we have to show that the sequence of actions $\mathsf{tr}'_{i_0} \ldots . \mathsf{tr}'_{j_0}$ can be executed without introducing any delay. Moreover, we show that the resulting configuration $\widehat{\mathcal{K}_{\mathtt{in}}}$ is such that $\mathsf{untimed}(\widehat{\mathcal{K}_{\mathtt{in}}}) = \widetilde{\mathcal{K}_{\mathtt{in}}^\star}$. Actually, the correspondence between timed and untimed configurations is maintained along the trace. The only difficult part is when the underlying action is an input. We know that this input is performed by $a \in \mathsf{Close}(v_0)$. Let $\mathtt{in}^\star(u)$ be an input occurring in the block and let Φ' the current frame in the untimed semantics when this action occurs and $\widehat{\Phi}$ its corresponding frame in the timed trace. By definition of the untimed semantics, we know that there exists a recipe R such that $R\Phi'\downarrow = u$. Thus, we know that $R\widehat{\Phi}\downarrow = u$. To conclude, it remains to show that the timing constraints are satisfied. We distinguish two cases:

- The input has been forged by an agent $b \in \mathsf{Close}(v_0)$. Any w used in the recipe R is either in $dom(\phi(\widetilde{\mathcal{K}_{\mathtt{reset}}^\star}))$ or output after tr'_{i_0} by an agent located at the same place as v_0. In both cases, since the global time has elapsed of $2t_0$ between $\widehat{\mathcal{K}}$ and $\widehat{\mathcal{K}_+}$, we know that all these w

will be available at time $\widehat{t} + 2t_0$ for b. Since a and b are located at the same place, we also have that this input can be done at time $\widehat{t} + 2t_0$.

- The input has been forged by an agent $b \notin \mathsf{Close}(v_0)$. In such a case, we know that $vars(R) \subseteq dom(\phi(\widetilde{\mathcal{K}^\star_{\mathsf{reset}}}))$, and thus thanks to the delay of $2t_0$ that has been applied between $\widehat{\mathcal{K}}$ and $\widehat{\mathcal{K}_+}$, we know that the input can be received at time $\widehat{t} + 2t_0$. Indeed, b can forge the message at time $\widehat{t} + t_0$ and thus it can be received at time $\widehat{t} + 2t_0$.

Note that, regarding the guarded input, the guard is trivially satisfied since no time has elapsed since the reset has been performed. In summary we have:

$$\widehat{\mathcal{K}_0} \xrightarrow{\widehat{\mathsf{tr}_1}.....\widehat{\mathsf{tr}_n}}_{\mathcal{T}'} \widehat{\mathcal{K}_n}$$

with $\widetilde{\mathcal{K}^\star_n}$ the untimed counterpart of $\widehat{\mathcal{K}_n}$ and thus, we have that:

$$\widehat{\mathcal{K}_n} = (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0}^{\widehat{t_v}} \cup \widehat{\mathcal{P}}; \widehat{\Phi_n}; \widehat{t_n}) \text{ for some } \widehat{\mathcal{P}}, \widehat{\Phi_n}, \widehat{t_v} \text{ and } \widehat{t_n}.$$

Step 5: To finish, it remains to reduce the topology $\mathcal{T}'$ to $\mathcal{T}^{t_0}_{\mathsf{DH}} = (\mathcal{A}_{\mathsf{DH}}, \mathcal{M}_{\mathsf{DH}}, \mathsf{Loc}_{\mathsf{DH}}, v_0, p_0)$. Let us consider the renaming

$$\rho(a) = \begin{cases} v_0 & \text{if } a \in \mathsf{Close}(v_0) \\ p_0 & \text{if } a \notin \mathsf{Close}(v_0) \text{ and } a \in \mathcal{M}_0 \\ e_0 & \text{if } a \notin \mathsf{Close}(v_0) \text{ and } a \notin \mathcal{M}_0 \end{cases}$$

Since $\mathsf{Loc}_{\mathsf{DH}}(\rho(a)) = \mathsf{Loc}'(a)$ for any $a \in \mathcal{A}_0$, and $\rho(a) \in \mathcal{M}_{\mathsf{DH}}$ if, and only if $a \in \mathcal{M}_0$, thanks to Lemma 4, we have that:

$$\widehat{\mathcal{K}_0\rho} \xrightarrow{\widehat{\mathsf{tr}_0\rho}.....\widehat{\mathsf{tr}_n\rho}}_{\mathcal{T}^{t_0}_{\mathsf{DH}}} \widehat{\mathcal{K}_n\rho}.$$

We can assume w.l.o.g. that $\widehat{\mathcal{K}_0}$ has global time 0, and only contain frame elements at time 0. Let $\Phi_0 = \phi(\widehat{\mathcal{K}_0})$. Thus, to conclude, it remains to show that the frame associated to $\widehat{\mathcal{K}_0\rho}$, i.e. $\phi(\widehat{\mathcal{K}_0\rho})$, is the expected one. Thus, we have to establish that:

- $img(\lfloor \phi(\widehat{\mathcal{K}_0\rho}) \rfloor^0_{v_0}) = img(\lfloor \phi(\widehat{\mathcal{K}_0\rho}) \rfloor^0_{e_0}) = \emptyset$, and
- $img(\lfloor \phi(\widehat{\mathcal{K}_0\rho}) \rfloor^0_{p_0}) = \mathsf{Knows}(\mathcal{I}_0, p_0, \{v_0, p_0, e_0\})$.

