Resource and Timing Aspects of Security Protocols

Abraão Aires Urquiza [a], Musab A. Alturki [b,c], Tajana Ban Kirigin [d,*], Max Kanovich [e,f], Vivek Nigam [g,a], Andre Scedrov [h,i] and Carolyn Talcott [j]

[a] *Federal University of Paraíba, João Pessoa, Brazil,*
E-mail: abraauc@gmail.com
[b] *KFUPM, Dhahran, Saudi Arabia,*
E-mail: musab@kfupm.edu.sa
[c] *Runtime Verification Inc., USA*
[d] *Department of Mathematics, University of Rijeka, Rijeka, Croatia,*
E-mail: bank@math.uniri.hr
[e] *University College London, London, UK,*
E-mail: m.kanovich@ucl.ac.uk
[f] *National Research University Higher School of Economics, Moscow, Russian Federation*
[g] *fortiss, Germany,*
E-mail: nigam@fortiss.org
[h] *University of Pennsylvania, Philadelphia, PA, USA,*
E-mail: scedrov@math.upenn.edu
[i] *National Research University Higher School of Economics, Moscow, Russian Federation, until July 2020.*
[j] *SRI International, Menlo Park, CA, USA,*
E-mail: clt@csl.sri.com

Abstract. Protocol security verification is one of the best success stories of formal methods. However, some aspects important to protocol security, such as time and resources, are not covered by many formal models. While timing issues involve *e.g.*, network delays and timeouts, resources such as memory, processing power, or network bandwidth are at the root of Denial of Service (DoS) attacks which have been a serious security concern. It is useful in practice and more challenging for formal protocol verification to determine whether a service is vulnerable not only to powerful intruders, but also to resource-bounded intruders that cannot generate or intercept arbitrarily large volumes of traffic. A refined Dolev-Yao intruder model is proposed, that can only consume at most some specified amount of resources in any given time window. Timed protocol theories that specify service resource usage during protocol execution are also proposed. It is shown that the proposed DoS problem is undecidable in general and is PSPACE-complete for the class of resource-bounded, balanced systems. Additionally, we describe a decidable fragment in the verification of the leakage problem for resource-sensitive timed protocol theories.

Keywords: Multiset Rewriting, Protocol Security, Complexity, Denial of Service

*Corresponding author. E-mail: bank@math.uniri.hr. Department of Mathematics, University of Rijeka, Radmile Matejcic 2, 51000 Rijeka, Croatia, tel. +385-51-584650

1. Introduction

Protocol security verification is one of the best success stories of formal methods. Indeed, a number of attacks and corrections have been discovered since Lowe found an attack on the Needham-Schroeder protocol [1, 2]. Valid verification relies on the careful formalization of all the relevant assumptions of a protocol execution. However, much of the use of formal methods does not consider some protocol aspects and assumptions such as the ones related to time. Timing aspects are particularly important in the verification of cyber-physical systems, including, *e.g.*, distance-bounding protocols.

Another important aspect of protocol verification are resources. For the past decades, Denial of Service (DoS) attacks and their distributed version (DDoS) have been a serious security concern, as no service is, in principle, protected against them. Indeed, if an intruder, such as the Dolev-Yao (DY) intruder [3], has enough resources, he may render any service unavailable by sending a large number of messages (flooding attacks) or by intercepting all messages in the network (jamming attacks). What makes the DoS threat even worse is that these attacks do not necessarily have to be carried out by (extremely) powerful intruders with almost unbounded resources. Attacks such as slow DoS attacks [4–11], asymmetric DoS attacks [12, 13], and amplification attacks [14, 15], can all be carried out by intruders using limited resources. For example, web-servers, such as Apache [4] and NGinx [6] can be successfully attacked by intruders with limited resources using, for instance, mobile phones (SlowDroid [16]).

However, it is hard to determine whether a service is vulnerable to such attacks. While a very powerful intruder with unbounded resources would simply flood the service rendering it unavailable, a resource-bounded intruder carries out an attack by exploiting the protocols used by the target service. He not only triggers a particular sequence of events to consume the service's resources, but also cleverly tries to minimize his effort by triggering events as lazily as possible, renewing service timeouts as late as possible, or by enlisting the help of other benign nodes in the network. Indeed, in many attacks [4, 16], the volume of traffic generated by the intruder is comparable to the volume of a legitimate client, thus making it hard for network administrators to even identify when the service is under attack. Therefore, determining such vulnerabilities in advance may help prevent attacks by installing suitable countermeasures.

Intruders can also exploit a wide range of types of resources. For example, Slowloris consumes the limited number of workers web-servers possess; Software Defined Network (SDN) TCAM exhaustion attacks [8, 9] consume the limited amount of TCAM memory of switches; TLS renegotiation DoS attacks [17] consume the server's processing power; SIP forking amplification attacks on Voice-over-IP (VoIP) systems [14, 15, 18, 19] consume the network bandwidth.

While security issues relevant for DoS attacks have been identified, *e.g.*, in [20], where a taxonomy of DoS attacks is given, the main contribution of this paper is on formal verification of DDoS, pioneered by the cost-based framework for analyzing DoS attacks from [21]. The proposed formalization includes timing aspects such as network and processing delays, as well as protocol timeouts that have specific applications in a variety of protocols.

The main contributions of the paper are the following:

(1) We formally define the *DoS problem*. In contrast to existing definitions, such as, the one proposed in [21], our DoS problem takes into account timing aspects, *i.e.*, duration of the attack. All contributions listed below build on this conceptual advance. The key technical challenge is to formally specify behaviors, formalized as traces, where services behave as expected, *e.g.*, triggering timeouts whenever applicable. This is accomplished by specifying configurations that are not allowed, called *critical configurations*. As we use dense time domains, the notion of a trace that does not involve critical configurations, called *non-critical traces*, becomes much more elaborate than in models

using discrete domains, since in dense time one cannot list all the moments that the trace covers. Therefore, ensuring that a trace is non-critical requires checking that all possible decompositions of time advancements, and there are *infinitely many*, do not contain critical configurations.

(2) *Protocol resource theories* are introduced, which refine protocol theories of [22, 23] with resource usage and timing aspects, including action duration and timeouts. For the formalization, the existing Timed Multiset Rewriting (MSR) model with real time of [24] is extended;

(3) *Resource-bounded intruder models* are introduced, which refine the DY intruder with resource usage and action duration. For formal verification, the traditional DY intruder [3] can trivially deny any service by, for example, blocking all communication. Furthermore, the traditional DY intruder is able to intercept and send messages anywhere at anytime, appearing, hence, faster than the speed of light. Hence, such an intruder is too powerful for the purposes of this paper. Inspired by the work [21], instead, a parametric resource-bounded intruder model is proposed, where intruder's actions consume resources and time;

(4) The *expressiveness of the model* is demonstrated by modeling different types of attacks. The expressiveness of the model with respect to timing aspects, is shown by specifying protocols with timeouts. Besides traditional DoS attacks, such as the SYN flooding attack, the model can also express the types of DoS attacks described above (Slow DoS, Asymmetric and Amplification attacks). Additionally, it is also shown how to model *countermeasures* based on traffic monitoring, such as, defenses [25] deployed in web-servers to mitigate the Slowloris attack;

(5) We prove that under suitable conditions, the non-critical reachability problem for general MSR models with dense time is PSPACE-complete, which non-trivially advances previous reachability problems [24] that did not consider critical configurations. This result is essential for the complexity results of verification problems involving non-critical traces.

(6) From the general result for non-critical reachability problem, we prove that our DoS problem is PSPACE-complete for a wide class of resource-bounded intruders. Moreover, it is proven that the DoS problem is undecidable in general;

(7) Verification of the *leakage problem* related to resource-sensitive timed protocol and intruder theories is also investigated. General undecidability and a PSPACE-completeness result for a variation of the leakage problem is shown. This builds on the past work in which a rich complexity theory for problems has been developed, formulated in terms of (Timed) MSR [24]. However, here the focus is on specific application of MSR models to resource-sensitive timed protocol specification and verification;

(8) Finally, we describe in detail the machinery implemented using the rewriting tool Maude [26] for discovering vulnerabilities. Our machinery takes as input an abstract specification of the resource usage and timeouts of protocols, in the form of a mode automaton, and performs symbolic search using rewriting modulo SMT [27] to discover attacks. Moreover, based on our theoretical study (balanced theories), we propose ways to improve search performance by performing bounded model-checking. This implementation is available at [28].

This paper combines and extends the conference papers [29, 30]. More specifically, the description of the automated verification has been greatly expanded w.r.t. [29], including the specification of methodology and new experiments. Also, detailed complexity proofs are provided and theories have been elaborated to better incorporate both time and resource aspects. In particular, we extend our previous papers with more general protocol theories that incorporate timeout based countermeasures given in Section 4.4. In addition, we consider the leakage problem, a refined version of the secrecy problem that involves resource

and time-sensitive protocol and intruder theories, which has not been previously investigated. Related complexity results are obtained.

The paper starts by describing the use of resources and timeouts in protocols, as well as presenting some examples of known DoS attacks in Section 2. In Section 3 the Timed MSR model of [24] is presented and the notion of non-critical traces for dense time MSR models is introduced. Applications of Timed MSR model to protocol specification and verification start with Section 4 where protocol resource theories are defined, while in Section 5 timed resource-bounded intruder model is introduced. It is described how the model can be used to specify intruders that can only generate bounded traffic or have bounded processing power. The expressivity of the theories is illustrated with some examples of protocol theories and attacks. In Section 6 verification problems are specified. The related complexities are studied in Section 7. Section 8 describes the results on automated search for DoS attacks. Finally, Section 9 concludes by discussing related work and points to future work.

2. Motivating Examples

The importance of expressing timing and resource aspects of protocols in the protocol verification is illustrated with several examples. The first one is on the use of timeouts. The second example illustrates the essence of DoS attacks, that is, the consumption of resources.

2.1. Timeouts

Protocol session timeouts become relevant when considering timing aspects, such as network communication delays. Http/Https protocols use timeouts to limit waiting time in multiple situations: idle connections, client waiting for server response, server waiting for client to complete a request etc. The Session Initiation Protocol (used by VoIP and other communication protocols) uses timers to limit the waiting time during different steps of the protocol. For example, if the called party is not available, the initialization should not ring forever! The ability to reset timers provides readily available attack surfaces.

Lifetime/time-to-live is another important time related concept. Networking protocols (for example, TLS [31], Kerberos) often use *tickets* to control access. These tickets typically have a lifetime after which they are no longer valid. Packets traveling through the network (for example TCP/IP) often have a time-to-live to avoid loops and problems delaying delivery.

Related verification using the traditional DY intruder may lead to false positives, as the intruder impersonates the network, *i.e.*, forwards messages instantaneously. In [30] a refinement of the DY intruder is proposed, which takes timing aspects, such as processing delays, into account.

2.2. Denial of Service Attacks

Traditional DoS attacks, known as flooding or brute-force attacks, target the main service resource by generating large amount of traffic. The most known example is the SYN flooding attack. Such attacks are always possible in the presence of very powerful intruders, who can send or intercept an unbounded number of messages. Resource-bounded intruders, on the other hand, exploit protocols used by the target service to perform attacks targeting specific service features related to, *e.g.*, protocols or applications used by the service, as detailed in below examples. Hence, to fully capture various types DoS attacks, all relevant resources should be included in the verification model.

Some existing DoS attacks that can be carried out by intruders with bounded resources are reviewed here. Attacks aim to consume different resources of the target service, such as the service's threads/workers, available memory, processing power, or even the network bandwidth. However, instead of generating a large amount of traffic, intruders exploit the protocol used by the target service to consume its resources in a lazy manner.

Attacks Consuming Workers/Threads. Web-servers, such as Apache and NGinx, and VoIP servers, such as Asterisk, are subject to attacks that consume all the available workers in their pool of threads. Examples of such attacks include Slowloris, RUDY, Slowread, and Coordinated Call attacks.

For example, Slowloris [4] is an attack on (connection-based) web-servers such as Apache. The intruder exploits the fact that when a web-server receives a GET request with an incomplete header, it allocates one of its workers to attend to this new request. However, since the GET request is incomplete, the worker is left idle and waits for a new piece of the request header until a timeout is triggered (typically 40 seconds). If a new piece of header arrives and the header is complete, the worker answers the request, but if the header continues to be incomplete, the timeout is reset and the worker waits for another piece until the timeout is triggered. To carry out the Slowloris attack, an intruder sends a burst of incomplete GET requests large enough to occupy all workers (typically 300-400 requests). The intruder does not have to send another burst until a timeout gets close, generating, as a result, very little traffic.

Attacks Consuming Memory. Intruders can target the service's memory. Examples include XML-bombs [12] and Second-Order DDoS attacks [13]. As demonstrated in [8, 9], even sophisticated networks, such as Software Defined Networks (SDN), can be attacked by resource-bounded intruders. In SDN, switches are general devices that use specialized memories called Ternary Content-Addressable Memory (TCAM) for storing routing rules which are installed by a (powerful) SDN controller. Since TCAMs are expensive and require much energy, SDN switches have a limited amount of TCAM memory capable of storing typically at most 5000 rules.

This makes SDN switches subject to Slow-TCAM attacks where the intruder consumes a switch's TCAM memory by forcing the installation of sufficiently many rules. Moreover, to avoid having rules removed, the intruder keeps them alive by sporadically sending new packets that trigger the forwarding rules installed in the switch. Indeed, the intruder can render an SDN switch unavailable by sending less than 4 packets per second.

Attacks Consuming Processing Power. Attacks can also target the processing power of servers. An example of such attack is the Transport Layer Security (TLS) renegotiation DoS attack [17]. TLS [31] is a cryptographic protocol widely used in communication over the Internet. A private connection is established between parties by sharing a private symmetric key created during TLS initialization through a handshake using available public keys. Parties can, however, renegotiate the symmetric key. The initialization process, however, requires more processing power from the server (10 times more effort) than from the client. By issuing a large enough number of renegotiation requests, the attacker can thus consume the processing power of the server, causing the TLS renegotiation DoS attack [17].

Attacks Consuming Network Bandwidth. Instead of targeting a specific resource on a designated server, an intruder with limited resources may target the entire network of servers by mounting an *amplification* DoS attack. The intruder floods the network with messages by expending *minimal* initial effort and exploiting the protocol to enlist the help of other servers in the network, causing the number of messages to amplify to an arbitrary large number while still requiring minimal (or no) further work by the intruder.

An example is the well-known amplification vulnerability in the Session Initiation Protocol (SIP) used to set up VoIP calls [14, 15, 18, 19]. SIP uses a network of SIP proxies to help locate VoIP clients and establish sessions. Generally, to set up a call from client A to another client B, A sends a SIP `INVITE` message (addressed to B) to A's domain SIP proxy, which forwards the message to a SIP proxy of the domain of B, which in turn locates the IP address of B and forwards the invite to B. Typically, however, A's invite message goes through multiple SIP proxies in between. Moreover, a SIP proxy may *fork* an invite message and forward it to multiple nodes (SIP proxies and/or users), a feature that, for example, enables a session to be established with one of several users who can act on behalf of B. However, forking of invite messages makes SIP vulnerable to amplification attacks [14, 15, 19]. Ill-configured or compromised SIP proxies may turn a single invite message into an arbitrarily large number of messages (*e.g.*, through forking loops), flooding the entire network and denying service to users.

3. Timed Multiset Rewriting

We briefly review Timed Multiset Rewriting (MSR) with dense time of [24] which is the language extended here to specify resource-bounded intruders and protocols. Assume a finite first-order typed alphabet, Σ, with variables, constants, function and predicate symbols. Additionally, an unbounded number of fresh values [22, 32] is allowed. Terms and facts are constructed as usual (see [33]), by applying symbols with correct type. 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 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.

Timestamped facts are used to specify systems that explicitly mention time. Timestamped facts are of the form $F@t$, where F is a fact and $t \in \mathbb{R}$ is a *non-negative real number* called *timestamp*. Similarly, time variables denoting timestamps, such as variable T in $F@T$, range over non-negative real numbers. For simplicity, timestamped facts are often referred to as facts.

A special predicate $Time$ with arity zero represents the global time. For example, the fact $Time@10.4$ denotes that the current global time of the system is 10.4. A *configuration*, $\mathcal{S}$, is a multiset of ground timestamped facts, $\mathcal{S} = \{Time@t, F_1@t_1, \ldots, F_n@t_n\}$, with a single occurrence of a $Time$ fact. In a configuration, facts may have timestamps ahead of the global time t, at the global time t or in the past moment. Intuitively, a configuration denotes a state of a system, while $F@t$ is an atomic information denoting that "fact F holds at time t in a configuration $\mathcal{S}$" if $F@t$ is included in $\mathcal{S}$.

Configurations are modified by multiset rewrite rules which can be interpreted as actions of the system. There is only one rule, $Tick$, that modifies global time:

$$Time@T \longrightarrow Time@(T + \varepsilon) \tag{1}$$

where T is a time variable and ε can be instantiated by any non-negative real number, also written $Tick_\varepsilon$ when referred to the $Tick$ rule Eq. (1) for a specific ε. Applied to a configuration, $\{ Time@t, F_1@t_1, \ldots, F_n@t_n \}$, $Tick_\varepsilon$ advances global time by ε, resulting in configuration $\{ Time@(t + \varepsilon), F_1@t_1, \ldots, F_n@t_n \}$. The $Tick$ rule changes only the timestamp of the fact $Time$, while the remaining facts in the configuration (those different from $Time$) are unchanged.

The remaining rules are instantaneous actions, which do not affect the global time, but may rewrite the remaining facts of configurations (those different from $Time$). Instantaneous rules have the form:

$$\begin{aligned} &Time@T,\ W_1@T_1, \ldots,\ W_p@T_p,\ F_1@T'_1, \ldots,\ F_n@T'_n \mid \mathcal{C} \longrightarrow \\ &\exists \vec{X}.\, [\ Time@T,\ W_1@T_1, \ldots,\ W_p@T_p,\ Q_1@(T + D_1), \ldots,\ Q_m@(T + D_m)\ ] \end{aligned} \tag{2}$$

where $D_1, \ldots, D_m$ are natural numbers, $\mathcal{W} = \{ W_1@T_1, \ldots, W_p@T_p \}$ is a multiset of timestamped facts, possibly containing variables, and $\mathcal{C}$ is a set of constraints involving the time variables appearing in the rule's pre-condition, *i.e.*, the variables $T, T_1, \ldots, T_p, T'_1, \ldots, T'_n$. All free variables appearing in the post-condition must appear in the pre-condition. Constraints are of the form:

$$T \geqslant T' \pm D, \quad T > T' \pm D \quad \text{and} \quad T = T' \pm D,$$

where T and T' are time variables, and D is a natural number. The symbol $\pm$ stands for either $+$ or $-$, that is, constraints may involve addition or subtraction. Whenever the set $\mathcal{C}$ of time constraints of a rule is empty, it is omitted.

Finally, the variables $\vec{X}$ that are existentially quantified in the rule Eq. (2) are to be replaced by fresh values, also called *nonces* in protocol security literature [22, 32]. As in the previous work [23], nonces are used whenever a unique identification is required, for example for a protocol session.

Facts $W_1@T_1, \ldots, W_k@T_k$ are preserved by the rule Eq. (2), while $F_1@T'_1, \ldots, F_n@T'_n$ are **replaced** by $Q_1@(T + D_1), \ldots, Q_m@(T + D_m)$. Following [22] a fact is *consumed* (resp. *created*) by some rule r if that fact occurs more (resp. less) times in r on the left side than on the right side. In a rule, the consumed facts are sometimes color red and the created facts blue. Notice that an instantaneous action Eq. (2) may consume facts with timestamps in the past, present or future and create facts with timestamps only in the present or in the (near) future. In that sense, rules Eq. (2) do not change the past.

