Towards Timed Models for Cyber-Physical Security Protocols

Max Kanovich*, Tajana Ban Kirigin†, Vivek Nigam‡, Andre Scedrov§ and Carolyn Talcott¶
* Queen Mary, University of London, UK, Email: mik@dcs.qmul.ac.uk
* University College London, UCL-CS, UK, Email: m.kanovich@ucl.ac.uk
† University of Rijeka, HR, Email: bank@math.uniri.hr
‡ Federal University of Paraíba, Brazil, Email: vivek.nigam@gmail.com
§University of Pennsylvania, USA, Email: scedrov@math.upenn.edu
¶SRI International, USA, Email: clt@csl.sri.com

Abstract*—Many security protocols rely on the assumptions on the physical properties in which its protocol sessions will be carried out. For instance, Distance Bounding Protocols take into account the round trip time of messages and the transmission velocity to infer an upper bound of the distance between two agents. We classify such security protocols as cyber-physical. The key elements of such protocols are the use of cryptographic keys, nonces and time. This paper investigates timed models for the verification of such protocols. Firstly, we introduce a multiset rewriting framework with continuous time and fresh values. We demonstrate that in this framework one can specify distance bounding protocols and intruder models for cyber-physical security protocols that take into account the physical properties of the environment. We then investigate how the models with continuous time relate to models with discrete time in protocol verification and show that there is a difference between these models in exposing security flaws. This is done by proposing a protocol and demonstrating that there is no attack to this protocol when using the model with discrete time, but there is an attack when using the model with continuous time. For the important class of Bounded Memory Cyber-Physical Security Protocols with a Memory Bounded Intruder the reachability problem is PSPACE-complete if the size of terms is bounded.

I. Introduction

With the development of pervasive cyber-physical systems and consequent security issues, it is often necessary to specify protocols that not only make use of cryptographic keys and nonces, but also take into account the physical properties of the environment where its protocol sessions are carried out. We call such protocols cyber-physical security protocols. For instance, Distance Bounding Protocols [5] is a class of cyber-physical security protocols which infers an upper bound on the distance between two agents from the round trip time of messages. In Alice (A) and Bob (B) notation, a session of a Distance Bounding Protocol exchanges two messages:

$$\begin{array}{l} A \longrightarrow B : N_A, A \\ B \longrightarrow A : \{N_A, B\}_{K_A} \end{array}$$

where N_A is a nonce and $\{N_A, B\}_{K_A}$ is the message obtained by encrypting N_A and B with the key K_A shared between Alice and Bob. In order to infer the distance to Bob, Alice remembers the time, t_0, when the message N_A, A was sent, and the time, t_1, when the message $\{N_A, B\}_{K_A}$ returns. From the difference $t_1 - t_0$ and the assumptions on the speed of the transmission medium, v, Alice can compute an upper bound on the distance to Bob, namely $(t_1 - t_0) \times v$.

This is just one example of cyber-physical security protocols. Other examples include Secure Neighbor Discovery, Secure Localization Protocols [6], [27], [29], and Secure Time Synchronization Protocols [15], [28]. The common feature in most cyber-physical security protocols is that they mention cryptographic keys, nonces and time. (For more examples, see [3], [24] and references therein.)

A major problem of the above description of the distance bounding protocol is that many assumptions about time, such as the time requirements for the fulfillment of a protocol session, are not formally specified. For instance, the text describes only informally that Alice remembers the time t_0 and t_1 and which exact moments these correspond to. Moreover, from the above description, it is not clear which assumptions about the network are used. For example, are the participants using some particular medium, *e.g.*, ultrasound or radio waves, or are different messages transmitted using different mediums with different transmission velocities? Furthermore, it is not formally specified which properties does the above protocol ensure and in which conditions and against which intruders.

It is easy to check that the above protocol is not safe against the standard Dolev-Yao intruder [12] which acts as the network capable of intercepting and sending messages anywhere at anytime. In particular, the Dolev-Yao intruder can easily convince A that B is closer than he actually is. The intruder first intercepts the message N_A, A and instantaneously sends it to B. Then he intercepts the message $\{N_A, B\}_{K_A}$ and instantaneously sends it to A. As the round-trip time will correspond to the time that B constructs the message $\{N_A, B\}_{K_A}$, A will believe that B is much closer than he actually is. Such an attack does not occur in practice as messages take time to travel from one point to a different point. Indeed, the standard Dolev-Yao intruder model is not a suitable model for the verification of cyber-physical protocols. Since he is able to intercept and send messages anywhere at anytime, he is faster than the speed of light.

This paper contains some first steps towards building general timed models for cyber-physical security protocols verification. In particular, the contributions of this paper are: (1) We propose a framework where one can specify cyber-physical security protocols and demonstrate its usefulness by specifying Distance Bounding Protocols; (2) We propose an intruder model for cyber-physical protocols; (3) We address the non-trivial relation between models with discrete time and models with continuous time and show that in protocol verification they behave differently. (4) We obtain the PSPACE-complete complexity result for the reachability problem for Bounded Memory Cyber-Physical Security Protocols in presence of a Memory Bounded Intruder.

For our first, second and fourth contributions, we propose a language based on multiset rewriting (MSR) for the specification of cyber-physical security protocols which extends the framework developed for security protocols [8], [13] with continuous time. We demonstrate how one can specify Distance Bounding Protocols in this language. We propose a novel intruder model based on the Dolev-Yao which takes into account the *physical properties* of the environment he is in. We show that such an intruder can be naturally specified in our MSR language with real time.

For our third contribution, we show that there are protocols for which no attack can be found when using a model with discrete time, but there is an attack when using a model with continuous time (or even dense times). This means that, in general, models with discrete time are not as suitable for protocol verification. In particular, for any way we attempt to discretize verification using seconds, miliseconds, etc, there is in principle a way for an attacker to carry out an attack when using the model with continuous time. This novel result illustrates the challenges of verifying timed models for cyber-physical security protocol verification.

Although discrete time might be suitable for some applications such as [25], it is just an abstraction of physical time. In other instances, such as cyber-physical security protocols, normal physical reality plays an essential role. In our formal models we can treat dense time as well as continuous time. Since dense time is rather unnatural, we consider models with continuous time.

This paper is organized as follows: In Section II, we introduce the MSR framework with real time and fresh values and we state our main complexity result. In Section III we explore the powers of discrete and continuous time models in protocol verification. In Section IV we use our model to specify Distance Bounding Protocols and the novel intruder model(s). In Sections V and VI we discuss related and future work. The Appendix contains in detail the execution of a protocol session and of an attack described in [3], as well as a discussion on lower bounds on processing and traversal times.

II. A Multiset Rewriting Framework with Continuous Time and Fresh Values

This section introduces a multiset rewriting (MSR) framework with continuous time and fresh values. Our language is similar to the usual languages used in the MSR literature, see for example [8], [13], [21]. However, for the decidability of the reachability problem dealt in this paper, we will impose some known restrictions [20] on the types of actions (also called rewrite rules, or simply rules) of our system.

A. *Facts, Timestamped Facts, and Configurations*

We assume a finite first-order typed alphabet, Σ, with variables, constants, function and predicate symbols. Terms and facts are constructed as usual (see [14]) by applying symbols with correct type (or sort). For instance, if P is a predicate of type $\tau_1 \times \tau_2 \times \cdots \times \tau_n \to o$, where o is the type for propositions, and $u_1, \ldots, u_n$ are terms with types $\tau_1, \ldots, \tau_n$, respectively, then $P(u_1, \ldots, u_n)$ is a *fact*. A fact is grounded if it does not contain any variables.

In order to specify systems that mention time, we use *timestamped facts* of the form $F@T$, where F is a fact and T is its timestamp. In our previous work [21], timestamps were only allowed to be natural numbers. Here, on the other hand, timestamps are allowed to be non-negative real numbers. We assume that there is a special predicate symbol $Time$ with arity zero, which will be used to represent the global time. A configuration is a multiset of ground timestamped facts, $\{Time@t, F_1@t_1, \ldots, F_n@t_n\}$, with a single occurrence of a $Time$ fact.

Configurations are to be interpreted as states of the system. For example, the following configuration

$$\{Time@7.5, Deadline@10.3, Task(1, done)@5.3, Task(2, pending)@2.13\}$$

specifies that the current global time is 7.5, the Task 1 was performed at time 5.3, Task 2 is still pending and issued at time 2.13, and the deadline to perform all tasks is 10.3. For simplicity we may sometimes denote the timestamp of a fact F in a given configuration as T_F.

B. *Actions and Constraints*

Actions are multiset rewrite rules and are either the Time Advancement Action or Instantaneous Actions. The action representing the advancement of time, called *Tick Action*, is the following:

$$Time@T \longrightarrow Time@(T + \varepsilon) \quad (1)$$

Here ε can be instantiated by any positive real number specifying that the global time of a configuration can advance by any positive number. For example, if we apply this action with $\varepsilon = 0.6$ to the configuration shown above we obtain the following configuration:

$$\{Time@8.1, Deadline@10.3, Task(1, done)@5.3, Task(2, pending)@2.13\}$$

where the global time advanced from 7.5 to 8.1.

