# Discrete vs. Dense Times in the Analysis of Cyber-Physical Security Protocols

Max Kanovich[1,5] Tajana Ban Kirigin[2] Vivek Nigam[3] Andre Scedrov[4,5] and Carolyn Talcott[6]

[1] Queen Mary, University of London & University College, UK, mik@dcs.qmul.ac.uk
[2] University of Rijeka, HR, bank@math.uniri.hr
[3] Federal University of Paraba, João Pessoa, Brazil, vivek@ci.ufpb.br
[4] University of Pennsylvania, Philadelphia, USA, scedrov@math.upenn.edu
[5] National Research University Higher School of Economics, Moscow, Russia
[6] SRI International, USA, 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. Time plays a key role in design and analysis of many of these protocols. This paper investigates the foundational differences and the impacts on the analysis when using models with discrete time and models with dense time. We show that there are attacks that can be found by models using dense time, but not when using discrete time. We illustrate this with a novel attack that can be carried out on most distance bounding protocols. In this attack, one exploits the execution delay of instructions during one clock cycle to convince a verifier that he is in a location different from his actual position. We propose a Multiset Rewriting model with dense time suitable for specifying cyber-physical security protocols. We introduce Circle-Configurations and show that they can be used to symbolically solve the reachability problem for our model. Finally, we show that for the important class of balanced theories the reachability problem is PSPACE-complete.

## 1 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 [4] 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 a distance bounding protocol session, the verifier ($V$) and the prover ($P$) exchange messages:

$$\begin{array}{l} V \longrightarrow P : m \\ P \longrightarrow V : m' \end{array} \tag{1}$$

where $m$ is a challenge and $m'$ is a response message (constructed using $m$'s components). To infer the distance to the prover, the verifier remembers the time, $t_0$, when the message $m$ was sent, and the time, $t_1$, when the message $m'$ returns. From the difference $t_1 - t_0$ and the assumptions on the speed of the transmission medium, $v$, the verifier can compute an upper bound on the distance to the prover, 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 [5,29,31], and Secure Time Synchronization Protocols [14,30]. The common feature in most cyber-physical security protocols is that they mention cryptographic keys, nonces and time. (For more examples, see [2,24] and references therein.)

A major problem of using the traditional protocol notation for the description of distance bounding protocols, as in Eg. 1, is that many assumptions about time, such as the time requirements for the fulfillment of a protocol session, are not formally specified. It is only informally described that the verifier 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, such as the transmission medium used by the participants. 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 [10] who is capable of intercepting and sending messages anywhere at anytime. The Dolev-Yao intruder can easily convince $V$ that $P$ is closer than he actually is. The intruder first intercepts the message $m$ and with zero transmission time sends it $P$. Then he intercepts the message $m'$ and instantaneously sends it to $V$, reducing the round-trip-time $(t_1 - t_0)$. Thus, $V$ will believe that $P$ 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 another. Indeed, the standard Dolev-Yao intruder model is not a suitable model for the analysis of cyber-physical protocols. Since he is able to intercept and send messages anywhere at anytime, he results faster than the speed of light. In fact, a major difference between cyber-physical protocols and traditional security protocols is that there is not necessarily a network in the traditional sense, as the medium is the network.

Existing works have proposed and used models with time for the analysis of distance bounding protocols where the attacker is constrained by some physical properties of the system. Some models have considered dense time [2], while others have used discrete time [3]. However, although these models have included time, the foundational differences between these models and the impacts to analysis has not been investigated in more detail. For example, they have not investigated the fact that provers, verifiers, and attackers may have different clock rates, *i.e.*, processing speeds, affecting security. This paper addresses this gap. While studying this problem, we have identified a novel attack called *Attack In-Between-Ticks*. We believe it can be carried out on most distance bounding protocols. The main observation is that while the verifier uses discrete clock ticking and thus measures time in discrete units, the environment and the attacker is not limited by a particular clock. In fact, a key observation of this paper is that models with dense time abstract the fact that attacker clocks may tick at any rate. The attacker can mask his location by exploiting the fact that a message may be sent at any point between two clock ticks of the verifier's clock, while the verifier believes that it was sent at a particular time. Depending on the speed of the verifier, *i.e.*, its clock rate, the attacker can *in principle* convince the verifier that he is very close to the verifier (less than a meter) even though he is very far away (many meters away).

Interestingly, however, from a foundational point of view, there is no complexity increase when using a model with dense time when compared to a model with discrete time. In our previous work [18], we proposed a rewriting framework which assumed discrete time. We showed that the reachability problem is PSPACE-complete. Here we show that if we extend the model with dense time, the reachability problem is still PSPACE-complete. For this result we introduce a novel machinery called Circle-Configurations.

Section 2 contains two motivational examples, including the novel attack in-between-ticks. In Section 3 we introduce a formal model based on Multiset Rewriting (MSR) which includes dense time. We also show how to specify distance bounding protocols in this language. Section 4 introduces a novel machinery, called Circle-Configurations, that allow one to symbolically represent configurations that mention dense time. Section 5 proves that the reachability problem for timed bounded memory protocols [16] is PSPACE-complete. Finally, in Section 6, we comment on related and future work.

## 2 Two Motivating Examples

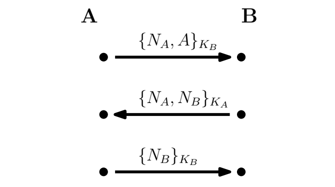
This section presents two examples of protocols to illustrates the differences between models with discrete and dense time. In the first example we present a timed version of the classical *Needham-Schroeder* protocol [25]. It shows that some attacks may only be found when using models with dense time. The second example is the novel attack in-between-ticks, which illustrates that for the analysis of distance bounding protocols it is necessary to consider time assumptions of the players involved.

### 2.1 Time-Bounding Needham-Schroeder Protocol

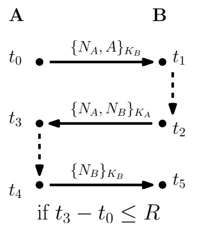
We first show some subtleties of cyber-physical protocol analysis by re-examining the original *Needham-Schroeder* public key protocol [25] (NS), see Fig 1a. Although this protocol is well known to be insecure [22], we look at it from another dimension, the dimension of time. We check whether Needham and Schroeder were right after all, in the sense that their protocol can be considered secure under some time requirements. In other words, we investigate whether NS can be fixed by means of time.

We timestamp each event in the protocol execution, that is, we explicitly mark the time of sending and receiving messages by a participant. We then propose a timed version of this protocol, called *Time-Bounding Needham-Schroeder Protocol* (Timed-NS), as depicted in Figure 1b. The protocol exchanges the same messages as in the original version, but the last protocol message, *i.e.* the confirmation message $\{N_B\}_{K_B}$, is sent by $A$ only if the time difference $t_3 - t_0$ is smaller or equal to the given *response bounding time* $R$.

The protocol is considered *secure* in the standard way, that is, if the "accepted" $N_A$ and $N_B$ may never be revealed to anybody else except Alice and Bob. Recall that the well known Lowe attack on NS [22] involves a third party, Mallory who is able to learn Bob's nonce. At the same time Bob believes that he communicated with Alice and that only Alice learned his nonce.