An instantaneous rule of the form $\mathcal{P} \mid \mathcal{C} \longrightarrow \exists \vec{X}.\mathcal{P}'$ may be applied to a configuration $\mathcal{S}$ if there is a subset $\mathcal{S}_0 \subseteq \mathcal{S}$ and a matching substitution θ, such that $\mathcal{S}_0 = \mathcal{P}\theta$ and $\mathcal{C}\theta$ evaluates to true. Substitution application ($\mathcal{S}\theta$) is defined as usual [33], *i.e.*, by mapping time variables in $\mathcal{S}$ to non-negative real numbers, nonce names to nonce names (renaming of nonces) and term variables to terms. The configuration resulting from the application of this rule is $(\mathcal{S} \setminus \mathcal{S}_0) \cup ((\mathcal{P}'\sigma)\theta)$, where σ is a substitution that maps the existentially quantified variables $\vec{X}$ to fresh constants, that is, constants different from any constant ever used.

An instance of an instantaneous rule can only be applied if all of its constraints are satisfied. For example, the rule $Time@T, F_1(X)@T, F_2(X,Y)@T_1 \mid T_1 \geqslant T + 1 \longrightarrow \exists N.[Time@T, F_1(X)@T, F_3(X,Y,N)@(T+2)]$ can be applied to configuration $\{Time@8.5, F_1(a)@8.5, F_2(a,b)@10.2\}$, yielding the configuration $\{Time@8.5, F_1(a)@8.5, F_3(a,b,n_1)@10.5\}$, where $F_2(a,b)@10.2$ is replaced by $F_3(a,b,n_1)@10.5$ with n_1 being a fresh constant.

Remark 3.1. *In a configuration, an instance of a timestamped fact $F@t$ is interpreted as F holds at time t. If the global time in a configuration is $Time@t_1$ and $t \leqslant t_1$, it only means that F held at t and says nothing about $F@t_1$. There may be other instances of F in the configuration, $F@t_3$, saying that F held at another time, t_3. If $t > t_1$, this is interpreted as F will hold at t (if it isn't removed). This is used to represent timers, reminders, etc. It is the responsibility of the rules to determine whether facts persist, change, or are removed. To see why we don't enforce that $F@T$ implied $F@(T+1)$, consider a message, m, being sent in the network at time t, denoted by the fact $N(m)@t$. It is not correct to imply from this fact that the fact is sent at time $t+1$. Furthermore, notice that rules may use such facts (in the past as well as in the future) to specify aspects such as receiving a message, as in protocol theories (Definition 4.1).*

Definition 3.2 (Timed MSR System)**.** A timed MSR system with dense time $\mathcal{R}$ *is a set of rules containing only instantaneous rules Eq. (2) and the Tick rule Eq. (1).*

Definition 3.3 (Trace)**.** *A* trace *of timed MSR system $\mathcal{R}$ from a given initial configuration $\mathcal{S}_0$ is a sequence of configurations $\mathcal{S}_0 \longrightarrow_{r_1} \mathcal{S}_1 \longrightarrow_{r_2} \cdots \longrightarrow_{r_n} \mathcal{S}_n$, such that for all $0 \leqslant i \leqslant n-1$, $\mathcal{S}_{i+1}$ is a configuration obtained by applying $r_{i+1} \in \mathcal{R}$ to $\mathcal{S}_i$.*

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 $\mathcal{S}'$. There is the additional non-determinism in the dense time model with respect to the discrete time model used in [34], provided by the choice of ε, representing the non-negative real value of time increase.

3.1. Goal and Critical Configurations, Non-Critical Traces

For protocol verification, not all possible traces are interesting, only those traces that do not contain undesired, critical configurations and reach some goal. The task of security verification is to check whether a system is vulnerable to an attack. For example, a goal configuration can denote that the service has suffered a DoS attack. The purpose of critical configurations will be to avoid traces where the service does not behave as expected, *e.g.*, denying service when there are enough available resources. Such system behaviour is accomplished by only considering non-critical traces defined below.

Definition 3.4 (Critical/Goal Configurations)**.** *A* critical configuration specification $\mathcal{CS}$ *(resp.* goal $\mathcal{GS}$*) is a set of pairs* $\{\langle \mathcal{S}_1, \mathcal{C}_1\rangle, \ldots, \langle \mathcal{S}_n, \mathcal{C}_n\rangle\}$*, with each pair* $\langle \mathcal{S}_j, \mathcal{C}_j\rangle$ *being of the form* $\langle \{F_1@T_1, \ldots, F_p@T_p\}, \mathcal{C}_j\rangle$*, where* $T_1, \ldots, T_p$ *are time variables,* $F_1, \ldots, F_p$ *are facts, and* $\mathcal{C}_j$ *is a set of time constraints involving only variables* $T_1, \ldots, T_p$*.*
A configuration $\mathcal{S}$ *is a* critical configuration *w. r. t.* $\mathcal{CS}$ *(resp. a* goal configuration *w.r.t.* $\mathcal{GS}$*) if for some* $1 \leqslant i \leqslant n$*, there is a grounding substitution,* σ*, such that* $\mathcal{S}_i\sigma \subseteq \mathcal{S}$ *and* $\mathcal{C}_i\sigma$ *evaluates to true.*

For example, the configuration $\{\, Time@3.5,\ F@3.5,\ G@0.2\}$ is critical w.r.t. the critical configuration specification $\{\langle\, \{Time@T,\ F@T_1\}, \{\, T_1 \geqslant T\,\}\rangle\}$. When it is clear from the context, the corresponding critical configuration specification or goal is elided, and the terminology *critical* or *goal configuration* is used.

Notice that nonce renaming is assumed as the particular nonce name should not matter for classifying a configuration as a critical or a goal configuration. Nonce names cannot be specified in advance, since these are freshly generated in a trace, *i.e.* during the execution of the process being modelled.

In discrete time setting [35], a trace is considered critical if it contains a critical configuration. This is not adequate for dense time models where the continuity of time flow is implicitly embedded in the MSR formalism. Namely, given arbitrary $\varepsilon > 0$ and any positive $\varepsilon_1 < \varepsilon$, there exists $\varepsilon_2 > 0$ such that the time $Tick$ for ε has the same effect as the $Tick$ for ε_1 followed by the $Tick$ for ε_2. That is, if $\mathcal{S}_0 \longrightarrow_{Tick_\varepsilon} \mathcal{S}_1$, then $\mathcal{S}_0 \longrightarrow_{Tick_{\varepsilon_1}} \mathcal{S}_2 \longrightarrow_{Tick_{\varepsilon_2}} \mathcal{S}_1$, where $\varepsilon = \varepsilon_1 + \varepsilon_2$ holds. Hence, the notion of non-critical traces in dense time setting needs to be refined. To illustrate this, consider, for example, a trace in a timed MSR with dense time, containing the following configurations and a $Tick$:

$$Time@1.5,\ F@3.5 \longrightarrow_{Tick_3} Time@4.5,\ F@3.5$$

which could potentially be considered as non-critical w.r.t. the critical configuration specification: $\{\langle\{Time@T,\ F@T_1\}, \{\, T_1 = T\,\}\rangle\}$, as it does not contain any critical configurations. However, a trace containing rules $Tick_1$ and $Tick_2$: $Time@1.5,\ F@3.5 \longrightarrow_{Tick_2} Time@3.5,\ F@3.5 \longrightarrow_{Tick_1} Time@4.5,\ F@3.5,$ would be critical w.r.t. the same critical configuration specification since it contains the critical configuration $\{\, Time@3.5,\ F@3.5\,\}$. Clearly this is not appropriate. A configuration that is not critical may turn into a critical configuration purely because the time is ticking. While in discrete time setting [35] one could list all (discrete time) moments in a time window, this is not possible when time is dense. The definition of non-critical traces in dense time setting is necessarily more involved.

Definition 3.5 (Non-Critical Traces)**.** *Let $\mathcal{R}$ be a timed MSR system and $\mathcal{CS}$ a critical configuration specification. A trace $\mathcal{P}$ of $\mathcal{R}$ is* non-critical *if it contains no critical configuration and if no critical configuration is reached along any trace obtained by matching any subtrace of $\mathcal{P}$ on the left below with the one on the right:* $\mathcal{S}_i \longrightarrow_{Tick_\varepsilon} \mathcal{S}_{i+1}$ $\quad$ $\mathcal{S}_i \longrightarrow_{Tick_{\varepsilon_1}} \mathcal{S}' \longrightarrow_{Tick_{\varepsilon_2}} \mathcal{S}_{i+1}$ *where ε_1 and ε_2 are arbitrary non-negative real numbers, such that, $\varepsilon = \varepsilon_1 + \varepsilon_2$ holds.*

That is, one has to consider all possible ways to decompose $Tick$s in a trace. Checking whether a given trace is non-critical potentially requires checking through an infinite number of traces. This could affect the complexity of the verification problems.

3.2. Balanced Systems

Balanced systems, introduced in [36], represent an important class of systems, for which several reachability problems are decidable [23, 24, 37]. A *rule* is *balanced* if the number of facts appearing in its pre-condition and in its post-condition is the same. Balanced systems contain only balanced rules and have the following important property [36]:

Proposition 3.6. *Let $\mathcal{R}$ be a set of balanced rules. Let $\mathcal{S}_0$ be a configuration with exactly m facts. Let $\mathcal{S}_0 \longrightarrow \cdots \longrightarrow \mathcal{S}_n$ be an arbitrary trace of rules from $\mathcal{R}$ starting with $\mathcal{S}_0$. Then for all $0 \leqslant i \leqslant n$, configuration $\mathcal{S}_i$ has exactly m facts.*

As described in [23], any unbalanced rule can be made balanced by using dummy facts, so-called empty facts. For example, the unbalanced rule $Time@T, F_1@T_1 \longrightarrow Time@T, F_1@T_1, F_2@T_2$ can be turned into a balanced rule by adding an empty fact D to its pre-condition, $Time@T, F_1@T_1, D@T_3 \longrightarrow Time@T, F_1@T_1, F_2@T_2$. However, the obtained balanced system is not equivalent to the original, unbalanced system as the set of reachable states and possible traces is reduced. Notice that the above balanced rule can only be applied if a D fact is available in the enabling configuration. That is not the condition for the application of the original, unbalanced rule.

For the complexity results related to balanced systems, it is important to additionally assume an upper-bound on the size of facts. The *size of a timed fact $F@t$* is the total number of symbols in F, written $|F@t|$. For example, $|M(a, \{a, b\}_k)@t| = 5$. This assumption bounds both the depth and the size of terms, *e.g.*, bounds nesting of pairs of submessages. This condition is necessary for decidability of the reachability problem, since the problem is undecidable even for (un-timed) balanced systems when the size of facts is unbounded [22, 32]. Balanced systems with facts of bounded size are suitable *e.g.*, for modeling scenarios with a fixed amount of memory. Namely, a configuration of m facts of size bounded by k, denotes $m \cdot k$ memory slots. Although this memory capacity is fixed for each such system, there is no a priori bound on the size of facts or on the number of facts in the initial configuration of a particular system.

As in [23], empty facts represent available free memory slots. In order to model systems and intruders with bounded memory, we consider empty facts $P(i)$ related to each specific intruder i and empty facts related to the system including agents, servers and the network, D and $N(*)$ facts.

There are important implications when using balanced systems in protocol specification and intruder models. Balanced systems used for protocol verification, as the ones in [23, 24], implicitly bound the number of protocol sessions that can be executed concurrently. However, the total number of protocol sessions in a trace is unbounded. Moreover, the number of facts (and thus symbols) that the balanced intruder can remember is bounded. As shown in [23], however, many of the known attacks on protocols can be carried out by balanced intruders.

Table 1
Summary of the complexity results for the reachability and non-critical reachability problems.

MSR		Reachability Problem	Non-critical Reachability
Balanced	untimed	PSPACE-complete [23, 44]	PSPACE-complete [23, 44]
	discrete time	PSPACE-complete [35]	PSPACE-complete [35]
	dense time	PSPACE-complete [24]	PSPACE-complete [new]
Not necessarily balanced		Undecidable [37]	Undecidable [37]

Remark 3.7. *This paper considers security protocol analysis in the symbolic (Dolev-Yao) model [22, 38–40] and takes into account resource limitations of the attacker. In order to deal with a bounded amount of memory, only facts of bounded size are considered, where the size of facts is measured by the total number of symbols contained. It is an interesting question for further research how to consider the issues we study here in the computational cryptographic model [41–43].*

3.3. Non-Critical Reachability Problem

The non-critical reachability problem for general MSR theories is recalled, which differently from the reachability problem, considers only non-critical traces. The verification problems we study in Sections 6 and 6.2 are special instances of non-critical reachability.

Definition 3.8 (Reachability Problem)**.** *Given a timed MSR system $\mathcal{R}$, a goal specification $\mathcal{GS}$, and an initial configuration $\mathcal{S}_0$, is there a trace, $\mathcal{P}$, that leads from $\mathcal{S}_0$ to a goal configuration?*

Definition 3.9 (Non-Critical Reachability Problem)**.** *Given a MSR system $\mathcal{R}$, a goal specification $\mathcal{GS}$, a critical configuration specification $\mathcal{CS}$ and an initial configuration $\mathcal{S}_0$, is there a non-critical trace, $\mathcal{P}$, that leads from $\mathcal{S}_0$ to a goal configuration?*

Reachability problems for MSR are undecidable in general [37]. However, by imposing some restrictions, such as using only balanced rules and bounding the size of facts, these problems become decidable, even in timed models with fresh values. A summary of known complexity results is shown in Table 1, together with the new result for the non-critical reachability for real-time MSR discussed in Section 7.

Remark 3.10. *It has been shown in [24, 35] that relaxing any of the main conditions on instantaneous rules leads to the undecidability of the reachability problems. For example, undecidability is obtained in systems with time constraints that involve three or more time variables. Furthermore, constants Ds and D_is that appear in time constraints and timestamps of created facts are restricted only to natural numbers for computational complexity issues. These values could be generalized to rationals. In fact, it suffices to assume that all these numerical constants mentioned within the above constraints and timestamps are commensurable.*

4. Resource-Sensitive Protocol Specifications

The MSR model is applied to protocol verification, formalizing verification scenarios, including protocol and intruder theories. A language is introduced for formal specification of how resources are

consumed during protocol execution and the timeouts, which specify when protocols may be terminated and resources recovered. This is obtained by extending protocol theories proposed in [22, 23].

4.1. MSR Signature for Protocol Verification

For the specification of the verification problems, signatures containing the constants, functions, and predicate symbols described below, are used.

Message Expressions. Assume a message signature Σ of constants and function symbols. Constants include nonces, symmetric keys and player names. Messages are constructed as usual, using constants, variables and function symbols including: $*$ denoting a dummy constant; $\mathsf{sk}(\mathsf{p})$ denoting the secret key of the agent p; $\mathsf{pk}(\mathsf{p})$denoting the public key of the agent p; $\{\mathsf{m}\}_\mathsf{k}$ denoting the encryption of m using key k; $\langle \mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_n \rangle$ is the tuple function denoting a list of messages $\mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_n$. Other crypto constructs, such as MAC and hash, can also be added as usual. It is defined that $(\mathsf{pk}(\mathsf{p}))^{-1} = \mathsf{sk}(\mathsf{p})$ and $\mathsf{k}^{-1} = \mathsf{k}$ if k is a symmetric key. Also, the singleton tuple $\langle \mathsf{m} \rangle$ and m are written interchangeably.

Resources. For simplicity, resources are represented with natural numbers. A more abstract definitions as the ones used in [21], *e.g.*, monoids, could also be used. The results in the paper are not affected by this change.

Each service provider (service, in short) is associated with an identifier id and a *minimal service resource* value, $r_{min}(\mathsf{id})$, specifying the bound on resources for which the service is considered exhausted. For example, for VoIP servers this would typically be 0 workers. Once resources reach $r_{min}(\mathsf{id})$, the service can not work and should therefore be considered denied. An *initial service resource* for a service id, written $r_{ini}(\mathsf{id})$ is also assumed.

Predicates. Besides the predicate *Time*, the signature contains the following specific predicate symbols, which, together with the specific rules of protocol in Definition 4.1 and intruder theories in Definition 5.1 introduced below, have the following intended meaning:

- $\mathsf{N}(\mathsf{m})@t$ denotes that message m has been sent at time t;
- $\mathsf{N}(*)@t$ denotes empty message slot available in the network from moment t;
- $\mathsf{S}_i(\mathsf{id}, s_{\mathsf{id}}, r_i, \vec{c}_i)@t$ where $0 \leqslant i \leqslant n$ specify $n+1$ protocol states for a protocol of service id. The number of states and their arguments, $\vec{c}_i$, are protocol-specific. s_{id} is a protocol session identification symbol, which is unique per protocol session, r_i is a natural number specifying the number of resources allocated by the service to maintain the protocol session in the state S_i, and the timestamp t of S_i specifies the moment at which the protocol moved to state S_i;
- $\mathsf{T}_i(\mathsf{id}, s_{\mathsf{id}})@t$ denotes that the protocol state $\mathsf{S}_i(\mathsf{id}, s_{\mathsf{id}}, r_i, \vec{x}_i)$ times out at moment t;
- $\mathsf{R}(\mathsf{id}, r)@t$ denotes that service/intruder id has r resources available at moment t;
- $\mathsf{Rec}(\mathsf{int}, r)@t$ denotes that the intruder int can recover r resources at moment t;
- $\mathsf{Av}(\mathsf{id})@t$ denotes that the service id is available from moment t;
- $\mathsf{Den}(\mathsf{id})@t$ denotes that the service id is unavailable from moment t;
- $\mathsf{M}(\mathsf{int}, \mathsf{m})@t$ denotes that the intruder id knows the message m from moment t;
- $P(\mathsf{id})@t$ denotes that a memory slot is available to bounded memory intruder id from moment t;
- $D(\mathsf{id})@t$ denotes empty memory slots available to service id from the moment t.

For readability and simplification of exposition of the model and the related complexity results, protocol and intruder specifications contain exactly one resource. Generalization to multiple resources is straightforward, by including specific resource predicates for each of the relevant resources, such as service's threads/workers, CPU time, network bandwidth, etc.

4.2. Protocol Resource Theory

The following definition specifies the types of rules used for the specification of protocol sessions carried out by services. They are constructed using the predicates described in Section 4.1.