Clearly such an action is a source of unboundedness as time can always advance by any positive real number. In particular we will need to deal with issues such as Zeno Paradoxes when considering how time should advance.

The remaining actions are the Instantaneous Actions, which do not affect the global time, thus instantaneous, but may rewrite the remaining facts. They have the following shape:

$$Time@T, W_1@T_1, \ldots, W_k@T_k, F_1@T_1', \ldots, F_n@T_n' \mid \mathcal{C} \longrightarrow \\ \exists \vec{X}.[Time@T, W_1@T_1, \ldots, W_k@T_k, Q_1@(T+D_1), \ldots, Q_m@(T+D_m)]$$

where $D_1, \ldots, D_m$ are natural numbers and $\mathcal{C}$ is the guard of the action which is a set of constraints involving the time variables appearing in the pre-condition, *i.e.*, the variables $T, T_1, \ldots, T_k, T_1', \ldots, T_n'$.

Constraints are of the form:[1]

$$T > T' \pm D \quad \text{and} \quad T = T' \pm D \tag{2}$$

where T and T' are time variables, and D is a natural number. An instantaneous action can only be applied if all the constraints in its guard are satisfied.

Notice that the global time does not change when applying an instantaneous action. Moreover, the timestamps of the facts that are created or whose timestamp is updated by the action, namely the facts $Q_1, \ldots, Q_m$, are of the form $T + D_i$, where D_i is a natural number and T is the global time. That is, their timestamps are in the present or the future.

For example, the following is an instantaneous action

$$Time@T, Task(1, done)@T_1, Deadline@T_2, Task(2, pending)@T_3 \mid \{T_2 \geq T + 2\} \\ \longrightarrow Time@T, Task(1, done)@T_1, Deadline@T_2, Task(2, done)@(T+1)$$

which specifies that one should complete Task 2, if Task 1 is completed, and moreover, if the Deadline is at least 2 units ahead of the current time. If these conditions are satisfied, then the Task 2 will be completed in one time unit. It is easy to check that this action can be applied to the configuration above, resulting in the following configuration:

$$\{Time@8.1, Deadline@10.3, Task(1, done)@5.3, Task(2, done)@9.1\}$$

where Task 2 will be completed by the time 9.1.

Finally, the variables $\vec{x}$ that are existentially quantified in the above action are to be replaced by fresh values, also called *nonces* in protocol security literature [8], [13]. For example, the following action specifies the creation of a new task with a fresh identifier id, which should be completed by time $T + D$:

$$Time@T \longrightarrow \exists Id.[Time@T, Task(Id, pending)@(T+D)]$$

Whenever this action is applied to a configuration, the variable Id is instantiated by a fresh value. In this way we are able to specify that the identifier assigned to the new task is different to the identifiers of all other existing tasks. In the same way it is possible to specify the use of nonces in Protocol Security [8], [13], as we do in Section IV.

Notice that by the nature of multiset rewriting there are various aspects of non-determinism in the model. For example, different actions and or even different instantiations of the same rule may be applicable to the same configuration $\mathcal{S}$, which may lead to different resulting configurations $\mathcal{S}'$.

Motivation for the use Natural Numbers D_i in Constraints and Actions As the specification of cyber-physical security protocols does not explicitly mention real numbers, such as the number π, but at most rational numbers, we can model these systems by using the Natural Numbers D_i in actions and constraints by normalizing the rational numbers mentioned in the system to obtain natural numbers.

Motivation for Instantaneous Actions Intuitively, the conditions on Instantaneous Actions, namely the constraints format and the restriction on the timestamps of the facts created by an action, specify systems that are not affected by time shifts [21]. That is, if we shift the timestamps of all facts in the configuration by the same value, the same actions still apply. For a more technical reason, these conditions are also necessary conditions for the decidability of the reachability problem *already for systems with discrete-time*. In particular, it is possible to show that if we allow actions with more relaxed constraints and/or actions that assign more complicated timestamps to created facts, then the reachability problem is undecidable [21].

C. Initial, Goal Configurations and The Reachability Problem

We write $\mathcal{S} \longrightarrow_r \mathcal{S}_1$ for the one-step relation where the configuration $\mathcal{S}$ is rewritten to $\mathcal{S}_1$ using an instance of action r. For a set of actions $\mathcal{R}$, we define $\mathcal{S} \longrightarrow_{\mathcal{R}}^* \mathcal{S}_1$ as the transitive reflexive closure of the one-step relation on all actions in $\mathcal{R}$. We elide the subscript $\mathcal{R}$, when it is clear from the context.

A *goal* $\mathcal{S}_G$ is a pair of a multiset of facts and a set of constraints:

$$\{F_1@T_1, \ldots, F_n@T_n\} \mid \mathcal{C}$$

where $T_1, \ldots, T_n$ are time variables, $F_1, \ldots, F_n$ are ground facts and $\mathcal{C}$ is a set of constraints involving only $T_1, \ldots, T_n$. We call a configuration $\mathcal{S}_1$ a *goal configuration* if there is a substitution σ replacing $T_1, \ldots, T_n$ by real numbers such that $\mathcal{S}_G\sigma \subseteq \mathcal{S}_1$ and all comparisons in $\mathcal{C}\sigma$ are satisfied. The reachability problem, $\mathcal{T}$, is then defined for a given initial configuration $\mathcal{S}_I$, a goal $\mathcal{S}_G$ and a set of actions $\mathcal{R}$ as follows:

Reachability Problem: Is there a goal configuration $\mathcal{S}_1$, such that $\mathcal{S}_I \longrightarrow_{\mathcal{R}}^* \mathcal{S}_1$?

Such a sequence of actions is called a *plan*. We assume that goals are invariant to nonce renaming, that is, a goal $\mathcal{S}_G$ is equivalent to the goal $\mathcal{S}_G'$ if they only differ on the nonce names (see [16] for more discussion on this).

D. Dealing with Continuous Time

In our technical report [17], we investigate the complexity of the reachability problem described above. The main challenge there is to deal with continuous time, *e.g.*, handle Zeno paradoxes. We introduced a novel machinery, called circle-configurations, to symbolically represent configurations and plans that explicitly mention real time. This abstraction is shown to be sound and complete. Therefore, we can search for solutions of the reachability problems symbolically, that is, within a finite space of circle-configurations. We no longer need to manipulate the infinite space of real numbers, *i.e.*, the concrete values of the timestamps in a plan. It also allows us to use/adapt existing machinery involving Multiset Rewriting, *e.g.*, [8], [13], [16], [21], to infer complexity results.

The details on this solution and the proofs of these claims can be found in [17]. Here we only state the main complexity

[1]We use $T' \geq T' \pm D$ to denote the disjunction of $T > T' \pm D$ and $T' = T' \pm D$

result. It mentions the condition that actions should be balanced, that is, all actions have the same number of facts in its pre and post-conditions (see [16] for more details).

Theorem 2.1: Let $\mathcal{T}$ be a reachability problem with balanced actions. Then $\mathcal{T}$ is in PSPACE with respect to m, k, and D_{max}, where m is the number of facts in the initial configuration, k is an upper bound on the size of facts, and D_{max} is an upper bound on the numbers appearing in $\mathcal{T}$.

For the result above, we assumed that the time values, namely the Ds in the rewrite rules and constraints, are given by the reachability problem. However, another problem that we are considering is the following: Is there a way of instantiating the Ds, that allow us to reach a goal configuration from a given initial configuration?

For some applications, such as for determining whether a distance bounding protocol has an attack, one can constraint the values of D: They do not have to be much greater than the time that a message takes to travel the distance bound specified by the protocol (twice this value is enough), which means that we can infer D_{max} from the protocol specification. Therefore, if our rewrite rules are balanced and assuming the upper bounds stated in the theorem above, then we can enumerate all the possible values of the Ds and check for each instance whether a goal configuration is reachable from a given initial configuration. This means that this problem is decidable when such constraints on the Ds are available. We leave for future work the task of finding other conditions for decidability and their lower bounds.

III. Continuous versus Discrete Time

This section investigates the motivation and the need of using continuous time models in protocol verification instead of the more simple discrete time models. We show that there is indeed a difference between these models. Namely, the model with discrete time is strictly less expressive than the model with continuous time in the sense that it is not able to expose as many protocol security flaws.

We address the basic issues as they arise in the formalization of protocols with explicit time:

- The time-sensitive features such as the *network delays* and *participants' processing time* are taken into account.
- Protocol execution depends on the round trip time of messages by means of *measuring the response time*. In particular, a participant has an opportunity to authenticate another one with the help of the corresponding time response.[2]

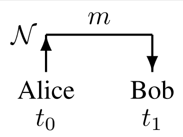
We illustrate the main subtleties by adding the time dimension to the original Needham-Schroeder protocol. The intriguing result is that this protocol is *secure* in the model with discrete time, but it is *insecure* in the model with continuous time already with respect to an adversary which is able to intercept and send messages, as well as encrypt and decrypt messages providing he has the corresponding keys. Such an adversary does not need to manipulate various submessages or even create fresh values. Here, as all the participants in the protocol execution, the adversary is subject to non-zero network delays and non-zero processing time. Although one could consider a more simple example, with instantaneous processing times, we opted for this quite realistic one. In the next section we propose a slightly different intruder model and discuss possible extensions and alternative models.

