Towards the Automated Verification of Cyber-Physical Security Protocols: Bounding the Number of Timed Intruders

Vivek Nigam[1(✉)], Carolyn Talcott[2], and Abraão Aires Urquiza[1]

[1] Federal University of Paraíba, João Pessoa, Brazil
vivek@ci.ufpb.br, abraauc@gmail.com
[2] SRI International, Menlo Park, USA
clt@csl.sri.com

Abstract. Timed Intruder Models have been proposed for the verification of Cyber-Physical Security Protocols (CPSP) amending the traditional Dolev-Yao intruder to obey the physical restrictions of the environment. Since to learn a message, a Timed Intruder needs to wait for a message to arrive, mounting an attack may depend on where Timed Intruders are. It may well be the case that in the presence of a great number of intruders there is no attack, but there is an attack in the presence of a small number of well placed intruders. Therefore, a major challenge for the automated verification of CPSP is to determine how many Timed Intruders to use and where should they be placed. This paper answers this question by showing it is enough to use the same number of Timed Intruders as the number of participants. We also report on some preliminary experimental results in discovering attacks in CPSP.

1 Introduction

The Dolev-Yao intruder model is one of the cornerstones for the success of protocol verification being used in most verification tools. The protocol security literature contains a number of properties about the Dolev-Yao intruder, many of them vital for automated verification. For instance, it has been shown that protocol security verification is complete when considering only a single Dolev-Yao intruder in the following sense: if there is an attack in the presence of one or more (colluding) Dolev-Yao intruders, then the same attack with a single Dolev-Yao intruder is possible [4]. Such result greatly simplifies the implementation of tools as it is enough to use only one Dolev-Yao intruder.

However, for the important class of Cyber-Physical Security Protocols (CPSP), the Dolev-Yao intruder model is not suitable. CSPS normally rely on the physical properties of the environment where sessions are carried out to establish some physical properties. For example, Distance Bounding Protocols are used to infer an upper-bound on the distance between two players V, the verifier, and P, the prover. It works as follows:

$$V \longrightarrow P : m$$
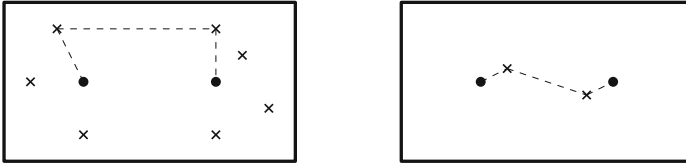$$P \longrightarrow V : m'$$

Fig. 1. The dots are protocol participants and the crosses are intruders.

The verifier sends a challenge m remembering the time t_1, when this message is sent. The prover responds to the challenge, m', and by measuring the round-trip time of the challenge response round, the verifier can compute (using assumptions on the transmission channel used) an upper bound on the distance to the prover.

It is easy to check that the Dolev-Yao intruder is not suitable for CPSP verification, as the Dolev-Yao intruder does not obey the physical properties of the system. As the Dolev-Yao intruder controls the network, he can receive the challenge m and instantaneously respond m' to the verifier's challenge. There have been, therefore, proposals to amend the Dolev-Yao intruder model to CSPS [3,16] in the form of Timed Dolev-Yao models. These have been used to prove general decidability of important properties of CSPS [3,15] and prove the security of protocols using theorem provers.

In contrast with the traditional Dolev-Yao intruder, who is the whole network, a timed intruder is placed at some location and in order to learn a message, must wait until the message arrives to that location. A consequence of this is that a greater number of colluding intruders may not do as much damage as a smaller number of intruders that are better placed. For example, consider Fig. 1. With a distribution of intruders shown to the left, there may not be an attack as it might take too long for intercepting and forwarding messages among intruders (illustrated by the dashed lines), while there may be an attack with the distribution of intruders shown to the right.

The main contribution of this paper is to answer the question: *How many intruders are enough for verification and where should they be placed?* We prove that it is enough to consider one intruder per protocol participant, thus bounding the number of timed intruders. This result greatly simplifies automated CSPS verification as the specifier no longer has to guess how many timed intruders to consider and where to place them.

Our second contribution is a general specification language, which extends strand spaces [25] by allowing for the symbolic representation of time. Instead of instantiating time variables and time constraints with explicit values, the semantics of our language accumulates symbolic time constraints. An execution using symbolic time constraints corresponds to a set of possible concrete executions, considerably reducing state-space. We implemented a prototype of our language in Maude [8] with SMT support. Our preliminary experiments show that it is possible to find attacks traversing few states. While we do not claim (yet) to have a complete tool, our first results are promising.

This paper is structured as follows: Sect. 2 specifies the syntax of our protocol specification language and its semantics extending Strand Spaces [25]. We introduce the Timed Intruder Model in Sect. 3. Section 4 contains the definition of the Timed Intruder Completeness problem and a solution to it. We revisit some examples in Sect. 5 briefly commenting on our prototype implementation. Finally we conclude by reviewing related and future work in Sect. 6.

2   A Specification Language for Cyber-Physical Security Protocols

We start first by specifying the syntax of our CPSP specification language with symbolic time variables and symbolic time constraints. We exemplify the specification of protocols using our language. Then, we formalize the operational semantics of our language by extending Strand Spaces [25] to include time variables.

2.1   Syntax

Message Expressions. We assume a message signature Σ of constants, and function symbols. Constants include nonces, symmetric keys and player names. The set of messages is constructed as usual using constants, variables and at least the following function symbols:

$\mathsf{sk}(\mathsf{p})$	Denoting the secret key of the player p;
$\mathsf{pk}(\mathsf{p})$	Denoting the public key of the player p;
$\mathsf{enc}(\mathsf{m}, \mathsf{k})$	Encryption function denoting the encryption of m using key k;
$\{\mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_n\}$	Tuple function denoting a list of messages $\mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_n$;

where $\mathsf{c}_1, \mathsf{c}_2, \ldots$ range over constants, $\mathsf{n}_1, \ldots, \mathsf{n}_n$ range over nonces, $\mathsf{k}_1, \mathsf{k}_2, \ldots$ range over symmetric keys, $\mathsf{p}_1, \mathsf{p}_2 \ldots$ range over player names, $\mathsf{v}_1, \mathsf{v}_2, \ldots$ range over variables, and $\mathsf{m}_1, \mathsf{m}_2, \ldots$ range over messages. For example, the message $\mathsf{enc}(\{\mathsf{v}_1, \mathsf{enc}(\mathsf{c}, \mathsf{k})\}, \mathsf{pk}(\mathsf{p}))$ denotes the encryption using the public key of p of the pair of messages v_1 (not instantiated) and $\mathsf{enc}(\mathsf{c}, \mathsf{k})$. We define $(\mathsf{pk}(\mathsf{p}))^{-1} = \mathsf{sk}(\mathsf{p})$ and $\mathsf{k}^{-1} = \mathsf{k}$ if k is a symmetric key. We also write interchangeably the singleton tuple $\{\mathsf{m}\}$ and m.

For a given scenario with some protocol session instances, we are going to distinguish the players that are participating in the protocol sessions, *e.g.*, as verifiers and as provers, which we call *protocol participants* (briefly *participants*), from the *Timed Intruders* which are not participating explicitly in the protocol sessions in the given scenario, but are simply manipulating messages and possibly interacting with the participants. The symbols $\mathsf{p}_1, \mathsf{p}_2 \ldots$ will range over participant names while $\mathsf{ti}_1, \mathsf{ti}_2, \ldots$ will range over the names of such Timed Intruders.

Time Expressions. We also assume a time signature Ξ which is disjoint to the message alphabet Σ. It contains:

$r_1, r_2, \ldots$	A set of numbers;
$tt_1, tt_2, \ldots,$	A set of time variables including the special variablecur;
$+, -, \times, /, \mathsf{floor}, \mathsf{ceiling}, \ldots$	A set of arithmetic symbols and other function symbols.

Time Expressions are constructed inductively by applying arithmetic symbols to time expressions. For example $\mathsf{ceiling}((2 + tt + \mathsf{cur})/10)$ is a Time Expression. The symbols $tr_1, tr_2, \ldots$ range over Time Expressions. We do not constrain the set of numbers and function symbols in Ξ. However, in practice, we allow only the symbols supported by the SMT solver used. All examples in this paper will contain SMT supported symbols (or equivalent). Finally, the time variable cur will be a keyword in our protocol specification language denoting the current global time.

Definition 1 (Symbolic Time Constraints). *Let Ξ be a time signature. The set of symbolic time constraints is constructed using time expressions. Let tr_1, tr_2 be time expressions, then*

$$tr_1 = tr_2, \qquad tr_1 \geq tr_2 \qquad tr_1 > tr_2, \qquad tr_1 < tr_2, \quad and \quad tr_1 \leq tr_2$$

are Symbolic Time Constraints.

For example, $\mathsf{cur} + 10 < \mathsf{floor}(tt - 5)$ is a Time Constraint. The symbols $tc_1, tc_2, \ldots$ will range over Time Constraints.

Finally, we let $b_1, b_2, \ldots$, range over boolean expressions, which include timed comparison constraints. We also allow for checking whether two messages m_1 and m_2 can be unified, *e.g.*, $\{v_1, v_2\} :=: \{p_1, k_1\}$ evaluates to true as they can be unified by the substitution $\{v_1 \mapsto p_1, v_2 \mapsto k_1\}$.