Definition 4.1 (Protocol Resource Theory)**.** *A* protocol resource theory $\mathcal{T}$ *of service* id *is specified by a set of state predicates* $\mathsf{S}_0, \mathsf{S}_1, \ldots, \mathsf{S}_n$*, its minimal resource* $r_{min}(\mathsf{id})$*, and the rules of the following form, where* $\mathcal{W}_1$ *and* $\mathcal{W}_2$ *denote multisets of facts with at most one* N *fact,* $\mathcal{C}$ *is a possibly empty list of time constraints and* $\vec{X}_i$ *denotes a possibly empty list of variables:*

- ***protocol initialization rule:***

$$Time@T, \mathsf{R}(\mathsf{id}, r_I + R + r_{min}(\mathsf{id}))@T_1, \mathcal{W}_1 \mid T_1 \leqslant T, \mathcal{C} \longrightarrow \exists S.[Time@T, \mathsf{S}_0(\mathsf{id}, S, r_I, \vec{X})@T, \mathsf{T}_0(\mathsf{id}, S)@(T + t_I), \mathsf{R}(\mathsf{id}, R + r_{min}(\mathsf{id}))@T, \mathcal{W}_2]$$

  where S_0 *is the initial state,* T_0 *denotes the initial state timeout,* r_I *is a natural number specifying the initialization resource cost for the protocol,* t_I *is a natural number specifying the timeout of the initial protocol state,* S *is a fresh protocol session identification token, and* $\vec{X}$ *specifies other parameters of the protocol session state;*

- ***protocol execution rules:***

$$\begin{aligned} &Time@T, \mathsf{S}_i(\mathsf{id}, S, r_i, \vec{X}_i)@T_1, \mathsf{T}_i(\mathsf{id}, S)@T_2, \mathsf{R}(\mathsf{id}, r_j - r_i + R + r_{min}(\mathsf{id}))@T_3, \mathcal{W}_1 \\ &\quad \mid T_1 \leqslant T, T_2 > T, T_3 \leqslant T, \mathcal{C} \longrightarrow \\ &\quad Time@T, \mathsf{S}_j(\mathsf{id}, S, r_j, \vec{X}_j)@T, \mathsf{T}_j(\mathsf{id}, S)@(T + t_j), \mathsf{R}(\mathsf{id}, R + r_{min}(\mathsf{id}))@T, \mathcal{W}_2 \qquad (3) \\ &Time@T, \mathsf{S}_n(\mathsf{id}, S, r_n, \vec{X}_n)@T_1, \mathsf{T}_n(\mathsf{id}, S)@T_2, \mathsf{R}(\mathsf{id}, R + r_{min}(\mathsf{id}))@T_3, \mathcal{W}_1 \\ &\quad \mid T_1 \leqslant T, T_2 > T, T_3 \leqslant T, \mathcal{C} \longrightarrow Time@T, \mathsf{R}(\mathsf{id}, r_n + R + r_{min}(\mathsf{id}))@T, \mathcal{W}_2 \end{aligned}$$

  where for $0 \leqslant i \leqslant n$*,* r_i *and* r_j *are natural numbers, specifying the execution resource costs associated to states* S_i *and* S_j*, respectively, and* t_j *is the timeout of the protocol of the service for state* S_j*. It is assumed here that* $r_j - r_i + R > 0$*, that is, the service has enough resources;*

- ***protocol state timeout rule:*** *where* $0 \leqslant i \leqslant n$ *:*

$$Time@T, \mathsf{R}(\mathsf{id}, R)@T_1, \mathsf{T}_i(\mathsf{id}, S)@T, \mathsf{S}_i(\mathsf{id}, S, r_i, \vec{c}_i)@T_2 \mid T_1 \leqslant T, T_2 \leqslant T, \mathcal{C} \longrightarrow Time@T, \mathsf{R}(\mathsf{id}, r_i + R)@T$$

The protocol initialization rule specifies that the creation of a new protocol session requires r_I resources. Notice the use of the existential quantifier that creates a fresh protocol session identifier. Protocol execution rules change the state of the protocol session from S_i to S_j, updating the resources allocated and the timeout. At the same time, validity of the current protocol state, *i.e.*, protocol state timeouts are checked through constraints involving timestamps of facts with protocol state timeout predicates T_i and the global time T, such as $T_2 > T$ in Eq. (3). The second protocol execution rule finalizes a protocol session. A $\mathsf{N}(M)$ fact, with M possibly containing variables, in $\mathcal{W}_1$ and/or $\mathcal{W}_2$ on each side of a protocol rule denotes receiving and/or sending of a message. The protocol state timeout rule specifies that a protocol session is forgotten when a timeout is reached and that the allocated resources are recovered.

Protocol Critical Configuration Specification. As rules can be applied in a non-deterministic fashion, undesirable traces where a service misbehaves are allowed. For example, it is possible to construct traces where the protocol state timeout rule is never applied, and thus the service never releases resources allocated to protocol sessions that should have been ended by a timeout. Such traces, where the service does not behave as expected, are classified as critical. This is formally achieved by protocol critical configuration specifications defined below:

Definition 4.2 (*Protocol $\mathcal{CS}$*)**.** *The* protocol critical configuration specification *(Protocol $\mathcal{CS}$) for a given protocol theory $\mathcal{T}$ of a service* id*, with protocol state timeout predicates* T_i*, with* $0 \leqslant i \leqslant n$*, is composed of the critical configuration specifications:*

- ***Timeout*** $\mathcal{CS}$*:* $\quad \langle \{Time@T, \mathsf{T}_i(\mathsf{id}, S)@T_1\}, \{T_1 < T\}\rangle, \quad$ *for all* $0 \leqslant i \leqslant n$.

Timeout $\mathcal{CS}$ specify that configurations with protocol sessions, for which some protocol state timeout has passed are critical. Notice that using variable S, all instances of protocol sessions are considered, while n copies of *Timeout $\mathcal{CS}$* cover all protocol states.

Intuitively, a service may run several protocols with a number of protocol sessions in parallel. Each instance of a protocol affects service resources, since, in order to maintain each protocol session, service id may need to allocate resources, such as memory, CPU time, workers, etc, for some time. Once the resources reach the lower bound $r_{min}(\mathsf{id})$, the service is not available until some resources are recovered.

Definition 4.3 (Service)**.** *A* service $\mathcal{A}$ *has the following components:*

- *A unique identification symbol* id*;*
- *A minimal service resource value* $r_{min}(\mathsf{id})$*;*
- *An initial service resource value* $r_{ini}(\mathsf{id})$*, such that* $r_{ini}(\mathsf{id}) > r_{min}(\mathsf{id})$*;*
- *A set of protocol theories* $\mathcal{T}_1, \ldots, \mathcal{T}_k$ *of service* id *. It is assumed that the set of protocol state predicates of different protocol theories are disjoint;*
- ***service availability rules:***

$$Time@T,\ \mathsf{R}(\mathsf{id}, r_{min}(\mathsf{id}))@T, \mathsf{Av}(\mathsf{id})@T_1 \mid T_1 \leqslant T \longrightarrow Time@T,\ \mathsf{R}(\mathsf{id}, r_{min}(\mathsf{id}))@T, \mathsf{Den}(\mathsf{id})@T$$

$$Time@T,\ \mathsf{R}(\mathsf{id}, r_{min}(\mathsf{id}) + R + 1)@T, \mathsf{Den}(\mathsf{id})@T_1 \mid T_1 \leqslant T \longrightarrow Time@T,\ \mathsf{R}(\mathsf{id}, r_{min}(\mathsf{id}) + R + 1)@T, \mathsf{Av}(\mathsf{id})@T$$

- Service Critical Configuration Specification *(Service $\mathcal{CS}$) defined as the union of Protocol $\mathcal{CS}$ of* $\mathcal{T}_1, \ldots, \mathcal{T}_k$ *and the following two types of critical configuration specifications:*
  - ***Denied*** $\mathcal{CS}$*:* $\quad \langle \{Time@T, \mathsf{R}(\mathsf{id}, r_{min}(\mathsf{id}))@T_1, \mathsf{Av}(\mathsf{id})@T_2\}, \{T > T_1, T > T_2\}\rangle$*;*
  - ***Available*** $\mathcal{CS}$*:* $\quad \langle \{Time@T, \mathsf{R}(\mathsf{id}, r_{min}(\mathsf{id}) + R + 1)@T_1, \mathsf{Den}(\mathsf{id})@T_2\}, \{T > T_1, T > T_2\}\rangle$.

Service availability rules specify that a service is denied when resources reach a minimum, and available if the resources are greater than the minimum. *Denied $\mathcal{CS}$* specifies that configurations are critical if a service is considered available at a time its resources have been exhausted. Similarly, as per *Available $\mathcal{CS}$*, a service should not be considered denied at anytime when sufficient resources are available.

Notice that critical configurations are necessary for the adequate specification of the expected behavior of services. Indeed, due to critical configuration specifications, such as *Timeout $\mathcal{CS}$*, protocol state timeout rules are necessarily applied, ending sessions and releasing resources. Otherwise, expired protocol states

could appear in a trace configuration requiring resources which in practice are released, thus leading to false denial of service attack traces. Similarly, *Denied $\mathcal{CS}$* and *Available $\mathcal{CS}$* specify that service availability rules are applied whenever the service's resources are depleted and therefore denied, or have enough resources and are available. Consider, for example, that at some point in a trace service resources are exhausted, *i.e.*, a configuration $\mathcal{S}$ containing the facts $Time@t, \mathsf{R}(\mathsf{id}, r_{min}(\mathsf{id}))@t, \mathsf{Av}(\mathsf{id})@t'$ is reached by a protocol execution rule. Then, only the service availability rule or the $Tick$ rule is applicable in $\mathcal{S}$. The application of service availability rule would replace the fact $\mathsf{Av}(\mathsf{id})@t'$ with $\mathsf{Den}(\mathsf{id})@t$, denoting that the service is denied. Otherwise, the application of $Tick$ rule would result in a configuration containing facts $Time@t'', \mathsf{R}(\mathsf{id}, r_{min}(\mathsf{id}))@t, \mathsf{Av}(\mathsf{id})@t'$ with $t'' > t, t'' > t'$, which is critical w.r.t. *Denied $\mathcal{CS}$*. Similarly, *Available $\mathcal{CS}$* would force the application of the other service availability rule signaling that the service is available. Hence, in a non-critical trace availability of service is properly modelled.

Definition 4.4 (Balanced Protocol Resource Theory, Balanced Service)**.** *A protocol resource theory is* balanced *if all of its rules are balanced. A service is* balanced*, if all its protocol theories are balanced. A balanced service* id *also contains a natural number* d_{id} *specifying the number of* $D(\mathsf{id})$ *facts available.*

Balanced services formalize services that can only maintain a bounded number of parallel protocol sessions. Balanced rules of protocol theories are obtained using dummy facts $D(s)$ (see Section 3) as needed. Since for each protocol session a D fact is consumed and there is a bounded number, d_s, of D facts available, as in [23], for balanced theories, verification relates to traces with a bounded number of concurrent protocol sessions. However, the total number of protocol sessions in a trace is unbounded.

Notice also that protocol sessions and state transitions are usually triggered by a message receipt, typically denoting requests and data updates. Balanced protocol initialization and execution rules denote receipt of at most one message (a $\mathsf{N}(*)$ or a $\mathsf{N}(M)$ fact in the rule precondition) which is followed either by a single message reply or by freeing a message slot in the network (by creating a $\mathsf{N}(*)$ fact). In this way, a constant, bounded network capacity (bandwidth) is modelled.

Remark 4.5. *Protocol and intruder verification models can be further enhanced by specifying assumptions about the network being used, such as the availability of some transmission channel to a particular agent for sending or receiving messages, network distances etc. Faithful timing of message transmission for a given network topology between agents can be obtained by enhancing the transmission rules with constraints that involve relevant distances,* e.g.*,:*

$$Time@T, Cap_S(A,C)@T_1, Cap_R(B,C,1)@T_2, \mathsf{N}_S(A,C,X)@T_3 \mid T \geqslant T_3 + D(A,B,C)$$
$$\longrightarrow Time@T, Cap_S(A,C)@T_1, Cap_R(B,C,1)@T_2, \mathsf{N}_R(B,C,X)@T$$

where $D(x,y,z)$ is a natural number denoting network delay time in communication from agent x to agent y when using transmission media z, $\mathsf{N}_S(x,y,m)@t$ and $\mathsf{N}_R(x,y,m)@t$ respectively denote that the message m was sent at moment t, or can be received from moment t, by agent x on transmission medium y, and $Cap_S(x,y)$ and $Cap_R(x,y)$ denote the capability of agent x of sending and receiving messages on transmission medium y, respectively. This way, transmission of messages is encoded as the transformation of N_S facts to N_R facts. For readability, we do not include above details in the theories, and use only N *facts. Also, in DoS verification, network delays are normally in a lower order of magnitude than service timeouts, hence they may not be that relevant for DoS attacks. This is in contrast with cyber-physical security protocols where transmission delays are very important [24, 45].*

Remark 4.6. *Execution and verification of security protocols may be affected by processing time. Specifying* duration *of actions is particularly important in verification of attacks that involve the variance of execution time, such as passport traceability attacks in [46].*

Duration can be expressed in our model through timestamps of created facts, where a fact $F@(T+d)$ created at moment T is available only from the moment $T+d$, after d time units. This can be further elaborated in protocol theory rules as in [30] using a specific function that takes into account various factors, such as length of plaintext affecting the encryption time.

Again, for readability, we omit such duration functions in the specification of protocol theories, but this feature is demonstrated in Section 5 in the specification of time- and resource-sensitive intruder theories.

4.3. Examples of Protocol Theories

This Section illustrates the expressiveness of the model and its suitability for the verification of protocols sensitive to time and resource consumption. Four examples of resource-sensitive protocol theories are described: GET-HTTP, SDN rule insertion, TLS renegotiation and SIP forking. While the GET-HTTP example is given in some detail along with its protocol theory specification, the other three examples are only briefly described. They can be modeled similarly to the GET-HTTP protocol, but using different types of resources and states.

GET-HTTP: (An abstract version of) the HTTP GET method is specified, which is subject to the Slowloris attack [4], exhausting web-server's workers, as described in Section 2. The initialization and execution rules are given in Figure 1. The protocol state timeout rule is elided.

The protocol has three states, $\mathsf{S}_0, \mathsf{S}_1$ and S_2, which, respectively, correspond to the state when the HTTP protocol session is initialized (S_0), *i.e.*, by performing the SYN-ACK which is omitted for brevity, the state where a incomplete GET request is received (S_1), and the state where the GET request is completed and the protocol session may end (S_2).

Each protocol state requires one resource, corresponding to one worker as is the case with connection based web-servers, such as Apache. The timeout of each protocol state is 40 time units, that is, if no further interaction is performed within 40 time units, then the protocol session is terminated.

Rule INIT specifies the protocol initialization, where one worker is allocated and a fresh protocol session S_{id} is created. Rules GET, INC, COM1 and COM2 specify transitions between the states. Notice

$$\text{INIT:}\quad Time@T,\ \mathsf{R}(Id, 1+R+r_{min}(Id))@T_1,\ \mathsf{N}(INIT)@T_2 \mid T_1 \leqslant T,\ T_2 \leqslant T \longrightarrow \exists S_{id}.[Time@T, \mathsf{S}_0(Id, S_{id}, 1)@T,\ \mathsf{T}_0(Id, S_{id})@(T+40), \mathsf{R}(Id, R+r_{min}(Id))@T]$$

$$\text{GET:}\quad Time@T,\ \mathsf{S}_0(Id, S_{id}, 1)@T_1,\ \mathsf{T}_0(Id, S_{id})@T_2,\ \mathsf{R}(Id, R)@T_3, \mathsf{N}(GET)@T_4 \mid T_1 \leqslant T,\ T_2 > T,\ T_3 \leqslant T,\ T_4 \leqslant T \longrightarrow Time@T,\ \mathsf{S}_1(Id, S_{id}, 1)@T,\ \mathsf{T}_1(Id, S_{id})@(T+40),\ \mathsf{R}(Id, R)@T$$

$$\text{INC:}\quad Time@T,\ \mathsf{S}_1(Id, S_{id}, 1)@T_1,\ \mathsf{T}_1(Id, S_{id})@T_2,\ \mathsf{R}(Id, R)@T_3, \mathsf{N}(Inc)@T_4 \mid T_1 \leqslant T,\ T_2 > T,\ T_3 \leqslant T,\ T_4 \leqslant T \longrightarrow Time@T,\ \mathsf{S}_1(Id, S_{id}, 1)@T,\ \mathsf{T}_1(Id, S_{id})@(T+40),\ \mathsf{R}(Id, R)@T$$

$$\text{COM1:}\quad Time@T,\ \mathsf{S}_0(Id, S_{id}, 1)@T_1,\ \mathsf{T}_0(Id, S_{id})@T_1,\ \mathsf{R}(Id, R)@T_2,\ \mathsf{N}(Com)@T_3 \mid T_1 \leqslant T,\ T_2 > T,\ T_3 \leqslant T,\ T_4 \leqslant T \longrightarrow Time@T,\ \mathsf{S}_2(Id, S_{id}, 1)@T,\ T_2(Id, S_{id})@(T+40),\ \mathsf{R}(Id, R)@T$$

$$\text{COM2:}\quad Time@T,\ \mathsf{S}_1(Id, S_{id}, 1)@T_1,\ \mathsf{T}_1(Id, S_{id})@T_1,\ \mathsf{R}(Id, R)@T_2,\ \mathsf{N}(Com)@T_3 \mid T_1 \leqslant T,\ T_2 > T\ T_3 \leqslant T,\ T_4 \leqslant T \longrightarrow Time@T,\ \mathsf{S}_2(Id, S_{id}, 1)@T,\ \mathsf{T}_2(Id, S_{id})@(T+40),\ \mathsf{R}(Id, R)@T$$

Fig. 1. Protocol Resource Theory for HTTP GET Protocol Method.

that in all states the allocated worker is kept allocated. Here *Inc* is used to represent a message which has not completed the GET header and *Com* is used for a message that has completed the header. In practice, a message header is complete if it ends with \r \n \r \n and incomplete otherwise. Thus, it is possible to move from state S_0 to the final state S_2 (rule COM1) by sending a complete message or from S_1 to S_2 by first sending an incomplete message (rule INC) and then a complete message (rule COM2).

SDN Rule Addition: Software Defined Networks (SDN) use protocols, such as OpenFlow [47], to install rules in SDN switches. These rules specify the behavior of flows, *e.g.*, allowing or blocking flow of packets. The number of available rules that can be stored in an SDN Switch is limited with typically a total number of rules in the range of at most 5000-8000 rules. This can be exploited by attackers to carry out DoS attacks and this can be formalized by our model [8, 9]. We use the number of available rule slots in a switch as the measure of resource.

Whenever an SDN switch receives a packet for which there is no applicable SDN forwarding rule, it installs a new rule after receiving rule installation approval from the centralized SDN controller. This is modeled by a protocol initialization rule. This rule reduces the number of available resources by one. If the number of resources reaches zero, the service is denied. This is modeled by the service availability rules.

Moreover, rules are associated with timeouts that work as follows: whenever a packet is received that triggers a rule in the switch, the timeout is reset. However, if no packet activates a rule after a timeout has elapsed, then the rule is deleted making the slot consumed by the rule available again. In our model, the reset of a timeout is specified by a protocol execution rule.

TLS Renegotiation: Transport Layer Security (TLS) renegotiation protocol can also be modeled using the number of handshakes being processed as the resource. For example, according to [17] a server can handle between 150-300 handshakes per second. This is proportional to the CPU power of the server. Whenever a new renegotiation is requested, the server consumes processing power. TLS protocols also illustrate the use of messages involving cryptographic operators for DoS attacks. The use of asymmetric keys in handshakes and the creation of fresh keys causes overheads to the server.

SIP invite forking: A protocol theory of SIP specifies the mechanism of forwarding and forking invite messages, which is known to be vulnerable to amplification attacks [14, 15]. As amplification targets the network's bandwidth, the service is identified as the network and the resource as the network's bandwidth (measured in terms of invite messages sent). When an invite message is first introduced, a session is created. A state in the protocol maintains the total number of forwarded or forked invite messages for a session, along with the session's timeout. While both forwarding and forking messages reset the timeout of their corresponding sessions, only forking increases the amount of resources allocated for the session.

4.4. Modeling Time-Based Countermeasures

This Section illustrates how the model can be extended to take into account countermeasures (CMs) for DDoS attacks based on timeouts, such as, ReqTimeOut [25] for mitigating the Slowloris attack. The basic idea of timeout based CM is to trigger a timeout whenever some condition on the traffic is satisfied, thus forcing that a connection is closed and the service's resources are made available.

