Cheap Intrusion-Tolerant Protection for CRUTIAL Things

Alysson Neves Bessani
Paulo Sousa
Miguel Correia
Nuno Ferreira Neves
Paulo Verissimo

DI–FCUL TR–2009–14

June 2009

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749–016 Lisboa
Portugal

Technical reports are available at `http://www.di.fc.ul.pt/tech-reports`. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

Cheap Intrusion-Tolerant Protection for CRUTIAL Things

Alysson Neves Bessani Paulo Sousa Miguel Correia
Nuno Ferreira Neves Paulo Verissimo
LASIGE, Faculdade de Ciências da Universidade de Lisboa – Portugal

June 2009

Abstract

Today's critical infrastructures like the *power grid* are essentially physical processes controlled by computers connected by networks. They are usually as vulnerable as any other interconnected computer system, but their failure has a high socio-economic impact. The report describes a new construct for the protection of these infrastructures, based on distributed algorithms and mechanisms implemented between a set of devices called CIS. CIS collectively ensure that incoming/outgoing traffic satisfies the security policy of an organization facing accidents and attacks. However, they are not simple firewalls but distributed protection devices based on a sophisticated access control model and designed with intrusion-tolerant capabilities. The report discusses the rationale behind the use of CIS to improve the resilience of critical infrastructures, and it describes and evaluates two CIS implementations, one using physical replicas, and another using virtual machine (VM) based replicas. Our intrusion-tolerant solution is cheap in four different ways: it uses less replicas than other intrusion-tolerant services; it does not requires expensive consensus protocols; the performance overhead is minimal; and it can be deployed in a single physical machine through the use of VM technology.

Keywords: Byzantine Fault Tolerance, Critical Infrastructures, Firewalls, Intrusion Tolerance, Security.

1 Introduction

Critical infrastructures like the power, water and gas distribution networks have a fundamental role in modern life. These infrastructures are essentially physical/mechanical processes controlled electronically. The control systems, usually called SCADA (Supervisory Control and Data Acquisition) or PCS (Process Control System), are composed by computers interconnected by computer networks [35, 47].

In recent years these systems evolved in several aspects that greatly increased their exposure to cyber-attacks coming from the Internet. Firstly, the computers, networks and protocols in those control systems are no longer proprietary but standard PCs and networks (such as wired and wireless Ethernet), and the protocols are often encapsulated on top of TCP/IP. Secondly, these networks are usually connected to the Internet indirectly through the corporate network or to other networks using modems and data links. Thirdly, several infrastructures are being interconnected creating a complexity that is hard to manage [44].

Therefore these infrastructures have a level of vulnerability similar to other systems connected to the Internet, but the socio-economic impact of their failure can be huge. This scenario, reinforced by several recent incidents [11, 43, 49], is generating a great concern about the security of these infrastructures, especially at government level.

A reference architecture was recently proposed to protect critical infrastructures, in the context of the CRUTIAL[1] EU-IST project [24, 47]. The whole infrastructure architecture is modeled as a WAN-of-LANs. This topology allows simple solutions to hard problems such as legacy control subnetworks, and interconnection of critical and non-critical traffic. Typically, a critical information infrastructure is formed by facilities, like power transformation substations or corporate offices, modeled as collections of LANs and interconnected by a wider-area network, modeled as a WAN, in the WAN-of-LANs model.

This architecture allows defining realms with different levels of trustworthiness. In this report we are interested in the problem of protecting realms from one another, i.e., a LAN from another LAN or from the WAN. However, given the ease of defining LANs in today's IP architectures (e.g., through virtual switched LANs), there is virtually no restriction to the level of granularity of protection domains, which can go down to a single host. In consequence, our model and architecture allow us to deal both with outsider threats (protecting a facility from the Internet) and insider threats (protecting a critical host from other hosts in the same physical facility, by locating them in different LANs).

[1] Critical UTility InfrastructurAL Resilience: `http://crutial.cesiricerca.it`.