Definition 2 (Timed Protocols). *The set of Timed Protocols, $\mathcal{PL}$, is composed of Timed Protocol Roles, pl, which are constructed by using commands as specified by the following grammar, where b is a boolean expression:*

pl	:=	nil	*Empty Protocol*
	|	(new v), pl	*Fresh Constant*
	|	(+m), pl	*Message Output*
	|	(+m # tc), pl	*Timed Message Output*
	|	(−m), pl	*Message Input*
	|	(−m # tc), pl	*Timed Message Input*
	|	(if b then pl_1 else pl_2)	*Conditional*
	|	(if b # tc then pl_1 else pl_2)	*Timed Conditional*

We explain some examples intuitively before we formalize the semantics of our language in the following section. We will elide nil whenever it is clear from the context.

Example 1. The following program specifies the verifier of a (very simple) distance bounding protocol:

$$(\mathsf{new\ v}), (+\mathsf{v}\ \#\ \mathsf{tt} = \mathsf{cur}), (-\mathsf{v}\ \#\ \mathsf{cur} \le \mathsf{tt} + 4)$$

It creates a fresh constant and sends it to the prover, remembering the current global time by assigning it to the time variable tt. Finally, when it receives the response v it checks whether the current time is less than $\mathsf{tt} + 4$.

Example 2. Timed conditionals can be used to specify the duration of operations, such as checking whether some message is of a given form. In practice, the duration of these operations can be measured empirically to obtain a finer analysis of the protocol [6].

For example, consider the following protocol role:

$$(\mathsf{new\ v}), (+\mathsf{v}), (-\{\mathsf{v}_{enc}, \mathsf{v}_{mac}\}\ \#\ \mathsf{tt}_0 = \mathsf{cur}),$$
$$\mathsf{if}\ (\mathsf{v}_{mac} :=: \mathsf{enc}(\mathsf{v}_{enc}, \mathsf{k}_M))\ \#\ \mathsf{tt}_1 = \mathsf{tt}_0 + \mathsf{tt}_{Mac}$$
$$\mathsf{then}\ (\mathsf{if}\ (\mathsf{v}_{enc} :=: \mathsf{enc}(\mathsf{v}, \mathsf{k}_E))\ \#\ \mathsf{tt}_2 = \mathsf{tt}_1 + \mathsf{tt}_{Enc})$$
$$\qquad \mathsf{then}\ (+done\ \#\ \mathsf{cur} = \mathsf{tt}_2)\ \mathsf{else}\ (+error\ \#\ \mathsf{cur} = \mathsf{tt}_2))$$
$$\mathsf{else}\ (+error\ \#\ \mathsf{cur} = \mathsf{tt}_1)$$

This role creates a fresh value v and sends it. Then it is expecting a pair of two messages v_{mac} and v_{enc}, remembering at time variable tt_0 when this message is received. It then checks whether the first component v_{mac} is of the form $\mathsf{enc}(\mathsf{v}_{enc}, \mathsf{k}_M))$, *i.e.*, it is the correct MAC. This operation takes tt_{mac} time units. The time variable tt_1 is equal to the time $\mathsf{tt}_0 + \mathsf{tt}_{mac}$, *i.e.*, the time when the message was received plus the MAC check duration. If the MAC is not correct, an *error* message is sent exactly at time tt_1. Otherwise, if the first component, v_{MAC}, is as expected, the role checks whether the second component, v_{enc}, is an encryption of the form $\mathsf{enc}(\mathsf{v}, \mathsf{k}_E))$, which takes (a longer) time tt_{enc}. If so it sends the *done* message, otherwise the *error* message, both at time tt_2 which is $\mathsf{tt}_1 + \mathsf{tt}_{enc}$.

We will need to identify a particular command in a Timed Protocol Role. We use a string of the form $i_1.i_2.i_3.\ldots.i_n$, called position and denoted by $\bar{i}$, where each $i_j \in \{1, 2\}$ to specify a path in the control flow of the Timed Protocol. For example, 1.1.1.1.2 in Example 2 leads to $(+error\ \#\ \mathsf{cur} = \mathsf{tt}_1)$. We denote by $\mathcal{PS}(\mathsf{pl})$ the set of strings representing the paths in the Timed Protocol Role pl.

2.2   Timed Strand Spaces and Bundles

We formalize the semantics of Timed Protocols by extending Strand Spaces and Bundles [25] to include time constraints and a network topology.

Network Topology. Messages take time to travel between agents, both honest players and intruders. The network model is specified by representing the time a message needs to travel from any agent a to any agent b, specified by $\mathsf{td}(a, b)$ using

a function that takes two names and returns a number. Typically, $\mathsf{td}(a,a) = 0$, that is the time for a message sent from a player to reach himself is 0, but we do not need to enforce this. We also assume the following axiom for all players $a, a_1, \ldots, a_n, a'$ (with $1 \leq n$):

$$\mathsf{td}(a, a') \leq \mathsf{td}(a, a_1) + \mathsf{td}(a_1, a_2) + \cdots + \mathsf{td}(a_n, a') \tag{1}$$

That is, it is faster for a message to travel directly from a to a', then to first travel through $a_1, \ldots, a_n$. This is similar to the usual triangle inequality in basic geometry.

A given scenario with some protocol session instances includes the protocol participants (or simply participants), $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ and a set of Timed Intruders $\mathcal{I} = \{\mathsf{ti}_1, \ldots, \mathsf{ti}_m\}$, who may be manipulating messages. The Network Topology is composed by two disjoint functions $\mathsf{td} = \mathsf{td}_{\mathcal{P}} \uplus \mathsf{td}_{\mathcal{I}}$ defined as follows:

$$\mathsf{td}(a, b) = \begin{cases} \mathsf{td}_{\mathcal{P}}(a, b) \text{ if } a, b \in \mathcal{P} \\ \mathsf{td}_{\mathcal{I}}(a, b) \text{ otherwise} \end{cases}$$

Thus, $\mathsf{td}_{\mathcal{P}}$ specifies the time messages take to travel among participants, while $\mathsf{td}_{\mathcal{I}}$ specifies the time messages take to travel between Timed Intruders, between a Timed Intruder and a participant and between a participant and a Timed Intruder.

Remark 1. Here we are assuming that two agents share a single transmission channel. We leave to future work how to incorporate different transmission channels. One way to do so is to add another parameter to td, which would imply the addition of more axioms. If multiple transmission channels are allowed, then it may well be the case that Eq. 1 does not hold as some participants might use much faster transmission channels. While we leave a more careful analysis of such cases to future work, we strongly believe that our completeness theorem (Theorem 1) still holds (see Remark 2) as one can assume that intruders communicate among themselves using the fastest available transmission medium.

The following definitions extend Strands and Bundles to include time variables capturing the semantics of Timed Protocols. A Timed Protocol Role is ground if it does not contain variables.

Definition 3. *A Timed Strand Space is a set Π and a trace mapping $tr : \Pi \longrightarrow \mathcal{P} \times \mathcal{GPL}$, where $\mathcal{P}$ is the set of player names $\{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ and $\mathcal{GPL}$ is the set of Ground Timed Protocol Roles. We denote by $tr(s)_1$ the player name and $tr(s)_2$ the Timed Protocol Role of a strand $s \in \Pi$.*

For the remainder we fix a Timed Strand Space $[\Pi, tr]$.

Definition 4. *The Timed Strand Space Graph, $\mathcal{G} = \langle \mathcal{N}, \Rightarrow \cup \rightarrow \rangle$, has nodes $\mathcal{N}$ and edges $\Rightarrow$ and $\rightarrow$ as defined below.*

1. *A node n is a tuple $\langle \mathsf{p}, s, \bar{i} \rangle @\mathsf{tt}$ with $s \in \Pi$, $\mathsf{p} = tr(s)_1$, $\bar{i} \in \mathcal{PS}(tr(s)_2)$ is a string identifying a command in the Timed Protocol, and tt is a time variable timestamping the node n. The set of nodes is denoted by $\mathcal{N}$;*

2. If $n = \langle \mathsf{p}, s, \bar{i} \rangle @\mathsf{tt}$, we denote by $term(n)$, the command at position $\bar{i}$ in $tr(s)_2$;
3. If $n_1 = \langle \mathsf{p}, s, \bar{i} \rangle @\mathsf{tt}_1$ and $n_2 = \langle \mathsf{p}, s, \bar{i}.j \rangle @\mathsf{tt}_2$ are in $\mathcal{N}$, then there is an edge $n_1 \Rightarrow n_2$;
4. For two nodes $\mathsf{n}_1, \mathsf{n}_2 \in \mathcal{N}$, there is an edge $n_1 \to n_2$ if and only if $term(n_1)$ is of the form $+\mathsf{m}$ or $+\mathsf{m} \# \mathsf{tc}_1$ and $term(n_2)$ is of the form $-\mathsf{m}$ or $-\mathsf{m} \# \mathsf{tc}_2$;
5. If a node $n \in \mathcal{N}$, $term(n) = \mathsf{new}\ \mathsf{c}$, then c originates on n, that is, all nodes n' such that $term(n')$ contains c are such that $n\ (\Rightarrow \cup \to)^*\ n'$, where $(\cdot)^*$ is the reflexive and transitive closure operator.