Relying on the fact that any agent a such that $\rho(a) = v_0$ is honest, we have that:

$$\begin{array}{rcl} img(\lfloor \phi(\widehat{\mathcal{K}_0\rho}) \rfloor^0_{v_0}) & = & img(\lfloor \Phi_0\rho \rfloor^0_{v_0} \\ & = & \bigcup_{\{a \in \mathcal{A}_0 \mid \rho(a) = v_0\}} img(\lfloor \Phi_0 \rfloor^0_a)\rho \\ & = & \bigcup_{\{a \in \mathcal{A}_0 \mid \rho(a) = v_0\}} \emptyset \\ & = & \emptyset \end{array}$$

Actually, the same reasoning applied regarding e_0. Then, we have that:

$$\begin{array}{rcl} img(\lfloor \phi(\widehat{\mathcal{K}_0\rho}) \rfloor^0_{p_0}) & = & \bigcup_{\{a \in \mathcal{A}_0 \mid \rho(a) = p_0\}} img(\lfloor \Phi_0 \rfloor^0_a)\rho \\ & = & \bigcup_{\{a \in \mathcal{A}_0 \mid \rho(a) = p_0\}} \mathsf{Knows}(\mathcal{I}_0, a, \mathcal{A}_0)\rho \\ & = & \bigcup_{\{a \in \mathcal{A}_0 \mid \rho(a) = p_0\}} \mathsf{Knows}(\mathcal{I}_0, \rho(a), \rho(\mathcal{A}_0)) \\ & = & \bigcup_{\{a \in \mathcal{A}_0 \mid \rho(a) = p_0\}} \mathsf{Knows}(\mathcal{I}_0, p_0, \{p_0, v_0, e_0\}) \end{array}$$

This allows us to conclude. □

C PROOFS FOR SECTION 6

Lemma 7. *Let t_0 be a term such that $t_0\downarrow =_{\mathsf{E}} \mathsf{f}(u_1,\dots,u_k)$ with f a quasi-free function symbol. We have that there exist $u'_1,\dots,u'_k$ such that $\mathsf{f}(u'_1,\dots,u'_k) \in st(t_0)$ and $u'_i\downarrow =_{\mathsf{E}} u_i$ for any $i \in \{1,\dots,k\}$.*

Proof. Let t_0 be a term, and $t_0\downarrow$ its normal form. Therefore, we have that $t_0 \to t_1 \to t_2 \dots \to t_n = t_0\downarrow$. We prove the result by induction on the length n of this derivation.

Base case: $n = 0$. We have that $t_0\downarrow = t_0$, and thus $t_0 =_{\mathsf{E}} \mathsf{f}(u_1,\dots,u_k)$. Since our theory is non-trivial, and since f does not occur in equations in E, we have that there exists $u'_1,\dots,u'_k$ such that $\mathsf{f}(u'_1,\dots,u'_k) \in st(t_0)$ and $u'_i =_{\mathsf{E}} u_i$ for any $i \in \{1,\dots,k\}$. Note that $u'_1,\dots,u'_k$ are in normal form since t_0 is in normal form, and thus the result holds.

Induction step. In such a case, applying our induction hypothesis, we know that there exist $u'_1,\dots,u'_k$ such that $\mathsf{f}(u'_1,\dots,u'_k) \in st(t_1)$ and $u'_i\downarrow =_{\mathsf{E}} u_i$ for any $i \in \{1,\dots,k\}$. We denote $\mathsf{g}(v'_1,\dots,v'_\ell) \to v'$ the rewrite rule applied at position p to rewrite t_0 in t_1. We have that there exists a substitution θ such that $t_0|_p =_{\mathsf{E}} g(v'_1,\dots,v'_k)\theta$ and $t_1 = t_0[v'\theta]_p$. We have that $\mathsf{f}(u'_1,\dots,u'_n) \in st(t_1)$, and we distinguish two cases depending on the position p_{f} at which this subterm occurs in $t_1 = t_0[v\theta]_p$:

(1) p_{f} is a position in $t_0[\_]_p$. In such a case, we have that either $\mathsf{f}(u'_1,\dots,u'_n) \in st(t_0[\_]_p)$ (in case p_{f} is not a prefix of p); or $t_0|_{p_{\mathsf{f}}} = \mathsf{f}(u''_1,\dots,u''_n)$ for some $u''_1,\dots,u''_n$, and we have that $t_0|_{p_{\mathsf{f}}} \to \mathsf{f}(u'_1,\dots,u'_n)$ with $u''_i = u'_i$ or $u''_i \to u'_i$. Therefore, we have that there exist $u''_1,\dots,u''_n$ such that $\mathsf{f}(u'_1,\dots,u'_n) \in st(t_0)$ and $u''_i\downarrow =_{\mathsf{E}} u_i$ for any $i \in \{1,\dots,k\}$.

(2) p_{f} is a position below p, i.e. p is a prefix of p_{f}. In such a case, since f does not occur in v' (by definition of quasi-free), we have that $\mathsf{f}(u'_1,\dots,u'_k) \in st(x\theta)$ for some $x \in vars(v') \subseteq vars(v'_1,\dots,v'_\ell)$. Therefore, we have that there exists $t' =_{\mathsf{E}} t_0|_p$ such that $\mathsf{f}(u'_1,\dots,u'_k) \in st(t')$. Since we only consider non-trivial theory, and since f does not occur in equations in E, we have that there exists $u''_1,\dots,u''_k$ such that $\mathsf{f}(u''_1,\dots,u''_k) \in st(t_0|_p)$ and $u''_i =_{\mathsf{E}} u'_i$ for any $i \in \{1,\dots,k\}$. Note that $u''_1,\dots,u''_k$ are in normal form since any subterm of $t_0|_p$ is in normal form. Therefore, we have that there exist $u''_1,\dots,u''_n$ such that $\mathsf{f}(u''_1,\dots,u''_n) \in st(t_0)$ and $u''_i\downarrow =_{\mathsf{E}} u_i$ for any $i \in \{1,\dots,k\}$.