[2]This is at the basis of distance-bounding protocols, see Section I.

- *Passing messages takes non-zero time.*

Agents communicate by *sending* and *receiving* messages m through the network $\mathcal{N}$. Even in protocol formalizations with no explicit time, there is a certain causality relation, namely the *receiving* event is caused by the corresponding *sending* event. In addition, here we have to deal with a possible time delay between these events.

We formalize *sending* a message m on the network $\mathcal{N}$ as $\mathcal{N}_S(m)$, and *receiving* m from $\mathcal{N}$ as $\mathcal{N}_R(m)$. Accordingly, $\mathcal{N}_S(m)@t_0$ stands for: "*m is sent on $\mathcal{N}$ at moment* t_0," and $\mathcal{N}_R(m)@t_1$ stands for: "*m is consumed from $\mathcal{N}$ at moment t_1*".



The fact that such a communication act, *i.e.* message delivery, is not instantaneous can be axiomatized by a multiset rewriting rule of the form:

$$Time@t_1,\ \mathcal{N}_S(m)@t_0 \mid \{\, t_1 > t_0 \,\} \longrightarrow Time@t_1,\ \mathcal{N}_R(m)@t_1 \tag{3}$$

where the attached time constraint $t_1 > t_0$ formalizes the network delay.

Notice that in the case of discrete time, $t_1 \geq t_0 + 1$.

- *Processing requests takes non-zero time.*

When an agent X receives a message m' at some moment t_1, he first stores m', and then starts to process m' to produce the output m''. For instance, he sends m'' on the network *at the later moment* $t_2 > t_1$, Here $t_2 - t_1$ represents X's processing time. We simulate these events with the following two rules:

$$\begin{array}{l} Time@t_1,\ \mathcal{N}_R(m')@t_1 \longrightarrow Time@t_1,\ X(m')@t_1, \\ Time@t_2,\ X(m')@t_1 \mid \{t_2 > t_1\} \longrightarrow \\ \qquad Time@t_2,\ X(m'')@t_2,\ \mathcal{N}_S(m'')@t_2. \end{array} \tag{4}$$

where for a participant X, the fact $X(z)@t$ denotes that *z is stored in X's memory at moment t*, and this information is included in X's private knowledge base.

- *"Continuous time"* versus *"Discrete time"*.

A *global continuous measurable quantity* `time` is assumed in which events occur in irreversible succession.

For the continuous time, we formalize the *time advance* with the following multiset rewriting rule:

$$Time@t \longrightarrow Time@(t+\varepsilon) \tag{5}$$

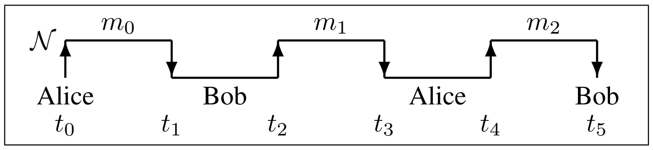
where ε is an arbitrary non-negative real number.

Discrete time views events as occurring at time moments which are multiples of a fixed 'time unit', so that `time` can be conceived of as be assigned only *integer values*. For the discrete time, the *time advance* is given by the rule

$$Time@t \;\longrightarrow\; Time@(t{+}1) \tag{6}$$

That is, time advances by one per tick.

We illustrate the main subtleties of time-sensitive security protocols and their formalization by proposing the following version of the *time-bounding Needham-Schroeder protocol*:



(a) *At some moment* t_0, Alice creates her nonce N_A, stores N_A in her memory, and sends to Bob (*in fact, on the network* $\mathcal{N}$) the message $m_0 = \{N_A, A\}_{K_B^{pub}}$, encrypted with Bob's public key K_B^{pub}.
We formalize this action with the multiset rewriting rule:

$$\begin{array}{l} Time@t_0 \;\longrightarrow \\ \quad \exists N_A\Big(Time@t_0,\; A(N_A)@t_0,\; \mathcal{N}_S(m_0)@t_0\Big) \end{array} \tag{7}$$

The postcondition of the rule reads that, in addition to sending the message m_0 on $\mathcal{N}$, Alice keeps, in her private knowledge base, the fact $A(N_A)@t_0$, which denotes that N_A is stored in her memory at moment t_0.

(b) When Bob receives such a message m_0, *at some moment* t_1, he first memorizes m_0, which is formalized as:

$$Time@t_1, \mathcal{N}_R(m_0)@t_1 \longrightarrow Time@t_1, B(m_0)@t_1. \tag{8}$$

Having received the above message from Alice, Bob decrypts it, creates his nonce N_B, stores N_A and N_B in his memory, and *at some later moment* t_2 sends to Alice the message $m_1 = \{N_A, N_B\}_{K_A^{pub}}$ encrypted with Alice's public key K_A^{pub}. Formally,

$$\begin{array}{l} Time@t_2,\; B(m_0)@t_1 \mid \{t_2 > t_1\} \;\longrightarrow \\ \quad \exists N_B\Big(Time@t_2,\; B(\langle N_A, N_B\rangle)@t_2,\; \mathcal{N}_S(m_1)@t_2\Big) \end{array} \tag{9}$$

Here $t_2 - t_1$ represents Bob's processing time.

(c) When Alice receives message m_1 *at some moment* t_3, she first stores m_1, which is formalized as:

$$Time@t_3, \mathcal{N}_R(m_1)@t_3 \longrightarrow Time@t_3, A(m_1)@t_3 \tag{10}$$

Then, in order to *authenticate Bob* and confirm Bob's request, she checks *the response time*: $t_3 - t_0$. If $t_3 - t_0 \leq 3$, and if nonces match, she sends to Bob the 'confirmation message' $m_2 = \{N_B\}_{K_B^{pub}}$ *at the later moment* t_4, where $t_4 - t_3$ is Alice's processing time. Formally:

$$\begin{array}{c} Time@t_4, A(N_A)@t_0, A(\{N_A, N_B\}_{K_A^{pub}})@t_3 \\ \mid\; \{t_3 - t_0 \leq 3,\; t_4 > t_3\} \;\longrightarrow \\ Time@t_4,\; A(\{N_A, N_B\}_{K_A^{pub}})@t_3,\; \mathcal{N}_S(m_2)@t_4 \end{array} \tag{11}$$

Otherwise, Alice sends the message $\{N_0\}_{K_B^{pub}}$ with a 'garbage nonce' N_0 to signal that N_A and N_B should not be accepted this time.

The protocol is declared *secure* if *the 'accepted'* N_A *and* N_B *may never be revealed to somebody else except Alice and Bob.*

There is a clear correspondence between protocol sessions and sequences of rewriting rules applied. E.g., let a session be the following:

(a) *At some moment* t_0, let Alice initiate a session with Bob by sending the message $m_0 = \{N_A, A\}_{K_B^{pub}}$ on the network.
(b) *At moment* $t_1 = t_0 + 1$, the message m_0 arrives to Bob;
(c) Bob stores Alice's nonce, creates his nonce and responds at time $t_2 = t_0 + 2$ with the message $m_1 = \{N_A, N_B\}_{K_A^{pub}}$
(d) *At moment* $t_3 = t_0 + 3$, the message m_1 arrives to Alice. Since $t_3 - t_0 \leq 3$, Alice responds to Bob with $m_2 = \{N_B\}_{K_B^{pub}}$.

Then the above session can be simulated with the sequence of our rewriting rules:

(a) First, we apply the rule (7) with sending m_0 on the $\mathcal{N}$:

$$Time@t_0 \longrightarrow Time@t_0,\; A(N_A)@t_0,\; \mathcal{N}_S(m_0)@t_0$$

(b) Time is ticking with (6):

$$Time@t_0 \longrightarrow Time@t_1$$

Then m_0 arrives to Bob by means of (3):

$$Time@t_1,\; \mathcal{N}_S(m_0)@t_0 \longrightarrow Time@t_1,\; \mathcal{N}_R(m_0)@t_1$$

(c) Bob stores m_0 in accordance with (8):

$$Time@t_1, \mathcal{N}_R(m_0)@t_1 \longrightarrow Time@t_1, B(m_0)@t_1.$$

While Bob works on his reply, time ticks again

$$Time@t_1 \longrightarrow Time@t_2$$

At moment t_2 Bob responds with m_1 as per (9).