Definition 5. *Let* td *be a Network Topology and let* $\mathcal{C} = \langle \mathcal{N}_\mathcal{C}, \to_\mathcal{C} \cup \Rightarrow_\mathcal{C} \rangle$ *be a subgraph of* $\mathcal{G} = \langle \mathcal{N}, \Rightarrow \cup \to \rangle$. *The Timed Constraint Set of* $\mathcal{C}$ *over* td, *denoted by* $\mathcal{TC}(\mathcal{C}, \mathsf{td})$, *is the smallest set of Time Constraints specified as follows:*

1. *If* $n = \langle \mathsf{p}, s, \bar{i} \rangle @\mathsf{tt} \in \mathcal{N}_\mathcal{C}$, *such that* $term(n)$ *is of the form* $\pm\mathsf{m} \# \mathsf{tc}$ *or if* $\mathsf{b} \# \mathsf{tc}$, *then* $\mathsf{tc}' \in \mathcal{TC}(\mathcal{C}, \mathsf{td})$ *where* tc' *is the Time Constraint obtained by replacing* cur *by* tt;
2. *If* $\langle \mathsf{p}, s, \bar{i} \rangle @\mathsf{tt}_1 \Rightarrow_\mathcal{C} \langle \mathsf{p}, s, \bar{i}.j \rangle @\mathsf{tt}_2$, *then* $\mathsf{tt}_2 \geq \mathsf{tt}_1 \in \mathcal{TC}(\mathcal{C}, \mathsf{td})$;
3. *If* $\langle \mathsf{p}_1, s_1, \bar{i}_1 \rangle @\mathsf{tt}_1 \to_\mathcal{C} \langle \mathsf{p}_2, s_2, \bar{i}_2 \rangle @\mathsf{tt}_2$, *then* $\mathsf{tt}_2 \geq \mathsf{tt}_1 + \mathsf{td}(\mathsf{p}_1, \mathsf{p}_2) \in \mathcal{TC}(\mathcal{C}, \mathsf{td})$.

Intuitively, the $\Rightarrow$ specifies the sequence of actions carried out by a protocol session participant, while the $\to$ specifies the interactions between protocol session participants. However, not all timed strand space graph will correspond to possible executions. Thus, we introduce Timed Bundle which is a subset of the Timed Strand space graph, playing a similar role of Bundles for Strand Spaces.

Definition 6. *Let* td *be a Network Topology. Let* $\to_\mathcal{C} \subseteq \to$ *and* $\Rightarrow_\mathcal{C} \subseteq \Rightarrow$ *and suppose* $\mathcal{C} = \langle \mathcal{N}_\mathcal{C}, \to_\mathcal{C} \cup \Rightarrow_\mathcal{C} \rangle$ *is a sub-graph of* $\langle \mathcal{N}, \Rightarrow \cup \to \rangle$. $\mathcal{C}$ *is a Timed Bundle over* td *if:*

1. $\mathcal{C}$ *is finite and acyclic;*
2. $n_2 \in \mathcal{N}_\mathcal{C}$ *is Message Input or a Timed Message Input, then there is a unique* $n_1 \in \mathcal{N}_\mathcal{C}$ *such that* $n_1 \to_\mathcal{C} n_2$;
3. $n_2 \in \mathcal{N}_\mathcal{C}$ *and* $n_1 \Rightarrow n_2$, *then* $n_1 \in \mathcal{N}_\mathcal{C}$, *and* $n_1 \Rightarrow_\mathcal{C} n_2$;
4. $n = \langle \mathsf{p}, s, \bar{i} \rangle$ *is a node such that* $term(n)$ *is of the form* if b *or* if $\mathsf{b} \# \mathsf{tc}$ *and* b *is evaluated to true, then* $n \Rightarrow_\mathcal{C} \langle \mathsf{p}, s, \bar{i}.1 \rangle$ *and* $n \not\Rightarrow_\mathcal{C} \langle \mathsf{p}, s, \bar{i}.2 \rangle$; *otherwise* $n \Rightarrow_\mathcal{C} \langle \mathsf{p}, s, \bar{i}.2 \rangle$ *and* $n \not\Rightarrow_\mathcal{C} \langle \mathsf{p}, s, \bar{i}.1 \rangle$;
5. *the Timed Constraint Set of* $\mathcal{C}$ *over* td *is satisfiable, i.e., there is a substitution* σ, *called model of* $\mathcal{TC}(\mathcal{C}, \mathsf{td})$, *replacing all time variables in* $\mathcal{TC}(\mathcal{C}, \mathsf{td})$ *by Real numbers so that all inequalities in* $\mathcal{TC}(\mathcal{C}, \mathsf{td})$ *are true.*

Example 3. The following is a graphical representation for a Timed Bundle using the Distance Bounding Protocol described in Example 1:

$$\langle \mathsf{p}_1, \mathsf{new}\ \mathsf{c} \rangle @\mathsf{tt}_1$$
$$\Downarrow$$
$$\langle \mathsf{p}_1, +\mathsf{c} \# \mathsf{tt} = \mathsf{cur} \rangle @\mathsf{tt}_2 \longrightarrow \langle \mathsf{p}_2, -\mathsf{c} \rangle @\mathsf{tt}_4$$
$$\Downarrow \qquad\qquad\qquad\qquad \Downarrow$$
$$\langle \mathsf{p}_1, (-\mathsf{c} \# \mathsf{cur} \leq \mathsf{tt} + 4) \rangle @\mathsf{tt}_3 \longleftarrow \langle \mathsf{p}_2, +\mathsf{c} \rangle @\mathsf{tt}_5.$$
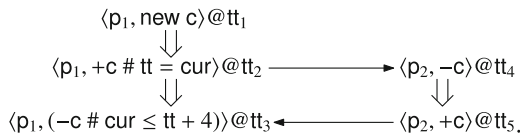
It involves two participants p_1 and p_2 which simply exchange a fresh value c.[1] Its Timed Constraint Set should be satisfiable for the assumed Network Topology specified by the function td:

$$\{\, \mathsf{tt}_5 \geq \mathsf{tt}_4, \mathsf{tt}_3 \geq \mathsf{tt}_2, \mathsf{tt}_2 \geq \mathsf{tt}_1, \mathsf{tt} = \mathsf{tt}_2, \mathsf{tt}_4 \geq \mathsf{tt}_2 + \mathsf{td}(\mathsf{p}_1, \mathsf{p}_2), \mathsf{tt}_3 \geq \mathsf{tt}_5 + \mathsf{td}(\mathsf{p}_2, \mathsf{p}_1), \mathsf{tt}_3 \leq \mathsf{tt} + 4 \,\}$$

Notice that the use of the time symbols in this representation means that this single object specifies a possibly infinite collection of executions of the Distance Bounding Protocol, where the time symbols are instantiated by concrete timestamps taken from the set of non-negative Real numbers $\mathbb{R}^+$. This compact representation greatly reduces the state space during automated protocol verification. In our prototype implementation, we use an SMT solver to check whether the set of Time Constraints is satisfiable or not.

Finally, consider the following specification $(\mathsf{new}\ \mathsf{v}), (-\mathsf{v})$. This specification creates a fresh constant and then expects v as input. Since this is a fresh constant and is never sent, it will never be received. The condition (5) in Definition 4 captures this restriction, as it disallows an $\rightarrow$ edge to $\langle \mathsf{p}, (-\mathsf{v}) \rangle$

3   Timed Intruder Model

The Timed Intruder Model is similar to the usual Dolev-Yao Intruder Model in the sense that it can compose, decompose, encrypt and decrypt messages provided it has the right keys. However, unlike the Dolev-Yao intruder, a Timed Intruder is constrained by the physical properties of the systems, namely, an intruder is not able to learn any message instantaneously, instead, must wait until the message arrives.

A Timed Intruder Set is a set of intruder names $\mathcal{I} = \{\mathsf{ti}_1, \ldots, \mathsf{ti}_n\}$ a set of initially known keys K_P, which contain all public keys, all private keys of all the intruders, all symmetric keys initially shared between intruders and honest players, and may contain "lost keys" that an intruder learned previously by, for instance, succeeding in some cryptoanalysis. Recall that Timed Intruders are situated at locations specified by the Network Topology. For instance, $\mathsf{td}(\mathsf{p}_1, \mathsf{ti}_1) = \mathsf{td}_{\mathcal{I}}(\mathsf{p}_1, \mathsf{ti}_1) = 4$ denotes that the timed needed for a message to travel from participant p_1 to intruder ti_1 is 4.

Definition 7. *An intruder trace is one of the following, where* ti *is a Timed Intruder Name,* $\mathsf{tt}, \mathsf{tt}_1, \mathsf{tt}_2, \mathsf{tt}_3$ *are time variables, and* $\mathsf{m}, \mathsf{m}_1, \ldots, \mathsf{m}_n, \mathsf{m}'_1, \ldots, \mathsf{m}'_p$ *are messages:*

– *Text Message:* $\langle \mathsf{ti}, +\mathsf{t} \rangle @\mathsf{tt}$, *where* t *is a text constant;*
– *Flushing:* $\langle \mathsf{ti}, -\mathsf{m} \rangle @\mathsf{tt}$;

---

[1] For readability we display graph nodes using the player's id paired with the node term, rather than using the strand identifier and trace position.