This concludes the proof. □

Lemma 8. *Let Φ be a frame and $c \in \mathcal{N}$ such that $c \notin st(img(\Phi))$, and $\Phi^+ = \Phi \cup \{w_c \xrightarrow{v_0,t_0} c\}$. Let R be a recipe such that $R\Phi^+\downarrow =_{\mathsf{E}} u$. Let C be a context of minimal size made of quasi-free public function symbols such that $u = C[c,u_1,\dots,u_p]$ for some $u_1,\dots,u_p$ and c does not occur in $st(\{u_1,\dots,u_p\})$. For any $i \in \{1,\dots,p\}$, we have that there exists R_i such that $R_i\Phi^+\downarrow =_{\mathsf{E}} u_i$.*

Proof. We prove this result by structural induction on the context C.

Base case: C is empty. In such a case, we have that either $p = 0$; or $p = 1$ with $u_1 = u$. In both case, the result trivially holds.

Inductive case: $C = \mathsf{f}(C_1,\dots,C_k)$. In such a case, by minimality of C, we know that c occurs in u. We have that $u = \mathsf{f}(u'_1,\dots,u'_k)$ and for all $i \in \{1,\dots,k\}$ we note $\{u^i_1,\dots,u^i_{p_i}\} \subseteq \{u_1,\dots,u_p\}$ the set of terms involved in the sub-context C_i i.e. $C_i[u^i_1,\dots,u^i_{p_i}] = u'_i$. Note that $\bigcup_{1\le i\le k}\{u^i_1,\dots,u^i_{p_i}\} = \{u_1,\dots,u_p\}$.

Applying Lemma 7 on $R\Phi^+$, we deduce that there exist $v_1,\dots,v_k$ such that $\mathsf{f}(v_1,\dots,v_k) \in st(R\Phi^+)$ and $v_i\downarrow =_{\mathsf{E}} u'_i$ for any $i \in \{1,\dots,k\}$. Now, since c occurs in u we have that there exists $i_0 \in \{1,\dots,k\}$ such that $u^{i_0}_1 = c$. Therefore we have that c occurs in $v_{i_0}\downarrow$ because C_{i_0} only contains quasi-free function symbols (i.e. function symbols which do not occur in E). By consequence we have that $\mathsf{f}(v_1,\dots,v_k) \notin img(\Phi^+)$ and thus there exists $R_1,\dots,R_k$ such that $\mathsf{f}(R_1,\dots,R_k) \in st(R)$ with $R_i\Phi^+\downarrow =_{\mathsf{E}} u'_i$ for any $1 \le i \le k$.

Our induction hypothesis applies for any $i \in \{1, \ldots, k\}$ and we obtain that for all $v \in \{u_1^i, \ldots, u_{p_i}^i\}$, there exists a recipe R_v such that $R_v\Phi^+\downarrow =_{\mathsf{E}} v$. Considering the previous remark stating that $\bigcup_{1\leq i\leq k}\{u_1^i, \ldots, u_{p_i}^i\} = \{u_1, \ldots, u_p\}$, this allows us conclude the proof. □

Lemma 9. *Let* (V, P) *be a well-formed distance-bounding protocol. The process* P^* *(as defined in Definition 13) is a semi-dishonest prover, called* the most general semi-dishonest prover. *Moreover, we can assume that the trace* tr^* *witnessing this fact is such that :*

$$\mathsf{tr}^* = \begin{cases} (a_1, \mathsf{out}(m_1)).(b_1, \mathsf{in}(m_1)) \ldots (a_{i_0-1}, \mathsf{out}(m_{i_0-1})).(b_{i_0-1}, \mathsf{in}(m_{i_0-1})). \\ (p_0, \mathsf{out}(m_{i_0+1}^1)). \ldots .(p_0, \mathsf{out}(m_{i_0+1}^l)). \\ (v_0, \mathsf{out}(m_{i_0})).(v_0, \mathsf{in}^{<2\cdot t_0}(m_{i_0+1})) \\ (p_0, \mathsf{in}(m_{i_0})).(p_0, \mathsf{out}(m_{i_0+1})). \\ (a_{i_0+2}, \mathsf{out}(m_{i_0+2})).(b_{i_0+2}, \mathsf{in}(m_{i_0+2})) \ldots (a_n, \mathsf{out}(m_n)).(b_n, \mathsf{in}(m_n)) \end{cases}$$

up to τ *actions where:*

- $\{a_i, b_i\} = \{v_0, p_0\}$ *for any* $i \in \{1, \ldots, n\} \setminus \{i_0, i_0 + 1\}$*;*
- $m_{i_0+1} =_{\mathsf{E}} C_{\mathsf{V},\mathsf{P}}[m_{i_0}, m_{i_0+1}^1, \ldots, m_{i_0+1}^l]$
- $(x_1, \ldots, x_k)\theta_{\mathcal{U}}\downarrow\sigma =_{\mathsf{E}} m_{i_1}, \ldots, m_{i_k}$ *where* $x_1, \ldots, x_k$ *are the variables occurring in input in the role* $\mathsf{V}(v_0, p_0)$ *and* $i_1, \ldots, i_k$ *are the indices among* $1, \ldots, n$ *corresponding to input performed by* v_0*,* $\mathcal{U} = \{x = u \mid$ "$\mathtt{let}\ x = u\ \mathtt{in}$" *occurs in* $\mathsf{V}(v_0, p_0)\}$*, and* σ *is a bijective renaming from variables to* $bn(\mathsf{P}^*)$*. This equality holds up to a bijective renaming of names freshly generated along the execution.*

Proof. This proof strongly relies on Definition 12. First, remark that tr^* corresponds to the trace tr in which the extra outputs of P^* are executed just before the $i_0{}^{\text{th}}$ communication action i.e. the output of the challenge; and the answer to the challenge received by v_0 is anticipated. We show that this sequence of actions tr^* is an execution w.r.t. our semantics:

- 1^{st} line of actions: The actions can be executed following our semantics applying a TIM rule with a delay $\delta = t_0$ before each input. Indeed this delay enables the agent b_i to receive the message m_i sent by a_i. Moreover, since there is no guarded input the IN rule always applies.
- 2^{nd} line of actions: It only contains outputs and thus can trivially be executed.
- 3^{rd} line of actions: Before executing the output, we apply a TIM rule to let available all the previous messages (including $m_{i_0+1}^1, \ldots, m_{i_0+1}^l$) for the malicious agent p. Since $C_{\mathsf{V},\mathsf{P}}$ only contains public symbols of functions (otherwise there is a contradiction with item (iv) of Definition 12), we have that $R = C_{\mathsf{V},\mathsf{P}}[w_{i_0}, w_{i_0+1}^1, \ldots, w_{i_0+1}^l]$ where w_{i_0} is the frame variable binding m_{i_0} and $w_{i_0+1}^j$, $(1 \leq j \leq l)$ is the frame variable binding $m_{i_0+1}^j$, is a recipe deducing m_{i_0+1}. Finally the guarded input can be executed because w.l.o.g. we may assume that the reset action has been made right before the output of m_{i_0}.
- 4^{th} and 5^{th} lines of actions: These actions can be executed for the same reason as the first line applying a TIM rule with a delay $\delta = t_0$ before each input.

Regarding the three items we have to establish, their are all consequences of Definition 12 and Definition 13. □

Proposition 3. *Let* (V, P) *be a well-formed distance-bounding protocol, and* P^* *be its most general semi-dishonest prover with* Φ^* *its associated frame. Let* $exec^*$ *be an execution witnessing the fact that* P^* *together with* Φ^* *is a semi-dishonest prover (as given in Lemma 9). We have that* $exec^*$ *is as follows:*

$$exec^* : (\{\lfloor \mathsf{V}(v_0, p_0)\rfloor_{v_0}^0, \lfloor \mathsf{P}^*\rfloor_{p_0}^0\}; \emptyset; 0) \xrightarrow{\mathsf{tr}^*}_{\mathcal{T}_{\text{simple}}^{t_0}} (\{\lfloor 0\rfloor_{v_0}^{t_v^*}, \lfloor 0\rfloor_{p_0}^{t_p^*}\}; \Phi^*; t^*).$$

Let P_{sd} *be a semi-dishonest prover for* (V,P) *together with its associated frame* Φ_{sd}*, and exec be the execution witnessing this fact, i.e.*

$$exec : (\{\lfloor \mathsf{V}(v_0,p_0)\rfloor^0_{v_0}, \lfloor \mathsf{P}_{\mathsf{sd}}\rfloor^0_{p_0}\};\emptyset;0) \xrightarrow{\mathsf{tr}}_{\mathcal{T}^{t_0}_{\mathsf{simple}}} (\{\lfloor 0\rfloor^{t_v}_{v_0}, \lfloor 0\rfloor^{t_p}_{p_0}\};\Phi_{\mathsf{sd}};t).$$

We have that there exists a substitution $\sigma : \mathcal{N} \to \mathcal{T}(\Sigma_c^+, \mathcal{N} \cup \mathcal{A})$ *from names freshly generated by* P^* *to constructor terms such that:*

(i) if $(v_0,\mathsf{in}(u)) \in \mathsf{tr}^*$ *(resp.* $(v_0,\mathsf{in}^{<t}(u)) \in \mathsf{tr}^*$*), then* $(v_0,\mathsf{in}(u\sigma)) \in \mathsf{tr}$ *(resp.* $(v_0,\mathsf{in}^{<t}(u\sigma)) \in \mathsf{tr}$*);*
(ii) if $(a,\mathsf{out}(u)) \in \mathsf{tr}^*$ *for some* $a \in \{v_0,p_0\}$*, then* $R\Phi_{\mathsf{sd}}{\downarrow} =_{\mathsf{E}} u\sigma$ *for some recipe* R*.*

Proof. Along an execution, variables occurring in input as well as those occurring in a let instruction are instantiated. We denote by τ_{tr} (resp τ_{tr^*}) the substitution associated to the execution *exec* (resp. *exec**).

Relying on Lemma 9, we note $\tau : \mathcal{X} \to \mathcal{N}$ the bijective renaming from variables to names freshly generated by P^* such that $(x_1,\dots,x_k)\theta_{\mathcal{U}}{\downarrow}\tau =_{\mathsf{E}} m_{i_1},\dots,m_{i_k}$ where $x_1,\dots,x_k$ are the variables occurring in input in $\mathsf{V}(v_0,p_0)$ and $m_{i_1},\dots,m_{i_k}$ are the inputted terms in tr^*. By definition of $\theta_{\mathcal{U}}$ we have that there exists σ' such that $x_j\tau_{\mathsf{tr}}{\downarrow} =_{\mathsf{E}} x_j\theta_{\mathcal{U}}\sigma'{\downarrow}$ for any $j \in \{1,\dots,k\}$. Since $x_j\tau_{\mathsf{tr}}$ and $x_j\theta_{\mathcal{U}}{\downarrow}$ are constructor terms, we have that $x_j\tau_{\mathsf{tr}} =_{\mathsf{E}} x_j\theta_{\mathcal{U}}{\downarrow}\sigma'$ for any $j \in \{1,\dots,k\}$.

We consider $\sigma = \tau^{-1}\sigma'$ and establish each item separately.

Item (i): We consider $\mathsf{V}(v_0,p_0) = \mathsf{act}_1.\dots.\mathsf{act}_{k'}$. On the first side, because $(v_0,\mathsf{in}^\star(u)) \in \mathsf{tr}^*$, we have that there exists $j \in \{1,\dots,k\}$ such that $x_j\tau_{\mathsf{tr}^*} = u\ (= m_{i_j})$. By definition of τ, we have that $u = x_j\theta_{\mathcal{U}}{\downarrow}\tau$.