For example, for SYN flooding attacks, timeout based CMs monitor whether a SYN-ACK handshake is completed within some specified timeout. If a timeout is triggered, then the connection is closed and the resources allocated to this connection can be used by another connection. Similarly, ReqTimeOut is a time based CM which monitors whether a packet header or packet body is completed within some

specified time. If a timeout is triggered, then the connection is closed and the allocated worker can be used to serve another connection. It is, therefore, used to mitigate attacks such as Slowloris which take too long to complete the header.

To formalize timeout based CM for service id, the fact $\mathsf{TimeCM}(\mathsf{id}, c)@t$ is used, denoting that the CM triggers a timeout at time t which closes the connection c of service id. The three rules below specify, respectively, that timeout countermeasures are initialized whenever a protocol session is initialized, whenever a protocol session enters some specific protocol state, and ends the protocol session whenever the countermeasure timeout is reached, thus releasing allocated service resources. Notice that if countermeasure timeouts are initialized on a protocol state S_j, then the protocol initialization or the protocol execution rule that moves to state S_j (given in Definition 4.1) shall be replaced by the appropriate rule with timeout countermeasure with the chosen timeout d^j_{CM}.

- **protocol initialization rule with timeout countermeasure:**

$$Time@T, \mathsf{R}(\mathsf{id}, r_I + R + r_{min}(\mathsf{id}))@T_1, \mathcal{W}_1 \mid T_1 \leqslant T,\ \mathcal{C} \longrightarrow \exists S.[Time@T, \mathsf{S}_0(\mathsf{id}, S, r_I, \vec{X})@T, \mathsf{T}_0(\mathsf{id}, S)@(T + t_I), \mathsf{R}(\mathsf{id}, R + r_{min}(\mathsf{id}))@T, \mathsf{TimeCM}(\mathsf{id}, S)@(T + d^0_{CM}), \mathcal{W}_2]$$

- **protocol execution rules with timeout countermeasure:**

$$Time@T, \mathsf{S}_i(\mathsf{id}, S, r_i, \vec{X}_i)@T_1, \mathsf{T}_i(\mathsf{id}, S)@T_2, \mathsf{R}(\mathsf{id}, r_j - r_i + R + r_{min}(\mathsf{id}))@T_3, \mathcal{W}_1 \mid T_1 \leqslant T, T_2 > T, T_3 \leqslant T,\ \mathcal{C} \longrightarrow Time@T, \mathsf{S}_j(\mathsf{id}, S, r_j, \vec{X}_j)@T, \mathsf{T}_j(\mathsf{id}, S)@(T + t_j), \mathsf{R}(\mathsf{id}, R + r_{min}(\mathsf{id}))@T, \mathsf{TimeCM}(\mathsf{id}, S)@(T + d^j_{CM}), \mathcal{W}_2 \tag{4}$$

$$Time@T, \mathsf{S}_i(\mathsf{id}, S, r_i, \vec{X}_i)@T_1, \mathsf{T}_i(\mathsf{id}, S)@T_2, \mathsf{R}(\mathsf{id}, r_j - r_i + R + r_{min}(\mathsf{id}))@T_3, \mathcal{W}_1 \mathsf{TimeCM}(\mathsf{id}, S)@T_4 \mid T_1 \leqslant T, T_2 > T, T_3 \leqslant T, T_4 > T,\ \mathcal{C} \longrightarrow Time@T, \mathsf{S}_j(\mathsf{id}, S, r_j, \vec{X}_j)@T, \mathsf{T}_j(\mathsf{id}, S)@(T + t_j), \mathsf{R}(\mathsf{id}, R + r_{min}(\mathsf{id}))@T, \mathsf{TimeCM}(\mathsf{id}, S)@(T + d^j_{CM}), \mathcal{W}_2 \tag{5}$$

$$Time@T, \mathsf{S}_n(\mathsf{id}, S, r_n, \vec{X}_n)@T_1, \mathsf{T}_n(\mathsf{id}, S)@T_2, \mathsf{R}(\mathsf{id}, R + r_{min}(\mathsf{id}))@T_3, \mathsf{TimeCM}(\mathsf{id}, S)@T_4, \mathcal{W}_1 \mid T_1 \leqslant T, T_2 > T, T_3 \leqslant T, T_4 > T,\ \mathcal{C} \longrightarrow Time@T, \mathsf{R}(\mathsf{id}, r_n + R + r_{min}(\mathsf{id}))@T, \mathcal{W}_2 \tag{6}$$

- **timeout countermeasure rule:**

$$Time@T, \mathsf{R}(\mathsf{id}, R)@T_1, \mathsf{S}_i(\mathsf{id}, S, r_i, \vec{x}_i)@T_2, T_i(\mathsf{id}, S)@T_3, \mathsf{TimeCM}(\mathsf{id}, S)@T \mid T_1 \leqslant T, T_2 \leqslant T, T_3 > T,\ \mathcal{C} \longrightarrow Time@T, \mathsf{R}(Id, r_i + R)@T$$

Protocol initialization rules and protocol execution rules of type Eq.(4) set CM timers, while rules of type Eq.(5) may update TimeCM for the following protocol phase. By rules of type Eq.(6) protocol sessions executed within allowed timeouts are finalized and TimeCM facts are removed. Timeout countermeasure rules are used to end sessions that expire due to CM timers. Notice that the CM timeout may relate to the entire session or to some of its particular phase, while timeouts in the protocol theories relate to a single protocol state.

Additionally, the following critical configuration for CM is specified:

- **Countermeasure Not Executed** $\mathcal{CS}$: $\langle\{Time@T, \mathsf{TimeCM}(\mathsf{id}, S)@T_1\}, \{T_1 < T\}\rangle$.

This is similar to the *Timeout* $\mathcal{CS}$ in Definition 4.2, as only traces where CMs trigger timeouts are considered. It relates to all protocol sessions, as variable S is used.

ReqTimeOut:. Using the above machinery, the ReqTimeOut [25] is specified, a time-based CM available in the Apache Web-Server for mitigating Slow-DoS attacks. Two natural numbers are specified: **Header Timeout** (h_{TO}) denoting the maximum elapsed time for the connection to send the complete header; **Body Timeout** (b_{TO}) denoting the maximum elapsed time for the connection to send the complete packet body. Thus, a protocol with ReqTimeOut has three states: S_{HI}, denoting a protocol state where the connection did not send the complete header; S_{HC-BI}, denoting a protocol state where the connection sent the complete header, but not the complete packet body; S_{BC}, denoting a protocol state where the connection sent a complete header and body.

The following protocol initialization rule with timeout countermeasure sets the TimeCM:

$$Time@T,\ \mathsf{R}(\mathsf{id}, 1+R+r_{min}(\mathsf{id}))@T_1,\ \mathsf{N}(INIT)@T_2 \mid T_1 \leqslant T,\ T_2 \leqslant T \longrightarrow \exists S.[Time@T,$$
$$\mathsf{S}_{HI}(\mathsf{id}, S, 1)@T, \mathsf{T}_{HI}(\mathsf{id}, S)@(T+40), \mathsf{R}(\mathsf{id}, R+r_{min}(\mathsf{id}))@T, \mathsf{TimeCM}(\mathsf{id}, S)@(T+h_{TO})]$$

The protocol execution rule with timeout countermeasure specifies that when the header is completed, the timeout is set for receiving the packet body:

$$Time@T,\ \mathsf{S}_{HI}(\mathsf{id}, S, 1)@T_1, \mathsf{T}_{HI}(\mathsf{id}, S, 1)@T_2,\ \mathsf{R}(\mathsf{id}, R)@T_3,\ \mathsf{N}(M)@T_4, \mathsf{TimeCM}(\mathsf{id}, S)@T_5$$
$$\mid T_1 \leqslant T, T_2 > T, T_3 \leqslant T, T_4 \leqslant T, T_5 > T \longrightarrow Time@T,$$
$$\mathsf{S}_{HC-BI}(\mathsf{id}, S, 1)@T, \mathsf{T}_{HC-BI}(\mathsf{id}, S)@(T+40), \mathsf{R}(\mathsf{id}, R)@T, \mathsf{TimeCM}(\mathsf{id}, S)@(T+b_{TO})$$

A similar rule specifies when the body of the packet is completed. This rule is elided.

It should be possible to model more refined time-based CMs by adding suitable facts and rules. For example, to model more advanced configurations of the ReqTimeOut, one can use a predicate that remembers the number of bytes received, which is activated depending on the traffic rate. The formalization of such extensions is left for future investigation of specific protocols and countermeasures.

5. Resource-Bounded Intruder Model

This Section introduces a novel parametric intruder model that is based on the powerful DY intruder [3], but has bounded resources. In contrast to the DY intruder, the resource-bounded intruder can only consume a bounded number of his resources in any given time window.

5.1. Definition of the Resource-Bounded Intruder

Provided he has enough resources, the resource-bounded intruder can compose, decompose, encrypt and decrypt messages for which he knows the appropriate key, and generate fresh values. The rules corresponding to these actions, depicted in Figure 2, are based on the DY intruder rules [22], but refined with the notion of time as in Timed DY intruders [45] and with resource consumption.

Each rule has an associated cost, specified by SPEC function, returning a triple of natural numbers $\langle \delta_L, \delta_R, r_R \rangle$, where δ_L is the time for carrying out the action, r_R denotes resources consumed by the action, which can only be re-used after δ_R time units. It is assumed that all SPEC functions are computable in polynomial time.

As with protocol theories introduced in Section 4, intruder resources may represent *e.g.*, traffic generation or CPU consumption. Again, for simplicity, resources are to be represented by natural numbers and only one resource is used.

In order to model the presence of multiple intruders, an identification int is associated to each of the intruders. This int is used to model resources, knowledge and memory of a particular intruder through facts $R(\mathsf{int}, r)$, $M(\mathsf{int}, \mathsf{m})$ and $P(\mathsf{int})$.

I/O Rules:

REC: $Time@T, \mathsf{N}(X)@T_1, \mathsf{R}(I, Z + r_R)@T_2 \mid T_1 \leqslant T, T_2 \leqslant T \longrightarrow$
$Time@T, \mathsf{M}(I, X)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{REC}(X, I) = \langle \delta_L, \delta_R, r_R \rangle$

SND: $Time@T, \mathsf{M}(I, X)@T_1, \mathsf{R}(I, Z + r_R)@T_2 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T \longrightarrow$
$Time@T, \mathsf{N}(X)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{SND}(X, I) = \langle \delta_L, \delta_R, r_R \rangle$

Message Composition and Decomposition Rules:

CMP: $Time@T, \mathsf{M}(I, X)@T_1, \mathsf{M}(I, Y)@T_2, \mathsf{R}(I, Z + r_R)@T_3 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T \longrightarrow$
$Time@T, \mathsf{M}(I, \langle X, Y \rangle)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{COMP}(X, Y, I) = \langle \delta_L, \delta_R, r_R \rangle$

DCM: $Time@T, \mathsf{M}(I, \langle X, Y \rangle)@T_1, \mathsf{R}(I, Z + r_R)@T_2 \mid T_1 \leqslant T, T_2 \leqslant T \longrightarrow$
$Time@T, \mathsf{M}(I, X)@(T + \delta_L), \mathsf{M}(I, Y)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{DCMP}(\langle X, Y \rangle, I) = \langle \delta_L, \delta_R, r_R \rangle$

USE: $Time@T, \mathsf{M}(I, X)@T_1, \mathsf{R}(I, Z + r_R)@T_2 \mid T_1 \leqslant T, T_2 \leqslant T \longrightarrow$
$Time@T, \mathsf{M}(I, X)@T_1, \mathsf{M}(I, X)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R),$
where $\text{SPEC}_{USE}(X, I) = \langle \delta_L, \delta_R, r_R \rangle$

ENC: $Time@T, \mathsf{M}(I, K)@T_1, \mathsf{M}(I, X)@T_2, \mathsf{R}(I, Z + r_R)@T_3 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T \longrightarrow$
$Time@T, \mathsf{M}(I, \{X\}_K)@(T + \delta_L), \mathsf{M}(I, K)@T_1, \mathsf{M}(I, X)@T_2, \mathsf{R}(I, Z)@T, \mathsf{Rec@}(I, r_R)(T + \delta_L)$
where $\text{SPEC}_{ENC}(K, X, I) = \langle \delta_L, \delta_R, r_R \rangle$

DEC: $Time@T, \mathsf{M}(I, K^{-1})@T_1, \mathsf{M}(I, \{X\}_K)@T_2, \mathsf{R}(I, Z + r_R)@T_3 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T \longrightarrow$
$Time@T, \mathsf{M}(I, X)@(T + \delta_L), \mathsf{M}(I, K^{-1})@T_1, \mathsf{M}(I, \{X\}_K)@T_2, \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{DEC}(K^{-1}, \{X\}_K, I) = \langle \delta_L, \delta_R, r_R \rangle$

GEN: $Time@T, \mathsf{R}(I, Z + r_R)@T_1 \mid T_1 \leqslant T \longrightarrow$
$\exists N. Time@T, \mathsf{M}(I, N)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{GEN}(I) = \langle \delta_L, \delta_R, r_R \rangle$

Resource Maintenance Rule:

RES: $Time@T, \mathsf{R}(I, Z)@T_1, \mathsf{Rec}(I, r_R)@T_2 \mid T_1 \leqslant T, T_2 \leqslant T \longrightarrow Time@T, \mathsf{R}(I, Z + r_R)@T$

Fig. 2. Resource-Bounded Intruder Theory

Bounded resource intruder rules, given in Figure 2, denote the following intruder actions:

- REC rule specifies the intruder action of receiving a message from the network. The rule's cost is given by $\text{SPEC}_{REC}(X, I) = \langle \delta_L, \delta_R, r_R \rangle$: the intruder I consumes r_R resources for receiving the message X, taking δ_L time units to learn X;
- SND rule specifies sending of a message to the network with the cost $\text{SPEC}_{SND}(X, I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for sending the message X, taking δ_L time units;
- CMP rule specifies composing two messages known to the intruder, costing $\text{SPEC}_{COMP}(X, Y, I) = \langle \delta_L, \delta_R, r_R \rangle$: the intruder I consumes r_R resources for composing messages X and Y into message $\langle X, Y \rangle$ in δ_L time units;
- DCM rule specifies decomposing, costing $\text{SPEC}_{DCMP}(X, Y, I) = \langle \delta_L, \delta_R, r_R \rangle$: the intruder I consumes r_R resources for decomposing $\langle X, Y \rangle$ into X and Y in δ_L time units;
- USE rule specifies copying of a known message, costing $\text{SPEC}_{USE}(X, I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for copying X in δ_L time units;

- ENC rule specifies encryption, having the cost $\text{SPEC}_{ENC}(K, M, I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for encrypting M using the key K, taking δ_L time units;
- DEC rule specifies decryption, costing $\text{SPEC}_{DEC}(K^{-1}, M_K, I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for decrypting M_K using the key K^{-1} and learning the message M, taking δ_L time units;
- GEN rule specifies creation of a fresh value, *e.g.*, a nonce or a fresh key. Its cost is $\text{SPEC}_{GEN}(I) = \langle \delta_L, \delta_R, r_R \rangle$: the intruder I consumes r_R resources, taking δ_L time units;
- RES rule denotes recovery of available resources.

Definition 5.1 (Resource-Bounded Intruder)**.** *A resource-bounded intruder, $\mathcal{I}$, consists of his unique identification symbol* int*, his maximal resource $r_{max}(\mathsf{int})$, a finite set $\mathcal{M} = \{\mathsf{M}(\mathsf{int}, \mathsf{m}_1)@0, \ldots, \mathsf{M}(\mathsf{int}, \mathsf{m}_n)@0\}$ of* M *facts specifying intruder's initial knowledge base and definitions of* SPEC_R *functions, for each rule R given in Figure 2.*

Notice that if for all rules R, $\text{SPEC}_R = \langle 0, 0, 0 \rangle$, then the (unbalanced) intruder model in Figure 2 is equivalent to the DY intruder [22]. All actions can be performed at no cost (in time or resources) to the intruder. Indeed, intruder can generate and intercept any number of messages. Finally, one can also imagine a lattice of intruder models with order defined by the number of resources where, clearly, the DY intruder is the most powerful. Such investigations are left for future work.

5.2. Types of Resource-Bounded Intruders

Different bounded intruders can be specified by using different definitions of SPEC_R functions. This feature is illustrated with some examples. Assume that the intruder has at most $r_{max}(I)$ resources.

Bounded Traffic Intruder Model. Bounded Traffic Intruder Model represents an intruder that can send only a number of messages at a given rate, *e.g.*, messages per second. This is achieved by specifying SPEC_{SND} and SPEC_{REC} accordingly, and setting $\text{SPEC}_R = \langle 0, 0, 0 \rangle$ for the remaining rules.

For example, setting $\text{SPEC}_{SND}(X, I) = \text{SPEC}_{REC}(X, I) = \langle 1, 1, 1 \rangle$ for all X means that the intruder can only generate/intercept traffic at a maximum rate of $r_{max}(I)$ messages per time unit. This is because sending or receiving a message consumes 1 of his resources for 1 time unit. Since he has only $r_{max}(I)$ resources, he can send at most $r_{max}(I)$ messages in a time unit.

One could refine this even further by specifying $\text{SPEC}_{SND}(X, I)$ to depend on X, so that, *e.g.*, the number of resources is proportional to the number of symbols of X, so that the intruder is only capable of sending $r_{max}(I)$ symbols per time unit.

Bounded Processing Intruder Model. One can also specify an intruder that can only carry out a bounded number of actions in a given time window according to his processing power. The maximum resources may be expressed in terms of percentage of available CPU, *i.e.*, $r_{max}(I) = 100$, or even in the number of CPU cycles for more precise models. Each action would then consume CPU resources for some given time. For example, the cost of encrypting and decrypting, SPEC_{ENC} and SPEC_{DEC}, will impact the CPU usage depending on the key and message being encrypted or decrypted.

Bounded Memory Intruder Models. Bounded memory intruder models represent intruders with fixed total memory. Such models are specified using balanced intruder models as described in [23]. An upper-bound on the size of facts is also assumed, so that all configurations denoting the intruder knowledge have a fixed number of facts, m, each of size bounded by k, modelling, hence, intruder memory of $m \cdot k$

slots. All the rules of the intruder model are transformed into balanced rules as described in Section 3, using empty facts $P(I)@T$.

For example, the balanced version of the *REC* rule is:

$$Time@T, \mathsf{N}(X)@T_1, \mathsf{R}(I, Z + r_R)@T_2, P(I)@T_3, P(I)@T_4 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T, T_4 \leqslant T \longrightarrow Time@T, \mathsf{N}(*)@T, \mathsf{R}(I, Z)@T, \mathsf{M}(I, X)@(T + \delta_L), \mathsf{Rec}(I, r_R)@(T + \delta_R)$$

Notice that empty facts are consumed when a message is received. Since the number of empty facts in a configuration is bounded, the number of messages that can be learned from the network is bounded. Hence, memory bounded intruders should also be able to manage their memory. This is specified by the following memory maintenance rule that enables the intruder to delete information form his memory:

DELM: $Time@T, \mathsf{M}(I, X)@T_1 \longrightarrow Time@T, P(I)@T$.

All the rules of a balanced bounded resource intruder theory are given in Figure 3.