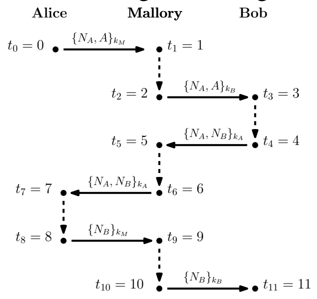


(a) Needham-Schroeder protocol

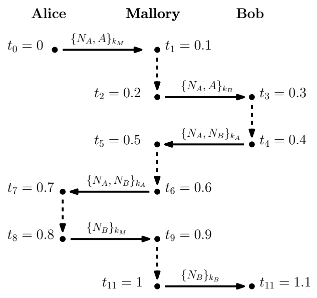


(b) Timed Needham-Schroeder Protocol

Fig. 1: Adding time to Needham-Schroeder Protocol



(a) Discrete Time Model



(b) Dense Time Model

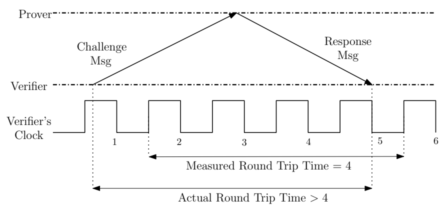
Fig. 2: Timed Version of Lowe Attack

The intriguing result of the analysis of Timed-NS is that one may not find an attack in the discrete time model, but can find one in the dense time model: Figure 2 depicts the Lowe attack scenario in Timed-NS. In particular, the attack requires that events marked with $t_0, \ldots, t_7$ take place and that the round trip time of messages, that is $t_7 - t_0$, does not exceed the given response bounding time $R$. Assuming that both network delay and processing time are non-zero, in the discrete time model the attack could be modeled only for response bounding time $R \geq 7$, see Figure 2a. In the discrete model, the protocol would seem safe for response bounding time $R < 7$. However, in the dense time model the attack is possible for any response bounding time $R$, see Figure 2b.

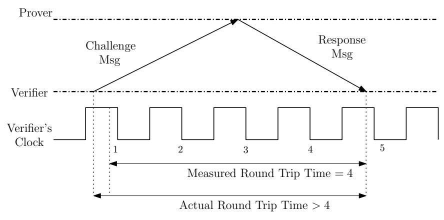
This simple example already illustrates the challenges of timed models for cyber-physical security protocol analysis and verification. No rescaling of discrete time units removes the presented difference between the models. For any discretization of time, such as seconds or any other infinitesimal time unit, there is a protocol for which there is an attack with continuous time and no attack is possible in the discrete case. This is further illustrated by the following more realistic example.

## 2.2 Attack In-Between-Ticks

Regardless of the design details of a specific distance bounding protocol a new type of anomaly can happen. We call it *Attack In-Between-Ticks*. This attack is particularly harmful when the verifier and the prover exchange messages using radio-frequency (RF), where the speed of transmission is the speed of light. In this case an error of a 1 nanosecond (*ns*) already results in a distance error of 30*cm*.



(a) In different ticks (Sequential Execution)



(b) In the same tick (Parallel Execution)

Fig. 3: Attack In-Between-Ticks. Here $R = 4$ ticks.

Consider the illustrations in Figure 3. They depict the execution of instructions by the verifier. The verifier has to execute two instructions: (1) the instruction that sends the signal to the prover and (2) the instruction that measures the time when this message is sent. Figure 3a illustrates the case when the verifier is running a sequential machine (that is, a single processor), which is the typical case as the verifier is usually a not very powerful device, *e.g.*, door opening device. Here we assume optimistically that an instruction can be executed in one cycle. When the first instruction is executed, it means that the signal is sent somewhere when the clock is up, say at time 0.6. In the following clock cycle, the verifier remembers the time when the message is sent. Say that this was already done at time 1.5. If the response message is received at time 5 it triggers an interruption so that the verifier measures the response time in the following cycle, *i.e.*, at time 5.5. Thus the measured round time is $4 = 5.5 - 1.5 = R$ ticks. Therefore, the verifier grants access to the prover although the actual round trip is $5 - 0.6 = 4.4 > R = 4$ ticks. This means that the verifier is granting access to the prover although the prover's distance to the verifier may not satisfy the distance bound and thus is a security flaw.[7]

Depending on the speed of verifier's processor, the difference of 0.4 tick results in a huge error. Many of these devices use very weak processors. The one proposed in [28], for example, executes at a frequency of at most $24MHz$. This means a tick is equal to $41ns$ (in the best case). Thus, an error of 0.4 tick corresponds to an error of $16ns$ or an error of *4.8 meters* when using RF. In the worst case, the error can be of 1.5 ticks when the signal is sent at the beginning of the cycle, *i.e.*, at time 0.5 tick, and the measurements at the end of the corresponding cycles, *i.e.*, at times 2 and 6 ticks. An error of 1.5 tick ($61.5ns$) corresponds to an error greater than 18 *meters* when using RF.

Consider now the case when the verifier can execute both instructions in the same cycle. Even in this case there might be errors in measurement as illustrated in Figure 3b. It may happen that the signal is sent before the measurement is taken thus leading to errors of at most 0.5 ticks (not as great as in the sequential case). (Here we are again assuming optimistically that an instruction can be executed in one cycle.)

Finally, we observe that these security flaws may happen *in principle*. In practice, distance bounding protocols carry out a large number of challenge and response rounds which we believe mitigates the chances of this attack occurring. We also point out that these attacks have been inspired by similar issues in the analysis of digital circuits [1].

[7]Notice that inverting the order of the instructions, *i.e.*, first collecting the time and then sending the signal, would imply errors of measurement but in the opposite direction turning the system impractical.

## 3 A Multiset Rewriting Framework with Dense Time

We assume a finite first-order typed alphabet, $\Sigma$, with variables, constants, function and predicate symbols. Terms and facts are constructed as usual (see [12]) by applying symbols with correct type (or sort). For instance, if $P$ is a predicate of type $\tau_1 \times \tau_2 \times \cdots \times \tau_n \rightarrow o$, where $o$ is the type for propositions, and $u_1, \ldots, u_n$ are terms of 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 [19], 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

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

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. We may sometimes denote the timestamp of a fact $F$ in a given configuration as $T_F$.

### 3.1 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:

$$Time@T \longrightarrow Time@(T + \varepsilon) \quad (3)$$

Here $\varepsilon$ 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 (2) we obtain the configuration

$$\{Time@8.1, Deadline@10.3, Task(1, done)@5.3, Task(2, pending)@2.13\} \quad (4)$$

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, but may rewrite the remaining facts. They have the following shape:

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

where $D_1, \ldots, D_m$ are natural numbers and $C$ 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:

$$T > T' \pm D \quad \text{and} \quad T = T' \pm D \quad (5)$$

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

Notice that the global time does not change when applying an instantaneous action. Moreover, the timestamps of the facts that are created 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. Applying this action to the configuration (4) yields

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

where Task 2 will be completed by the time 9.1.

Finally, the variables $\boldsymbol{X}$ that are existentially quantified in the above action are to be replaced by fresh values, also called *nonces* in protocol security literature [6, 11]. 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 [6, 11].

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

## 3.2 Initial, Goal Configurations, The Reachability Problem and Equivalence

We write $\mathcal{S} \longrightarrow_r \mathcal{S}_1$ for the one-step relation where 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:

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

where $T_1, \ldots, T_n$ are time variables, $F_1, \ldots, F_n$ are ground facts and $C$ 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 $\sigma$ replacing $T_1, \ldots, T_n$ by real numbers such that $\mathcal{S}_G\sigma \subseteq \mathcal{S}_1$ and all the constraints in $C\sigma$ 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 [15] for more discussion on this).

The following definition establishes the equivalence of configurations. Many formal definitions and results in this paper mention an upper bound $D_{max}$ on the numeric values of a reachability problem. This value is computed from the given problem: we set $D_{max}$ to be a natural number such that $D_{max} > n + 1$ for any number $n$ (both real or natural) appearing in the timestamps of the initial configuration, or the $D$s and $D_i$s in constraints or actions of the reachability problem.