(d) Time is ticking with (6):

$$Time@t_2 \longrightarrow Time@t_3$$

Message m_1 arrives to Alice by means of (3):

$$Time@t_3,\; \mathcal{N}_S(m_1)@t_2 \longrightarrow Time@t_3,\; \mathcal{N}_R(m_1)@t_3$$

Alice stores m_1 as per (10):

$$Time@t_3, \mathcal{N}_R(m_1)@t_3 \longrightarrow Time@t_3, A(m_1)@t_3$$

Time is ticking with (6):

$$Time@t_3 \longrightarrow Time@t_4$$

Then, since $t_3 - t_0 \leq 3$, Alice responds with m_2 in accordance with (11).

The Dolev-Yao intruder here is a participant with the standard Dolev-Yao capabilities who, in addition, can choose the moment to send his messages on the network.

Theorem 3.1: In the case of discrete time with the time advancement rule: $Time@t \rightarrow Time@(t{+}1)$, there is no Dolev-Yao attack on the timed Needham-Schroeder protocol.

Proof Sketch. Let t_0 be a moment when Alice sends a message of the form $\{N_A, A\}_{K_M^{pub}}$.

Alice can receive a message of the form $\{N_A, N''\}_{K_A^{pub}}$ at some moment t_3 such that necessarily $t_3 \geq t_0{+}3$.

Alice reveals a questionable message of the form $\{N''\}_{K_M^{pub}}$ only if $t_3{-}t_0 \leq 3$, which implies $t_3 \leq t_0{+}3$.

Hence, $t_3 = t_0{+}3$.

Therefore, M has no time to initiate any activity with Bob rather than to respond to Alice with $\{N_A, N''\}_{K_A^{pub}}$, where N'' is M's own nonce, not Bob's.

Theorem 3.2: In contrast, in the case of continuous time with the time advancement rule: $Time@t \to Time@(t{+}\varepsilon)$, there is a timed version of Lowe attack on the timed Needham-Schroeder protocol.

Proof. A possible attack scenario is as depicted in Figure 1:

- *At some moment* t_0, let Alice initiate a session with Mallory by sending the message $z_0 = \{N_A, A\}_{K_M^{pub}}$ on the network.
- At some moment, say $t_1 = t_0{+}0.1$, the message z_0 arrives to Mallory, and, at the moment $t_2 = t_1{+}0.1$, he sends to Bob the re-encrypted message $z_1 = \{N_A, A\}_{K_B^{pub}}$.
- At some moment, say $t_3 = t_2{+}0.1$, the message z_1 arrives to Bob, and at the moment $t_4 = t_3{+}0.1$, Bob responds with the message $z_2 = \{N_A, N_B\}_{K_A^{pub}}$
- Having received the message z_2 at some moment, say $t_5 = t_4 + 0.1$, Mallory resends this message to Alice at a later moment, say $t_6 = t_5 + 0.1$
- At some moment, say $t_7 = t_6 + 0.1$, message z_2 arrives to Alice.
  Since $t_7{-}t_0 \leq 3$, Alice responds to Mallory with $z_3 = \{N_B\}_{K_M^{pub}}$.
- Having received the message z_3, Mallory sends to Bob the 'confirmation' $z_4 = \{N_B\}_{K_B^{pub}}$, to force Bob to believe that *he communicated with Alice, and nobody else learned* N_B.

Remark 3.3: Here, the actual difference between discrete time and continuous time is that, between the moments t_0 and $t_0{+}3$, only three acts could have happened within discrete time, whereas an unbounded number of timed events are possible within continuous time.

Remark 3.4: No rescaling of discrete time units removes this issue. Namely, for any discretization of time, such as days, seconds or any other infinitesimal time unit, there is a protocol similar to the one given above, where there is an attack with continuous time and no attack is possible in the discrete case.

Remark 3.5: Notice that discrete time models implicitly impose lower bounds on transmission and processing time, see Remark 3.3. This is not the case in models with continuous time. Indeed, continuous time (or even dense time) allows us to not have such bounds. Nevertheless, lower bounds for delays for both processing time and for traversal time can be introduced in continuous time models (see Appendix C). However, one should question what such a bound would represent when considering *e.g.* various network distances, available and future technologies effecting both processing and traversal times. That is why in the above foundational comparison between discrete and continuous time models we intentionally did not impose lower bounds. Our specification is not driven by implementation.

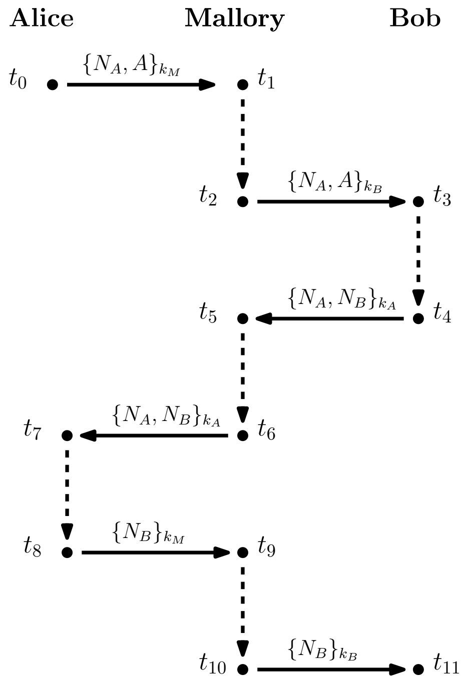


Fig. 1: Lowe style timed attack on the timed Needham-Schroeder protocol.

IV. Specifying Cyber-Physical Security Protocols

This section shows how one can specify cyber-physical security protocols. As an example we use the simple Distance Bounding Protocol described in the Introduction. Firstly we detail in Section IV-A the specification of Protocol Theories following the notation in [8], [13], [16]. Then in Section IV-B we describe an intruder model similar to the Dolev-Yao model that takes into account the physical properties of the environment.

In the following, assume that there are n agents. These agents are either honest participants, intruders, or verifiers. A verifier V is an agent that possesses a resource, such as access to a building, which may be granted to some other agent. For communication, agents may use different transmission mediums with possibly different transmission velocities, such as radio-frequency or ultrasound.

The verifier only grants its resource if following his challenge (message N_A, A in the Introduction) he receives a reply (message $\{N_A, B\}_{K_A}$ in the Introduction) from the requesting agent before some reference period of time has elapsed, called *distance bounding time*. The distance bounding time is given by the specification of the protocol and it specifies the upper bound on the distance at which the requesting agent should be positioned to have access to the verifier's resource.

The verification problem is to check whether an agent that does not comply with the distance bounding requirement cannot have access to a verifier's resource.

Rol Gen: $Time@T, Ver(A)@T_1, Ag(B)@T_2, P@T_3, P@T_4 \longrightarrow$
$Time@T, Ver(A)@T_1, Ag(B)@T_2, A_0(A, B, c_S, c_R)@T, B_0(A, B, c_S, c_R)@T$

Challenge: $Time@T, A_0(A, B, C_S, C_R)@T_1, P@T_2 \longrightarrow \exists N_A.[Time@T, A_1(A, B, C_R, N_A)@T, S_{A,C_S}(\langle N_A, A\rangle)@T]$

Response: $Time@T, Key(A, K_A)@T_1, B_0(A, B, C_S, C_R)@T_2, S_{A,C_S}(\langle N_A, A\rangle)@T_3 \mid T \geq T_3 + D(A, B, C_S) \longrightarrow$
$Time@T, Key(A, K_A)@T_1, B_1(A, B, N_A)@T, S_{B,C_R}(\{\langle N_A, B\rangle\}_{K_A})@T$

Accept: $Time@T, A_1(A, B, C_R, N_A)@T_1, S_{B,C_R}(\{\langle N_A, B\rangle\}_{K_A})@T_2 \mid T_1 + D_b^A \geq T \geq T_2 + D(B, A, C_R) \longrightarrow$
$Time@T, A_2(A, B, yes)@T, P@T$

Reject: $Time@T, A_1(A, B, C_R, N_A)@T_1, S_{B,C_R}(\{\langle N_A, B\rangle\}_{K_A})@T_2 \mid T > T_1 + D_b^A, T \geq T_2 + D(B, A, C_R) \longrightarrow$
$Time@T, A_2(A, B, no)@T, P@T$

Del A_2: $Time@T, A_2(A, B, X)@T_1 \longrightarrow Time@T, P@T$

Del B_1: $Time@T, B_1(A, B, M)@T_1 \longrightarrow Time@T, P@T$

Del $S_{i,c}$: $Time@T, S_{i,c}(M)@T_1 \longrightarrow Time@T, P@T$

Fig. 2: Bounded Memory Protocol Rules. Here c_S and c_R are constants specifying the transmission mediums used, respectively, to send the challenge and receive it.