I/O Rules:

REC: $Time@T, \mathsf{N}(X)@T_1, \mathsf{R}(I, Z + r_R)@T_2, P(I)@T_3, P(I)@T_4 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T, T_4 \leqslant T \longrightarrow Time@T, \mathsf{N}(*)@T, \mathsf{R}(I, Z)@T, \mathsf{M}(I, X)@(T + \delta_L), \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{REC}(X, I) = \langle \delta_L, \delta_R, r_R \rangle$

SND: $Time@T, \mathsf{N}(*)@T_1, \mathsf{R}(I, Z + r_R)@T_2, \mathsf{M}(I, X)@T_3, \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T \longrightarrow Time@T, \mathsf{N}(X)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{SND}(X, I) = \langle \delta_L, \delta_R, r_R \rangle$

Message Composition and Decomposition Rules:

COMP: $Time@T, \mathsf{M}(I, X)@T_1, \mathsf{M}(I, Y)@T_2, \mathsf{R}(I, Z + r_R)@T_3 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T \longrightarrow Time@T, \mathsf{M}(I, \langle X, Y \rangle)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{COMP}(X, Y, I) = \langle \delta_L, \delta_R, r_R \rangle$

DCMP: $Time@T, \mathsf{M}(I, \langle X, Y \rangle)@T_1, \mathsf{R}(I, Z + r_R)@T_2, P(I)@T_3, P(I)@T_4 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T, T_4 \leqslant T \longrightarrow Time@T, \mathsf{M}(I, X)@(T + \delta_L), \mathsf{M}(I, Y)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{DCMP}(\langle X, Y \rangle, I) = \langle \delta_L, \delta_R, r_R \rangle$

USE: $Time@T, \mathsf{M}(I, X)@T_1, \mathsf{R}(I, Z + r_R)@T_2, P(I)@T_3, P(I)@T_4 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T, T_4 \leqslant T \longrightarrow Time@T, \mathsf{M}(I, X)@T_1, \mathsf{M}(I, X)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R),$
where $\text{SPEC}_{USE}(X, I) = \langle \delta_L, \delta_R, r_R \rangle$

ENC: $Time@T, \mathsf{M}(I, K)@T_1, \mathsf{M}(I, X)@T_2, \mathsf{R}(I, Z + r_R)@T_3, P(I)@T_4, P(I)@T_5 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T, T_4 \leqslant T, T_5 \leqslant T \longrightarrow Time@T, \mathsf{M}(I, K)@T_1, \mathsf{M}(I, X)@T_2, \mathsf{M}(I, \{X\}_K)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec@}(I, r_R)(T + \delta_L)$
where $\text{SPEC}_{ENC}(K, X, I) = \langle \delta_L, \delta_R, r_R \rangle$

DEC: $Time@T, \mathsf{M}(I, K^{-1})@T_1, \mathsf{M}(I, \{X\}_K)@T_2, \mathsf{R}(I, Z + r_R)@T_3, P(I)@T_4, P(I)@T_5 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T, T_4 \leqslant T, T_5 \leqslant T \longrightarrow Time@T, \mathsf{M}(I, K^{-1})@T_1, \mathsf{M}(I, \{X\}_K)@T_2, \mathsf{M}(I, X)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{DEC}(K^{-1}, \{X\}_K, I) = \langle \delta_L, \delta_R, r_R \rangle$

GEN: $Time@T, \mathsf{R}(I, Z + r_R)@T_1, P(I)@T_2, P(I)@T_3 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T \longrightarrow \exists N. Time@T, \mathsf{M}(I, N)@(T + \delta_L), \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, r_R)@(T + \delta_R)$
where $\text{SPEC}_{GEN}(I) = \langle \delta_L, \delta_R, r_R \rangle$

Maintenance Rules:

RES: $Time@T, \mathsf{R}(I, Z)@T_1, \mathsf{Rec}(I, r_R)@T_2 \mid T_1 \leqslant T, T_2 \leqslant T \longrightarrow Time@T, \mathsf{R}(I, Z + r_R)@T, P(I)@T$

DELM: $Time@T, \mathsf{M}(I, X)@T_1 \mid T_1 \leqslant T \longrightarrow Time@T, P(I)@T$

Fig. 3. Balanced Resource-Bounded Intruder Theory

Definition 5.2 (Balanced Resource-Bounded Intruder)**.** *A* balanced resource-bounded intruder*, $\mathcal{I}$, consists of his unique identification symbol* int*, his maximal resource $r_{max}(\mathsf{int})$, a natural number dum_{int}, specifying the number of empty facts $P(\mathsf{int})$ available to intruder $\mathcal{I}$, a finite set of* M *facts specifying the initial knowledge base of intruder $\mathcal{I}$, and definitions of* SPEC_R *functions, for each rule R given in Figure 3.*

5.3. Attack Example: HTTP GET

This Section illustrates a DoS attack on a verification scenario involving the protocol theory of the HTTP GET described in Section 4.3. The attack is very close to the Slowloris attack [4]. The values for service timeout and number of packets sent by the intruder are the same as used in practice.

Assume a service with the HTTP GET protocol theory given in Figure 1 with initially 300 workers available, that is, $r_{ini}(\mathsf{id}) = 300$ and $r_{min}(\mathsf{id}) = 0$, so the service is denied when the service has no workers left. Assume the network capacity of at least 300 messages.

Consider a bounded traffic intruder I, with the function $\text{SPEC}_{SND} = \langle 1, 30, 1\rangle$, that is, intruder consumes 1 resource when he sends a message and this resource can only be used again after 30 time units, and assume $\text{SPEC}_R = \langle 0, 0, 0\rangle$ for the remaining rules R. Moreover, assume that $r_{max}(I) = 350$, that is, he has 350 resources. This means that the intruder can only send 350 messages in every 30 units time window. Finally, the intruder knows the relevant information from the GET protocol, namely, the set of timed facts: $\mathcal{M} = \{\, \mathsf{M}(I, INIT)@0.0, \mathsf{M}(I, GET)@0.0, \mathsf{M}(I, Inc)@0.0, \mathsf{M}(I, Com)@0.0\}$. Thus, the initial configuration is: $\mathcal{M} \cup \{\, Time@0, \mathsf{R}(s, 300)@0, \mathsf{R}(I, 350)@0, \mathsf{Av}(s)@0 \,\} \cup \{\mathsf{N}(*)@0\}^{300}$. Finally, let $mdur = 300$, *i.e.*, the service has to be down for 300 time units for a successful DoS attack.

The attack is performed by the intruder applying the USE rule with $M(I, INIT)$ 300 times, that is, making 300 copies of this message. This has no cost. Then, applying the SND rule on $M(I, INIT)$ 300 times, generating 300 copies of $\mathsf{N}(INIT)@1$. That is, the intruder sends a *burst of 300 messages*. Time then advances one time unit. Now the service applies the INIT rule 300 times generating 300 protocol sessions, *i.e.*, 300 facts S_0, and consuming all the service's resources. Thus the fact $\mathsf{Av}(s)$ is replaced by $\mathsf{Den}(s)$ using the corresponding service availability rule. At this point, 30 time units pass. The intruder can then recover his resources by applying 300 instances of the RES rule. The intruder then applies the USE rule with $M(Inc)$ 300 times and applies the SND rule 300 times generating 300 copies of $\mathsf{N}(Inc)$. That is, the intruder sends another burst of messages. These facts are then used by the service to move to state S_1. By waiting another 30 time units, and re-generating copies of $\mathsf{N}(Inc)$, *i.e.*, sending periodic bursts of messages, the intruder is able to consume the service's resources to 0 for indefinite time, leading to a DoS attack.

Notice that this attack, while captured by the model, has a great deal of non-determinism, *e.g.*, different rules can be applied to a configuration. Moreover, the length of the witness trace is quite large, making it challenging for automated verification to find. In Section 8, it is shown how Rewriting Modulo SMT [27] can help automate the search for DoS attacks by using symbolic search.

Finally, traditional flooding attacks, such as SYN flooding can also be modelled. For such attacks, the attacker(s) has resources that allow him to send a very large number of messages.

6. Verification Problems

This Section investigates some verification problems for resource-sensitive timed protocol and intruder theories. Given protocol and intruder theories and an initial configuration representing the knowledge of

participating agents and intruders, one looks for a trace representing an attack. For that purpose, a goal configuration will denote that a protocol has suffered an attack.

6.1. DoS Problem

DoS attacks are now formulated within the proposed framework, and contrasted with notions of DoS attacks from existing literature. In [21] a DoS attack is viewed as "the resource exhaustion attack, in which an attacker, by initiating a large number of instances of a protocol, causes a victim to exhaust his resources". According to [48], a DoS "is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service". As per US Department of Homeland Security official website [49], a DoS attack occurs "when legitimate users are unable to access information systems, devices, or other network resources due to the actions of a malicious cyber threat actor". Formulation of DoS attacks proposed here addresses the issues from the above definitions.

The notion of DoS attack from [21] is further refined by an additional duration parameter that is relevant for the following reasons. Flooding attacks are always possible in the presence of powerful attackers. However, very short service interruptions may be tolerated in practice, while prolonged unavailability of service would be considered as a successful DoS attack. As the access to server resources is not instant, due to, *e.g.*, network delays, users may tolerate short service delays, *i.e.*, short service interruptions may not be considered as actual denials of service.

Intuitively, a DoS attack on a service is successful if the service's resources are exhausted for some *duration*, *mdur*. More precisely, the verification task is to determine whether, in the presence of attackers, some service is subject to a DoS attack, by searching for a non-critical trace of the form:

$\mathcal{S}_0 \longrightarrow \mathcal{S}_1 \longrightarrow \cdots \longrightarrow \mathcal{S}_i \longrightarrow \cdots \longrightarrow \mathcal{S}_{i+m} \longrightarrow \cdots \longrightarrow \mathcal{S}_n$,

where, $\mathcal{S}_0$ is the initial configuration of the verification scenario, the global time, t_i, in the configuration $\mathcal{S}_i$, and the global time, t_{i+m} in the configuration $\mathcal{S}_{i+m}$, are such that $t_{i+m} - t_i \geqslant mdur$, and that for a service, id, its resources in all configurations between $\mathcal{S}_i$ and $\mathcal{S}_{i+m}$ are less or equal to $r_{min}(\mathsf{id})$.

In the definition of the DoS problem given below, we consider a number of services. This models an operation of a group of services, each of which should be available to clients. In addition, the DoS problem involves a number of intruders, thus specifying DDoS attacks as well.

Definition 6.1 (DoS Verification Scenario)**.** *A (respectively,* balanced*)* DoS verification scenario $\mathcal{V}$ *consists of the following components:*

- *A finite set of (respectively, balanced) services* $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ *(see Definition 4.3);*
- *A finite set of (respectively, balanced) resource-bounded intruders* $\{\mathcal{I}_1, \ldots, \mathcal{I}_m\}$ *(see Definition 5.1);*
- *natural numbers* $mdur_i$ *for,* $1 \leqslant i \leqslant n$*, specifying the minimal duration that the resources of the service* $\mathcal{A}_i$ *have to be consumed to represent a successful DoS attack.*

The initial configuration of DoS verification scenario $\mathcal{V}$*,* $\mathcal{S}_\mathcal{V}$*, contains exactly the following timed facts, for* $1 \leqslant i \leqslant n$ *and* $1 \leqslant j \leqslant m$*:*

- $Time@0$*, specifying that the initial global time is* 0*;*
- $\mathsf{R}(\mathsf{id}_i, r_{ini}(\mathsf{id}_i))@0$*, where* id_i *and* $r_{ini}(\mathsf{id}_i)$ *are the unique identification symbol and the initial service resources of service* $\mathcal{A}_i$*;*
- $\mathsf{Av}(\mathsf{id}_i)@0$*, specifying that the service* $\mathcal{A}_i$ *is available;*
- n *empty network facts* $\mathsf{N}(*)@0$ *representing network bandwidth;*

- $\mathsf{R}(\mathsf{int}_j, r_{max}(\mathsf{int}_j))@0$, *where* int_j *and* $r_{max}(\mathsf{int}_j)$ *are the unique identification symbol and maximal resource of the intruder* $\mathcal{I}_j$*;*
- *the facts in* $\mathcal{M}_j$*, the initial knowledge base of* $\mathcal{I}_j$*;*
- *the facts denoting the initial setting including key distribution,*
- *for balanced DoS verification scenario only,* dum_j *empty facts* $P(\mathsf{int}_j)@0$*, where* dum_j *is the number of available empty facts specified in* $\mathcal{I}_j$*, and* d_{id_i} *empty facts* $D(\mathsf{id}_i)@0$*, where* d_{id_i} *is the number of empty facts specified in* $\mathcal{A}_i$*.*

The goal of the DoS verification scenario $\mathcal{V}$ *is as follows:*

$$\mathcal{GS}_{\mathcal{V}} = \{\langle\{Time@T, \mathsf{Den}(\mathsf{id}_i)@T_1\}, \{T \geqslant T_1 + mdur_i\}\rangle \mid 1 \leqslant i \leqslant n\}$$

where $1 \leqslant i \leqslant n$*, and* id_i *is the unique identification symbol of service* $\mathcal{A}_i$*.*

Finally, the critical configuration specification of the DoS verification scenario $\mathcal{V}$*, (DoS scenario* $\mathcal{CS}$*),* $\mathcal{CS}_{\mathcal{V}}$*, is the union of the critical configuration specifications of all services* $\mathcal{A}_i$*,* $1 \leqslant i \leqslant n$*.*

The DoS problem associated to a DoS verification scenario is reduced to searching for non-critical traces as defined below. This involves critical configurations relating to protocol state timeouts of all protocols, and resource availability of all services in the scenario.

Definition 6.2 (DoS Problem)**.** *Let* $\mathcal{V}$ *be a DoS verification scenario. The* DoS problem *is to determine whether there is a non-critical trace w.r.t.* $\mathcal{CS}_{\mathcal{V}}$ *from the initial configuration* $\mathcal{S}_{\mathcal{V}}$ *to a goal configuration w.r.t.* $\mathcal{GS}_{\mathcal{V}}$*.*

Definition 6.3 (Balanced DoS Problem)**.** *The* balanced DoS problem *is a DoS problem with a balanced DoS verification scenario.*

Notice the role of *Service* $\mathcal{CS}$ (Definition 4.3). Without the *Timeout* $\mathcal{CS}$, false DoS attacks could be found as traces where the service simply does not garbage-collect protocol sessions that have expired due to a timeout. Similarly, without *Denied* $\mathcal{CS}$ and *Available* $\mathcal{CS}$, there would be traces where the facts Av and Den are not updated according to the level of resources of some service. For example, a trace would exist with a configuration where $\mathsf{Den}(\mathsf{id})$ is present although the service id has enough resources.

Notice that $mdur = 0$ in the DoS verification scenario, models the DoS attacks with no duration attached, as in [21].

6.2. Leakage Problem

While the availability of service due to resource consumption is the main verification problem of this paper, in principle, resource-sensitive timed protocol theories may lead to other types of attacks. Consider, for example *network coding*, which is a technique to improve information transfer, especially useful in bandwidth limited mobile networks [50, 51]. A related example scenario consists of two groups passing quickly with a large data object to share. Many senders in one group send small redundant parts of a data object, quickly captured by receivers in the other group. Later members of each group can cooperate to reassemble the received data objects. Related issues to consider are size of the object parts, amount of redundancy, depending factors such as time available for transfer and network loss rate, which clearly relate to the theories introduced in this paper.

In the future we intend to investigate such additional issues of security verification involving services and intruders that are resource- and time-sensitive. We start here by investigating the problem of whether

services may leak some confidential information, such as cryptographic keys. This is of importance for protocols, such as TLS, that involve both resource consumption and cryptography. Also, the operation of a service when its resources are limited may switch to some special working policies and disabled features, such as "safe mode" operation, where security issues may be further compromised. Furthermore, security properties of services may be verified w.r.t. parametric resource-bounded intruder models of different types.

The leakage problem is related to the standard *secrecy problem* in security protocol analysis [22], which is the problem of whether or not an intruder can discover a secret originally known to another protocol participant. The following definition of the secrecy problem is paraphrased from [22].

Definition 6.4 (Secrecy Problem)**.** *Let $\mathcal{R}$ be the set of MSR protocol rules, $\mathcal{I}$ be the set of intruder rules, $\mathcal{S}_0$ be an initial configuration denoting that a secret α is possessed by some participant. The* secrecy problem *is the problem of whether there is a trace of $\mathcal{R}$ and $\mathcal{I}$ rules from $\mathcal{S}_0$ to a configuration denoting that the intruder possesses the secret α.*

In our previous work, we studied the secrecy problem for untimed protocol and intruder MSR theories [23] and timed theories [30] (without resource aspects). We now consider a more subtle version of the problem involving resource-sensitive timed protocol and intruder theories, both for general and for balanced MSR theories. In order to avoid the confusion in terminology, we refer to this new and generalized problem as the *leakage problem*.

In the leakage problem, we assume there is a constant in the initial configuration which denotes confidential information, such as an encryption key, that should not be leaked to an intruder.

Definition 6.5 (Leakage Verification Scenario)**.** *A (respectively,* balanced*)* leakage verification scenario $\mathcal{L}$ *consists of the following components:*

- *A finite set of (respectively, balanced) services $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ (see Definition 4.3);*
- *A finite set of (respectively, balanced) resource-bounded intruders $\{\mathcal{I}_1, \ldots, \mathcal{I}_m\}$ (see Definition 5.1);*
- *A constant α known only to some agent.*

The initial configuration of leakage verification scenario $\mathcal{L}$, $\mathcal{S}_\mathcal{L}$, is the same as the initial configuration of DoS verification scenario $\mathcal{L}$.
The goal of the leakage verification scenario $\mathcal{L}$ is as follows:

$$\mathcal{GS}_\mathcal{L} = \{\langle\{Time@T, \mathsf{M}(\mathsf{int}_j, \alpha)@T_1\}, \emptyset\}\rangle \mid 1 \leqslant i \leqslant n\}$$

where $\mathcal{I}_j$, is an intruder from the scenario, $1 \leqslant j \leqslant m$.
Finally, the critical configuration specification of the leakage verification scenario $\mathcal{L}$ *(Leakage scenario $\mathcal{CS}$), $\mathcal{CS}_\mathcal{L}$, is the union of the protocol critical configuration specifications of all protocols from the services $\mathcal{A}_i$, $1 \leqslant i \leqslant n$.*

Leakage scenario $\mathcal{CS}$ only involves $\mathcal{CS}$ which relate to protocol timeouts, not the *Service $\mathcal{CS}$* related to resource availability. Hence, an attack trace related to the leakage problem denotes that some intruder has learned the secret α by running the service protocols, regardless of the levels of resources of the services involved. Leakage problem associated to a leakage verification scenario is now defined as a non-critical reachability problem.

Definition 6.6 (Leakage Problem)**.** *Let $\mathcal{L}$ be a leakage verification scenario. The* leakage problem *is to determine whether there is a non-critical trace w.r.t. $\mathcal{CS}_{\mathcal{L}}$ from the initial configuration $\mathcal{S}_{\mathcal{L}}$ to a goal configuration w.r.t. $\mathcal{GS}_{\mathcal{L}}$.*

Definition 6.7 (Balanced Leakage Problem)**.** *The* balanced leakage problem *is a leakage problem with a balanced leakage verification scenario.*

Notice that, differently from [23], the leakage problem not only generalizes the verification to resource and time sensitive theories, moreover, it may involve several different protocol theories from the relevant scenarios, enabling the verification of multi-protocol environments.

Similarly to the bounded-time problems introduced in [34], a bounded-time version of the leakage problem could be considered, *e.g.*, by bounding the total time in a trace or by bounding the total number of protocol sessions.

7. Complexity Results

Resource-sensitive timed protocol and intruder theories defined in Sections 4 and 5 represent a fragment of general timed MSR. Hence, some of the relating complexity results are obtained by relying on the relevant results for general MSR theories and related problems. However, the DoS problem is defined as non-critical reachability problem for dense time MSR (Section 6), for which the complexities were previously unknown. We first provide complexity results for non-critical reachability problem for MSR theories with dense time next. These general results are essential in providing complexities of the DoS problem.