On the other side, we know that the process has been entirely executed in tr and therefore there exists $(v_0,\mathsf{in}^\star(u')) \in \mathsf{tr}$ such that $u' = x_j\tau_{\mathsf{tr}}$. By definition of σ', we have $u' =_{\mathsf{E}} x_j\theta_{\mathcal{U}}{\downarrow}\sigma'$. By consequence we obtain that $u' = x_j\tau_{\mathsf{tr}} =_{\mathsf{E}} x_j\theta_{\mathcal{U}}{\downarrow}\sigma' = (u\tau^{-1})\sigma' = u\sigma$. This concludes the proof of item (i).

Item (ii): We have that $(a,\mathsf{out}(u)) \in \mathsf{tr}^*$. We distinguish several cases depending on the origin of this output.

In case $a = v_0$, it is an immediate corollary of item (i). Indeed we can prove by induction on the length of tr that for each configuration $\mathcal{K}$ in *exec*, there exists a configuration $\mathcal{K}^*$ in *exec** such that if we note V^* the process executed by v_0 in $\mathcal{K}^*$ then $V^*\sigma$ is the process executed by v_0 in $\mathcal{K}$.

Now, we assume that $a = p_0$, and we distinguish two cases depending on whether $u = m^j_{i_0+1}$ $(1 \le j \le l)$ or not. If not, relying on Lemma 9, we have that $(v_0,\mathsf{in}(u)) \in \mathsf{tr}^*$ (or $(v_0,\mathsf{in}^{<t}(u)) \in \mathsf{tr}^*$), and since item (i) holds we have now that $(v_0,\mathsf{in}(u\sigma)) \in \mathsf{tr}$ (or $(v_0,\mathsf{in}^{<t}(u\sigma)) \in \mathsf{tr}$). We can thus deduce that exists a recipe R such that $R\Phi_{\mathsf{sd}}{\downarrow} =_{\mathsf{E}} u\sigma$.

Now, we assume that $u = m^j_{i_0+1}$ for some $j \in \{1,\dots,l\}$. Thanks to Lemma 9, we know that $(v_0,\mathsf{in}^{<2\cdot t_0}(m_{i_0+1})) \in \mathsf{tr}^*$ and $m_{i_0+1} = C_{\mathsf{V},\mathsf{P}}[m_{i_0}, m^1_{i_0+1},\dots,m^l_{i_0+1}] = x\theta_{\mathcal{U}}{\downarrow}\tau$ where x in the variable occurring in the guarded input in $\mathsf{V}(v_0,p_0)$. According to item (i), we have that $(v_0,\mathsf{in}^{<2t_0}(m_{i_0+1}\sigma)) \in \mathsf{tr}$. Moreover, following the hypotheses on the structure of $\mathsf{V}(v_0,p_0)$, we know that there is a unique output in tr that is executed by v_0 before this guarded input and that contains the challenge m_{i_0} (note that $m_{i_0} = m_{i_0}\sigma$ because σ only applies on names generated by P^*). In addition, p_0 cannot receive the challenge soon enough to make an output containing m_{i_0} available to fill the guarded input. Indeed, assume that it was possible, and let $t_{\mathtt{reset}}$ (resp. t_{out} and t_{in}) the time when the reset action (resp. output of the challenge, reception of the guarded input) is executed in tr then we have that:

$$t_{\mathsf{in}} \ge t_{\mathsf{out}} + 2 \times \mathsf{Dist}_{\mathcal{T}^{t_0}_{\mathsf{simple}}}(v_0,p_0) \ge t_{\mathtt{reset}} + 2 \times \mathsf{Dist}_{\mathcal{T}^{t_0}_{\mathsf{simple}}}(v_0,p_0) = t_{\mathtt{reset}} + 2 \times t_0.$$

This is in contradiction with the constraint imposed by the guarded input: $t_{\mathsf{in}} - t_{\mathsf{reset}} < 2 \times t_0$. Finally, denoting Φ^+ the current frame when executing the guarded input in tr, we deduce that $\Phi^+ = \Phi \cup \{w \xrightarrow{v_0, t_{\mathsf{out}}} m_{i_0}\}$ for some sub-frame Φ such that $m_{i_0} \notin st(img(\Phi))$ and there exists a recipe R such that $R\Phi^+\downarrow =_{\mathsf{E}} m_{i_0+1}\sigma$. Lemma 8 applies and we conclude that there exists a recipe R_u such that $R_u\Phi^+\downarrow =_{\mathsf{E}} m^j_{i_0+1}\sigma = u\sigma$ and thus $R_u\Phi_{\mathsf{sd}}\downarrow =_{\mathsf{E}} u\sigma$ since Φ_{sd} contains Φ^+. □

Theorem 3. *Let $\mathcal{I}_0$ be a template, (V, P) be a well-formed distance-bounding protocol. Let Φ^* be the frame associated to the most general semi-dishonest prover of (V, P). We have that (V, P) is Terrorist fraud resistant w.r.t. t_0-proximity if, and only if, there exist a topology $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, Loc_0, v_0, p_0) \in \mathcal{C}^{t_0}_{\mathsf{MF}}$ and a valid initial configuration $\mathcal{K}_0$ for (V, P) w.r.t. $\mathcal{T}$ and $\Phi^* \cup \Phi^{\mathcal{T}}_{\mathcal{I}_0}$ such that $\mathcal{K}_0 \rightarrow^*_{\mathcal{T}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor^{t'}_{v_0} \uplus \mathcal{P}; \Phi; t)$.*

Proof. The direct implication is trivial since P^* is a semi-dishonest prover. We concentrate on the other implication, and we have to show that the property holds for any semi-dishonest prover. Let P_{sd} be a semi-dishonest prover for (V, P) with its associated frame Φ_{sd}, and $\mathsf{tr}_{\mathsf{sd}}$ be the trace witnessing this fact. We denote P^* the most general semi-dishonest prover, Φ^* its associated frame, and tr^* the trace witnessing this fact.