– *Forward:* $\langle \text{ti}, -\text{m}, +\text{m}\rangle @(\text{tt}_1, \text{tt}_2)$ *denoting the strand* $\langle \text{ti}, -\text{m}\rangle @\text{tt}_1 \Rightarrow \langle \text{ti}, +\text{m}\rangle$ $@\text{tt}_2$;

– *Concatenation:* $\langle \text{ti}, -\{\text{m}_1, \ldots, \text{m}_n\}, -\{\text{m}'_1, \ldots, \text{m}'_p\}, +\{\text{m}_1, \ldots, \text{m}_n, \text{m}'_1, \ldots,$ $\text{m}'_p\}\rangle @(\text{tt}_1, \text{tt}_2, \text{tt}_3)$ *denoting the strand*
$\langle \text{ti}, -\{\text{m}_1, \ldots, \text{m}_n\}\rangle @\text{tt}_1 \Rightarrow \langle \text{ti}, -\{\text{m}'_1, \ldots, \text{m}'_p\}\rangle @\text{tt}_2 \Rightarrow \langle \text{ti}, +\{\text{m}_1, \ldots, \text{m}_n, \text{m}'_1, \ldots, \text{m}'_p\}\rangle @\text{tt}_3$

– *Decomposing:* $\langle \text{ti}, -\{\text{m}_1, \ldots, \text{m}_n\}, +\{\text{m}_1, \ldots, \text{m}_i\}, +\{\text{m}_{i+1}, \ldots, \text{m}_n\}\rangle @(\text{tt}_1, \text{tt}_2,$ $\text{tt}_3)$ *denoting the strand*
$\langle \text{ti}, -\{\text{m}_1, \ldots, \text{m}_i, \text{m}_{i+1}, \ldots, \text{m}_n\}\rangle @\text{tt}_1 \Rightarrow \langle \text{ti}, +\{\text{m}_1, \ldots, \text{m}_i\}\rangle @\text{tt}_2 \Rightarrow \langle \text{ti}, +\{\text{m}_{i+1}, \ldots, \text{m}_n\}\rangle @\text{tt}_3$

– *Key:* $\langle \text{ti}, +\text{k}\rangle @\text{tt}$ *if* $\text{k} \in K_P$;

– *Encryption:* $\langle \text{ti}, -\text{k}, -\text{m}, +\text{enc}(\text{m}, \text{k})\rangle @(\text{tt}_1, \text{tt}_2, \text{tt}_3)$ *denoting the strand*
$\langle \text{ti}, -\text{k}\rangle @\text{tt}_1 \Rightarrow \langle \text{ti}, -\text{m}\rangle @\text{tt}_2 \Rightarrow \langle \text{ti}, +\text{enc}(\text{m}, \text{k})\rangle @\text{tt}_3$

– *Decryption:* $\langle \text{ti}, -\text{k}^{-1}, -\text{enc}(\text{m}, \text{k}), +\text{m}\rangle @(\text{tt}_1, \text{tt}_2, \text{tt}_3)$.
$\langle \text{ti}, -\text{k}^{-1}\rangle @\text{tt}_1 \Rightarrow \langle \text{ti}, -\text{enc}(\text{m}, \text{k})\rangle @\text{tt}_2 \Rightarrow \langle \text{ti}, +\text{m}\rangle @\text{tt}_3$

As with the the usual Dolev-Yao intruder model as, e.g., in [25], the Timed Intruder can send text messages and known keys, receive a message, replay a message, concatenate and decompose messages, and finally encrypt and decrypt messages. There are, however, two differences with respect to the usual Dolev-Yao intruder model as defined in [25]. Each node of the trace is associated with an intruder name ti and a time variable tt. These are necessary for extracting the Time Constraints of a Strand Graph (as described in Definition 5), specifying the physical restrictions of the Timed Intruder.

As the time when timed intruders receive and manipulate messages cannot be measured by the protocol participants, they do not have control over the time variables of timed intruder strands. The following assumption captures this intuition:

Time Variable Disjointness Assumption. For any Bundle $\mathcal{B}$, the set of time variables appearing in protocol participant strands in $\mathcal{B}$ is disjoint from the set of time variables appearing in timed intruder strands in $\mathcal{B}$.

Example 4. Let us return to the distance bounding protocol described in Example 1. The following is an attack, where two colluding intruders ti_1, who is close to p_1, and ti_2, who is close to p_2, collude by sharing a fast channel to fool p_1 into thinking that p_2 is closer than he actually is.
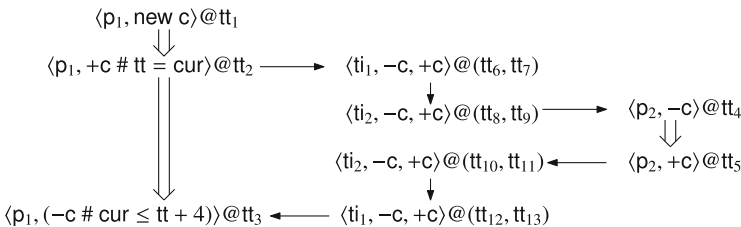
The intruders ti_1 and ti_2 simply forward messages between each other and the players p_1 and p_2. However, this is a Bundle only if the following Time Constraint Set is satisfiable:

$$\left\{ \begin{array}{c} \mathsf{tt}_2 \geq \mathsf{tt}_1, \mathsf{tt} = \mathsf{tt}_2, \mathsf{tt}_6 \geq \mathsf{tt}_2 + \mathsf{td}(\mathsf{p}_1, \mathsf{ti}_1), \mathsf{tt}_7 \geq \mathsf{tt}_6, \mathsf{tt}_8 \geq \mathsf{tt}_7 + \mathsf{td}(\mathsf{ti}_1, \mathsf{ti}_2), \mathsf{tt}_9 \geq \mathsf{tt}_8, \\ \mathsf{tt}_4 \geq \mathsf{tt}_9 + \mathsf{td}(\mathsf{ti}_1, \mathsf{p}_2), \mathsf{tt}_5 \geq \mathsf{tt}_4, \mathsf{tt}_{10} \geq \mathsf{tt}_5 + \mathsf{td}(\mathsf{p}_2, \mathsf{ti}_1), \mathsf{tt}_{11} \geq \mathsf{tt}_{10}, \mathsf{tt}_{12} \geq \mathsf{tt}_{11} + \mathsf{td}(\mathsf{ti}_2, \mathsf{ti}_1), \\ \mathsf{tt}_{13} \geq \mathsf{tt}_{12}, \mathsf{tt}_3 \geq \mathsf{tt}_{13} + \mathsf{td}(\mathsf{ti}_1, \mathsf{p}_1), \mathsf{tt}_3 \leq \mathsf{tt}_2 + 4 \end{array} \right\}$$

This set of constraints represents a set of concrete executions, where the Timed Intruders ti_1 and ti_2 collude. There is a concrete execution only if the set of Time Constraints is satisfiable, which depends on the Network Topology, that is, on the function td.

4   Timed Intruder Completeness

Standard Security Protocol Verification is already very challenging. However, automated verification has been very successful in discovering new attacks. A good part of this success is due to the Dolev-Yao intruder model, which greatly simplifies the design of verification tools. Tools can rely on the important result that just a single Dolev-Yao intruder is enough, in the sense that if there is an attack in the presence of multiple (colluding) Dolev-Yao intruders, then there is also an attack in the presence of a single Dolev-Yao intruder [4].

Unfortunately, for Cyber-Physical Security Protocols, it is not the case that a single Timed Intruder is enough for verification. Consider the attack illustrated in Example 4. There may be a great number of Timed Intruders, but none of them situated between p_1 and p_2, as illustrated by Fig. 1. In such a scenario there might not be an attack as the round time to receive and return a message between such a display of intruders may never be less than the distance bound (4). On the other hand, two strategically placed Timed Intruders, as in the second picture in Fig. 1, may lead to an attack.

Clearly there is an unbounded number of choices based on deciding:

– How many Timed Intruders are there?
– Where are these Timed Intruders located?

This is similar to the challenge in usual security protocol verification of determining how many protocol sessions running in parallel should the scenario have, which is undecidable [20]. Fortunately, we are able to prove a completeness result which answers the two questions above. In order to formalize the completeness statement, we introduce some notation.
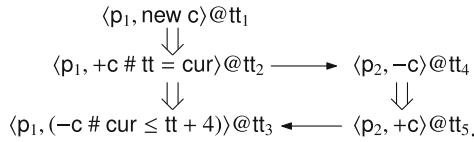
Definition 8. *Let $\mathcal{B}$ be a Timed Bundle over the Network Topology td involving the participants $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ and the Timed Intruders $\mathcal{I} = \{\mathsf{ti}_1, \ldots, \mathsf{ti}_n\}$. The graph $\mathcal{B}$ restricted to participants $\mathcal{P}$, written $\mathcal{B}_{\mathcal{P}}$, is the graph $\langle \mathcal{N}_{\mathcal{B}}^{\mathcal{P}}, (\Rightarrow_{\mathcal{B}}^{\mathcal{P}} \cup \rightarrow_{\mathcal{B}}^{\mathcal{P}}) \rangle$ specified as follows:*

– *$\mathcal{N}_{\mathcal{B}}^{\mathcal{P}}$ contains only the nodes in $\mathcal{B}$ belonging to a participant in $\mathcal{P}$, i.e., of the form $\langle \mathsf{p}, s, \bar{i} \rangle$ where $\mathsf{p} \in \mathcal{P}$;*
– *For two nodes n_1, n_2 in $\mathcal{N}_{\mathcal{B}}^{\mathcal{P}}$, if $n_1 \Rightarrow n_2$ in $\mathcal{B}$, then $n_1 \Rightarrow_{\mathcal{B}}^{\mathcal{P}} n_2$;*