7.1. Complexity Results for Non-Critical Reachability Problem for Dense Time MSR

When dealing with the complexity of verification problems in timed MSR with dense time there are several challenges that need to be addressed, starting with the underlying nondeterministic nature of the multiset rewriting formalism. Then, an unbounded number of fresh values can appear in a trace. Additionally, in our timed MSR there is no bound on the global time value, *i.e.*, on represented time periods. Furthermore, in the dense time model, there is the additional non-determinism in the choice of ε in time advancement $Tick$ rule.

Some of the above issues have been addressed for balanced MSR theories with a bound on the size of facts, by using the abstractions called *circle-configurations* [24]. It has been shown in [24] that, with respect to the reachability problem (without critical configurations) for such MSR, traces over circle-configurations are a sound and complete representation of traces with dense time. In particular, the $Tick$ rule is represented by a collection of rules defined over circle-configurations.

However, additional challenges appear when non-critical traces in dense time setting are considered. As discussed in Section 3, the notion of non-critical traces in dense time setting is much more elaborate than in the untimed and discrete time models. Recall that, as per Definition 3.5 showing that a trace of a dense time MSR is non-critical involves not only the configurations it contains, but also an infinite number of configurations obtained by decomposing $Tick$ rules in order to faithfully capture the continuity of time. We now present the obtained complexity results for the non-critical reachability problem for dense time MSR, including the balanced case. Undecidability of the general case of non-critical reachability problem for dense time MSR models follows directly form the undecidability of the reachability problem [37].

Theorem 7.1 (Non-Critical Reachability Problem)**.** *The non-critical reachability problem is undecidable in general.*

Handling the complexity of the balanced non-critical reachability problem involves a new auxiliary notion of *immediate successor configurations*, related to satisfiability of relevant time constraints. Using immediate successor configurations reduces the search space when checking that a trace of a dense time MSR is non-critical. Furthermore, there is only a finite number of abstractions related to a given balanced non-critical reachability problem, which bounds the length of solution traces, resulting in a polynomial space search space. Full details of this approach appear in the Technical Report [52]. For space restrictions, here we only present the main ideas behind the proof.

Given a balanced non-critical reachability problem, a natural number d is syntactically inferred as an upper-bound on the numbers appearing in time constraints and timestamps of the problem specification, *i.e.*, in the initial configuration $\mathcal{S}_0$, rules $\mathcal{T}$, critical configuration specification $\mathcal{CS}$ and goal $\mathcal{GS}$. Such a bound was used in the definition of circle-configurations and is now related to time constraints in the definition of immediate successor relation between configurations.

Definition 7.2 (Immediate Successor Configurations)**.** *Given a timed MSR $\mathcal{T}$ with dense time, and a natural number d, let $\mathcal{C}_d$ be a set of all constrains containing natural numbers up to d:*

$$\mathcal{C}_d = \{\ T > T' \pm N,\ T \geqslant T' \pm N,\ T = T' \pm N\ \mid\ N \leqslant\ d\ \}.$$

A configuration $\mathcal{S}_2$ is an immediate successor *of configuration $\mathcal{S}_1$ w.r.t. d if the following holds:*

i) There exists $\varepsilon > 0$ such that $\mathcal{S}_1 \longrightarrow_{Tick_\varepsilon} \mathcal{S}_2$;
ii) $\mathcal{S}_1$ and $\mathcal{S}_2$ do not satisfy the same set of constraints from $\mathcal{C}_d$, where variables T and T' refer to timestamps of the same facts from $\mathcal{S}_1$ and $\mathcal{S}_2$;
iii) For all $\varepsilon' > 0$, $\varepsilon' < \varepsilon$ if $\mathcal{S}_1 \longrightarrow_{Tick_{\varepsilon'}} \mathcal{S}'$ then $\mathcal{S}'$ satisfies the same constraints from $\mathcal{C}_d$ either as $\mathcal{S}_1$ or as $\mathcal{S}_2$.

When $\mathcal{S}_2$ is an immediate successor of $\mathcal{S}_1$ w.r.t. d the notation $\mathcal{S}_1 \longrightarrow_{Tick^d_{IS}} \mathcal{S}_2$ is used.

For example, configuration $\mathcal{S}' = \{\ Time@2.4,\ F@0.5,\ G@2.4,\ H@1.9\ \}$ is an immediate successor of configuration $\mathcal{S} = \{\ Time@2.1,\ F@0.5,\ G@2.4,\ H@1.9\ \}$. While $\mathcal{S}$ satisfies time constraint $T < T_1$, $\mathcal{S}'$ satisfies $T = T_1$ instead, where time variables T and T_1 relate to facts $Time@T$ and $G@T_1$, respectively. On the other hand, configuration $\{\ Time@2.45,\ F@0.5,\ G@2.4,\ H@1.9\ \}$ is not an immediate successor of $\mathcal{S}$ because, *e.g.*

$\{\ Time@2.1,\ F@0.5,\ G@2.4,\ H@1.9\ \} \longrightarrow_{Tick_{0.3}}$
$\{\ Time@2.4,\ F@0.5,\ G@2.4,\ H@1.9\ \} \longrightarrow_{Tick_{0.05}} \{\ Time@2.45,\ F@0.5,\ G@2.4,\ H@1.9\ \}$

where all three of the above configurations satisfy different sets of time constraints.

There is a clear connection between non-critical traces and immediate successor configurations. Notice that if neither $\mathcal{S}_i$ nor its immediate successor configuration $\mathcal{S}_{i+1}$ is critical, then the condition on non-critical traces given in Definition 3.5 is satisfied.

Proposition 7.3. *Let $\mathcal{T}$ be a timed MSR with dense time, and d a natural number. Let $\mathcal{S} \longrightarrow_{Tick_\varepsilon} \mathcal{S}'$, where $\mathcal{S}'$ is an immediate successor of $\mathcal{S}$ w.r.t. d. If $\mathcal{S}$ and $\mathcal{S}'$ are not critical w.r.t. some critical configuration specification $\mathcal{CS}$ involving constraints from $\mathcal{C}_d$, then for any $\varepsilon_1 > 0$, $\varepsilon_1 < \varepsilon$, the configuration $\mathcal{S}''$, such that $\mathcal{S} \longrightarrow_{Tick_{\varepsilon_1}} \mathcal{S}'' \longrightarrow_{Tick_{\varepsilon_2}} \mathcal{S}'$, is not critical.*

Proof. Let $\mathcal{S} \longrightarrow_{Tick_{IS}} \mathcal{S}'$, and assume neither $\mathcal{S}$ nor $\mathcal{S}'$ is critical. Let $\mathcal{S} \longrightarrow_{Tick} \mathcal{S}'' \longrightarrow_{Tick} \mathcal{S}'$. Since $\mathcal{S}'$ is an immediate successor of $\mathcal{S}$, as per Definition 7.2, such configuration $\mathcal{S}''$ satisfies the same set of constraints from $\mathcal{C}_d$ as either $\mathcal{S}$ or $\mathcal{S}'$. This includes the constrains used in $\mathcal{CS}$. Since both $\mathcal{S}$ and $\mathcal{S}'$ are not critical, $\mathcal{S}''$ is not critical as well. □

The above result is then used to show bisimulation of non-critical traces with traces over circle-configurations. This allows the search for a solution non-critical trace symbolically, reducing the search space from an infinite number of traces (recall the $Tick$ rule decomposition) to traces over a finite number of abstractions. The search can be done in space polynomial to the inputs. For full details see [52].

Theorem 7.4 (Balanced Non-Critical Reachability Problem)**.** *The non-critical reachability problem for balanced timed MSR with dense time is PSPACE-complete when assuming a bound on the size of facts.*

7.2. Complexity Results for DoS Problems

The undecidability of the general version of the DoS problem follows from the undecidability of secrecy problem for general untimed MSR theories [23, 37]. It is shown how to encode the secrecy problem as an instance of a DoS problem in the proof of Lemma 7.6.

The complexity of the balanced DoS problem relies on the PSPACE-completeness of the secrecy problem for bounded memory intruder and balanced untimed MSR protocol theories [23] and on the complexity of the non-critical reachability problem for dense time MSR presented in Section 7.1. It clearly follows from Definition 6.2 and Definition 6.3 that the non-critical reachability problem is the DoS verification scenario.

Lemma 7.5. *The (balanced) DoS problem is an instance of the (balanced) non-critical reachability problem for MSR with real time.*

The following lemma gives the lower bound for the complexity of DoS problems. We recall that the secrecy problem is undecidable in general [22], and PSPACE-complete for bounded memory intruder and balanced untimed MSR protocol theories [23] when a bound on the size of facts is assumed.

Lemma 7.6. *The secrecy problem for Dolev-Yao intruder and MSR protocol theories is an instance of the DoS problem. The secrecy problem for bounded memory Dolev-Yao intruder and balanced MSR protocol theories is an instance of the balanced DoS problem.*

Proof. The secrecy problem is encoded as an instance of the DoS problem. In particular, it follows from the encoding that the DoS problem occurs if and only if the intruder discovers the secret. The case of balanced intruder and protocol theories is described. The general case follows by simply ignoring the empty facts.

Let $\mathcal{S}_0$ be the initial configuration of the given secrecy problem $\mathcal{D}$. It is assumed $\mathcal{S}_0$ contains the facts representing initial knowledge, such as participants names and keys, a fact denoting that a secret α is known to some participant, as well as the initial intruder knowledge, and a number of empty facts $P(*)$ and $D(*)$ representing intruder and system memory, respectively.

$$
\begin{array}{ll}
\textit{REC:} & \mathsf{N}(X), P(*) \to \mathsf{M}(X), D(*) \\
\textit{SND:} & \mathsf{M}(X), D(*) \to \mathsf{N}(X), P(*) \\
\textit{COMP:} & \mathsf{M}(X), M(Y) \to \mathsf{M}(\langle X, Y\rangle), P(*) \\
\textit{DCMP:} & \mathsf{M}(\langle X, Y\rangle), P(*) \to \mathsf{M}(X), \mathsf{M}(Y) \\
\textit{USE:} & \mathsf{M}(X), P(*) \to \mathsf{M}(X), \mathsf{M}(X) \\
\textit{ENC:} & \mathsf{KP}(K_d, K_e), \mathsf{M}(K_e), \mathsf{M}(X) \to \mathsf{KP}(K_d, K_e), \mathsf{M}(K_e), \mathsf{M}(enc(K_e, X)) \\
\textit{DEC:} & \mathsf{M}(K_d), \mathsf{KP}(K_e, K_d), \mathsf{M}(enc(K_e, X)), P(*) \to \mathsf{M}(K_d), \mathsf{KP}(K_e, K_d), \mathsf{M}(x), \mathsf{M}(enc(K_e, X)) \\
\textit{GEN:} & P(*) \to \exists N.\mathsf{M}(N) \\
\textit{DELM:} & \mathsf{M}(X) \to P(*)
\end{array}
$$

Fig. 4. (Untimed) Bounded Memory Intruder Theory [23]

Let $\mathcal{T}$ be the given protocol theory. Its rules have one of the following forms:

$$
\begin{array}{l}
W, D(*) \to W, S_0(\vec{x}) \\
S_0(\ldots), D(*), W \to \exists \vec{z}.S_l(\ldots), \mathsf{N}(\ldots), W' \\
S_i(\ldots), \mathsf{N}(\ldots), W \to \exists \vec{z}.S_j(\ldots), \mathsf{N}(\ldots), W' \\
S_h(\ldots), \mathsf{N}(\ldots), W \to \exists \vec{z}.S_k(\ldots), D(*), W' \\
S_k \to D(*)
\end{array} \tag{7}
$$

where $l > 0$, $j > i$, $k > h$, S_k is the final state of one of the protocol role theories, and W and W' are multisets of facts containing no role states nor N facts. Intruder theory $\mathcal{I}$ corresponds to the bounded memory intruder rules from [23], as depicted in Figure 4.

To the secrecy problem given above, the following DoS verification scenario $\mathcal{V}$ is related, containing one intruder $\mathcal{I}'$ with the identifier i and one service with the identifier s. Attack duration $mdur = 0$ is set, so the goal configuration is specified by $\mathcal{GS}_{\mathcal{V}} = \{\langle\{Time@T, \mathsf{Den}(s)@T_1\}, \{T \geqslant T_1\}\rangle\}$.

To the intruder theory $\mathcal{I}$ (Figure 4) naturally corresponds the balanced resource-bounded intruder theory $\mathcal{I}'$ (Figure 3) with $\text{SPEC}_R = \langle 0, 0, 0\rangle$, for all intruder rules R, and $r_{max}(i) = 1$. Notice that the intruder spends no time nor resources for his actions. For example, the obtained GEN rule of $\mathcal{I}'$:

$$
Time@T, \mathsf{R}(I, Z)@T_1, P(I)@T_2, P(I)@T_3 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T \longrightarrow \\
\quad \exists N.Time@T, \mathsf{R}(I, Z)@T, \mathsf{Rec}(I, 0)@T, \mathsf{M}(I, N)@T
$$

followed by the RES rule of $\mathcal{I}'$:

$$
Time@T, \mathsf{R}(I, Z)@T_1, \mathsf{Rec}(I, X)@T_2 \mid T_1 \leqslant T, T_2 \leqslant T \longrightarrow Time@T, \mathsf{R}(I, Z + X)@T, P(I)@T,
$$

results in facts $Time@T, \mathsf{R}(i, r)@t_1, P(i)@t_2, P(i)@t_3$ being replaced by facts $Time@T, \mathsf{R}(i, r)@t$, $\mathsf{M}(i, n)@t$, $P(i)@t$, ie, generating a nonce, while keeping the same level of resources.

Initial configuration contains all the facts of $\mathcal{S}_0$ timestamped with 0, $P(*)$ and $D(*)$ facts replaced by facts $P(i)@0$ and $D(s)@0$, respectively, and additional facts $Time@0$, $Av(s)@0$, $\mathsf{R}(s, 1)@0$ and $\mathsf{R}(i, 0)@0$.

Protocol resource theory $\mathcal{T}'$ of the service with identifier s is obtained by translating the rules Eq. (7) of protocol theory $\mathcal{T}$, simply by adding *e.g.*, resource facts $\mathsf{R}(s, X)$, protocol state timeout facts, $\mathsf{T}_i(s, Y)$, obtaining thus protocol initialization and execution rules. Additionally, protocol state timeout and service availability rules are added, as per Definition 4.1. All the facts in the obtained rules are timestamped with time variables, the fact $Time@T$ is added, as well as the corresponding time constraints, as per Definition 4.1. Here all state timeout values are set to 1, $r_{min}(s) = 0$, as well as setting initial resources $r^s_{ini} = 1$, and zero resource cost for all the rules. Finally, an additional protocol resource theory of service

s is added, which is enabled whenever the secret α is released on the network:

$$Time@T, \mathsf{R}(s,1)@T_1, \mathsf{N}(\alpha)@T_2, D(s)@T_3, D(s)@T_4 \mid T_1 \leqslant T, T_2 \leqslant T, T_3 \leqslant T, T_4 \leqslant T \longrightarrow \exists S_{id}.[Time@T, \mathsf{S}'_0(s,S_{id},1)@T, \mathsf{T}'_0(s,S_{id})@(T+1), \mathsf{R}(s,0)@T, \mathsf{N}(\alpha)@T] \tag{8}$$

$$Time@T, \mathsf{S}'_0(s,S_{id},1)@T_1, \mathsf{T}'_0(s,S_{id})@T_2, \mathsf{R}(s,R)@T_3 \mid T_1 \leqslant T, T_2 > T, T_3 \leqslant T \longrightarrow Time@T, \mathsf{R}(s,R+1)@T, D(s)@T, D(s)@T \tag{9}$$

$$Time@T, \mathsf{R}(s,R)@T_1, \mathsf{T}_0(s,S)@T, \mathsf{S}_0(s,S,1)@T_2 \mid T_1 \leqslant T, T_2 \leqslant T \longrightarrow Time@T, \mathsf{R}(s,1+R)@T \tag{10}$$

Notice that the above protocol initialization rule Eq. (8) is the only rule with a non-zero resource cost. It has the cost 1, and is applicable only when the secret α is released to the network, available to the intruder to learn the secret. Recall that the related *DoS scenario* $\mathcal{CS}$ contains *Denied* $\mathcal{CS}$: $\langle\{Time@T, \mathsf{R}(s,0)@T_1, \mathsf{Av}(s)@T_2\}, \{T > T_1, T > T_2\}\rangle$. Once the rule Eq. (8) reduces the service resources to zero, *Denied* $\mathcal{CS}$ forces the application of the service availability rule that creates the $\mathsf{Den}(s)@t$ fact, reaching the goal.

It follows that the secrecy problem $\mathcal{D}$ has a solution iff the related scenario $\mathcal{V}$ allows a DoS attack. □

Theorem 7.7 (DoS problem)**.** *The DoS problem is undecidable in general.*

Proof. The lower bound is inferred from Lemma 7.6 and the undecidability of the secrecy problem for DY intruder and untimed MSR protocol theories [23]. □

From Lemma 7.5 and Lemma 7.6, we obtain the following complexity result for the balanced DoS problem.

Theorem 7.8 (Balanced DoS problem)**.** *Assuming a bound on the size of facts, the balanced DoS problem is PSPACE-complete.*

7.3. Complexity Results for Leakage Problems

By relying on the previous complexity results for the secrecy problem [23] and for the non-critical reachability problem for timed MSR, the complexity result for the resource-sensitive timed version of secrecy problem is obtained.

Lemma 7.9. *The (balanced) leakage problem for resource-sensitive timed theories is an instance of the (balanced) non-critical reachability problem for MSR with real time.*

Proof. This trivially follows from Definition 6.6 and Definition 6.7. Namely, the non-critical reachability problem is the leakage verification scenario. □

Lemma 7.10. *The secrecy problem for Dolev-Yao intruder and MSR protocol theories is an instance of the leakage problem for resource-sensitive timed theories. The secrecy problem for bounded memory Dolev-Yao intruder and balanced MSR protocol theories is an instance of the balanced leakage problem for resource-sensitive timed theories.*

Proof. The secrecy problem $\mathcal{T}$ for untimed version of bounded memory intruder and balanced MSR protocol theories can be encoded as balanced leakage problem $\mathcal{T}'$. The encoding of the general case of the secrecy problem is obtained by ignoring the empty facts.

The encoding is similar to the encoding given in the proof of Lemma 7.6, but without the protocol theory defined in Eq. (8)-Eq. (10), and relating to the leakage verification scenario instead of the DoS scenario. In particular, the same constant α denotes the secret. Only one intruder i and one service s is involved. All the timestamps in the initial configuration are set to 0. Protocol states of timed protocol theories have timeouts of 1 time unit and zero resource cost for all protocol rules, and with $r_{min}(s) = 0$. Resource cost for the intruder is also zero, *i.e.*, $\text{SPEC}_R = \langle 0, 0, 0 \rangle$, for all intruder rules R.

Exact values of timestamps have no particular impact to the attack trace because the resource and time cost of all intruder rules is zero. Indeed, the attack trace does not have to contain a single $Tick$ rule. In such a trace all the timestamps would be 0, apart from the timestamps of protocol timeout facts T_i. Hence, all constraints attached to rules of intruder and protocol theories are always satisfied. Since the goal:

$\mathcal{GS}_{\mathcal{L}} = \{\langle\{Time@T, \mathsf{M}(i, \alpha)@T_1\}, \emptyset\}\rangle\}$

involves no time constraints, it specifies that the secret is discovered by an intruder, taking any amount of time. Consequently, there is an attack relating to problem $\mathcal{T}$ if and only if there is an attack relating to problem $\mathcal{T}'$. □

From Lemma 7.10 and the undecidability of secrecy problem for DY intruder and untimed MSR protocol theories [23], the following result is obtained.