Applying Proposition 3, there exists a substitution $\sigma : \mathcal{N} \to \mathcal{T}(\Sigma^+_c, \mathcal{N} \cup \mathcal{A})$, from names freshly generated by P^* in tr^* to constructor terms such that:

(i) if $(v_0, \mathsf{in}(u)) \in \mathsf{tr}^*$, then $(v_0, \mathsf{in}(u\sigma)) \in \mathsf{tr}_{\mathsf{sd}}$ (and similarly for the guarded input).
(ii) if $(a, \mathsf{out}(u)) \in \mathsf{tr}^*$ for some $a \in \{v_0, p_0\}$, then $R\Phi_{\mathsf{sd}}\downarrow =_{\mathsf{E}} u\sigma$ for some recipe R.

By hypothesis, there exist a topology $\mathcal{T} \in \mathcal{C}^{t_0}_{\mathsf{MF}}$ and a valid initial configuration $\mathcal{K}_0$ for (V, P) w.r.t. $\mathcal{T}$ and $\Phi^* \cup \Phi^{\mathcal{T}}_{\mathcal{I}_0}$ such that

$$\mathcal{K}_0 = (\mathcal{P}_{\mathsf{init}}; \Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi^*; 0) \xrightarrow{\mathsf{tr}}_{\mathcal{T}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor^{t'}_{v_0} \uplus \mathcal{P}; \Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi^* \cup \Phi_{\mathsf{out}}; t)$$

for some frame Φ_{out}. Without loss of generality, we may assume that $\mathcal{T} = (\mathcal{A}_0, \mathcal{M}_0, \mathsf{Loc}_0, v_0, p_0)$ with $\mathcal{M}_0 \neq \emptyset$. Otherwise, we add such a malicious agent, and the trace remains executable. Applying the substitution σ along this execution, we obtain a valid execution (remember that our calculus does not feature else branches):

$$(\mathcal{P}_{\mathsf{init}}\sigma; \Phi^{\mathcal{T}}_{\mathcal{I}_0}\sigma \cup \Phi^*\sigma; 0) \xrightarrow{\mathsf{tr}\sigma}_{\mathcal{T}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor^{t'}_{v_0} \uplus \mathcal{P}\sigma; \Phi^{\mathcal{T}}_{\mathcal{I}_0}\sigma \cup \Phi^*\sigma \cup \Phi_{\mathsf{out}}\sigma; t).$$

Actually, since names occurring in $dom(\sigma)$ are names freshly generated by P^*, we have that $\mathcal{P}_{\mathsf{init}}\sigma = \mathcal{P}_{\mathsf{init}}$, $\Phi^{\mathcal{T}}_{\mathcal{I}_0}\sigma = \Phi^{\mathcal{T}}_{\mathcal{I}_0}$, and therefore, we have that:

$$(\mathcal{P}_{\mathsf{init}}; \Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi^*\sigma; 0) \xrightarrow{\mathsf{tr}\sigma}_{\mathcal{T}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor^{t'}_{v_0} \uplus \mathcal{P}\sigma; \Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi^*\sigma \cup \Phi_{\mathsf{out}}\sigma; t).$$

Finally, from item (ii), we have that for any $u \in img(\Phi^*)$, there exists a recipe R such that $R\Phi_{\mathsf{sd}}\downarrow =_{\mathsf{E}} u\sigma$. We can thus deduce that for any term v and recipe R_v such that $R_v(\Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi^*\sigma)\downarrow =_{\mathsf{E}} v$ we have that there exists R'_v such that $R'_v(\Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi_{\mathsf{sd}})\downarrow =_{\mathsf{E}} v$. Starting by applying a TIM rule with a delay δ equal to twice the greatest distance between two agents in $\mathcal{T}$, we have:

$$(\mathcal{P}_{\mathsf{init}}; \Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi_{\mathsf{sd}}; 0) \xrightarrow{\mathsf{tr}\sigma}_{\mathcal{T}} (\lfloor \mathsf{end}(v_0, p_0) \rfloor^{t'}_{v_0} \uplus \tilde{\mathcal{P}}; \Phi^{\mathcal{T}}_{\mathcal{I}_0} \cup \Phi_{\mathsf{sd}} \cup \Phi'_{\mathsf{out}}\sigma; t + \delta).$$

with $\Phi'_{\mathsf{out}} = \{w \xrightarrow{a, t+\delta} u \mid w \xrightarrow{a,t} u \in \Phi_{\mathsf{out}}\}$. The delay δ enables a dishonest agent (there is one by assumption) to build any term occurring in $\Phi^*\sigma$ from Φ_{sd}. □

Corollary 3. *Let $\mathcal{I}_0$ be a template, (V, P) be a well-formed distance-bounding protocol. Let Φ^* be the frame associated to the most general semi-dishonest prover of (V, P). We have that (V, P) is Terrorist*

fraud resistant w.r.t. t_0-proximity if, and only if, there exists a valid initial configuration $\mathcal{K}_0$ for (V,P) w.r.t. $\mathcal{T}_{\mathsf{MF}}^{t_0}$ and $\Phi^* \cup \Phi_{\mathcal{I}_0}^{\mathcal{T}_{\mathsf{MF}}^{t_0}}$ such that $\mathcal{K}_0 \to^*_{\mathcal{T}_{\mathsf{MF}}^{t_0}} (\lfloor \mathsf{end}(v_0,p_0)\rfloor_{v_0}^{t'} \uplus \mathcal{P};\Phi;t)$.