– If n is a node in $\mathcal{N}_{\mathcal{B}}^{\mathcal{P}}$ whose term is a message receive, $-\mathsf{m}$ or $-\mathsf{m}\ \#\ \mathsf{tc}$, and n' is a maximal element of the set of predecessors of n in $\mathcal{N}_{\mathcal{B}}^{\mathcal{P}}$ under the relation $(\Rightarrow \cup \rightarrow)^*; \rightarrow$ then $n' \rightarrow_{\mathcal{B}}^{\mathcal{P}} n$. We let $\mathcal{P}(n, \mathcal{B})$ denote this set of predecessors.
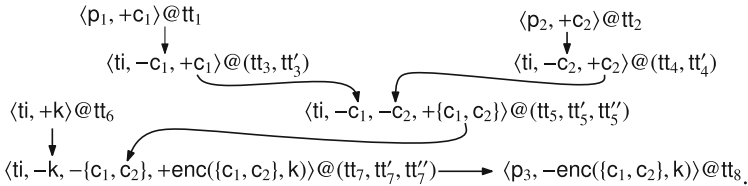
Intuitively, a Bundle restricted to the set of participants specifies the events observable by the participants without including the moves corresponding to the timed intruders. It includes all the edges of the original bundle connecting two nodes of $\mathcal{N}_{\mathcal{B}}^{\mathcal{P}}$. The "maximal predecessor" in $\mathcal{N}_{\mathcal{B}}^{\mathcal{P}}$ is the first element of $\mathcal{N}_{\mathcal{B}}^{\mathcal{P}}$ encountered when following edges in the predecessor direction. It is maximal in the partial order on nodes induced by the edges of the bundle. Thus the terms of nodes in $\mathcal{P}(n, \mathcal{B})$ contain all the terms used by the intruders to derive the term at node n.

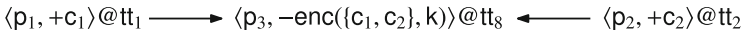The Bundle shown in Example 4 restricted to the participants $\{\mathsf{p}_1, \mathsf{p}_2\}$ is

$$\langle \mathsf{p}_1, \mathsf{new}\ \mathsf{c} \rangle @ \mathsf{tt}_1$$
$$\Downarrow$$
$$\langle \mathsf{p}_1, +\mathsf{c}\ \#\ \mathsf{tt} = \mathsf{cur} \rangle @ \mathsf{tt}_2 \longrightarrow \langle \mathsf{p}_2, -\mathsf{c} \rangle @ \mathsf{tt}_4$$
$$\Downarrow \qquad\qquad\qquad \Downarrow$$
$$\langle \mathsf{p}_1, (-\mathsf{c}\ \#\ \mathsf{cur} \leq \mathsf{tt} + 4) \rangle @ \mathsf{tt}_3 \longleftarrow \langle \mathsf{p}_2, +\mathsf{c} \rangle @ \mathsf{tt}_5.$$

The edge $\langle \mathsf{p}_1, +\mathsf{c}\ \#\ \mathsf{tt} = \mathsf{cur} \rangle @ \mathsf{tt}_2 \rightarrow \langle \mathsf{p}_2, -\mathsf{c} \rangle @ \mathsf{tt}_4$ in this figure simply specifies that using the message, c, sent by p_1, the timed intruders were able to send the message c to the participant p_2.

For another example, consider the following Bundle, where timed intruder ti uses his key $\mathsf{k} \in K_P$ and the messages c_1 and c_2 to compose the message $\mathsf{enc}(\{\mathsf{c}_1, \mathsf{c}_2\}, \mathsf{k})$ to p_3:

$$\langle \mathsf{p}_1, +\mathsf{c}_1 \rangle @ \mathsf{tt}_1 \qquad\qquad\qquad\qquad \langle \mathsf{p}_2, +\mathsf{c}_2 \rangle @ \mathsf{tt}_2$$
$$\downarrow \qquad\qquad\qquad\qquad\qquad\qquad \downarrow$$
$$\langle \mathsf{ti}, -\mathsf{c}_1, +\mathsf{c}_1 \rangle @ (\mathsf{tt}_3, \mathsf{tt}'_3) \qquad\qquad \langle \mathsf{ti}, -\mathsf{c}_2, +\mathsf{c}_2 \rangle @ (\mathsf{tt}_4, \mathsf{tt}'_4)$$
$$\langle \mathsf{ti}, +\mathsf{k} \rangle @ \mathsf{tt}_6 \qquad\qquad \langle \mathsf{ti}, -\mathsf{c}_1, -\mathsf{c}_2, +\{\mathsf{c}_1, \mathsf{c}_2\} \rangle @ (\mathsf{tt}_5, \mathsf{tt}'_5, \mathsf{tt}''_5)$$
$$\langle \mathsf{ti}, -\mathsf{k}, -\{\mathsf{c}_1, \mathsf{c}_2\}, +\mathsf{enc}(\{\mathsf{c}_1, \mathsf{c}_2\}, \mathsf{k}) \rangle @ (\mathsf{tt}_7, \mathsf{tt}'_7, \mathsf{tt}''_7) \longrightarrow \langle \mathsf{p}_3, -\mathsf{enc}(\{\mathsf{c}_1, \mathsf{c}_2\}, \mathsf{k}) \rangle @ \mathsf{tt}_8.$$

The corresponding bundle restricted to the participants $\mathsf{p}_1, \mathsf{p}_2$ and p_3 is:

$$\langle \mathsf{p}_1, +\mathsf{c}_1 \rangle @ \mathsf{tt}_1 \longrightarrow \langle \mathsf{p}_3, -\mathsf{enc}(\{\mathsf{c}_1, \mathsf{c}_2\}, \mathsf{k}) \rangle @ \mathsf{tt}_8 \longleftarrow \langle \mathsf{p}_2, +\mathsf{c}_2 \rangle @ \mathsf{tt}_2$$

It captures the fact that the messages sent by p_1 and p_2 are used to generate the message received by p_3 without explicitly showing how intruders manipulated these messages.

Notice that unlike bundles, a receive node in a restricted bundle may have multiple incoming edges, reflecting the possibility of processing by multiple intruders.

The next two lemmas follow directly from the definition of Bundles and restricted Bundles.

Lemma 1. *Let* $p = \mathsf{n} \rightsquigarrow_1 \mathsf{n}_1 \rightsquigarrow_2 \mathsf{n}_2 \rightsquigarrow_3 \cdots \rightsquigarrow_{j-1} \mathsf{n}_j \rightsquigarrow_j \mathsf{n}'$ *be a path from* n *in* $\mathcal{P}(\mathsf{n}', \mathcal{B})$ *to* n', *where* $\rightsquigarrow_i$ *is either* $\rightarrow$ *or* $\Rightarrow$ *for* $1 \leq i \leq j$. *Then* p *is necessarily of the form:*

$$\langle \mathsf{p}, snd \rangle @\mathsf{tt} \rightarrow \langle \mathsf{ti}_1, s_1 \rangle @\mathsf{tt}_1 \rightsquigarrow_2 \langle \mathsf{ti}_2, s_2 \rangle @\mathsf{tt}_2 \rightsquigarrow_2 \cdots \rightsquigarrow_{j-1} \langle \mathsf{ti}_j, s_j \rangle @\mathsf{tt}_j \rightarrow \langle \mathsf{p}', rcv \rangle @\mathsf{tt}'$$

where snd is a message send $(+\mathsf{m})$ *or a timed message send* $(+\mathsf{m} \# \mathsf{tc})$, *rcv is a message receive* $(-\mathsf{m})$ *or a timed message receive* $(-\mathsf{m} \# \mathsf{tc})$, *and for* $1 \leq i \leq j$, $\langle \mathsf{ti}_i, s_i \rangle$ *are timed intruder strands.*

Lemma 2. *Let* $\mathcal{T}(\mathcal{B}, \mathsf{td})$ *be the Time Constraint Set of* $\mathcal{B}$ *for a given Network Topology* td. *Let* p *be a path in* $\mathcal{B}$ *as described in Lemma 1 of the form:*

$$\langle \mathsf{p}, snd \rangle @\mathsf{tt} \rightarrow \langle \mathsf{ti}_1, s_1 \rangle @\mathsf{tt}_1 \rightsquigarrow_1 \langle \mathsf{ti}_2, s_2 \rangle @\mathsf{tt}_2 \rightsquigarrow_2 \cdots \rightsquigarrow_{j-1} \langle \mathsf{ti}_j, s_j \rangle @\mathsf{tt}_j \rightarrow \langle \mathsf{p}', rcv \rangle @\mathsf{tt}'$$

Then any satisfying model of $\mathcal{T}(\mathcal{B}, \mathsf{td})$ *will also satisfy the constraint:*

$$\mathsf{tt}' \geq \mathsf{tt} + \mathsf{td}(\mathsf{p}, \mathsf{ti}_1) + \mathsf{td}(\mathsf{ti}_1, \mathsf{ti}_2) + \cdots + \mathsf{td}(\mathsf{ti}_{j-1}, \mathsf{ti}_j) + \mathsf{td}(ti_j, \mathsf{p}').$$

The following specifies the equivalence of two Bundles.