**Definition 1.** *Given a reachability problem $\mathcal{T}$, let $D_{max}$ be an upper bound on the numeric values appearing in $\mathcal{T}$. Let*

$$\mathcal{S} = Q_1@t_1, Q_2@t_2, \ldots, Q_n@t_n \qquad \text{and} \qquad \widetilde{\mathcal{S}} = Q_1@\widetilde{t_1}, Q_2@\widetilde{t_2}, \ldots, Q_n@\widetilde{t_n}$$

*be two configurations written in canonical way where the two sequences of timestamps $t_1, \ldots, t_n$ and $\widetilde{t_1}, \ldots, \widetilde{t_n}$ are non-decreasing. Then $\mathcal{S}$ and $\widetilde{\mathcal{S}}$ are equivalent if they satisfy the same constraints, that is: $t_i > t_j \pm D$ iff $\widetilde{t_i} > \widetilde{t_j} \pm D$ and $t_i = t_j \pm D$ iff $\widetilde{t_i} = \widetilde{t_j} \pm D$, for all $1 \leq i, j \leq n$ and $D < D_{max}$.*

The following proposition states that the notion of equivalence defined above is coarse enough to identify applicable actions and thus the reachability problem.

**Proposition 1.** *Let $\mathcal{S}$ and $\mathcal{S}'$ be two equivalent configurations for a given reachability problem $\mathcal{T}$ and the upper bound $D_{max}$. There is a transition $\mathcal{S} \longrightarrow_r \mathcal{S}_1$ for an action $r$ in $\mathcal{T}$ if and only if there is a transition $\mathcal{S}' \longrightarrow_r \mathcal{S}'_1$ using a possibly different instance of the same action $r$ and furthermore $\mathcal{S}_1$ and $\mathcal{S}'_1$ are also equivalent.*

**Theorem 1.** *Let $\mathcal{S}_I$ and $\mathcal{S}'_I$ be two equivalent initial configurations, $\mathcal{S}_G$ be a goal and $\mathcal{R}$ a set of actions. Let $D_{max}$ be an upper bound on the numbers in $\mathcal{R}$, $\mathcal{S}_I$, $\mathcal{S}'_I$ and $\mathcal{S}_G$. Then the reachability problem with $\mathcal{S}_I, \mathcal{S}_G$ and $\mathcal{R}$ is solvable if and only if the reachability problem with $\mathcal{S}'_I, \mathcal{S}_G$ and $\mathcal{R}$ is solvable.*

### 3.3 Distance Bounding Protocol Formalization

To demonstrate how our model can capture the attack in-between-ticks, consider the following protocol, called DB, This protocol captures the time challenge of distance bounding protocols.[8] Verifier should allow the access to his resources only if the measured round trip time of messages in the distance-bounding phase of the protocol does not exceed the given bounding time $R$. We assume that the verifier and the prover have already exchanged nonces $n_P$ and $n_V$:

$$\begin{array}{ll} V \longrightarrow P : n_P & \text{at time } t_0 \\ P \longrightarrow V : n_V & \text{at time } t_1 \\ V \longrightarrow P : OK(P) & \text{iff } t_1 - t_0 \leq R \end{array}$$

*Encoding of verifier's clock* The fact $Clock_V@T$ denotes the local clock of the verifier *i.e.* the discrete time clock that verifier uses to measure the response time in the distance bounding phase of the protocol.

[8]Another specification that includes an intruder model, keys, and the specification of the attack described in [2] can be found in our workshop paper [17].

$$
\begin{array}{l}
Time@T, V_0(P, N_P, N_V)@T_1, E@T_2, E@T_3 \longrightarrow \\
\qquad\qquad Time@T, V_1(pending, P, N_P, N_V)@T, \mathcal{N}_V^S(N_P)@T, Start(P, N_P, N_V)@T \\
Time@T, V_1(pending, P, N_P, N_V)@T_1, Clock_V@T, P@T_2 \mid T \geq T_1 \longrightarrow \\
\qquad\qquad Time@T, V_1(start, P, N_P, N_V)@T, Clock_V@T, Start_V(P, N_P, N_V)@T \\
Time@T, P_0(V, N_V, N_P)@T_1, \mathcal{N}_P^R(N_P)@T_2 \mid T \geq T_2 \longrightarrow Time@T, P_1(V, N_V, N_P)@T, \mathcal{N}_P^S(N_V)@T \\
Time@T, V_1(start, P, N_P, N_V)@T_1, \mathcal{N}_V^R(N_V)@T_2 \longrightarrow Time@T, V_2(pending, P, N_P, N_V)@T, Stop(P, N_P, N_V)@T \\
Time@T, V_2(pending, P, N_P, N_V)@T_1, Clock_V@T, E@T_2 \mid T \geq T_1 \longrightarrow \\
\qquad\qquad Time@T, V_2(stop, P, N_P, N_V)@T, Clock_V@T, Stop_V(P, N_P, N_V)@T \\
Time@T, Start_V(P, N_P, N_V)@T_1, Stop_V(P, N_P, N_V)@T_2, V_2(stop, P, N_P, N_V)@T_3 \mid T_2 - T_1 \leq R, T \geq T_3 \longrightarrow \\
\qquad\qquad Time@T, V_3(P)@T, \mathcal{N}_V^S(Ok(P))@T, E@T
\end{array}
$$

Fig. 4: Protocol Rules for DB protocol

We encode ticking of verifier's clock in *discrete units of time*. Action (6) represents the ticking of verifier's clock:

$$Time@T,\ Clock_V@T_1 \mid T = T_1 + 1 \ \longrightarrow Time@T,\ Clock_V@T \qquad (6)$$

Notice that if this action is not executed and $T$ advances too much, *i.e.*, $T > T_1$, it means that the verifier clock stopped as it no longer advances.

*Network* Let $D(X, Y) = D(Y, X)$ be the integer representing the minimum time needed for a message to reach $Y$ from $X$. We also assume that participants do not move. Rule (7) models network transmission from $X$ to some $Y$:

$$Time@T,\ \mathcal{N}_X^S(m)@T_1,\ E@T_2 \ \mid\ T \geq T_1 + D(X, Y) \longrightarrow Time@T,\ \mathcal{N}_X^S(m)@T_1,\ \mathcal{N}_Y^R(m)@T \quad (7)$$

Facts $\mathcal{N}_X^S(m)$ and $\mathcal{N}_X^R(m)$ specify that the participant $X$ has sent and may receive the message $m$, respectively. Once $X$ has sent the message $m$, that message can only be received by $Y$ once it traveled from $X$ to $Y$. The fact $E$ is an empty fact which can be interpreted as a slot of resource. This is a technical device used to turn a theory balanced. It can safely be ignored until Section 5 (see as well [16]).

*Measuring the round trip time of messages* A protocol run creates facts denoting times when messages of the distance bounding phase are sent and received by the verifier. Predicates *Start* and *Stop* denote the actual (real) time of these events so that the round trip time of messages is $T_2 - T_1$ for timestamps $T_1, T_2$ in $Start(m)@T_1$, $Stop(m)@T_2$. On the other hand predicates $Start_V$ and $Stop_V$ model the verifier's view of time: $T_2 - T_1$, for $T_1, T_2$ in $Start_V(m)@T_1$, $Stop_V(m)@T_2$.

*Protocol Theory* Our example protocol $DB$ is formalized in Figure 4. The first rule specifies that the verifier has sent a nonce and still needs to mark the time, specified by the fact $V_1(pending, P, N_P, N_V)@T$. The second rule specifies verifier's instruction of remembering the current time. The third rule specifies prover's response to the verifier's challenge. The fourth and fifth rules are similar to the first two, specifying when verifier actually received prover's response and when he executed the instruction to remember the time. Finally, the sixth rule specifies that the verifier grants access to the prover if he believes that the distance to the prover is under the given bound.

*Attack In-Between-Ticks*

We now show how attack in-between-ticks is detected in our formalization.