Protection of LANs from the WAN or other LANs is made by a device called *CRUTIAL Information Switch* (CIS). A CIS provides two basic services: the *Protection Service (PS)* and the *Communication Service (CS)*. The PS ensures that the incoming and outgoing traffic in/out of the LAN satisfies the security policy of the infrastructure. The CS supports secure communication between CIS and, ultimately, between LANs. Moreover, one of the most challenging goals of the CS is to improve the timeliness of the communication between LANs, even under network failures (due to (D)DoS attacks, for instance) [19]. Although the CIS supports these two services, this report presents only the protection service, not the communication service. Therefore, from now on "the CIS" means "the CIS *Protection Service*". Figure 1 ilustrates the use of CIS protecting several LANs of a critical infrastructure.

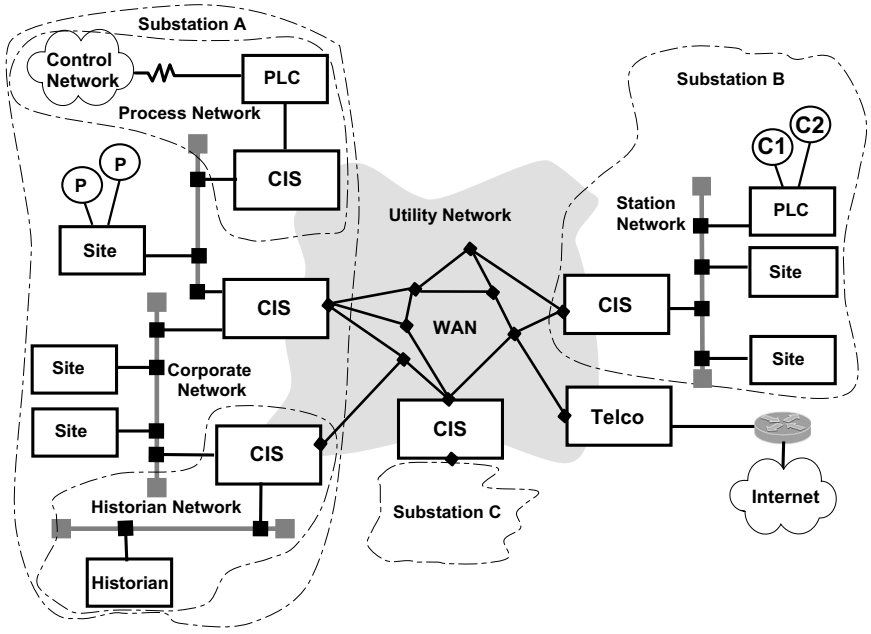


Figure 1: WAN-of-LANs connected by CIS.

A CIS can not be a simple firewall since that would put the infrastructure at most at the level of security of current Internet systems, which is not acceptable since intrusions in those systems are constantly being reported [25, 29]. Instead, a CIS is a distributed protection device based on a sophisticated access control model and designed with intrusion-tolerant capabilities. The main characteristics of the CIS are the following. Firstly, it has similarities to a *distributed firewall* [5],

since CIS can be deployed not only on the network border but inside the networks to better protect critical equipment. Secondly, the CIS uses a *rich access control model* that takes into account the involvement of different organizations and allows the access control rules to depend on context information [1]. Thirdly, the CIS is *intrusion-tolerant* [23, 48], which means that it operates correctly even if there are intrusions in some of its components.

The report is essentially about the last topic, the design of an intrusion-tolerant CIS. The CIS is replicated in a set of machines. If there are intrusions in some of the replicas, the CIS masks these faults and follows its specification. Moreover, CIS intrusion-tolerance mechanisms are *cheap* in four different ways: $2f + 1$ replicas are sufficient to tolerate f Byzantine servers; it does not require expensive consensus protocols (and consequently, does not need any timing assumption); the performance overhead related to a non intrusion-tolerant design is minimal; and we describe a VM-based prototype that uses a single physical machine.