I/O Rules:

REC: $Time@T, Int(I)@T_1, Cap(I, C)@T_2, S_{A,C}(X)@T_3 \mid T \geq T_3 + D(A, I, C) \longrightarrow$
$Time@T, Int(I)@T_1, Cap(I, C)@T_2, M_I(X)@T$

SND: $Time@T, Int(I)@T_1, Cap(I, C)@T_2, M_I(X)@T_3 \mid T \geq T_3 \longrightarrow$
$Time@T, Int(I)@T_1, Cap(I, C)@T_2, S_{I,C}(X)@T$

Decomposition and Composition Rules:

DCMP: $Time@T, M_I(\langle X, Y\rangle)@T_1, P@T_2 \longrightarrow Time@T, M_I(X)@T, M_I(Y)@T$

DEC: $Time@T, M_I(K)@T_1, M_I(\{X\}_K)@T_2, P@T_3 \longrightarrow Time@T, M_I(K)@T_1, M_I(\{X\}_K)@T_2, M_I(X)@T$

COMP: $Time@T, M_I(X)@T_1, M_I(Y)@T_2 \rightarrow Time@T, M_I(\langle X, Y\rangle)@T, P@T$

USE: $Time@T, M_I(X)@T_1, P@T_2 \rightarrow Time@T, M_I(X)@T_1, M_I(X)@T$

ENC: $Time@T, M_I(K)@T_1, M_I(X)@T_2, P@T_3 \rightarrow Time@T, M_I(K)@T_1, M_I(X)@T_2, M_I(\{X\}_K)@T$

GEN: $Time@T, P@T_1 \rightarrow \exists N.Time@T, M_I(N)@T$

Memory Maintenance Rule:

DELM: $Time@T, M_I(X)@T_1 \rightarrow Time@T, P@T$

Fig. 3: Bounded Memory Dolev-Yao Adversary Theory.

There are, in fact, two properties that need to be checked: the authentication property, *i.e.*, checking whether the intruder cannot convince the verifier that he is an honest participant, and the distance bounding requirement, described above. As the first problem does not involve the physical properties of the system, standard techniques [8], [13], [16] which can be modeled by instantaneous actions, can be used to check this property, so we only show how to specify the distance bounding requirement.

To formalize the distance bounding property we assume given the following values:

- $D(i, j, c)$ denotes the time it takes for a message to travel from agent i to agent j using the transmission medium c. For simplicity, we assume that $D(i, j, c) = D(j, i, c)$.
- D_b^A is the distance bounding time for the verifier A.

A. *Protocol Theories*

We use the following predicates in our specifications:

- $S_{i,c}(m)$ denotes that the message m was sent by agent i using transmission medium c;
- $A_0(A, B, c_S, c_R), A_1(A, B, c_R, N_A), B_0(A, B, c_S, c_R)$, $B_1(A, B, N_A)$ and $A_2(A, B, yes/no)$ are role state predicates used in a protocol session between agents A and B using the transmission channels c_S and c_R to, respectively, send the challenge nonce N_A and receive the response. We assume that c_S and c_R are given by the protocol specification. Moreover the predicate $A_2(A, B, yes)$ (resp. $A_2(A, B, no)$) denotes that the protocol succeeded (resp. failed) and that A did (resp. did not) grant access to the resource it owns to B;
- $Ag(B)$ specifies that B is an honest participant, $Ver(A)$ that A is a Verifier, and $Int(I)$ that I is an intruder;

- $Key(A, K)$ specifies that K is the public key of agent A;
- $Cap(I, C)$ specifies that the intruder I is capable of using the transmission medium C;
- $M_I(N)$ specifies that the intruder I knows N;
- P is a zero arity predicate representing an empty fact. It is a place holder in the configuration used to limit the number of concurrent protocol sessions and to specify the Bounded Memory Protocols and Bounded Memory Intruders (see [19]). In order to obtain specifications that are not Memory Bounded, one simply needs to delete all occurrences of P facts.

The initial configuration is composed of a fact $Ag(B)$ for each honest participant B, a fact $Ver(V)$ for each verifier V, $Int(I)$ for each intruder I, a fact $Key(A, K)$ for each key K belonging to an agent A, $Cap(I, C)$ for each transmission medium C that the intruder I is capable of using and a number of empty facts P. For example, the following initial configuration specifies a setting with two honest participants, h_1 and h_2, one verifier v with key k_v and one intruder i which is capable of using RF and ultrasound, denoted by r and s:

$$\left\{ \begin{array}{c} Time, Ag(h_1), Ag(h_2), Ver(v), Key(v, k_v), \\ Int(i), Cap(i, r), Cap(i, s) \end{array} \right\} \cup \mathcal{P}$$

where $\mathcal{P} = \{P, \ldots, P\}$ is a multiset of P facts and we elide their timestamps which are all 0.

The protocol theory is composed by the actions shown in Figure 2. In fact, the actions in Figure 2 are schema rules that should be instantiated by all possible values of $D_b^j, D(i, j, c)$, for agents i and j and transmission medium c.

The action **Rol Gen** generates a new protocol session involving the verifier A and the honest participant B using the transmission medium c_S and c_R, which are given by the protocol specification. The actions **Challenge** and **Response** correspond to the protocol steps. The **Challenge** action creates a nonce N_A and sends it to the Network using the transmission medium C_S. The **Response** action specifies that B can only respond once the time, $D(A, B, C_S)$, for the message to travel to him using the medium C_S has elapsed. This is specified by the constraint $T \geq T_3 + D(A, B, C_S)$. Thus, the message sent by A is not immediately available to B as expected when taking into account the physical properties of the system. Similarly, the action **Accept** specifies the case when B succeeded the Distance Bounding Challenge and is granted access to the resource, specified by the fact $A_2(A, B, yes)$, while the action **Reject** specifies the case when B did not succeed the Distance Bounding Challenge and the protocol is terminated without granting B access to the resource. Finally, the remaining **Del** actions are used to delete protocol roles (see discussion in [19]). One is also allowed to garbage-collect messages sent (using **DEL** $S_{i,c}$).

B. An Intruder Model

In symbolic protocol verification one normally assumes a powerful intruder, called Dolev-Yao intruder [12], which acts as the network. Since such an intruder can intercept and send a message anywhere in the network at anytime, he results faster than the speed of light. Thus using the traditional Dolev-Yao intruder makes the physical properties of the system irrelevant.

This paper proposes an intruder that behaves similarly to the Dolev-Yao intruder, but that has to obey the physical properties of the system. The Bounded Memory version of this intruder is specified in Figure 3. For the Unbounded Memory Version, one simply needs to erase all occurrences of P facts and delete the Memory Maintenance Rule (see [16] for the differences between these intruder models). While the Composition and Decomposition actions are similar to the Dolev-Yao intruder, the **I/O** actions are the interesting ones as they impose the physical properties of the system. In particular, action **Rec** specifies that to learn the contents of a message sent by agent A, the intruder has to wait the time, $D(A, I, C)$, for this message to reach him. This means that he is not faster than the speed of light. The **Snd** action specifies that the intruder can only send a message in a medium in which he has the capability to do so.

Notice that all the facts created by above intruder rules have the timestamp equal to the current time. Therefore we do not need to use additional time constraints in the rules, such as *e.g.* $T_1 \leq T$, $T_2 \leq T$, $T_3 \leq T$ in **DEC** rule.

In Section III we presented a protocol anomaly w.r.t an intruder which is subject both to network delays and processing time. However, for simplicity of the presentation, in the intruder model given in Figure 3 we disregard processing time. This could also be viewed as a characteristic of a very powerful intruder which is equipped with some supreme technology. On the other hand, we demonstrate how to represent network delays more precisely, specifying concrete traversal times.

Also notice that we are able to formalize both symmetric and asymmetric encryption. In the intruder model given in Figure 3 we represent symmetric encryption, while in Section III we show how to model public key encryption.

Alternative and/or Extended Intruder Models We can imagine alternative intruder models. For example, besides using both types of encryption, a more powerful intruder should be able to use any medium, that is, may use all transmission mediums available. Such an intruder, however, would still be subject to the timing constraints of the transmission mediums to receive and send messages.

Another possible intruder model is the intruder that is able to read messages without intercepting them. This would be specified by a rule of the following shape:

$$Time@T, S_{A,C}(X)@T_1, P@T_2 \mid T \geq T_1 + D(A, I, C) \longrightarrow \\ Time@T, S_{A,C}(X)@T_1, M_I(X)@T$$

where the fact $S_{A,C}(X)@T_1$ is not consumed by the rule , *i.e.* it appears both in the pre-condition and in the post-condition of the rule. It would therefore still be possible for the agent C to receive the original message X as well. This would correspond to using different communication technologies, such as radio transmission and ultrasound, while the intruder model shown in Figure 3 represents for example communication through a physical unreliable network or using radio frequency directed antennas.

Similarly, we could adapt the protocol specification to model simple reading of messages where the same message could be received by another agent including the intruder at some other time. This type of changes in both intruder model and protocol specification could affect the nature of possible anomalies in such scenarios.

Moreover, we have not specified a communication topology in our model. This is implicit by the $D(a, b, c)$ facts, specifying the time that a message needs to transit from agent a to agent b using the transmission medium c. However, more information can be inserted into the model, such as the routes used for transmission. This information should have important impact on the types of attacks that are possible.

Finally, the agents are assumed to be static. That is, during the execution of the protocol they are not allowed to move, *i.e.*, the values for $D(a, b, c)$ are always the same. Depending on the medium, *e.g.*, slower mediums, it might be the case that the intruder can exploit the fact that a protocol session runs in a slower medium and have time to carry out attacks when he is able to move.