The initial configuration contains facts $Time@0$, $Clock_V@0$ denoting that global time and time on verifier's discrete time are initially set to 0.

Given the protocol specification in Figure 4, attack in-between-ticks is represented with the following configuration:

$$Start(P, N_P, N_V)@T_1, Stop(P, N_P, N_V)@T_2, \mathcal{N}_V^S(Ok(P))@T_3 \mid T_2 - T_1 > R$$

It denotes that in the session involving nonces $N_P, N_V$ the verifier $V$ has allowed the access to prover $P$ although the distance requirement has been violated.

Notice that such an anomaly is really possible in this specification. Consider the following example: between moments 1.7 and 4.9, there would be 3 ticks on the verifier's clock. The verifier would consider starting time of 2 and finishing time of 5, and confirm with the time bound $R = 3$. Actually, the real round trip time is greater than the time bound, namely $4.9 - 1.7 = 3.2$. Following facts would appear in the configuration: $Start_V(n)@2, Stop_V(n)@5, Start(n)@1.7, Stop(n)@4.9$. Since $5 - 2 = 3$ the last rule from Figure 4, the accepting rule, would apply resulting in the configuration containing the facts: $Start(p, n_P, n_V)@1.7$, $Stop(p, n_P, n_V)@4.9$, $\mathcal{N}_V^S(Ok(p))@5$. Since $4.9 - 1.7 = 3.2$ is greater than $R = 3$, this configuration constitutes an attack.

*Protocol Formalization in Maude* We have formalized this scenario in an extension with SMT-solver of the rewriting logic tool, Maude. The tool was able to automatically find this attack. A main advantage of using an SMT solver in conjunction with Maude proof search is that one can reduce considerably the search-space involved. For example, when using the specification above, we do not provide a specific value for $D(p, v)$, but simply state that $D(p, v) > R$, that is, the prover is outside the distance bound. Due to space constraints, we do enter into the details of this implementation. We leave as future work the challenges of building a tool that can find verify cyber-physical protocols. We believe that we can integrate time constraints with the machinery used by MaudeNPA [13].

## 4 Circle-Configurations

This section introduces the machinery, called Circle-Configurations, that can symbolically represent configurations and plans that mention dense time. Dealing with dense time leads to some difficulties, which have puzzled us for some time now, in particular, means to handle Zeno paradoxes. When we use discrete domains to represent time, such as the natural numbers, time always advances by one, specified by the rule:

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

There is no other choice.[9] On the other hand, when considering systems with dense time, the problem is much more involved, as the non-determinism is much harder to deal with: the value that the time advances, the $\varepsilon$ in $Time@T \longrightarrow Time@(T + \varepsilon)$ (Eq. 3), can be instantiated by any positive real number.

Our claim is that we can symbolically represent any plan involving dense time by using a canonical form called circle-configurations. We show that circle-configurations provide a sound and complete representation of plans with dense time (Theorem 2).

[9] However, as time can always advance, a plan may use an unbounded number of natural numbers. This source of unboundedness was handled in our previous work [19]. This solution, however, does not scale to dense time.

A circle-configuration consists of two components: a *$\delta$-Configuration*, $\Delta$, and a *Unit Circle*, $\mathcal{U}$, written $\langle \Delta, \mathcal{U} \rangle$. Intuitively, the former accounts for the integer part of the timestamps of facts in the configuration, while the latter deals with the decimal part of the timestamps.

In order to define these components, however, we need some additional machinery. For a real-number, $r$, $int(r)$ denotes the integer part of $r$ and $dec(r)$ its decimal part. For example, $int(2.12)$ is 2 and $dec(2.12)$ is 0.12. Given a natural number $D_{max}$, the *truncated time difference* between two facts $P@t_P$ and $Q@t_Q$ such that $t_Q \geq t_P$ is defined as follows

$$\delta_{P,Q} = \begin{cases} int(t_Q) - int(t_P), \text{ if } int(t_Q) - int(t_P) \leq D_{max} \\ \infty, \text{ otherwise} \end{cases}$$

For example, if $D_{max} = 3$ and $F@3.12, G@1.01, H@5.05$, then $\delta_{F,H} = 2$ and $\delta_{G,H} = \infty$. Notice that whenever $\delta_{P,Q} = \infty$ for two timestamped facts, $P@t_P$ and $Q@t_Q$, we can infer that $t_Q > t_P + D$ for any natural number $D$ in the theory. Thus, we can truncate time difference without sacrificing soundness and completeness. This was pretty much the idea used in [19] to handle systems with discrete-time.

*$\delta$-Configuration* We now explain the first component, $\Delta$, of circle-configurations, $\langle \Delta, \mathcal{U} \rangle$, namely the $\delta$-configuration, to only later enter into the details of the second component in Section 4.1. Given a configuration $\mathcal{S} = \{F_1@t_1, \ldots, F_n@t_n, Time@t\}$, we construct its $\delta$-configuration as follows: We first sort the facts using the integer part of their timestamps, obtaining the sequence of timestamped facts $Q_1@t'_1, \ldots, Q_{n+1}@t'_{n+1}$, where $t'_i \leq t'_{i+1}$ for $1 \leq i \leq n+1$ and $\{Q_1, \ldots, Q_{n+1}\} = \{F_1, \ldots, F_n, Time\}$. We then aggregate in classes facts with the same integer part of the timestamps obtaining a sequence of classes $\{Q_1^1, \ldots, Q_{m_1}^1\}, \{Q_1^2, \ldots, Q_{m_2}^2\}, \ldots, \{Q_1^j, \ldots, Q_{m_j}^j\}$, where $\delta_{Q_i^k, Q_j^k} = 0$ for any $1 \leq i \leq m_k$ and $1 \leq k \leq j$. The $\delta$-configuration for $\mathcal{S}$ is then:

$$\Delta = \left\langle \{Q_1^1, \ldots, Q_{m_1}^1\}, \delta_{1,2}, \{Q_1^2, \ldots, Q_{m_2}^2\}, \ldots \{Q_1^{j-1}, \ldots, Q_{m_{j-1}}^{j-1}\}, \delta_{j-1,j}, \{Q_1^j, \ldots, Q_{m_j}^j\} \right\rangle$$

where $\delta_{i,i+1} = \delta_{Q_1^i, Q_1^{i+1}}$ is the truncated time difference between the facts in class $i$ and class $i+1$. For such a $\delta$-configuration, $\Delta$, we define

$$\Delta(Q_i^l, Q_j^h) = \begin{cases} \sum\limits_{k=l}^{k=h-1} \delta_{k,k+1} & \text{if } h \geq l \\ -\sum\limits_{k=h}^{k=l-1} \delta_{k,k+1} & \text{otherwise} \end{cases}$$

which is the truncated time difference between any two facts $Q_i^l$ and $Q_j^h$ from the classes $l$ and $h$, respectively, of $\Delta$. Here we assume $\infty$ is the addition absorbing element, *i.e.*, $\infty + D = \infty$ for any natural number $D$ and $\infty + \infty = \infty$.

Notice that, for a given upper bound $D_{max}$, different configurations may have the same $\delta$-configuration. For example, with $D_{max} = 4$, configurations

$$\begin{array}{c} \mathcal{S}_1 = \{M@3.01, R@3.11, P@4.12, Time@11.12, Q@12.58, S@14\} \quad \text{and} \\ \mathcal{S}'_1 = \{M@0.2, R@0.5, P@1.6, Time@6.57, Q@7.12, S@9.01\} \end{array} \tag{8}$$

have both the following $\delta$-configuration: $\Delta_{\mathcal{S}_1} = \langle \{M, R\}, 1, \{P\}, \infty, \{Time\}, 1, \{Q\}, 2, \{S\} \rangle$. This $\delta$-configuration specifies the truncated time differences between the facts. For example, $\Delta_{\mathcal{S}_1}(R, P) = 1$, that is, the integer part of the timestamp of the fact $P$ is ahead one unit with respect to the integer part of the timestamp of the fact $R$. Moreover, the timestamp of the fact $Time$ is more than $D_{max}$ units ahead with respect to the timestamp of $P$. This is indeed true for both configurations $\mathcal{S}_1$ and $\mathcal{S}_1'$ given above.