Several intrusion-tolerant services have been proposed in the literature (e.g., storage [13, 23, 37], certification authorities [40, 51], and DNS [12]), either based on Byzantine quorum systems (BQS) [36, 51] or state machine replication (SMR) [3, 13, 41]. However, the CIS design presents two very interesting challenges that make it essentially different from those services. The first is that a firewall-like component has to be transparent to protocols that pass through it, so it can not modify the protocols themselves to obtain intrusion tolerance. This also means that recipient nodes will ignore any internal CIS intrusion tolerance mechanisms, and as such they cannot protect themselves from messages forwarded by faulty replicas not satisfying the security policy. The report shows that these two challenges are not trivial and presents a solution that is based neither on BQS nor on SMR.

Although the proposed service is designed for critical infrastructures protection, many other systems could also be protected by a CIS. However, the design takes into consideration critical infrastructures since *(i.)* the solution must comply with legacy/standard components that cannot be easily replaced in critical infrastructures; and *(ii.)* the solution we propose is more costly and has less packet processing capacity than common (non-replicated) firewalls, so it is more adequate for

protecting low-traffic critical facilities than high-bandwidth corporate networks.

The report has the following two main contributions:

1. It presents the design of an intrusion-tolerant firewall-like component adequate for protecting networked infrastructures and services from outsider and insider accidental or malicious threats, such as critical information infrastructures. This design is formalized, proved correct, implemented (two prototypes) and evaluated. To our knowledge it is the *first proposal* for an intrusion-tolerant firewall that can deal with generic protocols.

2. It presents the first implementation of an intrusion-tolerant service in a single machine, using logical replication based on virtual machines. The evaluation of this prototype has shown that the limited resources of a single machine constrain the performance of the service, but lead to an economic solution. However, more hardware-lavish (and more performant) solutions are not precluded, as it will be shown.

2 The CIS Protection Service

The CIS works mainly like a firewall. It captures packets, checks if they satisfy the security policy being enforced, and forwards the approved packets, discarding those that do not satisfy the policy. However, several other characteristics of the CIS make it a unique protection device. These characteristics are presented in this section.

Distributed firewall. CIS can be used in a distributed way, enforcing the same policies in different points of the network. An extreme case in the SCADA/PCS side is to have a CIS in each gateway interconnecting each substation network, and a CIS specifically protecting each critical component of the SCADA/PCS network. The concept is akin to using firewalls to protect hosts instead of only network borders [5], and is specially useful for critical information infrastructures given their complexity and criticality, with many routes into the control network that can not be easily closed (e.g., Internet, dial-up modems, VPNs, wireless access points) [11].

Application-level firewall. Critical infrastructures have many legacy components that were designed without security in mind, and thus do not employ security mechanisms like access control and cryptography [21]. Since these security mechanisms are not part of the SCADA/PCS protocols and systems, which *must still be protected*, protection must be deployed in some point between the infrastructure and the hosts that access it. The CIS has to inspect and evaluate the messages considering application-level semantics because, as already said, the application (infrastructure) itself does not verify it.

Rich access control model. Besides the capacity to inspect application-level messages, the CIS needs to support a rich access control policy that takes into account the multi-organizational nature of the critical infrastructures as well as their different operational states. Taking the Power System as an example, there are several companies involved in generation, transmission and distribution of energy, as well as regulatory agencies, and several of these parties can execute operations in the power grid. Moreover, almost all power system operation is based on a classical state model of the grid [22]. In each state of this model, specific actions must be taken (e.g., actions defined in a defense plan, to avoid or recover from a power outage) and many of these actions are not allowed in other states (e.g., a generator can not be separated automatically when the power grid is in its normal state). These two complex facets of access control in critical infrastructures require more elaborated models than basic discretionary, mandatory, or role-based access control. To deal with this, in the architecture of CRUTIAL we adopt a more elaborated model, OrBAC (Organization Based Access Control) [1]. It allows the specification of security policies containing permissions, prohibitions, obligations and recommendations, taking into account the role of the subject, who is part of an organization, the action it wants to execute, the target object of this action, and the context in which it is executed. An example: *"In context 'emergency', operators from company C can execute maintenance operations on device D."*