V. Related Work

Others have also investigated the formalization of timed models and some have used these models for the verification of cyber-physical security protocols. We review this literature.

Meadows *et al.* [24] and Pavlovic and Meadows in [26] 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 from our work is that their PDL specification is not an executable specification, while our specification can be in principle directly used in computational tools such as Maude [9]. Finally, they do not investigate the complexity of verifying protocols nor investigate the expressiveness of formalizations using discrete and continuous time.

Another approach similar to [24] in the sense that it uses a theorem proving approach is given by Schaller *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 is a source of inspiration for our intruder model specified in Section IV, but there are some differences. As they are using a theorem prover to verify such protocols, they did not investigate executable models for the intruder model. Also they did not investigate the complexity of protocol verification nor the expressiveness differences of using different models. We believe that there can be much synergy between our approaches. For example, while our model seems suitable for model-checking using for instance Maude [9], we believe that our model can be also used in theorem provers to prove more general results such as those in [3], *e.g.*, safety results for an unbounded number of protocol sessions.

The Timed Automata [2] community has also proposed models for cyber-physical protocol verification. For example, Corin *et al.* [10] formalize protocols and the standard Dolev-Yao intruder as timed automata and demonstrate that these can be used for verification. They are able to formalize the generation of nonces by using timed automata, but they need to assume that there is a bound on the number of nonces. This means that they assume a bound on the total number of protocol sessions. Our model based on rewrite theory, on the other hand, allows for an unbounded number of nonces, even in the case of balanced theories [16]. Also they do not investigate the complexity of the verification problems nor the expressiveness difference between models with discrete and continuous time.

Another approach for specifying and verifying protocols using Timed Automata was given by Lanotte *et al.* [22]. They do not specify cyber-physical protocols, but protocols where messages can be re-transmitted or alternatively a protocol session can be terminated, *i.e.*, timeouts, in case a long time time elapses. They formalize the standard Dolev-Yao intruder. Finally, they also obtain a decidability result for their formalism and an lower bound of EXPSPACE-hard for the reachability problem. It seems possible to specify features like timeouts and message re-transmission, in our rewriting formalism. Moreover, we obtain a better complexity (PSPACE-completeness) result proved in our technical report [17] for a realistic fragment of cyber-physical protocols, namely of Bounded Memory Protocols [18], [19].

Recently [4] proposed a discrete time model for formalizing distance-bounding protocols and their security requirements. Their approach is probabilistic and the adversary model is a probabilistic Turing machine, which does not directly correspond to our intruder model with Dolev-Yao style rules. In their model computation time is not taken into account. Furthermore, they model only one transmission speed, therefore anomalies that arise from different transmission velocities are not considered. We on the other hand do specify various transmissions through capabilities of using specific mediums and the corresponding traversal times. We leave the deeper comparison of our models for some future work.

Petri nets (PN) have been used both for security protocol specification and analysis [11] as well as for the formalization of real-time systems [1]. Our reachability problem is related to the reachability and coverability problems for Petri nets and our model has clear similarities to Timed-Arc Petri Nets (TAPN) [7], *e.g.*, connections between token age and timestamps of facts as well as between time intervals labeling the arcs in TAPNs and our time constraints. Despite similarities, it does not seem possible to provide direct, faithful reductions between our model and PNs. Moreover, we are not aware of any work on PN that includes both real-time and fresh values.

Finally, Malladi *et al.* [23] formalize distance bounding protocols in strand spaces. They then construct an automated tool for protocol verification using a constraint solver. They do not investigate the complexity of their verification nor the expressiveness of their model. We believe, however, that the idea of using constraint solver, or alternatively SMT solvers, is an interesting direction for implementing tools for cyber-physical protocol verification, in particular, for reducing search space. In fact, we are currently investigating how to use SMT solvers together with Maude to do so.

VI. Conclusions and Future Work

This paper proposes a timed model for cyber-physical security protocols. We showed that a model with discrete time is strictly less powerful than a model with continuous time in the sense that there are protocols for which there is no attack in the former model but there is an attack in the latter model. We then proposed an intruder model which respects the physical properties of his environment and showed that our formalism can capture known attacks.

There is a number of directions which we are currently working on. Firstly, we are investigating the power of our intruder model, that is, how much damage can he do and under which conditions. We are also investigating alternative intruder models such as those described at the end of Section IV and their properties and differences. We are also investigating clever ways to implement tools for verifying cyber-physical protocols based on our intruder model. We are currently investigating the use of SMT solvers together with Maude.

One assumption in our model is that all agents share a global clock. Although this assumption is reasonable for some applications, such as for the distance bounding protocols, it is not the case for others such as in Network Time Protocols. We are investigating suitable models with local clocks, that is, models where each agent has its own clock, as well as the corresponding properties, such as complexity results.

Acknowledgments: Nigam is supported by the Brazilian Research Agency CNPq. Scedrov is supported in part by the AFOSR MURI ”Science of Cyber Security: Modeling, Composition, and Measurement” as AFOSR Grant No. FA9550-11-1-0137. Additional support for Scedrov from NSF Grant CNS-0830949 and from ONR grant N00014-11-1-0555. During the work on the revised version Kanovich and Scedrov were visiting the National Research University Higher School of Economics, Moscow. They would like to thank Sergei O. Kuznetsov for providing a very pleasant environment for work.

References

[1] P. A. Abdulla and A. Nylén. Timed petri nets and bqos. In *ICATPN*, pages 53–70, 2001.

[2] R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *SFM*, pages 1–24, 2004.

[3] D. A. Basin, S. Capkun, P. Schaller, and B. Schmidt. Formal reasoning about physical properties of security protocols. *ACM Trans. Inf. Syst. Secur.*, 14(2):16, 2011.

[4] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical & provably secure distance-bounding. *IACR Cryptology ePrint Archive*, 2013:465, 2013.

[5] S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In T. Helleseth, editor, *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, pages 344–359. Springer, 1993.

[6] S. Capkun and J.-P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):221–232, 2006.

[7] F. Cassez and O. H. Roux. Structural translation from time petri nets to timed automata. *Journal of Systems and Software*, 79(10):1456–1468, 2006.

[8] I. Cervesato, N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *CSFW*, pages 55–69, 1999.

[9] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Mart´-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer.

[10] R. Corin, S. Etalle, P. H. Hartel, and A. Mader. Timed analysis of security protocols. *J. Comput. Secur.*, 15(6):619–645, Dec. 2007.

[11] F. Crazzolara and G. Winskel. Petri nets in cryptographic protocols. In *IPDPS*, page 149, 2001.

[12] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[13] N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[14] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

[15] S. Ganeriwal, C. Pöpper, S. Capkun, and M. B. Srivastava. Secure time synchronization in sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(4), 2008.

[16] M. Kanovich, T. B. Kirigin, V. Nigam, and A. Scedrov. Bounded memory Dolev-Yao adversaries in collaborative systems. *Inf. Comput.* Accepted for Publication.

[17] M. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, and C. Talcott. On the complexity of cyber-physical security protocols, available on Nigam's homepage.

[18] M. I. Kanovich, T. B. Kirigin, V. Nigam, and A. Scedrov. Bounded memory protocols. *Computer Languages, Systems & Structures*. Accepted for Publication.

[19] M. I. Kanovich, T. B. Kirigin, V. Nigam, and A. Scedrov. Bounded memory protocols and progressing collaborative systems. In J. Crampton, S. Jajodia, and K. Mayes, editors, *ESORICS*, volume 8134 of *Lecture Notes in Computer Science*, pages 309–326. Springer, 2013.

[20] M. I. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, C. L. Talcott, and R. Perovic. A rewriting framework for activities subject to regulations. In A. Tiwari, editor, *RTA*, volume 15 of *LIPIcs*, pages 305–322. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[21] M. I. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, C. L. Talcott, and R. Perovic. A rewriting framework and logic for activities subject to regulations. Submitted, available in Nigam's homepage, 2013.

[22] R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Reachability results for timed automata with unbounded data structures. *Acta Inf.*, 47(5-6):279–311, 2010.

[23] S. Malladi, B. Bruhadeshwar, and K. Kothapalli. Automatic analysis of distance bounding protocols. *CoRR*, abs/1003.5383, 2010.

[24] C. Meadows, R. Poovendran, D. Pavlovic, L. Chang, and P. F. Syverson. Distance bounding protocols: Authentication logic analysis and collusion attacks. In *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, pages 279–298. 2007.

[25] V. Nigam, T. B. Kirigin, A. Scedrov, C. Talcott, M. Kanovich, and R. Perovic. Towards an automated assistant for clinical investigations. In *Second ACM SIGHIT International Health Informatics Symposium*, 2012.

[26] D. Pavlovic and C. Meadows. Deriving ephemeral authentication using channel axioms. In B. Christianson, J. A. Malcolm, V. Matyas, and M. Roe, editors, *Security Protocols Workshop*, volume 7028 of *Lecture Notes in Computer Science*, pages 240–261. Springer, 2009.