### 4.1 Unit Circle and Constraint Satisfaction

In order to handle the decimal part of the timestamps, we use intervals instead of concrete values. These intervals are represented by a circle, called Unit Circle, which together with a $\delta$-configuration composes a circle-configuration. The unit circle of a configuration $\mathcal{S} = \{F_1@t_1, \ldots, F_n@t_n, Time@t\}$ is constructed by first ordering the facts according to the *decimal part* of their timestamps, obtaining the sequence of facts $Q_1, \ldots, Q_{n+1}$, where $\{Q_1, \ldots, Q_{n+1}\} = \{F_1, \ldots, F_n, Time\}$. Then the unit circle of the given configuration $\mathcal{S}$ is obtained by aggregating facts that have the same *decimal part* obtaining a sequence of classes:

$$\mathcal{U} = [\{Q_1^0, \ldots, Q_{m_0}^0\}_{\mathcal{Z}}, \{Q_1^1, \ldots, Q_{m_1}^1\}, \ldots, \{Q_1^j, \ldots, Q_{m_j}^j\}]$$

where the first class $\{Q_1^0, \ldots, Q_{m_1}^0\}_{\mathcal{Z}}$, marked with the subscript $\mathcal{Z}$ contains all facts whose timestamp's decimal part is zero, *i.e.*, $dec(Q_i^0) = 0$ for $1 \leq i \leq m_0$. We call it the *Zero Point*. Notice that the zero point may be empty. For a unit circle, $\mathcal{U}$, we define: $\mathcal{U}(Q_j^i) = i$ to denote the class in which the fact $Q_j^i$ appears in $\mathcal{U}$.

For example, the unit circle of configuration $\mathcal{S}_1$ given in Eq. 8 is the sequence: $\mathcal{U}_{\mathcal{S}_1} = [\{S\}_{\mathcal{Z}}, \{M\}, \{R\}, \{P, Time\}, \{Q\}]$. Notice that $P$ and $Time$ are in the same class as the decimal parts of their timestamps are the same, namely 0.12. Moreover, we have that $\mathcal{U}_{\mathcal{S}_1}(S) = 0 < 2 = \mathcal{U}_{\mathcal{S}_1}(R)$, specifying that the decimal part of the timestamp of the fact $R$ is greater than the decimal part of the timestamp of the fact $S$.

We will graphically represent a unit circle as shown in Figure 5. The (green) ellipse at the top of the circle marks the zero point, while the remaining classes are placed on the circle in the (red) squares ordered clockwise starting from the zero point. Thus, from the above graphical representation, the decimal part of the timestamp of the fact $Q_1^1$ is smaller than the decimal of the timestamp of the fact $Q_1^2$, while the decimal part of the timestamps of the facts $Q_1^i$ and $Q_2^i$ are equal. The exact point where the squares are placed is not important, only their relative positions matter, *e.g.*, the square for the class containing the fact $Q_1^1$ should be placed on the circle somewhere in between the zero point and the square for the class containing the fact $Q_1^2$, clockwise.

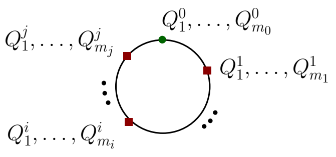
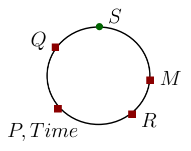

$Q_1^0, \ldots, Q_{m_0}^0$
$Q_1^j, \ldots, Q_{m_j}^j$
$Q_1^1, \ldots, Q_{m_1}^1$
$Q_1^i, \ldots, Q_{m_i}^i$

Fig. 5: Unit Circle

*Constraint Satisfaction*

A circle-configuration $\langle \varDelta, \mathcal{U} \rangle$ contains all the information needed in order to determine whether a constraint of the form given in Eq. 5 is satisfied or not. Consider the circle-configuration in Figure 6 which corresponds to configuration $\mathcal{S}_1$, Eq. 8. To determine, for instance, whether $t_Q > t_{Time} + 1$, we compute the integer difference between $t_Q$ and $t_{Time}$ from the $\delta$-configuration. This turns out to be 1 and means that we need to look at the decimal part of these timestamps to



$\langle \{M, R\}, 1, \{P\}, \infty, \{Time\}, 1, \{Q\}, 2, \{S\} \rangle$

Fig. 6: Circle-Configuration

determine whether the constraint is satisfied or not. Since the decimal part of $t_Q$ is greater then the decimal part of $t_{Time}$, as can be observed in the unit circle, we can conclude that the constraint is satisfied. Similarly, one can also conclude that the constraint $t_Q > t_{Time} + 2$ is not satisfied as $int(t_Q) = int(t_{Time}) + 1$. The following definition formalizes this intuition.

**Definition 2.** *Let $\langle \Delta, \mathcal{U} \rangle$ be a circle-configuration. We say that $\langle \Delta, \mathcal{U} \rangle$ satisfies the constraint involving the timestamps of two arbitrary facts $P$ and $Q$ in the circle-configuration, where $D$ is a natural number, as defined by cases:*
- $t_P > t_Q + D$ *iff* $\Delta(Q, P) > D$ *or* $\Delta(Q, P) = D$ *and* $\mathcal{U}(P) > \mathcal{U}(Q)$;
- $t_P > t_Q - D$ *iff* $\Delta(P, Q) < D$ *or* $\Delta(P, Q) = D$ *and* $\mathcal{U}(P) > \mathcal{U}(Q)$;
- $t_P = t_Q + D$ *iff* $\Delta(Q, P) = D$ *and* $\mathcal{U}(Q) = \mathcal{U}(P)$;
- $t_P = t_Q - D$ *iff* $\Delta(P, Q) = D$ *and* $\mathcal{U}(Q) = \mathcal{U}(P)$;

**Proposition 2.** *For a given upper bound $D_{max}$, the configuration $\mathcal{S}$ satisfies a constraint $c$ of the form $t_P > t_Q \pm D$ or $t_P = t_Q \pm D$, for any facts $P, Q \in \mathcal{S}$ and $D \leq D_{max}$ iff its circle-configuration also satisfies the same constraint $c$.*

## 4.2 Rewrite Rules and Plans with Circle-Configurations

This section shows that given a reachability problem with a set of rules, $\mathcal{A}$, involving dense time, and an upper bound on the numbers appearing in the problem, $D_{max}$, we can compile a set of rewrite rules, $\mathcal{C}$, over circle-configurations. Moreover, we show that any plan generated using the rules from $\mathcal{A}$ can be soundly and faithfully represented by a plan using the set of rules $\mathcal{C}$. We first explain how we apply instantaneous rules to circle-configurations and then we explain how to handle the time advancement rule.