Intrusion-tolerant firewall. As discussed in the introduction, the level of security of current systems connected to the Internet is not adequate for the infrastructures we are concerned with, given

their criticality. To improve the security and dependability of the CIS, it is designed to be intrusion-tolerant [48]: it is replicated in n machines and follows its specification as long as at most f of these machines are attacked and have their behavior corrupted. Obviously, such intrusion tolerance is only useful if there is independence in the way machines are corrupted. This independence of the corruptions or intrusions requires that the machines do not share the same vulnerabilities, since an intrusion is always the result of an attack that activates a vulnerability (or more). The usually accepted way to enforce this property is by having diversity in the machines [33, 38]. Therefore, the intrusion-tolerant CIS is designed with diversity in mind.

In this report, we address the problem of designing an intrusion-tolerant distributed firewall. Other complex questions related with the CRUTIAL security architecture, like policy dissemination and consistency between different CIS in the same security domain, were left as future work.

3 CIS Intrusion Tolerance

In this section we describe the design of the intrusion-tolerant CIS, starting with the design rationale, then define the algorithms it executes, and finally we prove their correctness.

3.1 Design Rationale

To understand the *rationale of the design* of the intrusion-tolerant CIS, consider the problem of implementing a replicated firewall between a non-trusted WAN and the trusted LAN that we want to protect. Further assume that we wish to ensure that only the correct messages (according to the deployed policy) go from the WAN side, through the CIS, to the *station computers*[2] in the LAN. A first problem is that the traffic has to be received by all n replicas, instead of only 1 (as in a normal firewall), so that messages can be evaluated by all replicas. A second problem is that up to f replicas can be faulty and behave maliciously, both towards other replicas and towards the station computers.

[2]Station computers in SCADA/PCS networked systems are the front-ends of control devices.

Our solution to the first problem is to use a device (e.g., an Ethernet hub) to broadcast the traffic to all replicas. These verify whether the messages comply with the OrBAC policy and do a vote, approving the messages if and only if at least $f+1$ different replicas vote in favor. A message approved by the CIS is then forwarded to its destination by a randomly selected replica, so there is no unnecessary traffic multiplication inside the LAN. The way we deal with omissions in the broadcast is addressed later in the report.

The second problem is usually addressed with masking protocols of the Byzantine type, which extract the correct result from the n replicas, despite f maliciously faulty: only messages approved by $f+1$ replicas should go through (one of which must be correct since at most f can be faulty). Since the result must be sent to the station computers, either it is consolidated at the source, or at the destination.

The simplest and most usual approach is to implement a front-end in the destination host that accepts a message if: (1) $f+1$ different replicas send it; or (2) the message has a certificate showing that $f+1$ replicas approve it [8]; or (3) the message has a signature generated by $f+1$ replicas using a threshold cryptography scheme [20]. These solutions would imply modifying the hosts' software. However, modifying the software of the SCADA/PCS system can be complicated, and the traffic inside the protected LAN would be multiplied by n in certain cases (every replica would send the message to the LAN), so this solution is undesirable.

So we should turn ourselves to consolidation at the source, and sending *only one, but correct, forwarded message*, in a way similar to an active replication scheme of Delta-4 [14]. However, what is innovative here is that source-consolidation mechanisms should be transparent to the (standard) station computers. Moreover, a faulty replica has access to the LAN (contrarily to the proposal of [14] where only the fail-silent adapters had access to the LAN) so it can send incorrect traffic to the station computers, which typically can not distinguish faulty from correct replicas. This makes consolidation at the source a hard problem.