Definition 9. *Let* $\mathcal{P}$ *be a set of participants and* $\mathcal{I}, \mathcal{I}'$ *be two possibly equal sets of Timed Intruders. Let* $\mathsf{td}_1 = \mathsf{td}_{\mathcal{P}} \uplus \mathsf{td}_{\mathcal{I}}$ *and* $\mathsf{td}_2 = \mathsf{td}_{\mathcal{P}} \uplus \mathsf{td}_{\mathcal{I}'}$ *be Network Topologies. Then we say that a Timed Bundle* $\mathcal{B}_1$ *over* td_1 *is equivalent to a Timed Bundle* $\mathcal{B}_2$ *over* td_2, *written* $\mathcal{B}_1 \cong_{\mathsf{td}_1}^{\mathsf{td}_2} \mathcal{B}_2$, *if their Bundles restricted to* $\mathcal{P}$ *are (syntactically) identical, i.e.,* $\mathcal{B}_1^{\mathcal{P}} = \mathcal{B}_2^{\mathcal{P}}$.[2]

Intuitively, the condition $\mathcal{B}_1^{\mathcal{P}} = \mathcal{B}_2^{\mathcal{P}}$ specifies that for the honest participants the two Bundles are equivalent, although they may have different timed intruders in different locations manipulating messages in different ways. Thus, if such a $\mathcal{B}_1$ constitutes an attack, then $\mathcal{B}_2$ also constitutes an attack.

Timed Intruder Completeness Problem:

Let $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ be a set of participants and $\mathcal{I} = \{\mathsf{ti}_1, \ldots, \mathsf{ti}_m\}$ be a set of timed intruders. Let $\mathsf{td}_{\mathcal{P}}$ be a Network Topology of the participants. Is there a subset $\mathcal{I}' \subseteq \mathcal{I}$ and $\mathsf{td}_{\mathcal{I}'}$ such that for any $\mathsf{td}_{\mathcal{I}}$ and any Bundle $\mathcal{B}_1$ over $\mathsf{td}_1 = \mathsf{td}_{\mathcal{P}} \uplus \mathsf{td}_{\mathcal{I}}$, there is a Bundle $\mathcal{B}_2$ over $\mathsf{td}_2 = \mathsf{td}_{\mathcal{P}} \uplus \mathsf{td}_{\mathcal{I}'}$ such that $\mathcal{B}_1 \cong_{\mathsf{td}_1}^{\mathsf{td}_2} \mathcal{B}_2$?

In other words, given a particular scenario with $\mathcal{P}$ participants and a Network Topology for these participants $\mathsf{td}_{\mathcal{P}}$, is there a Network Topology $\mathsf{td}_{\mathcal{I}}'$ involving a collection of Timed Intruders $\mathcal{I}'$ that can be used to carry out the same observable events for any other Network Topology $\mathsf{td}_{\mathcal{I}}$ with a possibly larger number of Timed Intruders?

If such an $\mathcal{I}'$ and $\mathsf{td}_{\mathcal{I}'}$ exists then an automated verification tool does not have to guess how many timed intruders there are, and where they are located, but simply can use $\mathcal{I}'$ and $\mathsf{td}_{\mathcal{I}'}$.

---

[2] It is possible to relax this definition so that they are identical modulo time variable names, but this is not needed here.

4.1 Completeness Proof

We are given a set of participants $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$, a set of Timed Intruders $\mathcal{I} = \{\mathsf{ti}_1, \ldots, \mathsf{ti}_m\}$, and a Network Topology $\mathsf{td}_{\mathcal{P}}$ specifying the time messages take to travel between participants.

A Solution for the Timed Intruder Completeness Problem: For our solution, we assume that there are as many timed intruders as participants. If this is not the case, we can safely add more dummy timed intruders. We associate with each participant p_i one Timed Intruder $\mathsf{ti}_{\mathsf{p}_i}$. Thus:

$$\mathcal{I}' = \{\mathsf{ti}_{\mathsf{p}_1}, \ldots, \mathsf{ti}_{\mathsf{p}_n}\}.$$

Moreover, we assume that the time a message takes to travel between p_i to ti_i is 0 (or negligible). Moreover, the time for a message to travel between two Timed Intruders $\mathsf{ti}_{\mathsf{p}_i}$ and $\mathsf{ti}_{\mathsf{p}_j}$ is the same as the time it takes to travel between their corresponding participants p_i and p_j. Thus:

$$\begin{aligned}
\mathsf{td}_{\mathcal{I}'}(\mathsf{p}_i, \mathsf{ti}_{\mathsf{p}_i}) = \mathsf{td}_{\mathcal{I}'}(\mathsf{ti}_{\mathsf{p}_i}, \mathsf{p}_i) &= 0 &&\text{for all } \mathsf{p}_i \in \mathcal{P}; \\
\mathsf{td}_{\mathcal{I}'}(\mathsf{ti}_{\mathsf{p}_i}, \mathsf{ti}_{\mathsf{p}_j}) &= \mathsf{td}_{\mathcal{P}}(\mathsf{p}_i, \mathsf{p}_j) &&\text{for all } \mathsf{p}_i, \mathsf{p}_j \in \mathcal{P}.
\end{aligned}$$

The Timed Intruders in $\mathcal{I}'$ collude in the following form: whenever a Timed Intruder $\mathsf{t}_{\mathsf{p}_i}$ learns a message m sent by p_i, it broadcasts this message m to the remaining Timed Intruders in $\mathcal{I}' \setminus \{\mathsf{ti}_{\mathsf{p}_i}\}$. For example, the Strand for when p_1 sends a message is then as follows:
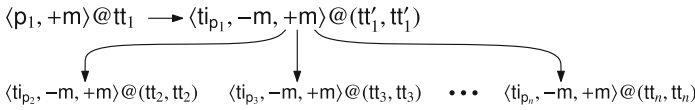
Notice that the message m reaches to a Timed Intruder $\mathsf{ti}_{\mathsf{p}_i}$ at time tt_i which is subject to the Time Constraints $\mathsf{tt}_i \geq \mathsf{tt}'_1 + \mathsf{td}_{\mathcal{P}}(\mathsf{p}_1, \mathsf{p}_i)$ and $\mathsf{tt}'_1 \geq \mathsf{tt}_1 + \mathsf{td}(\mathsf{p}_1, \mathsf{ti}_{\mathsf{p}_1})$, which reduces to $\mathsf{tt}'_1 \geq \mathsf{tt}_1$ as $\mathsf{td}(\mathsf{p}_1, \mathsf{ti}_{\mathsf{p}_1}) = 0$. Thus, $\mathsf{tt}_i \geq \mathsf{tt}_1 + \mathsf{td}_{\mathcal{P}}(\mathsf{p}_1, \mathsf{p}_i)$. Moreover, if the Timed Intruder $\mathsf{ti}_{\mathsf{p}_i}$ forwards this message to the participant p_i, then this message will be received at a time $\mathsf{tt}'_i \geq \mathsf{tt}_1 + \mathsf{td}_{\mathcal{P}}(\mathsf{p}_1, \mathsf{p}_i)$, that is, as if the message had traveled directly from p_1 to p_i without passing through intruders ti_{p_1} and ti_{p_i}

Proof. We will now show that the $\mathcal{I}'$ and $\mathsf{td}_{\mathcal{I}'}$ defined above provide a solution for the Timed Intruder Completeness Problem. For this, assume given a $\mathsf{td}_{\mathcal{I}}$ and a Bundle $\mathcal{B}_1$ over $\mathsf{td}_1 = \mathsf{td}_{\mathcal{P}} \uplus \mathsf{td}_{\mathcal{I}}$.

We will construct a Bundle $\mathcal{B}_2$ over $\mathsf{td}_2 = \mathsf{td}_{\mathcal{P}} \uplus \mathsf{td}_{\mathcal{I}'}$ such that $\mathcal{B}_1 \cong_{\mathsf{td}_1}^{\mathsf{td}_2} \mathcal{B}_2$. We do so by transforming $\mathcal{B}_1$ into $\mathcal{B}_2$.

Let the following be a sub-graph of $\mathcal{B}_1$ restricted to $\mathcal{P}$:
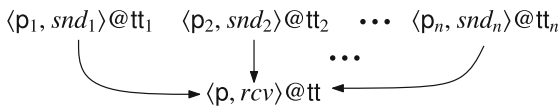
where for all $1 \leq i \leq n$, snd_i is a Message Output $(+m_i)$ or a Timed Message Output $(+m_i \# tc)$, rcv is a Message Input $(-m)$ or a Timed Message Input $(-m \# tc)$.

Let p be an arbitrary path from node $\langle p_i, snd_i \rangle @tt_i$ to $\langle p, rcv \rangle @tt$ path in $\mathcal{B}_1$. From Lemma 1, p has the shape:

$$\langle p_i, snd_i \rangle @tt_i \rightarrow \langle ti_1, s_1 \rangle @tt_1 \rightsquigarrow_1 \langle ti_2, s_2 \rangle @tt_2 \rightsquigarrow_2 \cdots \rightsquigarrow_{j-1} \langle ti_j, s_j \rangle @tt_j \rightarrow \langle p', rcv \rangle @tt$$

Moreover, from Lemma 2, any model satisfying $\mathcal{B}_1$ will also satisfy the constraint:

$$tt \geq tt_i + td(p_i, ti_1) + td(ti_1, ti_2) + \cdots + td(ti_{j-1}, ti_j) + td(ti_j, p). \qquad (2)$$

Given our assumption on the Network Topology (Eq. 1), we also have that

$$td(p_i, p) \leq td(p, ti_1) + td(ti_1, ti_2) + \cdots + td(ti_{j-1}, ti_j) + td(ti_j, p)$$

That is, the time it takes to travel directly from p_i to p is less than or equal to the time it takes to travel from p_i to p via the timed intruders $ti_1, \ldots, ti_j$.