*Instantaneous Actions*

Let $D_{max}$ be an upper bound on the numeric values in the given problem and let the following rule be an instantaneous rule (see Section 3.1) in the set of actions $\mathcal{A}$:

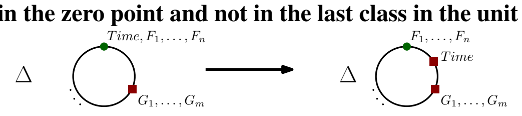
$$\begin{array}{l} Time@T, W_1@T_1, \ldots, W_k@T_k, F_1@T_1', \ldots, F_n@T_n' \mid C \longrightarrow \\ \quad \exists \boldsymbol{X}.[Time@T, W_1@T_1, \ldots, W_k@T_k, Q_1@(T + D_1), \ldots, Q_m@(T + D_m)] \end{array}$$

The above rule is compiled into a sequence of operations that may rewrite a given circle-configuration $\langle \Delta, \mathcal{U} \rangle$ into another circle-configuration $\langle \Delta_1, \mathcal{U}_1 \rangle$ as follows:

1. Check whether there are occurrences of $W_1, \ldots, W_k$ and $F_1, \ldots, F_n$ in $\langle \Delta, \mathcal{U} \rangle$ such that the guard $C$ is satisfied by $\langle \Delta, \mathcal{U} \rangle$. If it is the case, then continue to the next step; otherwise the rule is not applicable;
2. We obtain the circle-configuration $\langle \Delta', \mathcal{U}' \rangle$ by removing the occurrences $F_1, \ldots, F_n$ in $\langle \Delta, \mathcal{U} \rangle$ used in step 1, and recomputing the truncated time differences so that for all the remaining facts $P$ and $R$ in $\Delta$, we have $\Delta'(P, R) = \Delta(P, R)$, *i.e.*, the truncated time difference between $P$ and $R$ is preserved;
3. Create fresh values, $\boldsymbol{e}$, for the existentially quantified variables $\boldsymbol{X}$;
4. We obtain the circle-configuration $\langle \Delta_1, \mathcal{U}_1 \rangle$ by adding the facts $Q_1[\boldsymbol{e}/\boldsymbol{X}], \ldots, Q_m[\boldsymbol{e}/\boldsymbol{X}]$ to $\Delta'$ so that $\Delta_1(Time, Q_i) = D_i$ for $1 \leq i \leq m$ and that $\Delta_1(P, R) = \Delta'(P, R)$ for all the remaining facts $P$ and $R$ in $\Delta'$. Moreover, we obtain $\mathcal{U}_1$ by adding $Q_1, \ldots, Q_m$ to the class of $Time$ in the unit circle $\mathcal{U}'$;
5. Return the circle-configuration $\langle \Delta_1, \mathcal{U}_1 \rangle$.

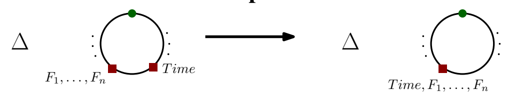
- **Time in the zero point and not in the last class in the unit circle, where $n \geq 0$:**

**Rule 0:**



- **Time alone and not in the zero point nor in the last class in the unit circle:**

**Rule 1:**



- **Time not alone and not in the zero point nor in the last class in the unit circle:**

**Rule 2:**

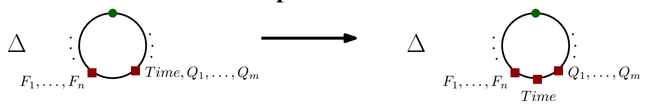


Fig. 7: Rewrite Rules for Time Advancement using Circle-Configurations.

The sequence of operations described above has the effect one would expect: replace the facts $F_1, \ldots, F_n$ in the pre-condition of the action with facts $Q_1, \ldots, Q_m$ appearing in the post-condition of the action but taking care to update the truncated time differences in the $\delta$-configuration. Moreover, all steps can be computed in polynomial time.

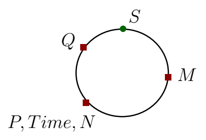
For example, consider the configuration $\mathcal{S}_1$ given in Eq. 8 and the rule:

$Time@T, R@T_1, P@T_2 \rightarrow Time@T, P@T_2, N@(T + 2)$

If we apply this rule to $\mathcal{S}_1$, we obtain the configuration

$\mathcal{S}_2 = \{M@3.01, P@4.12, Time@11.12, Q@12.58, N@13.12, S@14\}$.

On the other hand, if we apply the above steps to the circle-configuration of $\mathcal{S}_1$, shown in Figure 6, we obtain the circle-configuration shown to the right. It is easy to check that this is indeed the circle-configuration of $\mathcal{S}_2$. The truncated time differences are updated and the fact $N$ is added to the class of $Time$ in the unit circle.



$\langle \{M\}, 1, \{P\}, \infty, \{Time\}, 1, \{Q\}, 1, \{N\}, 1, \{S\} \rangle$

*Time Advancement Rule*

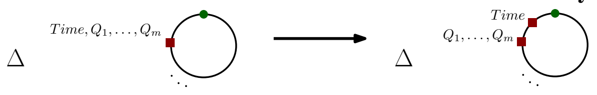
Specifying the time advancement rule (Eq. 3 shown in Section 3.1) over circle-configurations is more interesting. This action is translated into the rules depicted in Figures 7 and 8. There are eight rules that rewrite a circle-configuration, $\langle \varDelta, \mathcal{U} \rangle$, depending on the position of the fact $Time$ in $\mathcal{U}$.

Rule 0 specifies the case when the fact $Time$ appears in the zero point of $\mathcal{U}$. Then $\mathcal{U}$ is re-written so that a new class is created immediately after the zero point clockwise, and $Time$ is moved to that class. This denotes that the decimal part of $Time$ is greater than zero and less than the decimal part of the facts in the following class $G_1, \ldots, G_n$.

Rule 1 specifies the case when $Time$ appears alone in a class on the unit circle and not in the last class. This means that there are some facts, $F_1, \ldots, F_n$, that appear in a class immediately after $Time$, *i.e.*, $\mathcal{U}(F_i) > \mathcal{U}(Time)$ and for any other fact $G$, such that $\mathcal{U}(G) > \mathcal{U}(Time)$, $\mathcal{U}(G) > \mathcal{U}(F_i)$ holds. In this case, then time can advance so that it ends up in the same class as $F_i$, *i.e.*, time has advanced so much that its decimal part is the same as the decimal part of the timestamps of $F_1, \ldots, F_n$. Therefore a constraint of the form $T_{F_i} > T_{Time} + D$ that was satisfied by $\langle \varDelta, \mathcal{U} \rangle$ might no longer be satisfied by the resulting circle-configuration, depending on $D$ and the $\delta$-configuration $\varDelta$.

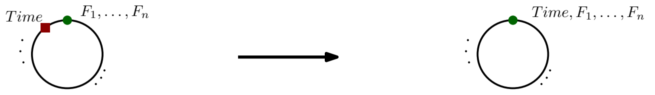
- **Time not alone and in the last class in the unit circle which may be at the zero point:**

Rule 3:



- **Time alone and in the last class in cnit circle - Case 1:** $m > 0, k \geq 0, n \geq 0$ **and** $\delta_1 > 1$**:**

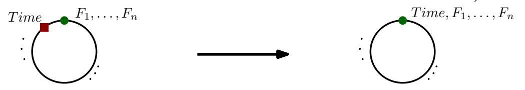
Rule 4:



$\Delta = \langle \dots, \mathcal{P}_{-1}, \delta_{-1}, \{Time, Q_1, \dots, Q_m\}, \delta_1, \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$ $\Delta' = \langle \dots, \mathcal{P}_{-1}, \delta_{-1}, \{Q_1, \dots, Q_m\}, 1, \{Time\}, \delta_1 - 1, \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$

- **Time alone and in the last class in unit circle - Case 2:** $m > 0, k \geq 1$ **and** $n \geq 0$**:**

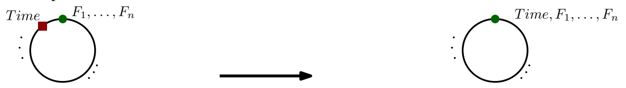
Rule 5:



$\Delta = \langle \dots, \mathcal{P}_{-1}, \delta_{-1}, \{Time, Q_1, \dots, Q_m\}, 1, \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$ $\Delta' = \langle \dots, \mathcal{P}_{-1}, \delta_{-1}, \{Q_1, \dots, Q_m\}, 1, \{Time\} \cup \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$

- **Time alone and in the last class in unit circle - Case 3:** $k \geq 0, \delta_1 > 1$ **and** $\gamma_{-1}$ **is the truncated time of** $\delta_{-1} + 1$**:**

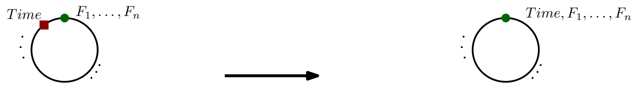
Rule 6:



$\Delta = \langle \dots, \mathcal{P}_{-1}, \delta_{-1}, \{Time\}, \delta_1, \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$ $\Delta' = \langle \dots, \mathcal{P}_{-1}, \gamma_{-1}, \{Time\}, \delta_1 - 1, \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$

- **Time alone and in the last class in unit circle - Case 4:** $k \geq 1$ **and** $\gamma_{-1}$ **is the truncated time of** $\delta_{-1} + 1$**:**

Rule 7:

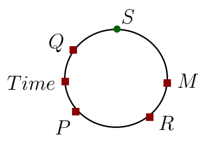


$\Delta = \langle \dots, \mathcal{P}_{-1}, \delta_{-1}, \{Time\}, 1, \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$ $\Delta' = \langle \dots, \mathcal{P}_{-1}, \gamma_{-1}, \{Time\} \cup \mathcal{P}_1, \dots, \mathcal{P}_k \rangle$

Fig. 8: (Cont.) Rewrite Rules for Time Advancement using Circle-Configurations.

Rule 2 is similar, but is only applicable when $Time$ is not alone in the unit circle class, *i.e.*, there is at least one fact $F_i$ such that $\mathcal{U}(Time) = \mathcal{U}(F_i)$ and this class is not the last one, as in Rule 1. Rule 2 advances time enough so that its decimal part is greater than the decimal part of the timestamps of $F_i$, but not greater than the decimal part of the timestamps of the facts in the class that immediately follows on the circle.

For example, Rule 2 could be applied to the circle-configuration shown in Figure 6. We obtain the following circle-configuration, where the $\delta$-configuration does not change, but the fact $Time$ is moved to a new class on the unit circle, obtaining the circle-configuration $C_{\mathcal{S}_2}$ shown to the right.

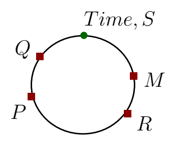


$\langle \{M, R\}, 1, \{P\}, \infty, \{Time\}, 1, \{Q\}, 2, \{S\} \rangle$

Rule 3 is similar to Rule 2, but it is applicable when $Time$ is in the last equivalence class, in which case a new class is created and placed clockwise immediately before the zero point of the circle.

Notice that the $\delta$-configuration is not changed by Rules 0-3. The only rules that change the $\delta$-configuration are the Rules 4, 5, 6 and 7, as in these cases $Time$ advances enough to complete the unit circle, *i.e.*, reach the zero point. Rules 4 and 5 handle the case when $Time$ initially has the same integer part as timestamps of other facts $Q_1, \dots, Q_m$, in which case it might create a new class in the $\delta$-configuration (Rule 4) or merge with the following class $\mathcal{P}_1$ (Rule 5). Rules 6 and 7 handle the case when $Time$ does not have the same integer part as the timestamp of any other fact, *i.e.*, it appears alone in $\Delta$, in which case it might still remain alone in the same class (Rule 6) or merge with the following class $\mathcal{P}_1$ (Rule 7). Notice that the time difference, $\delta_{-1}$, to the class, $\mathcal{P}_{-1}$, immediately before the class of $Time$ is incremented by one and truncated by the value of $D_{max}$ if necessary.

For example, it is easy to check that applying Rule 1, followed by Rule 3 to circle-configuration $C_{\mathcal{S}_2}$ shown above, we obtain a circle-configuration for which the Rule 7 is applicable. After applying Rule 7 we obtain the configuration shown to the right.



$\langle\{M,R\},1,\{P\},\infty,\{Time,Q\},2,\{S\}\rangle$

Given a reachability problem $\mathcal{T}$ and an upper bound $D_{max}$ on the numeric values of $\mathcal{T}$ with the set of rules $\mathcal{R}$ containing an instantaneous rule $r$, we write $[r]$ for the corresponding rewrite rule of $r$ over circle-configurations as described above. Moreover, let $Next$ be the set of 8 time advancing rules shown in Figures 7 and 8. Notice that for a given circle-configuration only one of these rules is applicable. We use $C \longrightarrow_{rl} C_1$ for the one-step reachability relation using the rewrite rule $rl$, *i.e.*, the circle-configuration $C$ may be rewritten to the circle-configuration $C_1$ using the rewrite rule $rl$. Finally, $C \longrightarrow^* C_1$ (respectively, $C \longrightarrow^*_{\mathcal{R}'} C_1$) denotes the reflexive transitive closure relation of the one-step relation (respectively, using only rules in the set $\mathcal{R}' \subseteq \mathcal{R}$).

**Lemma 1.** *Let $\mathcal{T}$ be a reachability problem and $D_{max}$ be an upper bound on the numeric values in $\mathcal{T}$. Let $\mathcal{S}_1$ be a configuration, whose circle-configuration is $C_1$, and $r$ be an instantaneous action in $\mathcal{T}$. Then $\mathcal{S}_1 \longrightarrow_r \mathcal{S}_2$ if and only if $C_1 \longrightarrow_{[r]} C_2$ and $C_2$ is the circle-configuration of $\mathcal{S}_2$. Moreover, $\mathcal{S}_1 \longrightarrow_{Tick} \mathcal{S}_2$ if and only if $C_1 \longrightarrow^*_{Next} C_2$ and $C_2$ is the circle-configuration of $\mathcal{S}_2$.*

**Theorem 2.** *Let $\mathcal{T}$ be a reachability problem, $D_{max}$ be an upper bound on the numeric values in $\mathcal{T}$. Then $\mathcal{S}_I \longrightarrow^* \mathcal{S}_G$ for some initial and goal configurations, $\mathcal{S}_I$ and $\mathcal{S}_G$, in $\mathcal{T}$ if and only if $C_I \longrightarrow^* C_G$ where $C_I$ and $C_G$ are the circle-configurationss of $\mathcal{S}_I$ and $\mathcal{S}_G$, respectively.*

This theorem establishes that the set of plans over circle-configurations is a sound and complete representation of the set of plans with dense time. This means that we can search for solutions of problems symbolically, that is, without writing down the explicit values of the timestamps, *i.e.*, the real numbers, in a plan.

## 5 Complexity Results

This section details some of the complexity results for the reachability problem.

*Conditions for Decidability*

From the Literature, we can infer some conditions for decidability of the reachability problem in general:

1. *Upper Bound on the Size of Facts*: In general, if we do not assume an upper bound on the size of facts appearing in a plan, where the size of facts is the total number of predicate, function, constant and variable symbols it contains (*e.g.* the size of

$P(f(a), x, a)$ is 5), then it is easy to encode the Post-Correspondence problem which is undecidable, see [6,11].[10] Thus we will assume an upper bound on the size of facts, denoted by the symbol $k$.

2. *Balanced Actions*: An action is balanced if its pre-condition has the same number of facts as its post-condition [20]. The reachability problem is undecidable for (un-timed) systems with possibly unbalanced actions even if the size of facts is bounded [6, 11]. In a balanced system, on the other hand, the number of facts in any configuration in a plan is the same as the number of facts of the initial configuration, allowing one to recover decidability under some additional conditions. We denote the number of facts in the configuration by the symbol $m$.