[27] V. Shmatikov and M.-H. Wang. Secure verification of location claims with simultaneous distance modification. In *ASIAN*, pages 181–195, 2007.

[28] K. Sun, P. Ning, and C. Wang. Tinysersync: secure and resilient time synchronization in wireless sensor networks. In *ACM Conference on Computer and Communications Security*, pages 264–277, 2006.

[29] N. O. Tippenhauer and S. Capkun. Id-based secure distance bounding and localization. In *ESORICS*, pages 621–636, 2009.

Appendix

A. Protocol Theories: A protocol session

We show in detail a normal execution of a protocol session of the Distance Bounding Protocol described in Section IV. Assume that there is only one verifier v and one honest participant h and no intruder. Let the distance bounding time for v be $D_b^v = 6$ time units. Moreover, let $D(v, h, c_S) = 1$ and $D(h, v, c_R) = 4$, *i.e.*, the transmission velocity of the medium c_S is four times higher than the velocity of c_R. The upper bound $D_{max} = 7$ and the initial configuration is:

$$I_0 = \{Time@0, Ag(h)@0, Ver(v)@0, Key(v, k_v)@0\} \cup \mathcal{P}$$

where $\mathcal{P}$ is a multiset of P facts. We assume that there are enough P facts, namely at least 4 P facts in $\mathcal{P}$.

We can apply the action **Rol Gen**, which rewrites I to the following configuration:

$$I_1 = \left\{ \begin{array}{c} Time@0, Ag(h)@0, Ver(v)@0, Key(v, k_v)@0, \\ A_0(v, h, c_S, c_R)@0, B_0(v, h, c_S, c_R)@0 \end{array} \right\} \cup \mathcal{P}_1$$

where $\mathcal{P}_1$ is $\mathcal{P} \setminus \{P@0, P@0\}$.

At this point we can apply the action **Challenge** which rewrites I_1 to:

$$I_2 = \left\{ \begin{array}{c} Time@0, Ag(h)@0, Ver(v)@0, Key(v, k_v)@0, \\ A_1(v, h, c_S, c_R)@0, B_0(v, h, c_S, c_R)@0, \\ S_{v,c_S}(\langle n_v, v\rangle)@0 \end{array} \right\} \cup \mathcal{P}_2$$

where $\mathcal{P}_2$ is $\mathcal{P}_1 \setminus \{P@0\}$. Until now, time has not played any important role as the guard of the applied actions were empty. However, for the message $S_{v,c_S}(\langle n_v, v\rangle)@0$ to be received by the participant h, one needs time to elapse $D(v, h, c_S) = 1$ time units. Say that Time elapses by 1.15, *i.e.*, we instantiate ε by 1.15 in the Tick action obtaining the configuration:

$$I_3 = \left\{ \begin{array}{c} Time@1.15, Ag(h)@0, Ver(v)@0, Key(v, k_v)@0, \\ A_1(v, h, c_S, c_R)@0, B_0(v, h, c_S, c_R)@0, \\ S_{v,c_S}(\langle n_v, v\rangle)@0 \end{array} \right\} \cup \mathcal{P}_2$$

Now the action **Response** can be applied as the constraint $T \geq T_3 + D(v, h, c_S)$ is satisfied, namely $1.15 \geq 0 + 1$. We obtain the following configuration

$$I_4 = \left\{ \begin{array}{c} Time@1.15, Ag(h)@0, Ver(v)@0, Key(v, k_v)@0, \\ A_1(v, h, c_S, c_R)@0, S_{v,c_S}(\langle n_v, v\rangle)@0, \\ B_1(v, h, n_v)@1.15, S_{h,c_R}(\langle n_v, h\rangle_{k_v})@1.15 \end{array} \right\} \cup \mathcal{P}_3$$

where $\mathcal{P}_3 = \mathcal{P}_2 \setminus \{P@0\}$.

Again, we cannot apply the **Accept** nor **Reject** actions as the time for the message $S_{h,c_R}(\langle n_v, h\rangle_{k_v})$ to reach v has not elapsed. We thus apply the Tick rule once again, say with $\varepsilon = 4.45$, obtaining the configuration:

$$I_5 = \left\{ \begin{array}{c} Time@5.6, Ag(h)@0, Ver(v)@0, Key(v, k_v)@0, \\ A_1(v, h, c_S, c_R)@0, S_{v,c_S}(\langle n_v, v\rangle)@0, \\ B_1(v, h, n_v)@1.15, S_{h,c_R}(\langle n_v, h\rangle_{k_v})@1.15 \end{array} \right\} \cup \mathcal{P}_3$$

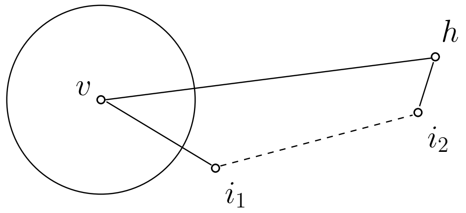
Since the guard of the action **Accept**, $T_1 + D_b^v \geq T \geq T_2 + D(h, v, c_R)$, is satisfied, namely $0 + 6 \geq 5.6 \geq 1.15 + 4$, the protocol is completed by granting h access to v's resource as specified by the following configuration:

$$I_6 = \left\{ \begin{array}{c} Time@5.6, Ag(h)@0, Ver(v)@0, Key(v, k_v)@0, \\ B_1(v, h, n_v)@1.15, A_2(v, h, yes)@5.6, \\ S_{v,c_S}(\langle n_v, v\rangle)@0, \end{array} \right\} \cup \mathcal{P}_4$$

where $\mathcal{P}_4 = \mathcal{P}_3 \cup \{P@5.6\}$.

B. An attack

We detail how the attack described in [3] can be captured in our framework. The attack is illustrated by the following diagram:



Here v is a verifier, h is a honest participant and i_1, i_2 are intruders. The idea is that i_1 and i_2 are colluding to convince v that h is closer than he actually is. This is done by using a much faster communication channel (depicted by the dashed line). It is used to forward the reply message from h intercepted by i_2 using the faster path $h - i_2 - i_1 - v$ instead of directly using the slower path $h - v$.

As in the previous example, let $D_b^v = 6$ time units, but, let $D(v, h, c_S) = 1$ and $D(h, v, c_R) = 10$, *i.e.*, c_R be ten times slower than c_S. Notice that with these values h would not succeed the Distance Bounding Requirement, as $1+10 > 6$. Moreover, let $D(h, i_2, c_R) = 1 = D(i_2, i_1, c_I)$ and $D(i_1, v, c_R) = 2$, that is, i_2 is closer to h and the transmission channel c_I is much faster than c_R.

The initial configuration is:

$$A_0 = \left\{ \begin{array}{c} Time@0, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int(i_2)@0, Key(v, k_v)@0, Cap(i_1, c_R)@0, \\ Cap(i_1, c_I)@0, Cap(i_2, c_R)@0, Cap(i_2, c_I)@0 \end{array} \right\} \cup \mathcal{P}$$

where $\mathcal{P}$ is a multiset of P facts. It specifies the two intruders which are capable of using the transmission mediums c_R and c_I. (They do not need to use the transmission medium c_S for this attack.)

As before, the protocol starts by first generating the role predicates using the action **Rol Gen**. We obtain the configuration:

$$A_1 = \left\{ \begin{array}{c} Time@0, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int_(i_2)@0, Key(v, k_v)@0, Cap(i_1, c_R)@0, \\ Cap(i_1, c_I)@0, Cap(i_2, c_R)@0, Cap(i_2, c_I)@0 \\ A_0(v, h, c_S, c_R)@0, B_0(v, h, c_S, c_R)@0 \end{array} \right\} \cup \mathcal{P}_1$$

where $\mathcal{P}_1$ is $\mathcal{P} \setminus \{P@0, P@0\}$.

Say that time advances by 2.37 and this is when the verifier sends the challenge, *i.e.*, we have a sequence of actions Tick and **Challenge** obtaining the following configuration:

$$A_2 = \left\{ \begin{array}{c} Time@2.37, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int(i_2)@0, Key(v,k_v)@0, Cap(i_1,c_R)@0, \\ Cap(i_1,c_I)@0, Cap(i_2,c_R)@0, Cap(i_2,c_I)@0, \\ B_0(v,h,c_S,c_R)@0, A_1(v,h,c_S,c_R)@2.37, \\ S_{v,c_S}(\langle n_v, v\rangle)@2.37 \end{array} \right\} \cup \mathcal{P}_2$$

where $\mathcal{P}_2$ is $\mathcal{P}_1 \setminus \{P@0\}$.