From our solution, we obtain for the sub-graph shown above the following subgraph where all the messages $m_1, \ldots, m_n$ are broadcast to all Timed Intruders including the Timed Intruder ti_p:

$$\langle p_1, +snd_1 \rangle @tt_1 \qquad \cdots \qquad \langle p_n, +snd_n \rangle @tt_n$$
$$\downarrow \qquad\qquad\qquad\qquad\qquad \downarrow$$
$$\langle ti_{p_1}, -m_1, +m_1 \rangle @(tt_1', tt_1') \qquad \cdots \qquad \langle ti_{p_n}, -m_n, +m_n \rangle @(tt_n', tt_n')$$
$$\downarrow \qquad\qquad\qquad\qquad\qquad \downarrow$$
$$\langle ti_p, -m_1, +m_1 \rangle @(tt_1'', tt_1'') \qquad \cdots \qquad \langle ti_p, -m_n, +m_n \rangle @(tt_n'', tt_n'')$$

where the intruder ti_p receives the messages $m_1, \ldots, m_n$. Notice that for $1 \leq i \leq n$, we have that $tt_i'' \geq tt_i + td(p_i, p)$. At this point the intruder ti_p has all the information he needs to compose the message m. Moreover, he can do so without losing time. Thus he is able to deliver the message m to p at time tt satisfying the constraints:

$$tt \geq tt_1 + td(p_1, p) \quad tt \geq tt_2 + td(p_2, p) \quad \cdots \quad tt \geq tt_n + td(p_n, p). \qquad (3)$$

As any model of the Time Constraints Set of $\mathcal{B}_1$ satisfies Eq. 2, the same assignment for $tt_1, \ldots, tt_n, tt$ will also satisfy the time constraints in Eq. 3. Moreover, if any of $snd_1, \ldots, snd_n$ is a Timed Output $(p_i, +m_i \# tc_i)$ or rcv is a Timed Input $(p, -m \# tc)$ the same assignment will also satisfy tc_i and tc because protocol participant strands and timed intruder strands do not share time variable (Time Variable Disjointness Assumption).

By repeating this procedure for each sub-graph in $\mathcal{B}_1$ restricted to $\mathcal{P}$ as shown above, we are able to construct $\mathcal{B}_2$ using $td_{\mathcal{I}'}$ where the only timed intruder strands are those of the intruders $\mathcal{I}'$ leading to the following result.

Theorem 1. *Let $\mathcal{P}$ be participant names and $\mathcal{I}$ be Timed Intruders, such that $|\mathcal{I}| \geq |\mathcal{P}|$. Let $\mathcal{I}'$ and $td_{\mathcal{I}'}$ be as described above. Then $\mathcal{I}'$ and $td_{\mathcal{I}'}$ solve the Timed Intruder Completeness Problem.*

Remark 2. It should be possible to extend our completeness result for cases with different transmission channels (See Remark [1]). For this, our solution would assume that the intruders communicate with the fastest transmission speed available. This is enough to prove Eq. [2] without relying on Eq. [1]. We leave a more careful analysis to future work.

Remark 3. Our solution of placing a timed intruder close to each participant might be unrealistic for some scenarios, *e.g.*, when a participant, A, is guaranteed to be alone due to some physical barrier which ensures that intruders are at least d units away. We speculate that by instead of placing a single intruder for A, we would need n intruders, where n is the number of participants. Each intruder would be placed d units away from A in the direction of each other participant. This should be enough to prove a corresponding completeness theorem. We leave this interesting problem to future work.

5   Examples and Preliminary Experimental Results

We illustrate with some examples that our solution is able to identify attacks on CPSP. We are using the terminology of attacks described in [10].

External Distance Fraud. Assume two honest participants p_1 (Verifier) and p_2 (Prover). They exchange some information, normally to authenticate p_2, for example [24], using a standard Needham-Schroeder-Lowe protocol session [17], and then carry-out a distance bounding protocol session. The following Timed Strand captures the attack where the intruder ti_{p_1} fools player p_1 that p_2 is closer than he actually is by completing the distance bounding challenge:
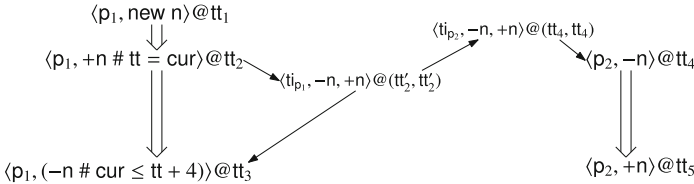
Notice that the timed intruder ti_{p_1} is able to complete the distance bounding session as he is very close to the verifier p_1. This is captured by the Time Constraint Set of this Bundle. Moreover, here we assume that they exchange a nonce, but if we allow equational theories specifying, for example xor operations $\oplus$ as done in [13], a similar Timed Bundle would be obtained.

Attack-in-Between-Ticks. The In-Between-Ticks attack [15] is an instance of a Lone Distance Fraud attack [10], where the prover is dishonest but is not colluding with other Timed Intruders. This attack exploits the fact that real verifiers are running on a processor with a slow clock speed. When the verifier receives the response from the prover, he is only able to record the time of receival in the following clock cycle. This is captured by using the Time Constraint $(floor(cur) + 1)$ as illustrated by the following Timed Strand:

$\langle p_1, \text{new } n \rangle @ tt_1$

$\langle p_1, +n \# tt = \text{floor(cur)} + 1) \rangle @ tt_2 \longrightarrow \langle ti_{p_1}, -n, +n \rangle @ (tt_2, tt_2) \longrightarrow \langle ti_{p_2}, -n, +n \rangle @ (tt_4, tt_4) \longrightarrow \langle p_2, -n \rangle @ tt_4$

$\langle p_1, (-n \# \text{floor(cur)} + 1 \le tt + 4) \rangle @ tt_3 \longleftarrow \langle ti_{p_1}, -n, +n \rangle @ (tt_3, tt_3) \longleftarrow \langle ti_{p_2}, -n, +n \rangle @ (tt_5, tt_5) \qquad \langle p_2, +n \rangle @ tt_5$
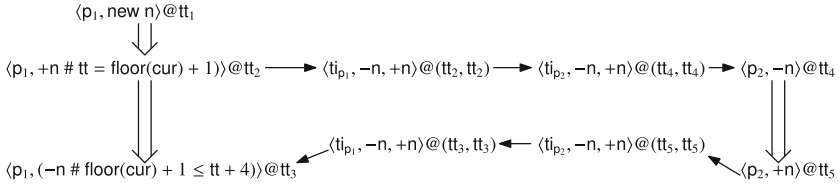
It is possible to show that the Time Constraint Set of this Timed Strand, $\mathcal{T}$, is satisfiable although the distance between p_1 and p_2 is greater than the distance bound 4. That is, it is possible to show that the set $\mathcal{T} \cup \{ \text{td}(p_1, p_2) > 4, \text{td}(p_2, p_1) > 4 \}$ is satisfiable.

Distance Hijacking. In our technical report [21], we show the Timed Bundle with the Distance Hijacking attack described in [24] on the protocol that combines the traditional Needham-Schroeder-Lowe protocol and a distance bounding session.

5.1 Prototype Implementation

We developed a prototype implementation, which can be found at [22], of this strategy in a version of Maude [8] integrated with the SMT solver CVC4 [2]. Our preliminary results seem quite promising.

In addition to symbolic time constraints we implemented a symbolic constraint solver in order to tackle the state-space explosion due to the fact that a timed intruder can generate an unbounded number of messages. It works along the same lines as in usual implementations of such constraint solvers used by tools assuming the standard Dolev-Yao intruder by not instantiating messages generated by the intruder, but rather using symbolic constraints.

Our prototype used and implements mechanisms for the main contributions of this paper:

- **Network Topology as a Constraint Set:** While here we assume that the Network Topology is given by a function td which completely determines the time messages take to travel between agents, our implementation allows the user to specify the Network Topology as a set of constraints. For example, the constraint $\text{td}(p_1, p_2) > 4$ specifies the set of Network Topologies where the time it takes for a message to travel from p_1 to p_2 is greater than 4. This reduces even further the decision choices needed when specifying some scenario as one does not need to consider grounded Network Topologies.
- **Time Variables and Time Constraints:** As described here, we use time variables and keep track of the Time Constraints of the constructed Timed Strand, which is initially empty. Whenever a command in our protocol language is executed, we add the corresponding constraint to the set of constraints following Definition 5. We then call the SMT solver to check whether the set of constraints is satisfiable. If it is not, then search on this branch of the search tree is aborted.

– **Timed Intruders:** Our prototype also implements the solution described in Sect. 4.1 for the configuration of timed intruders. This greatly simplifies the number of decisions needed when specifying a verification scenario. Whenever a message is sent by a participant, his corresponding timed intruder broadcasts this message to all other Timed Intruders. A timed intruder is only able to learn such a message when enough time has elapsed. This is implemented also using the SMT solver and adding appropriate time constraints.

Table 1 summarizes some preliminary experimental results.

Table 1. Preliminary experimental results

Scenario	Size of protocols	No of states	Search time
External Distance Fraud	5	12	31ms
Attack-in-Between-Ticks	5	70	55ms
Simplified Paywave	14	3224	8s
Paywave	22	20807	78s
NSL + Distance Bounding ⋆	15	86	108ms

The External Distance Fraud and Attack-in-Between-Ticks are as described above. The number of states traversed is quite small for finding these. The distance bounding protocol scheme is used by many other protocols, such as the protocol described in [24] (NSL + Distance Bounding) and the lack of its use leads to an attack on the Paywave protocol [6]. We implemented these to check how our tool scales to larger protocols. We implemented a simplified version of the Paywave protocol omitting some of the steps taken and only concentrating on the core part of the protocol. Our tool was able to find the attack in 8s traversing around 3.2k states. Finally, we implemented the whole Paywave protocol and our tool was also able to find the attack, but now in 78s traversing 20.8k states.