Proof. We consider the first implication. If (V,P) is Terrorist fraud resistant w.r.t. t_0-proximity then there exist a topology $\mathcal{T} \in C_{\mathsf{MF}}^{t_0}$ and a valid initial configuration $\mathcal{K}$ for (V,P) w.r.t. $\mathcal{T}$ and $\Phi^* \cup \Phi_{\mathcal{I}_0}^{\mathcal{T}}$ such that $\mathcal{K} \xrightarrow{\mathsf{tr}}_{\mathcal{T}} (\lfloor \mathsf{end}(v_0,p_0)\rfloor_{v_0}^{t_v} \uplus \mathcal{P};\Phi;t)$.

By definition of a protocol, we know that V (resp. P) is $\mathcal{I}_0^V$-executable (resp. $\mathcal{I}_0^P$-executable). Hence, no agent names but v_0 and p_0 may occur in the initial configuration leading to Φ^*. Since this initial configuration has an empty frame, agent names are not publicly available, and thus the attacker p cannot introduce new identities along the execution leading to Φ^*. Therefore, we have that $\mathit{names}(\Phi^*) \cap \mathcal{A} \subseteq \{v_0,p_0\}$ and Theorem 1 applies. We deduce that there exits a valid initial configuration $\mathcal{K}_0$ for (V,P) w.r.t. $\mathcal{T}_{\mathsf{MF}}^{t_0}$ and $\Phi^* \cup \Phi_{\mathcal{I}_0}^{\mathcal{T}}$ such that:

$$\mathcal{K}_0 \xrightarrow{\mathsf{tr}'}_{\mathcal{T}_{\mathsf{MF}}^{t_0}} (\lfloor \mathsf{end}(v_0,p_0)\rfloor_{v_0}^{t'_v} \uplus \mathcal{P}';\Phi';t').$$

The other direction is an immediate application of Theorem 3 since $\mathcal{T}_{\mathsf{MF}}^{t_0} \in C_{\mathsf{MF}}^{t_0}$. □

D PROOF OF PROPOSITION 4

Proposition 4. *Let $t_0 \in \mathbb{R}_+$, $\mathcal{T}_0 = (\mathcal{A}_0,\mathcal{M}_0,Loc_0,v_0,p_0)$ be a topology, and (V,P) be a protocol such that $\mathsf{V}_{\mathsf{end}}(v_0,p_0)$ has the following form:*

$$\mathsf{block}_1 \,.\, \mathtt{reset} \,.\, \mathtt{out}(u) \,.\, \mathtt{in}^{<t}(x) \,.\, \mathsf{block}_2 \,.\, \mathsf{end}(v_0,p_0) \quad \textit{with } t \le 2\cdot t_0$$

Let $\mathcal{K}_0$ be a valid initial configuration for (V,P) w.r.t. $\mathcal{T}$ and Φ_0. If $\mathcal{K}_0 \xrightarrow{\mathsf{tr}}_{\mathcal{T}_0} (\lfloor \mathsf{end}(v_0,p_0)\rfloor_{v_0}^{t_v} \uplus \mathcal{P};\Phi;t)$ with $Dist_{\mathcal{T}_0}(v_0,p_0) \ge 2\cdot t_0$ then we have that:

$$\mathcal{F}(\mathcal{T}_0,(\mathsf{V},\mathsf{P}),\Phi_0,t_0) \overset{\mathsf{tr}'}{\Longrightarrow} (\{2 : \mathsf{end}(v_0,p_0)\} \uplus \mathcal{P}';\phi;2).$$

Moreover, in case there is no $a \in \mathcal{M}_0$ such that $Dist_{\mathcal{T}}(a,v_0) < t_0$, 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 output in phase 0.

Proof. In the same spirit as the proof of Theorem 2, we are able to prove that:

$$\widetilde{\mathcal{K}_0^\star} \overset{\mathsf{tr}'_1.....\mathsf{tr}'_n}{\rightsquigarrow}_{\mathcal{T}_0} (\lfloor \mathsf{end}(v_0,p_0)\rfloor_{v_0} \uplus \widetilde{\mathcal{P}};\widetilde{\Phi})$$

where $\widetilde{\mathcal{K}_0^\star}$ (resp. $\widetilde{\mathcal{P}}$, $\widetilde{\Phi}$) is the untimed counterpart of $\mathcal{K}_0$ (resp. $\mathcal{P}$, Φ) in which `reset` commands have been removed and guarded inputs have been replaced by simple inputs except in $\mathsf{V}_{\mathsf{end}}(v_0,p_0)$. Let i_0 (resp. j_0) be the index of the `reset` (resp. guarded input) occurring in $\mathsf{V}_{\mathsf{end}}(v_0,p_0)$. Moreover, still following the proof of Theorem 2, we have that:

(i) for all $i \in \{i_0,\dots,j_0\}$, the action tr'_i is executed by an agent $a_i \in \mathsf{Close}(v_0)$;
(ii) for all $i \in \{i_0,\dots,j_0\}$, if tr'_i is an input then the agent b_i responsible of the output is such that $b_i \in \mathsf{Close}(v_0)$, or the recipe that is used to trigger the input only contains handles output before the `reset` command.

In the following we say that a process is initial if it starts by an input. Let s_0 be the session identifier of the process $\mathsf{V}_{\mathsf{end}}(v_0,p_0)$. In the following we will prove that there exists a trace preserving items (i) and (ii) and satisfying the two following additional properties:

(iii) all processes but s_0 are either initial when executing the `reset` action or let unchanged until the end on the execution, i.e. there is no action in trace corresponding to this process after the `reset` action.

(iv) all processes but s_0 are either initial when executing the $\mathtt{in}^{<t}(u)$ action or let unchanged until the end on the execution.