Now, we need to wait for time to elapse so that the message sent by the verifier reaches h, *i.e.*, that more than 1 time unit elapses. Say it elapses 1.55 and this is when h responds. That is we have a sequence of the Tick and **Response** actions obtaining the following configuration:

$$A_3 = \left\{ \begin{array}{c} Time@3.92, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int(i_2)@0, Key(v,k_v)@0, Cap(i_1,c_R)@0, \\ Cap(i_1,c_I)@0, Cap(i_2,c_R)@0, Cap(i_2,c_I)@0, \\ A_1(v,h,c_S,c_R)@2.37, S_{v,c_S}(\langle n_v, v\rangle)@2.37 \\ B_1(v,h,n_v)@3.92, S_{h,c_R}(\langle n_v, h\rangle_{k_v})@3.92 \end{array} \right\} \cup \mathcal{P}_3$$

where $\mathcal{P}_3$ is $\mathcal{P}_2 \setminus \{P@0\}$.

The message sent by h is then intercepted by the intruder i_2. However, differently from the Dolev-Yao intruder, which could do this immediately, i_2 has to wait for this message to reach him, *i.e.*, wait for the time to elapse at least $D(h, i_2, c_R)$. Say that it elapses 1.18, and that intruder i_2 intercepts this message using **REC** action. We obtain the following configuration, where i_2 learns $\langle n_v, h\rangle_{k_v}$:

$$A_4 = \left\{ \begin{array}{c} Time@4.1, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int(i_2)@0, Key(v,k_v)@0, Cap(i_1,c_R)@0, \\ Cap(i_1,c_I)@0, Cap(i_2,c_R)@0, Cap(i_2,c_I)@0, \\ A_1(v,h,c_S,c_R)@2.37, S_{v,c_S}(\langle n_v, v\rangle)@2.37 \\ B_1(v,h,n_v)@3.92, M_{i_2}(\langle n_v, h\rangle_{k_v})@4.1 \end{array} \right\} \cup \mathcal{P}_3$$

Intruder i_2 immediately sends this learned message to i_1 using the transmission channel c_I. That is, we apply the action **SND** obtaining the following configuration:

$$A_5 = \left\{ \begin{array}{c} Time@4.1, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int(i_2)@0, Key(v,k_v)@0, Cap(i_1,c_R)@0, \\ Cap(i_1,c_I)@0, Cap(i_2,c_R)@0, Cap(i_2,c_I)@0, \\ A_1(v,h,c_S,c_R)@2.37, S_{v,c_S}(\langle n_v, v\rangle)@2.37 \\ B_1(v,h,n_v)@3.92, S_{i_2,c_I}(\langle n_v, h\rangle_{k_v})@4.1 \end{array} \right\} \cup \mathcal{P}_3$$

Intruder i_1 has to wait for this message to reach him, *i.e.*, wait time elapse $D(i_2, i_1, c_I) = 1$. Say that time elapses 1.2, when the intruder i_1 receives the message sent by i_2. That is, we apply the Tick action and then **REC** action, obtaining the following configuration:

$$A_6 = \left\{ \begin{array}{c} Time@5.3, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int(i_2)@0, Key(v,k_v)@0, Cap(i_1,c_R)@0, \\ Cap(i_1,c_I)@0, Cap(i_2,c_R)@0, Cap(i_2,c_I)@0, \\ A_1(v,h,c_S,c_R)@2.37, S_{v,c_S}(\langle n_v, v\rangle)@2.37 \\ B_1(v,h,n_v)@3.92, M_{i_1}(\langle n_v, h\rangle_{k_v})@5.3 \end{array} \right\} \cup \mathcal{P}_3$$

Similarly as before, the intruder i_1 immediately sends the learned message but using the transmission channel c_R as expected by the protocol. We obtain the following configuration:

$$A_7 = \left\{ \begin{array}{c} Time@5.3, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int(i_2)@0, Key(v,k_v)@0, Cap(i_1,c_R)@0, \\ Cap(i_1,c_I)@0, Cap(i_2,c_R)@0, Cap(i_2,c_I)@0, \\ A_1(v,h,c_S,c_R)@2.37, S_{v,c_S}(\langle n_v, v\rangle)@2.37 \\ B_1(v,h,n_v)@3.92, S_{i_1,c_R}(\langle n_v, h\rangle_{k_v})@5.3 \end{array} \right\} \cup \mathcal{P}_3$$

This message should take at least $D(i_1, v, c_R) = 2$ time units to reach the verifier. Say that time elapses 2.3, obtaining the following configuration:

$$A_8 = \left\{ \begin{array}{c} Time@7.6, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int(i_2)@0, Key(v,k_v)@0, Cap(i_1,c_R)@0, \\ Cap(i_1,c_I)@0, Cap(i_2,c_R)@0, Cap(i_2,c_I)@0, \\ A_1(v,h,c_S,c_R)@2.37, S_{v,c_S}(\langle n_v, v\rangle)@2.37 \\ B_1(v,h,n_v)@3.92, S_{i_1,c_R}(\langle n_v, h\rangle_{k_v})@5.3 \end{array} \right\} \cup \mathcal{P}_3$$

At this point the guard of the action **Accept**, $T_1 + D_b^v \geq T \geq T_2 + D(h, v, c_R)$, is satisfied as $2.37 + 6 \geq 7.6 \geq 5.3 + 2$, thus, the verifier grants h access to its resource, specified by the following configuration which is the results of applying the **Accept** action:

$$A_9 = \left\{ \begin{array}{c} Time@7.6, Ag(h)@0, Ver(v)@0, Int(i_1)@0, \\ Int(i_2)@0, Key(v,k_v)@0, Cap(i_1,c_R)@0, \\ Cap(i_1,c_I)@0, Cap(i_2,c_R)@0, Cap(i_2,c_I)@0, \\ A_2(v,h,yes)@7.6, S_{v,c_S}(\langle n_v, v\rangle)@2.37 \\ B_1(v,h,n_v)@3.92, S_{i_1,c_R}(\langle n_v, h\rangle_{k_v})@5.3 \end{array} \right\} \cup \mathcal{P}_4$$

$\mathcal{P}_4 = \mathcal{P}_3 \cup \{P@7.6\}$. That is verifier v granted access to h although from the specification of the protocol h does not comply with the Distance Bounding Requirement. It is therefore an attack.

C. Time-bounding Needham-Schroeder Protocols: Attacks in Discrete Time versus Attacks in Continuous Time

Consider given a distance bounding protocol with the distance bounding time R, denoting the treshold for the response time. Here R is a positive integer.

Let non-negative integers a and b be strict lower bounds for passing messages and processing requests, that is the network delay is greater than a, and the internal processing time is greater than b.

The case $a = 0,\ b = 0$ considered in Section III represents the general knowledge that the passing messages and processing requests each takes a non-zero amount of time. The fact that here we do not invoke any *explicit* lower bounds provides the stability of our analysis for the current protocols with respect to the improvements in the machine and web technologies in the future.

In particular, in the case of continuous time, whatever positive threshold R we take, there is a Dolev-Yao attack on the protocol as stated in the next theorem.

Theorem C.1: For the case where $a = 0,\ b = 0$,

- For discrete time, there is no Dolev-Yao attack on the time-bounding Needham-Schroeder protocol with the distance bounding time R:
$$R \leq 6$$
- For continuous time, whatever positive distance bounding time R we take, there is a Dolev-Yao attack on the protocol.

However, in the case where either of the strict lower bounds for passing messages and processing requests is positive, the protocol *may remain secure* even within continuous time. Security of the protocol in the continuous time model depends on the realtion between the non-negative integers a and b representing lower bounds, and the positive integer R denoting the treshhold, as stated in the following theorem.

Theorem C.2: For the case where $a \geq 1$, or $b \geq 1$,

- For discrete time, there is no Dolev-Yao attack on the time-bounding Needham-Schroeder protocol with distance bounding time R:
$$R \leq 4a + 3b + 6$$
- For continuous time, there is no Dolev-Yao attack on the time-bounding Needham-Schroeder protocol with the distance bounding time R:
$$R < 4a + 3b$$

Rewrite rules with lower bounds

Non-zero lower bounds for passing messages and processing requests are formalized in our multiset rewriting model as follows. We slightly modify rules (3) and (4) given in Section III. In particular, we include lower bounds a and b in the time constraints of the corresponding rules.

The following action formalizes non-zero traversal time with a lower bound a:

$$\begin{aligned} &Time@t_1,\ \mathcal{N}_S(m)@t_0 \mid \{\, t_1 > t_0 + a \,\} \ \longrightarrow \\ &\qquad\qquad Time@t_1,\ \mathcal{N}_R(m)@t_1 \end{aligned} \tag{12}$$

while this pair of actions represents non-zero processing time with a lower bound b:

$$\begin{aligned} &Time@t_1,\ \mathcal{N}_R(m')@t_1 \ \longrightarrow\ Time@t_1,\ X(m')@t_1, \\ &Time@t_2,\ X(m')@t_1 \mid \{t_2 > t_1 + b\} \ \longrightarrow \\ &\qquad\qquad Time@t_2,\ X(m'')@t_2,\ \mathcal{N}_S(m'')@t_2. \end{aligned} \tag{13}$$

The rules specifying concrete protocol theories would need to be adjusted analogously, such as the rules (9), (11) of the protocol given in Section III.