The use of the SMT solver was essential to reduce the number of states. However, it seems that it is possible to reduce the overhead caused by each call of the SMT solver.

We also experimented with protocols that fall outside of our language fragment. The NSL + Distance Bounding protocol described in [24] with a small modification carries out a standard Needham-Schroeder-Lowe protocol session, followed by a distance bounding protocol using xor. Since our tool does not support yet equational theories, a subject for future work, we modeled the distance bounding session with a pair. Our tool was able to find a terrorist attack in 108 ms traversing 86 states. This attack was not reported in [24] as they did not assume that intruders are close to the participants.

Finally, we also obtained preliminary results on using the tool for checking whether there is a privacy attack on a protocol [7]. In order to check for such an attack, we need to enumerate all possible executions. (The formal definitions

are out of the scope of this paper.) In order to have an idea of how big this set of executions is, we implemented the protocol used for RFID in European passports. The total number of states was only 10 states. This is a promising result for extending this work to check for properties that rely on observational equivalence [5].

6   Related and Future Work

Meadows *et al.* [19] and Pavlovic and Meadows in [23] propose and use a logic called Protocol Derivation Logic (PDL) to formalize and prove the safety of a number of cyber-physical protocols. In particular, they specify the assumptions and protocol executions in the form of axioms, specifying the allowed order of events that can happen, and show that safety properties are implied by the axiomatization used. They do not formalize an intruder model. Another difference between their work and ours is that their PDL specification is not an executable specification.

Another approach similar to [19], in the sense that it uses a theorem proving approach, is given by Basin *et al.* [3]. They formalize an intruder model that is similar to ours in Isabelle, and also formalize some cyber-physical security protocols. They then prove the correctness of these protocols under some specific conditions and also identify attacks when some conditions are not satisfied. Their work has been a source of inspiration for our intruder model specified in Sect. 3. However, they do not propose or investigate the Timed Intruder Completeness Problem.

Chothia *et al.* [6] investigate empirically the execution times of commands of CPSP which are carried out by limited resource devices and then, based on these measurements, they propose the inclusion of a distance bounding session to mitigate relay attacks. They proved the security of CPSP by modeling the protocol in different phases. As we illustrate in Example 2, our language allows the inclusion of the measurements themselves. We leave a more detailed analysis to future work.

Cheval and Cortier [5] propose a way to prove the observational equivalence with time by reducing it to the observational equivalence based on the length of inputs. They are able to automatically show that RFID protocols used by passports suffer a privacy attack. Their approach is, therefore, different as they do not investigate the Timed Intruder Completeness Problem. Also it is not clear whether from their language one can capture attacks such as the Attack-in-Between Ticks which exploits the time constraints of the verifier. Finally, from our initial experiments with the Passport RFID protocol, we believe that it is also feasible to check for privacy attacks given the very low number of states encountered by our tool. This is left for future work.

Corin *et al.* [9] propose using timed automata to model check security protocols taking into account retransmission and error states of security protocols. This is an early contribution on the analysis of timing aspects of security protocols which did not consider CPSP nor the completeness of timed intruders.

Finally, Malladi *et al.* [18] formalize distance bounding protocols in strand spaces. They then construct an automated tool for protocol verification using a constraint solver to verify a number of examples. There are some similarities between their goals and the goal we want to achieve, namely, the automated verification of CPSP and in the use of SMT solvers to do so. However, there are some important differences. Firstly, we formalize and provide a solution to the Timed Intruder Completeness Problem and, secondly, our language seems to have more expressive features, *e.g.*, our time constraints.

The definition of restricted bundle to characterize executions from the protocol participants perspective is inspired by the notions of skeleton and shape in strand space based protocol analysis [11,12].

Arnaud *et al.* [1] propose a model for specifying and reasoning about secured routing protocols where nodes communicate in a direct way with their neighbors. It seems possible to represent our network model using time constraints as they do and not only reason about the routing of packets, but also the time when these arrive, which is important for cyber-physical systems where agents use some routing protocol to communicate. We leave this to future work.

We are currently investigating methods to control even further the state space explosion, for example, using more elaborate symbolic constraint systems for messages and investigating how to support backward Narrowing as in Maude-NPA [14]. Moreover, we are extending our implementation to support message signatures with equational theories using the library available in Maude [13]. Finally, we are investigating definitions of observational equivalence which involve time and that can be implemented using SMT-solvers.

References

1. Arnaud, M., Cortier, V., Delaune, S.: Modeling, verifying ad hoc routing protocols. Inf. Comput. **238**, 30–67 (2014). Special Issue on Security and Rewriting Techniques
2. Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 171–177. Springer, Heidelberg (2011)
3. Basin, D.A., Capkun, S., Schaller, P., Schmidt, B.: Formal reasoning about physical properties of security protocols. ACM Trans. Inf. Syst. Secur. **14**(2), 16 (2011)
4. Cervesato, I.: Data access specification and the most powerful symbolic attacker in MSR. In: Okada, M., Babu, C.S., Scedrov, A., Tokuda, H. (eds.) ISSS 2002. LNCS, vol. 2609, pp. 384–416. Springer, Heidelberg (2003)
5. Cheval, V., Cortier, V.: Timing attacks in security protocols: symbolic framework and proof techniques. In: Focardi, R., Myers, A. (eds.) POST 2015. LNCS, vol. 9036, pp. 280–299. Springer, Heidelberg (2015)

6. Chothia, T., Garcia, F.D., de Ruiter, J., van den Breekel, J., Thompson, M.: Relay cost bounding for contactless EMV payments. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 189–206. Springer, Heidelberg (2015)
7. Chothia, T., Smirnov, V.: A traceability attack against e-passports. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 20–34. Springer, Heidelberg (2010)
8. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude: A High-Performance Logical Framework. Springer, Heidelberg (2007)
9. Corin, R., Etalle, S., Hartel, P.H., Mader, A.: Timed model checking of security protocols. In: FMSE. ACM (2004)
10. Cremers, C.J.F., Rasmussen, K.B., Schmidt, B., Capkun, S.: Distance hijacking attacks on distance bounding protocols. In: IEEE Symposium on Security and Privacy, SP (2012)
11. Doghmi, S.F., Guttman, J.D., Thayer, F.J.: Searching for shapes in cryptographic protocols. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 523–537. Springer, Heidelberg (2007)
12. Doghmi, S.F., Guttman, J.D., Thayer, F.J.: Skeletons, homomorphisms, shapes, characterizing protocol executions. In: Mathematical Foundations of Program Semantics (2007)
13. Durán, F., Eker, S., Escobar, S., Martí-Oliet, N., Meseguer, J., Talcott, C.: Built-in variant generation and unification, and their applications in Maude 2.7. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNCS, vol. 9706, pp. 183–192. Springer, Heidelberg (2016). doi:10.1007/978-3-319-40229-1_13
14. Escobar, S., Meadows, C.A., Meseguer, J.: Maude-NPA, cryptographic protocol analysis modulo equational properties. In: Aldini, A., Barthe, G., Gorrieri, R. (eds.) FOSAD 2007/2008/2009. LNCS, vol. 5705, pp. 1–50. Springer, Heidelberg (2009)
15. Kanovich, M., Kirigin, T.B., Nigam, V., Scedrov, A., Talcott, C.: Discrete vs. dense times in the analysis of cyber-physical security protocols. In: Focardi, R., Myers, A. (eds.) POST 2015. LNCS, vol. 9036, pp. 259–279. Springer, Heidelberg (2015)
16. Kanovich, M.I., Kirigin, T.B., Nigam, V., Scedrov, A., Talcott, C.L.: Towards timed models for cyber-physical security protocols. Available in Nigam's homepage (2014)
17. Lowe, G.: Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In: Margaria, T., Steffen, B. (eds.) TACAS 1996. LNCS, vol. 1055, pp. 147–166. Springer, Heidelberg (1996)
18. Malladi, S., Bruhadeshwar, B., Kothapalli, K.: Automatic analysis of distance bounding protocols. CoRR, abs/1003.5383 (2010)
19. Meadows, C., Poovendran, R., Pavlovic, D., Chang, L., Syverson, P.F.: Distance bounding protocols, authentication logic analysis and collusion attacks. In: Poovendran, R., Roy, S., Wang, C. (eds.) Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks. Advances in Information Security, vol. 30, pp. 279–298. Springer, New York (2007)
20. Millen, J.K.: A necessarily parallel attack. In: Workshop on Formal Methods and Security Protocols (1999)
21. Nigam, V., Talcott, C., Urquiza, A.A.: Towards the automated verification of cyber-physical security protocols, Bounding the number of timed intruders. CoRR, abs/1605.08563 (2016)
22. Nigam, V., Talcott, C., Urquiza, A.A.: https://github.com/SRI-CSL/VCPublic.git (2016)
23. Pavlovic, D., Meadows, C.: Deriving ephemeral authentication using channel axioms. In: Security Protocols, Workshop, pp. 240–261 (2009)

24. Santiago, S., Escobar, S., Meadows, C.A., Meseguer, J.: Effective sequential protocol composition in Maude-NPA. CoRR, abs/1603.00087 (2016)
25. Thayer, F.J., Herzog, J.C., Guttman, J.D.: Strand spaces: proving security protocols correct. J. Comput. Secur. **7**(1), 191–230 (1999)