Item (iii): Assume that there exists a process in $\widetilde{\mathcal{K}}^{\star}_{\mathtt{reset}}$, the configuration just before the `reset` command, with a session identifier $s' \neq s_0$ which is not initial. Let $k \in \{i_0, \ldots, n\}$ be the index of the first action corresponding to this process, i.e. tagged by the session identifier s' in the remaining of the trace. If k does not exist then the process is let unchanged and item (iii) is satisfied. Otherwise, we prove the following claim.

Claim 3. *For all $i \in \{i_0, \ldots, k-1\}$ we have that $\mathsf{tr}'_k \not\hookrightarrow \mathsf{tr}'_i$.*

Proof. We note $\mathsf{tr}'_k = (a_k, \alpha_k, s', r_k)$ and $\mathsf{tr}'_i = (a_i, \alpha_i, s_i, r_i)$. Assume that $\mathsf{tr}'_k \hookrightarrow \mathsf{tr}'_i$. If $\mathsf{tr}'_k \hookrightarrow_s \mathsf{tr}'_i$ then we have that $s_i = s'$, leading to a contradiction. Otherwise, we have that $\mathsf{tr}'_k \hookrightarrow_d \mathsf{tr}'_i$ and thus $\alpha_k = \mathtt{in}^{\star}(v)$. Contradiction since the process identified by s' in $\widetilde{\mathcal{K}}^{\star}_{\mathtt{reset}}$ is assumed not initial. □

Repeatedly applying Lemma 6, we are able to move the action tr'_k just before $\widetilde{\mathcal{K}}^{\star}_{\mathtt{reset}}$. Applying the same reasoning for all actions corresponding to s' until reaching an initial process, and then doing the same to all processes that are not initial in $\widetilde{\mathcal{K}}^{\star}_{\mathtt{reset}}$, we obtain an execution:

$$\widetilde{\mathcal{K}}^{\star}_0 \xrightarrow{\mathsf{tr}'_1 \ldots . \mathsf{tr}'_{i_0-1}}_{\mathcal{T}_0} \xrightarrow{\widetilde{\mathsf{tr}_{i_0}} \ldots . \widetilde{\mathsf{tr}_{j'_0-1}}}_{\mathcal{T}_0} \mathcal{K}'_{\mathtt{reset}} \xrightarrow{\widetilde{\mathsf{tr}_{i'_0}} \ldots . \widetilde{\mathsf{tr}_{j'_0-1}}}_{\mathcal{T}_0} \mathcal{K}'_{\mathtt{in}} \xrightarrow{\widetilde{\mathsf{tr}_{j'_0}} \ldots . \widetilde{\mathsf{tr}_n}}_{\mathcal{T}_0} (\lfloor \mathsf{end}(v_0, p_0) \rfloor_{v_0} \uplus \widetilde{\mathcal{P}}; \widetilde{\Phi})(\star)$$

satisfying item (iii) by construction. Moreover, items (i) and (ii) holds since we do not introduce new actions between the `reset` and the guarded input (actually, we only move actions before the `reset`).

Item (iv): We follow the same reasoning as for item (iii). Assume there is a process that is not initial in $\mathcal{K}'_{\mathtt{in}}$ and note s' its session identifier. Note k the index of the first action corresponding to session s' after $\mathcal{K}'_{\mathtt{in}}$. If k does not exist then it means that the process is kept unchanged until the end on the execution, thus the result holds. Otherwise, similarly as before, we establish the following claim:

Claim 4. *For all $i \in \{j'_0, \ldots, k-1\}$ we have that $\widetilde{\mathsf{tr}_k} \not\hookrightarrow \widetilde{\mathsf{tr}_i}$.*

Again, applying Lemma 6 we are able to move action $\widetilde{\mathsf{tr}_k}$ right before $\mathcal{K}'_{\mathtt{in}}$. We obtain a trace that satisfies item (iv). Moreover, the three items (i), (ii) and (iii) are still satisfied:

(ii) we do not introduce inputs between the `reset` and the guarded input;

(iii) the beginning of the trace (before the `reset`) is not modified;

(i) all the actions we introduce between the `reset` and the guarded input are executed by agents $a \in \mathsf{Close}(v_0)$. Indeed, we have seen that item (ii) holds and thus the process identified by s' is initial when executing the `reset` action. Since it is no longer initial when executing the guarded input, agent a must have executed an action in the meantime. Therefore, relying on item (ii) that holds on the execution ($\star$), we deduce that $a \in \mathsf{Close}(v_0)$;

To conclude, it remains to show that such a trace can actually be mimicked from the configuration $\mathcal{F}(\mathcal{T}_0, (\mathsf{V}, \mathsf{P}), \Phi_0, t_0)$. Actually, the `reset` action is replaced by phase 1, and the guarded input is replaced by a simple input followed by a phase 2 action.

Let $\lfloor P \rfloor^t_a$ be a process occurring in $\widetilde{\mathcal{K}}^{\star}_0$, we know that $P = \mathsf{V}(a, b)$, or $P = \mathsf{P}(a, b)$ for some $b \in \mathcal{A}_0$; or $P = \mathsf{V}_{\mathsf{end}}(v_0, p_0)$. We now consider all the actions performed by this process along the execution, and in particular, we pay attention on the slicing of all these actions w.r.t. `reset` action and the guarded input. This gives us the corresponding process that we have to consider in our translation, so that it will be able to mimic all the actions of $\lfloor P \rfloor^t_a$. Items (iii) and (iv) allow one

to ensure that our slicing (just before the inputs) is indeed sufficient. Our transformation $\mathcal{F}^{\geq}$ also forbid actions to be executed during phase 1, and this is justified by our item (i). Finally, item (ii) allows us to prevent the ProVerif attacker to act in phase 1 in case there is no dishonest participants in the vicinity of v_0. □