Theorem 7.11 (Leakage Problem)**.** *The leakage problem is undecidable in general.*

Theorem 7.12 (Balanced Leakage Problem)**.** *The balanced leakage problem is PSPACE-complete when assuming a bound on the size of facts.*

Proof. The lower bound is inferred from Lemma 7.10. We recall the PSPACE-completeness of the secrecy problem $\mathcal{T}$ for untimed version of bounded memory intruder and balanced MSR protocol theories [23].

The upper bound relies on Lemma 7.9 and the PSPACE-completeness of balanced the non-critical reachability problem, obtained in Section 7.1. □

8. Towards Automated Verification

While the main focus of this paper is on laying the foundations for the specification and verification of resource-sensitive timed protocol theories, we elaborate in this section how existing symbolic machinery enables automated verification. We also point out challenges we faced which shall be dealt in future work towards the construction of verification tools.

As with protocol security, the challenge is to reduce automated search by using symbols constrained by a set of constraints instead of enumerating traces with concrete terms and values. In this way, a trace containing such symbols denotes a possibly infinite set of traces each obtained by replacing these symbols by concrete values that satisfy the set of attached constraints. However, differently from the symbolic methods used for protocol verification, that use symbols denoting messages that can be constructed by the intruder, our problems involve constraints on the amount of resources used by services and by the intruder.

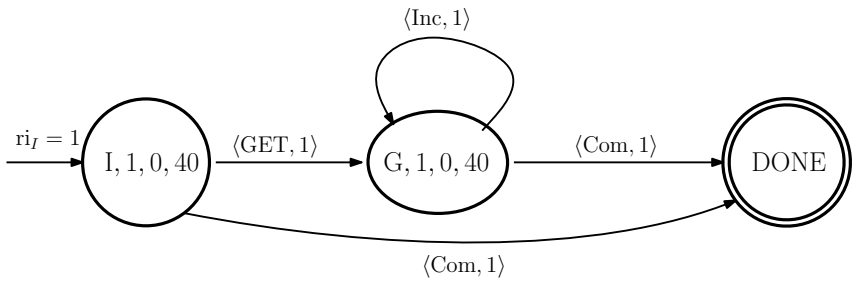

Fig. 5. Example of a Mode Automaton for the HTTP GET protocol extended with resource usage.

For example, to find the Slowloris attack in the scenario used in the example in Section 5.3, one would need to search a tree of depth of at least 500. Moreover, it is not possible to instantiate all values of ε in the time advancement rule as there are infinite possibilities.

We have built a prototype in Maude, available at [28], to validate the hypothesis that, despite the high complexity of the verification problems, in practice it is possible to verify whether services are well configured to mitigate some of the DDoS attacks described above. For example, our prototype can answer whether services have appropriate timeout values for the given assumptions on the intruder. In the following sections, we describe our Maude implementation and the experimental results focusing on how design options improve the scalability of the tool, and point out future work directions.

8.1. Formal Specification Model

Specification of Protocol Resource Usage. The first challenge encountered is how to specify services and their resource usage. While MSR is suitable for building the foundations given its simple and clear semantics, it is not suitable as a programming language as it is hard to enforce good software engineering practices, such as modularity and separation of concerns. Instead of encoding protocol behavior as MSR theories, we specify protocols and their resource usage by extending Mode Automata, which are finite state machines typically designed for specifying the modes of operation of services.

A mode automaton is a finite state machine, *i.e.*, it contains a set of states, $\mathcal{Q}$, an initial state, q_I, possibly a final state, q_F, and transitions, $\delta : \mathcal{Q} \times \mathcal{M} \to \mathcal{Q} \times \mathcal{R}$. Each state, q, is associated with a unique name of the state, two resource values, r_S, r_c, specifying the number of resources required by, respectively, the service and the client to stay in this mode of operation, and a timeout, t, specifying the timeout of the protocol session when in this mode of operation. The initial state includes an additional resource value, ri_I, specifying the number of resources required by the client to initialize the protocol session. A transition $\delta(q_i, m) = (r_c, q_j)$ denotes that the mode of operation may change from state q_i to state q_j provided the service receives the message m and in the process the client consumes r_c resources.

Figure 5 illustrates the Mode Automaton for the HTTP Get Service extended with resource consumption. Service resources are measured by the number of workers, while client resources by the number of messages. For example, the protocol starts at state I after the client spends one resource. To maintain this state, the service requires one worker and sets the timeout at 40 time units. If a Com message denoting a complete header is received, protocol session moves to state DONE, *i.e.*, it ends. Alternatively, if the service receives a GET message, it moves to mode G and keeps at this state if the header is incomplete. Once it is complete the protocol session ends.

There is a relation between Mode Automaton and protocol resource theory (Definition 4.1). The Automaton initial state corresponds to the protocol initialization rule, the state transitions correspond

to the protocol execution rules and the timeout rules. Thus, the Mode Automaton depicted in Figure 5 corresponds exactly to the specification of the GET protocol shown in Section 4.3.

As mentioned above, keeping track of all protocol sessions does not scale, *e.g.*, requiring all 500 protocol sessions for denying a web-server. Instead, we keep track of the bursts of protocol sessions initiated by the intruder. A burst of protocol sessions has the form $(\mathsf{psid}, \mathsf{num}, q)$ where psid is a unique session identifier, num is a constrained symbol specifying the number of protocol sessions initiated in this burst, and q the state in which all protocol sessions of the burst are. For example, the sessions $(\mathsf{psid}_0, \mathsf{num}_0, I)$ and $(\mathsf{psid}_1, \mathsf{num}_1, G)$ are protocol sessions executing the GET protocol and are, respectively, at states I and G. The possible values for the symbols num_0 and num_1 are specified by a set of constraints which are solved during search by calling an SMT-solver, as we detail below. In this way, we do not need to specify the exact amount of parallel protocol sessions in a burst, but only keep it symbolically.

Finally, notice that the mode automaton could be extended by keeping track of more resources, such as the number of resources consumed by the service to transit from one mode to another, or by including a timeout in the transitions. However, for the examples we encountered such extensions were not required.

Service and Intruder Configurations. Our model is an actor-based model containing at least two types of actors, services and intruders, that interact among themselves, by generating and maintaining burst of protocol sessions.

A service configuration has the following form $[\mathsf{id} \mid \text{pxs} \mid \mathsf{ps} \mid r_s \mid r_{min}]$ where:

- id is the service unique identifier;
- pxs is a multiset of protocol sessions currently being executed;
- ps is a set of protocols known by the service;
- r_s is a symbol specifying how many resources are currently available;
- r_{min} is the resource value specifying when the service is unavailable.

Intuitively, a service configuration corresponds to all the facts in a MSR configuration related to a service. For example, it collects all protocol sessions in which the service is involved in, the protocols that it can execute and its available resources.

For example, $[s_0 \mid (\mathsf{psid}_0, \mathsf{num}_0, I), (\mathsf{psid}_1, \mathsf{num}_1, G) \mid GET \mid r_s \mid 0]$ specifies a service currently executing two bursts of protocol sessions, having r_s workers available, and whose service is denied once it has no workers available. As with the number of parallel protocol sessions in a burst, the number of available resources in a service is kept symbolically where the possible values are specified by a set of constraints.

An intruder configuration has the form $[\mathsf{id} \mid \text{pxs} \mid \mathsf{ps} \mid r_I \mid t_{rec}]$ where

- id is the intruder unique identifier;
- pxs is a multiset of protocol sessions currently being executed by the intruder;
- ps is a set of protocols known by the intruder;
- r_I is a symbol specifying the number of resources currently available by the intruder;
- t_{rec} is a value specifying when the intruder can recover its used resources.

Similarly as with the service configurations, intruder configurations corresponds to all the facts in an MSR configuration, such as the protocol sessions in which the intruder is participating, and the number of available resources. For example, $[i_0 \mid (\mathsf{psid}_0, \mathsf{num}_0, I), (\mathsf{psid}_1, \mathsf{num}_1, G) \mid GET \mid r_I \mid 30]$ specifies an intruder maintaining two bursts of protocol sessions, having r_I resources available, and recovering resources 30 time units after spending a resource.

Notice that one could imagine to represent the recovery time t_{rec} as a symbol as well. However, this would mean that it should be constrained by a lower bound, t, *e.g.*, $t_{rec} \geqslant t$. Otherwise, the intruder could, just like the DY intruder, easily deny any service because he would be able to recover instantaneously any resource used. But then, since it is already bounded by a lower-bound, there is not really a need to use a symbol, but use lower-bound, t, directly, as it is the most powerful intruder under the given constraint.

Scheduler. The intruder can start a protocol session burst of size num at any time, provided he has enough resources available to do so. This means that the time when these bursts occur are also kept symbolic. It is the role of the scheduler to keep track of the timeouts and resource recovery of the intruder. For example, whenever a protocol session burst, psid, is initiated at a service sid, a timeout message is inserted in the scheduler. It has the form $\mathsf{sid}, \mathsf{psid} \leftarrow \mathsf{recover} r_i$, where r_i is a symbol specifying the number of resources allocated for psid. Similarly, there is also a type of message for when the intruder can recover an used resource.

The scheduler has the form $[\mathsf{id} \mid \mathrm{msgs}_1 \mid \mathrm{msgs}_2]$ where:

- id is the scheduler unique identifier;
- msgs_1 is a multiset of messages that can be delivered immediately;
- msgs_2 is a multiset of messages that shall be delivered at a future time.

Intuitively, the scheduler enforces the critical configuration in the MSR theory by forcing timeouts whenever they are due and that the intruder recovers resources whenever the recovery time has passed.

System Configurations and Symbolic Operational Semantics. A system configuration consists of some number of service and intruder actors and a single scheduler. Moreover, as already anticipated above, to improve search space, symbolic search is used through Rewriting Modulo SMT [27]. This allows one to perform symbolic search relying on the power of off-the-shelf SMT solvers. There are the following three types of symbols:

- **Time Symbols:** Instead of using concrete values for global time, time symbols are used. Time symbols, ts, may be used in expressions, such as in $\mathrm{ts} + 2.0$;
- **Intruder and Service Resource Symbols:** Instead of using concrete values for intruder and service resources, intruder resource symbols and service resource symbols are used, *e.g.*, r_I in $[i_0 \mid (\mathsf{psid}_0, \mathsf{num}_0, I), (\mathsf{psid}_1, \mathsf{num}_1, G) \mid GET \mid r_I \mid 30]$;
- **Number of Protocol Instance Symbols:** Instead of creating one protocol session at a time, the intruder is allowed to create several instances of a protocol session representing a burst from the intruder, *e.g.*, num_0 in $(\mathsf{psid}_0, \mathsf{num}_0, I)$.

The symbolic rewrite rules accumulate constraints on the values and the search stops whenever the set of accumulated constraints is not satisfiable. Therefore, the system configuration also contains the set of constraints that specify the values that the symbols appearing in the configuration can have. Intuitively, a (symbolic) system configuration denotes a possibly infinite set of (concrete) configurations.

Formally, a system configuration has the following form [players | ts | tcons | rcons] where:

- players is a collection of services, intruders and contains exactly one scheduler;
- ts is a time symbol denoting the current time;
- tcons is a set of arithmetic constraints involving only time symbols, *e.g.*, $\mathrm{ts} \leqslant 10$;
- rcons is a set of arithmetic constraints involving only a number of protocol instances and resource symbols, *e.g.*, $\mathrm{rs}_1 \times \mathsf{num} \leqslant \mathrm{rs}_2$.

Notice that the constraints on time symbols are disjoint from the constraints on the remaining symbols. This means that the SMT-solver has less effort in checking the consistency of the constraints, as it can check the constraints independently.

We specified the rules that rewrite system configurations. Intuitively, rewrite rules can generate new symbols and generate new constraints. We informally describe the implemented rewrite rules. They closely match the rules of resource protocol theories and a subset of the intruder theories, namely, the send and recover rules. Each rule creates a fresh time symbol, ts^{ν} to represent the new global time, and adds a constraint $\mathrm{ts}^{\nu} \geqslant \mathrm{ts}$, where ts is the current time symbol. Moreover, a rewrite rule is only applicable if both sets of symbolic constraints, tcons and rcons, are satisfiable.

We describe the rewrite rules informally:

- **New Protocol Burst Instance Rule:** This rewrite rule creates a new burst of protocol session instances. This rule creates a new protocol instance symbol, num^{ν}, and constrains it according to the number of resources, r_I, available to the intruder and the amount of resources, ri_I, required to initialize the protocol: $\mathsf{num}^{\nu} \times ri_I \leqslant r_I$. Moreover, a new resource symbol, r_I^{ν}, is created for the intruder and it is constrained accordingly: $r_I^{\nu} = r_I - \mathsf{num}^{\nu} \times ri_I$. If the resources of the service are depleted, a flag, $\mathsf{depleted}(\mathrm{ts})$, is inserted in the service configuration, that is, if the resource constraints entail that the number of resources available by the service is less or equal to r_{min}.
- **Protocol Burst Advancement:** This rewrite rule advances the state of a protocol instance burst. (Notice that all bursts are advanced at the same time.) It manages the resources in a similar way as in the rule **New Protocol Burst Instance Rule**;
- **Timeout:** This rewrite rule processes a timeout scheduled in the scheduler by terminating all the timed out protocol instance bursts. New resource symbols are generated and constrained specifying the de-allocation of resources in a similar way as in rule **New Protocol Burst Instance Rule**;
- **Intruder Recover:** This rewrite rule processes a recover message in the scheduler by re-allocating resources to the intruder in a similar way as in rule **New Protocol Burst Instance Rule**.
- **Scheduling a Message**: Recall that the scheduler keeps track of the messages (msgs_1) that are to be processed at the current time and the messages (msgs_2) that are to be processed in the future. This rewrite rule plays the role of moving messages from msgs_2 to msgs_1 whenever the time advances.

Given the rules above, we can use Maude's search engine to check whether a system configuration can be reached containing service whose resources are depleted for some given duration *dur*. This is done by checking whether there is the flag $\mathsf{depleted}(\mathrm{ts}_0)$ and checking whether $\mathrm{ts} - \mathrm{ts}_0 \geqslant dur \wedge \mathsf{tcons}$ is satisfiable, where ts and tcons are the current time and the current set of time constraints.

8.2. Experimental Results

We carried out a collection of experiments involving the scenarios for the Slowloris, Slow-TCAM and TLS Renegotiation attacks. Our focus was to understand how well does our symbolic approach scale. Besides the symbolic reasoning and to further improve performance, we considered bounded model-checking based on the following parameters:

- **Number of Parallel Symbolic Bursts of Protocol Sessions** (pxs)**:** Allowing the intruder to create an unbounded number of sessions increases greatly the size of search space and the complexity of the verification problem. Following our balanced theory approach (Section 3.2), we bound the number of parallel symbolic protocols bursts. That is, there is a bound on the size of the multiset ps in intruder configurations of the form $[\mathsf{id} \mid \mathrm{pxs} \mid \mathsf{ps} \mid r_I \mid t_{rec}]$. Notice that this does not mean that

Table 2

Verification results: Results in terms of number of states and time taken for our machinery to find an attack. The scenarios are parametric where a greater number specifies a scenario where the service is more resilient to DDoS attacks. Search was interrupted after 10 mins and we indicate with – whenever no attack has been found within this time.

	No Bounding		**Bounded** msgs_1		**Bounded** pxs		**Bounded** msgs_1 **and** pxs	
Attack	States	Time (s)	States	Time (s)	States	Time (s)	States	Time (s)
SL [1]	18	0.2	16	0.2	13	0.1	11	0.1
SL [2]	409	7.1	277	6.8	51	0.6	29	0.4
SL [3]	–	–	–	–	775	12	147	6.4
STCAM [2]	17	0.2	15	0.2	12	0.1	9	0.1
STCAM [3]	387	6.7	266	6.1	265	4.6	164	3.9
STCAM [4]	13552	422.1	6946	363.8	10153	310.1	4519	264.9
TLS [1]	50	0.9	36	0.8	22	0.2	14	0.2
TLS [2]	–	–	–	–	5077	127	1199	82.4

the number of parallel sessions is bounded because symbolic protocols bursts still carry the number of instance symbol, that is, there is no bound on the value of num in a symbolic bursts of protocol sessions of the form $(\mathsf{psid}, \mathsf{num}, I)$;

- **Number of Scheduler Messages Processed at a Time** (msgs_1)**:** The state-space is also affected by the number of scheduler messages that can be processed at a given time. Bounding this number reduces the state-space as it reduces inter-leavings caused by choosing the order in which scheduler messages are processed.

As with bounded model-checking, the values of these parameters are not known in advance. So typically, one starts with lower values and increases the bound until either an attack is found or one is satisfied with the evidence collected supporting the security of the services.

Table 2 summarizes our results. For the attacks, we considered different scenarios discussed in Section 4.3 with increasing difficulty for finding the attacks. For the Slowloris scenario, we set different duration parameters (*mdur*, see Definition 6.1) for the DDoS problem definition. For example, **SL [1]** denotes a scenario using the HTTP Get protocol, that is vulnerable to the Slowloris attack, with *mdur* being equal to one timeout of the protocol. The greater the *mdur* the harder it is for the intruder to cause DDoS, as he would need to exhaust the service's resources for a longer period. For the SlowTCAM scenarios, we considered scenarios where more bursts of protocol sessions are required. For example, **STCAM [2]** is a scenario where no attack is possible if there is only one burst of protocol sessions, but there is an attack if there are two parallel burst of protocol sessions. Finally, for the scenarios involving the TLS-renegotiation attack, we increased the CPU capacity of the attacked services. For example, **TLS [2]** specifies that the service has twice more processing power than the intruder.

The Slowloris attack is discovered using different values for *mdur* up to 3 times the protocol timeout. This information can be used by specifiers to build defenses to mitigate such attacks. Similarly, for the SlowTCAM attacks, our machinery was able to discover non-trivial attacks where the intruder maintains protocol sessions using to up to 4 parallel bursts of protocol sessions. This information can be used by specifiers to generalize attacks to greater instances and investigate adequate defenses. Finally, for the TLS

renegotiation attack, our experiments were able to determine up to what type of intruder a service can be secure against a DDoS attack.

Regarding the effect of bounding the number of messages or the number of parallel protocol sessions, it seems that bounding the number of parallel protocol sessions has a greater improvement to search, but not always as witnessed by the experiments for **STCAM [4]**.

For determining which bounds were necessary, we carried out a sequence of experiments with increasing bounds. For most experiments, the bound of 1 for msgs_1 and 1 for pxs was enough. However, for the STCAM experiments, we needed to increase the bound of pxs. For example, our machinery could only discover an attack when the bound on pxs was set to 4 or greater. This may provide hints to engineers that they should be monitoring multiple parallel bursts to defend against attacks.

9. Conclusions and Related Work

This paper introduces a new framework for analyzing the security of systems against DoS and related resource and timing attacks, which allows a finer analysis then the existing verification models. With respect to formalization of time, the paper builds on [24, 53–55] and introduces a uniform and extensible framework for expressing a wide range of timing properties of protocols enabling the investigation of the complexity of different verification problems. Thus, this work is complementary to the related works that focus on more limited languages in order to automate analyses.

The framework also allows reasoning about service's and intruder's resources and service timeouts. The power of the model is illustrated with a number of examples and intruder models. The complexity of the DoS problem and the leakage problem is studied. Finally, the use of Rewriting Modulo SMT for efficiently automating the verification task is demonstrated.