As all these undecidability results are time irrelevant, they carry over to systems with dense time.

**Corollary 1.** *The reachability problem for our model is undecidable in general.*

*PSPACE-Completeness* We show that the reachability problem for our model with dense time and balanced actions is PSPACE-complete. Interestingly, the same problem is also PSPACE-complete when using models with discrete time [18].

Given the machinery in Section 4, we can re-use many results in the Literature to show that the reachability problem is also PSPACE-complete for balanced systems with dense time that can create fresh values, given in Section 3, assuming an upper bound on the size of facts. For instance, we use the machinery detailed in [15] to handle the fact that a plan may contain an unbounded number of fresh values.

The PSPACE lower bound can be inferred from [15]. The interesting bit is to show PSPACE membership of the reachability problem. The following lemma establishes an upper bound on the number of different circle-configurations:

**Lemma 2.** *Given a reachability problem $\mathcal{T}$ under a finite alphabet $\Sigma$, an upper bound on the size of facts, $k$, and an upper bound, $D_{max}$, on the numeric values appearing in $\mathcal{T}$, then the number of different circle-configurations, denoted by $L_{\mathcal{T}}(m, k, D_{max})$, with $m$ facts (counting repetitions) is $L_{\mathcal{T}}(m, k, D_{max}) \leq J^m (D+2mk)^{mk} m^m (D_{max}+2)^{(m-1)}$, where $J$ and $D$ are, respectively, the number of predicate and the number of constant/function symbols in $\Sigma$.*

Intuitively, our upper bound algorithm keeps track of the length of the plan it is constructing and if its length exceeds $L_{\mathcal{T}}(m, k, D_{max})$, then it knows that it has reached the same circle-configuration twice. This is possible in PSPACE since the above number, when stored in binary, occupies only polynomial space with respect to its parameters. The proof of the result below is similar to the one in given in [15].

**Theorem 3.** *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 $\mathcal{T}$.*

[10] We leave for Future Work the investigation of specific cases, *e.g.*, protocol with tagging mechanisms, where this upper bound may be lifted [27].

## 6 Related and Future Work

The formalization of timed models and their use in the analysis of cyber-physical security protocols has already been investigated. 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 we have implemented our specification in Maude [7]. Finally, they do not investigate the complexity of protocol analysis 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.* [2]. They formalize an intruder model and some cyber-physical security protocols in Isabelle. They then prove the correctness of these protocols under some specific conditions and also identify attacks when some conditions are not satisfied. Their work was a source of inspiration for our intruder model specified in [17], which uses the model described in Section 3. Although their model includes time, their model is not refined enough to capture the attack in-between-ticks as they do not consider the discrete behaviour of the verifier.

Recently [3] proposed a discrete time model for formalizing distance bounding protocols and their security requirements. Thus they are more interested in the computational soundness of distance bounding protocols by considering an adversary model based on probabilistic Turing machines. They claim that their SKI protocol is secure against a number of attacks. However, their time model is discrete where all players are running at the same clock rate. Therefore, their model is not able to capture attacks that exploit the fact that players might run at different speeds.

The Timed Automata [1] (TA) literature contains models for cyber-physical protocol analysis. Corin *et al.* [8] formalize protocols and the standard Dolev-Yao intruder as timed automata and demonstrate that these can be used for the analysis. 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 [15]. Also they do not investigate the complexity of the analysis problems nor the expressiveness difference between models with discrete and continuous time. Lanotte *et al.* [21] 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 EXPSPACE-hard lower bound for the reachability problem. It seems possible to specify features like timeouts and message re-transmission, in our rewriting formalism.

We also point out some important differences between our PSPACE-completeness proof and PSPACE-completeness proof for timed automata [1]. A more detailed account can be found in the Related Work section of [19]. The first difference is that we do not impose any bounds on the number of nonces created, while the TA proof normally assumes a bound. The second difference is due to the first-order nature of rewrite rules. The encoding of a first-order system in TA leads to an exponential blow-up on the number of states of the automata as one needs take into account all instantiations of rules. Finally, the main abstractions that we use, namely circle-configurations, are one-dimensional, while regions used in the TA PSPACE proof are multidimensional.

Malladi *et al.* [23] formalize distance bounding protocols in strand spaces. They then construct an automated tool for protocol analysis using a constraint solver. They did not take into account the fact that the verifier is running a clock in their analysis and therefore are not able to detect the attack in-betweeen-ticks.

Finally, [9] introduces a taxonomy of attacks on distance bounding protocols, which include a new attack called Distance Hijacking Attack. This attack was caused by failures not in the time challenges phase of distance bounding protocols, but rather in the autenthication phases. It would be interesting to understand how these attacks can be combined with the attack in-between-ticks to build more powerful attacks. We are investigating completeness theorems for the analysis of protocols against types of attacks in the taxonomy. For example, how many colluding intruders is enough.

Another well known formalism that involves time is Time Petri Nets and we plan to investigate the relationship to our model in the future.

*Acknowledgments:* Nigam is supported by the Brazilian Research Agencies CNPq and Capes. Talcott is partially supported by NSF grant CNS-1318848. Kanovich is supported in part by EPSRC. 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 ONR. Part of the work was done while 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. R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *SFM*, pages 1–24, 2004.
2. 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.
3. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical & provably secure distance-bounding. *IACR Cryptology ePrint Archive*, 2013:465, 2013.
4. S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *EUROCRYPT*, pages 344–359, 1993.
5. S. Capkun and J.-P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):221–232, 2006.
6. 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.
7. 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*. 2007.
8. R. Corin, S. Etalle, P. H. Hartel, and A. Mader. Timed analysis of security protocols. *J. Comput. Secur.*, 15(6):619–645, dec 2007.
9. C. J. F. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *SP*, 2012.
10. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

11. 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.
12. H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.
13. S. Escobar, C. Meadows, J. Meseguer Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties. *Foundations of Security Analysis and Design V*, 2009.
14. S. Ganeriwal, C. Pöpper, S. Capkun, and M. B. Srivastava. Secure time synchronization in sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(4), 2008.
15. M. Kanovich, T. B. Kirigin, V. Nigam, and A. Scedrov. Bounded memory Dolev-Yao adversaries in collaborative systems. *Inf. Comput.*, 2014.
16. M. I. Kanovich, T. B. Kirigin, V. Nigam, and A. Scedrov. Bounded memory protocols and progressing collaborative systems. In *ESORICS*, pages 309–326, 2013.
17. M. I. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, and C. L. Talcott. Towards timed models for cyber-physical security protocols. Available on Nigam's homepage, 2014.
18. 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 *RTA*, pages 305–322, 2012.
19. 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 on Nigam's homepage, 2014.
20. M. I. Kanovich, P. Rowe, and A. Scedrov. Collaborative planning with confidentiality. *J. Autom. Reasoning*, 46(3-4):389–421, 2011.
21. 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.
22. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *TACAS*, pages 147–166, 1996.
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. R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.
26. D. Pavlovic and C. Meadows. Deriving ephemeral authentication using channel axioms. In *Security Protocols Workshop*, pages 240–261, 2009.
27. R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. In *FST TCS*, 2003.
28. K. Ravi, G.H.Varun, and P. T.Vamsi. Rfid based security system. *International Journal of Innovative Technology and Exploring Engineering*, 2, 2013.
29. V. Shmatikov and M.-H. Wang. Secure verification of location claims with simultaneous distance modification. In *ASIAN*, pages 181–195, 2007.
30. K. Sun, P. Ning, and C. Wang. Tinysersync: secure and resilient time synchronization in wireless sensor networks. In *CCS*, pages 264–277, 2006.
31. N. O. Tippenhauer and S. Capkun. Id-based secure distance bounding and localization. In *ESORICS*, pages 621–636, 2009.