While inspired by the work [21], the model mentions time explicitly, enabling the reasoning about service timeouts, essential for discovering vulnerabilities to Slow and Asymmetric Attacks. This leads to a more refined definition of DoS attacks than the one proposed in [21]. The DoS attack is defined as exhausting the target service's resource for a certain period of time, reflecting the intuitive notion of a DoS attack, which is to take down a service temporarily or indefinitely. Finally, the proposed model can also specify time-based counter-measures that issue timeouts whenever some condition is applicable.

In [20] a taxonomy of DoS attacks and defense mechanisms is presented. Some of the relevant security issues have clearly been covered in the model presented in this paper. For example, attack rate dynamics issue resulting in constant or variable rate attacks is captured by recovery of resources within associated time through R and Rec facts. However, the model did not go into details of all issues identified in [20], *e.g.*, means used to prepare and perform the attack (manual, semi-automatic and automatic DDoS attacks) nor into source address validity issue (spoofed vs. valid IPs), impact on the victim (recoverable vs. non-recoverable attacks) etc. Nevertheless, additional details could be introduced in the model in relation to such security issues. By including various resources of the service in the specification, one is able to represent and differentiate between flooding attacks and more sophisticated semantic attacks. This also applies to modelling of DoS defense mechanisms.

MSR frameworks [22–24, 30] have been proposed for security verification. However, they relate to authentication and secrecy-related problems, to distance-bounding protocols and not to DoS attacks, which is the main goal here. This paper also builds on the work of [23] which considers bounded memory intruders. Intruders proposed here can be bounded with respect to a wider range of types of resources.

Authors in [15] demonstrate a model checking technique, called measure checking, for finding amplification attacks on VoIP using rewriting logic (implemented in Maude). They do not provide, however,

general intruder models, nor the corresponding complexity results. Finally, this paper also considers a wider range of attacks, such as slow and CPU exhaustion attacks.

A number of other frameworks have been developed for the verification of timing properties of systems. Early examples include [56–58]. Basin et.al [59] and Cremers et.al [60] present a formalism for representing and analyzing cyber-physical security protocols that is implemented in Isabelle/HOL. They model physical properties of communication, location, and time. Similar to the approach taken here, both honest players and intruders are subject to physical constraints. Cheval et al. [61] present a decidability result relating to timing attacks in security protocols. The result is based on the reduction of time-trace equivalence to length-trace equivalence, and is applied, in particular, on verification of privacy properties. Similar to the parametrized action execution, formalized using SPEC function, they also deal with computation time w.r.t. to length of inputs. The model allows representation of other “side-channel" resources that can be leaked by the execution, such as power consumption. It remains to be investigated whether such an approach may be applicable to the general protocol theories with varied correct execution time. A benefit of formalization in the Timed MSR is the ability to leverage a variety of complexity results developed for different fragments as illustrated in the previous sections.

Timed automata (TA) [62] have been used for the verification of many systems involving real-time. The framework proposed in this paper is more closely related to security, as resources, intruder models, and DoS problems are considered. This is obtained by means of adding explicit notions of timeouts and also of resources to rules. Using first-order rules in MSR versus finite number of states and rules in TA, further affects comparisons of complexity results. Also, the model of this paper has the additional feature of non-critical traces, distinguishing among potential reachability solutions only those traces that have no negative properties, *i.e.*, are non-critical.

Protocol verification in [63], using timed automata, also involves timing aspects of security protocols in the presence of DY intruder, as well as automated tools. They investigate timed authentication properties, based on expected time intervals for completion of successful protocol sessions. Such an approach may not be as adequate for our DoS problems, as the service resources may also be consumed by sessions that have not successfully completed. Also, this paper considers more general protocol theories with varied execution time of correct sessions, due to, *e.g.*, possibility of sending incomplete headers in a correct protocol execution, or even loops in protocols which are, differently from [63], allowed in the protocol theories etc. Also, we investigate the computational complexity related to the verification.

The paper [45] introduces a timed protocol language and addresses the issue of timed intruder models, showing that one DY intruder per honest player is sufficient. This work was built on earlier formalizations in Timed MSR [53, 64] and in turn has suggested new modeling challenges addressed in the present paper. We believe it may be possible to extend the verification model from a concrete topology of agents and intruders of [30] to general topologies by combining the models with SMT solvers, similarly to [45].

Also, we expect that in our model we are able to capture a larger class of security problem closely related to DoS attacks, including other types of attacks that may include a DoS as a component, attempts to prevent a particular individual from accessing a service, attempts to disrupt service to a specific system or person, as well as poor service performance resulting from intruder interference.

Another direction for application of the model could be on the analysis of Distance-Bounding protocols [55, 65, 66], possibly by extending the model with probabilities. One of the ways of such probabilistic extensions of the models might involve branching actions introduced in [35]. Statistical Model Checking has also been used to investigate the effectiveness of attack defense [55, 67–71]. We believe that the search space reduction due to the use of symbolic search can improve the performance of these methods for the verification of defense.

Finally, we would like to investigate how different resource-bounded intruder models can be compared. For example, whether it is possible to define a partial order relating the strength of intruders. This is left for future work.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions. Part of this work was done during the visits to the University of Pennsylvania by Alturki, Ban Kirigin, Kanovich, Nigam, and Talcott, which were partially supported by ONR grant N00014-15-1-2047 and by the University of Pennsylvania. Ban Kirigin is supported in part by the Croatian Science Foundation under the project UIP-05-2017-9219. The work of Max Kanovich was partially supported by EPSRC Programme Grant EP/R006865/1: "Interface Reasoning for Interacting Systems (IRIS)." Nigam is partially supported by NRL grant N0017317-1-G002, and CNPq grant 303909/2018-8. Scedrov is partially supported by ONR grants N00014-20-1-2635 and N00014-18-1-2618. Talcott was partially supported by ONR grants N00014-15-1-2202 and N00014-20-1-2644, and NRL grant N0017317-1-G002. Nigam has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 830892

References

[1] R.M. Needham and M.D. Schroeder, Using encryption for authentication in large networks of computers, *Commun. ACM* **21**(12) (1978), 993–999. doi:http://doi.acm.org/10.1145/359657.359659.
[2] G. Lowe, Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR, in: *TACAS*, 1996, pp. 147–166.
[3] D. Dolev and A. Yao, On the security of public key protocols, *IEEE Transactions on Information Theory* **29**(2) (1983), 198–208.
[4] slowloris, http://ha.ckers.org/slowloris/, 2013.
[5] r-u-dead-yet, https://code.google.com/p/r-u-dead-yet/, 2013.
[6] slowread, https://code.google.com/p/slowhttptest/, 2013.
[7] E. Cambiaso, G. Papaleo, G. Chiola and M. Aiello, Mobile executions of Slow DoS attacks, *Logic Journal of IGPL* (2015), 54–67.
[8] T.A. Pascoal, Y.G. Dantas, I.E. Fonseca and V. Nigam, Slow TCAM Exhaustion DDoS Attack, in: *ICT Systems Security and Privacy Protection (IFIP SEC)*, 2017.
[9] T.A. Pascoal, I.E. Fonseca and V. Nigam, Slow Denial-of-Service Attacks on Software Defined Networks, *Computer Networks* **173** (2020), 107223. doi:10.1016/j.comnet.2020.107223.
[10] E. Cambiaso, G. Papaleo, G. Chiola and M. Aiello, Slow DoS attacks: definition and categorisation, *International Journal of Trust Management in Computing and Communications* **1**(3–4) (2013), 300–319.
[11] M.O.O. Lemos, Y.G. Dantas, I. Fonseca, V. Nigam and G. Sampaio, A Selective Defense for Mitigating Coordinated Call Attacks, in: *34th Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, 2016.
[12] B. Sullivan, Application-Level Denial of Service Attacks and Defenses, 2011, [Online; Accessed 16-December-2016]. https://media.blackhat.com/bh-dc-11/Sullivan/BlackHat_DC_2011_Sullivan_Application-Level_Denial_of_Service_Att_&_Def-wp.pdf.
[13] O. Olivo, I. Dillig and C. Lin, Detecting and Exploiting Second Order Denial-of-Service Vulnerabilities in Web Applications, in: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, ACM, New York, NY, USA, 2015, pp. 616–628. ISBN 978-1-4503-3832-5. doi:10.1145/2810103.2813680.
[14] R. Sparks, S. Lawrence, A. Hawrylyshen and B. Campen, Addressing an Amplification Vulnerability in Session Initiation Protocol (SIP) Forking Proxies, *Request for Comments*, IETF, 2008. http://www.ietf.org/rfc/rfc5393.txt.
[15] R. Shankesi, M. AlTurki, R. Sasse, C.A. Gunter and J. Meseguer, Model-Checking DoS Amplification for VoIP Session Initiation, in: *ESORICS*, 2009, pp. 390–405.
[16] E. Cambiaso, G. Papaleo and M. Aiello, SlowDroid: Turning a Smartphone into a Mobile Attack Vector, in: *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, IEEE, 2014, pp. 405–410.
[17] IETF, [TLS] SSL Renegotiation DOS, Available at https://www.ietf.org/mail-archive/web/tls/current/msg07553.html.

[18] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, SIP: Session Initiation Protocol, *Request for Comments*, IETF, 2002, Updated by RFCs 3265, 3853, 4320, 4916, 5393. http://www.ietf.org/rfc/rfc3261.txt.
[19] P. Gupta and V. Shmatikov, Security Analysis of Voice-over-IP Protocols, in: *20th IEEE Computer Security Foundations Symposium, Venice, Italy*, IEEE Computer Society, 2007, pp. 49–63.
[20] J. Mirkovic and P. Reiher, A taxonomy of DDoS attack and DDoS defense mechanisms, *ACM SIGCOMM Computer Communication Review* **34**(2) (2004), 39–53.
[21] C.A. Meadows, A Cost-Based Framework for Analysis of Denial of Service Networks, *Journal of Computer Security* **9**(1/2) (2001), 143–164. http://content.iospress.com/articles/journal-of-computer-security/jcs143.
[22] 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) (2004), 247–311.
[23] M.I. Kanovich, T. Ban Kirigin, V. Nigam and A. Scedrov, Bounded memory Dolev-Yao adversaries in collaborative systems, *Inf. Comput.* **238** (2014), 233–261. doi:10.1016/j.ic.2014.07.011.
[24] M.I. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov and C.L. Talcott, Time, computational complexity, and probability in the analysis of distance-bounding protocols, *Journal of Computer Security* **25**(6) (2017), 585–630. doi:10.3233/JCS-0560.
[25] ReQTimeOut, https://httpd.apache.org/docs/2.4/mod/mod_reqtimeout.html, 2014.
[26] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and C. Talcott, *All About Maude: A High-Performance Logical Framework*, LNCS, Vol. 4350, Springer, 2007.
[27] C. Rocha, J. Meseguer and C.A. Muñoz, Rewriting modulo SMT and open system analysis, *J. Log. Algebr. Meth. Program.* **86**(1) (2017), 269–297. doi:10.1016/j.jlamp.2016.10.001.
[28] M.P. for DDoS Verification, https://github.com/viveknigam/boundedIntruder, 2020.
[29] A.A. Urquiza, M.A. AlTurki, M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov and C. Talcott, Resource-Bounded Intruders in Denial of Service Attacks, in: *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, IEEE, 2019, pp. 382–38214.
[30] M.A. Alturki, T. Ban Kirigin, M. Kanovich, V. Nigam, A. Scedrov and C. Talcott, A Multiset Rewriting Model for Specifying and Verifying Timing Aspects of Security Protocols, in: *Foundations of Security, Protocols, and Equational Reasoning*, Springer, 2019, pp. 192–213.
[31] IETF, The Transport Layer Security (TLS) Protocol Version 1.3, Available at https://tools.ietf.org/html/rfc8446.
[32] I. Cervesato, N.A. Durgin, P. Lincoln, J.C. Mitchell and A. Scedrov, A Meta-Notation for Protocol Analysis, in: *CSFW*, 1999, pp. 55–69.
[33] H.B. Enderton, *A mathematical introduction to logic*, Academic Press, 1972, pp. I–XIII, 1-295. ISBN 978-0-12-238450-9.
[34] M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov and C. Talcott, Timed Multiset Rewriting and the Verification of Time-Sensitive Distributed Systems, in: *14th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2016.
[35] M.I. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, C.L. Talcott and R. Perovic, A rewriting framework and logic for activities subject to regulations, *Mathematical Structures in Computer Science* **27**(3) (2017), 332–375. doi:10.1017/S096012951500016X.
[36] P. Rowe, Policy compliance, confidentiality and complexity in collaborative systems, PhD thesis, University of Pennsylvania, 2009.
[37] M. Kanovich, P. Rowe and A. Scedrov, Policy Compliance in Collaborative Systems, in: *CSF '09: Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*, IEEE Computer Society, Washington, DC, USA, 2009, pp. 218–233. ISBN 978-0-7695-3712-2. doi:http://dx.doi.org/10.1109/CSF.2009.19.
[38] S. Escobar, C. Meadows and J. Meseguer, Maude-NPA: Cryptographic protocol analysis modulo equational properties, in: *Foundations of Security Analysis and Design V*, Springer, 2009, pp. 1–50.
[39] S.F. Doghmi, J.D. Guttman and F.J. Thayer, Searching for shapes in cryptographic protocols, in: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2007, pp. 523–537.
[40] B. Blanchet et al., An efficient cryptographic protocol verifier based on prolog rules., in: *csfw*, Vol. 1, Citeseer, 2001, pp. 82–96.
[41] M. Backes, B. Pfitzmann and M. Waidner, A composable cryptographic library with nested operations, in: *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, October 27-30, 2003*, S. Jajodia, V. Atluri and T. Jaeger, eds, ACM, 2003, pp. 220–230. ISBN 1-58113-738-9. doi:10.1145/948109.948140.
[42] R. Canetti, Universally composable security: A new paradigm for cryptographic protocols, in: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, IEEE, 2001, pp. 136–145.
[43] B. Blanchet, A computationally sound mechanized prover for security protocols, *IEEE Transactions on Dependable and Secure Computing* **5**(4) (2008), 193–207.
[44] M.I. Kanovich, P. Rowe and A. Scedrov, Collaborative Planning with Confidentiality, *J. Autom. Reasoning* **46**(3–4) (2011), 389–421.

[45] V. Nigam, C. Talcott and A.A. Urquiza, Towards the Automated Verification of Cyber-Physical Security Protocols: Bounding the Number of Timed Intruders, in: *European Symposium on Research in Computer Security (ESORICS)*, 2016.
[46] T. Chothia and V. Smirnov, A traceability attack against e-passports, in: *International Conference on Financial Cryptography and Data Security*, Springer, 2010, pp. 20–34.
[47] OpenFlow, *Open Networking Foundation (ONF)*, https://www.opennetworking.org/. Accessed in: 02 de Setembro de 2016.
[48] CERT Coordination Center, Denial of Service Attacks, www.cs.columbia.edu/~danr/courses/6761/Fall00/week14/cert.ps.
[49] Department of Homeland Security, Understanding Denial-of-Service Attacks, https://www.us-cert.gov/ncas/tips/ST04-015.
[50] M. Médard and A. Sprintson, Elsevier, Academic Press, 2012. doi:doi.org/10.1016/C2009-0-30680-7.
[51] U. Lee, J.-S. Park, J. Yeh, G. Pau and M. Gerla, Code torrent: content distribution using network coding in vanet, in: *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, ACM New York, NY, 2006, pp. 1–5.
[52] M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov and C. Talcott, Compliance in Real Time Multiset Rewriting Models, Available at https://arxiv.org/abs/1811.04826.
[53] M.I. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov and C.L. Talcott, Towards Timed Models for Cyber-Physical Security Protocols, 2014, Available in Nigam's homepage.
[54] M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov and C. Talcott, Can we mitigate the attacks on Distance-Bounding Protocols by using challenge-response rounds repeatedly?, in: *FCS*, 2016.
[55] M.A. AlTurki, M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov and C. Talcott, Statistical Model Checking of Distance Fraud Attacks on the Hancke-Kuhn Family of Protocols, in: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, ACM, 2018, pp. 60–71.
[56] G. Bella and L.C. Paulson, Kerberos Version 4: Inductive Analysis of the Secrecy Goals, in: *Computer Security - ESORICS 98, 5th European Symposium on Research in Computer Security, Louvain-la-Neuve, Belgium, September 16-18, 1998, Proceedings*, 1998, pp. 361–375. doi:10.1007/BFb0055875.
[57] N. Evans and S. Schneider, Analysing Time Dependent Security Properties in CSP Using PVS, in: *Computer Security - ESORICS 2000, 6th European Symposium on Research in Computer Security, Toulouse, France, October 4-6, 2000, Proceedings*, 2000, pp. 222–237.
[58] R. Gorrieri, E. Locatelli and F. Martinelli, A Simple Language for Real-time Cryptographic Protocol Analysis, in: *Proceedings of the 12th European Conference on Programming*, ESOP'03, Springer-Verlag, Berlin, Heidelberg, 2003, pp. 114–128. ISBN 3-540-00886-1. http://dl.acm.org/citation.cfm?id=1765712.1765723.
[59] 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) (2011), 16.
[60] C.J.F. Cremers, K.B. Rasmussen, B. Schmidt and S. Capkun, Distance Hijacking Attacks on Distance Bounding Protocols, in: *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, 2012, pp. 113–127. doi:10.1109/SP.2012.17.
[61] V. Cheval and V. Cortier, Timing attacks in security protocols: symbolic framework and proof techniques, in: *International Conference on Principles of Security and Trust*, Springer, 2015, pp. 280–299.
[62] R. Alur and D.L. Dill, A theory of timed automata, *Theoretical Computer Science* **126**(2) (1994), 183–235. doi:https://doi.org/10.1016/0304-3975(94)90010-8. http://www.sciencedirect.com/science/article/pii/0304397594900108.
[63] G. Jakubowska and W. Penczek, Modelling and checking timed authentication of security protocols, *Fundamenta Informaticae* **79**(3–4) (2007), 363–378.
[64] M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov and C. Talcott, Discrete vs. Dense Times in the Analysis of Cyber-Physical Security Protocols, in: *Principles of Security and Trust - 4th International Conference, POST*, 2015, pp. 259–279.
[65] C.A. 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*, 2007, pp. 279–298.
[66] S. Mauw, Z. Smith, J. Toro-Pozo and R. Trujillo-Rasua, Distance-bounding protocols: Verification without time and location, in: *2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2018, pp. 549–566.
[67] M. AlTurki, J. Meseguer and C.A. Gunter, Probabilistic Modeling and Analysis of DoS Protection for the ASV Protocol, *Electr. Notes Theor. Comput. Sci.* **234** (2009), 3–18.
[68] J. Eckhardt, T. Mühlbauer, J. Meseguer and M. Wirsing, Statistical Model Checking for Composite Actor Systems, in: *WADT*, 2012, pp. 143–160.
[69] J. Eckhardt, T. Mühlbauer, M. AlTurki, J. Meseguer and M. Wirsing, Stable Availability under Denial of Service Attacks through Formal Patterns, in: *FASE*, 2012, pp. 78–93.
[70] Y.G. Dantas, V. Nigam and I.E. Fonseca, A Selective Defense for Application Layer DDoS Attacks, in: *IEEE JISIC 2014*, 2014, pp. 75–82. doi:10.1109/JISIC.2014.21.
[71] M.O.O. Lemos, Y.G. Dantas, I.E. Fonseca and V. Nigam, On the accuracy of formal verification of selective defenses for TDoS attacks, *J. Log. Algebr. Meth. Program.* **94** (2018), 45–67. doi:10.1016/j.jlamp.2017.09.001.