The solution to the second problem (the existence of up to f faulty replicas) lies on using IPSEC [31], a set of standard protocols that are expected to be generalized in SCADA/PCS sys-

tems, according to best practice recommendations from expert organizations and governments [43]. Henceforth, we assume that the IPSEC Authentication Header (AH) protocol [30] runs both in the station computers and in the CIS replicas. The basic idea is that station computers will only accept messages with a valid IPSEC/AH Message Authentication Code (MAC), which can only be produced if the message is approved by $f+1$ different replicas. However, IPSEC/AH MACs are HMACs (*Keyed-Hashing for Message Authentication* [32]), generated using a shared key[3] and a hash function, so it is not possible to use threshold cryptography. As the attentive reader will note, the shared key storage becomes a vulnerability point that can not be overlooked in a high resilience design, therefore, *there must be some secure component that stores the shared key and produces MACs for messages approved by* $f+1$ *replicas*.

The requirement in the previous paragraph implies a set of trustworthy (secure) components immersed in non-trustworthy (Byzantine-on-failure) environment. Recent research points to the need for representing these scenarios where different fault models coexist, through hybrid (and not homogeneous) distributed system models and architectures. In such architectures, *stronger* components also nick-named *wormholes*, provide services to the rest of the system following weaker assumptions, through a well-defined interface [45].

Figure 2 represents the intrusion-tolerant CIS architecture. Local wormholes (represented by the small W boxes) provide services for a secure voting protocol that produces a MAC for a message if at least $f+1$ replicas approved it. Each CIS replica is deployed in a different operating system (e.g., Linux, FreeBSD, Windows XP), and the operating systems are configured to use different passwords and different internal firewalls (e.g., iptables, ipf). A second traffic replication device (see figure, right hand side) is used precisely for the replicas to receive whatever the others send to the LAN. This enables us to implement controls to reduce the probability of a message being forwarded by more than one replica.

The CIS does not provide exactly-once semantics, i.e., messages can be lost when the traffic is high and the reception buffers of the replicas become full. This is not different from regular

[3] We assume that IPSEC/AH is used with manual key management [31].

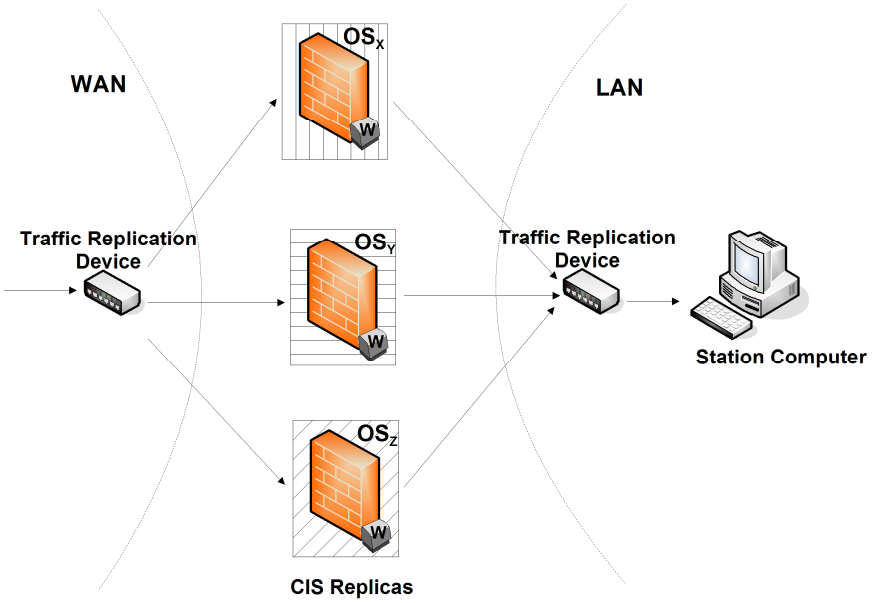


Figure 2: Intrusion-tolerant CIS architecture.

firewalls, except that the CIS operation is more complex so the throughput is expected to be lower. However, it is important to notice that almost all wide area control protocols, and specially the ones designed for power systems like ICCP [27] and IEC 61850 [28], are designed on top of TCP, which ensures reliable communication even if the network (in this case the CIS) loses messages.

3.2 System Model

The system is composed by n CIS replicas $CIS = \{CIS_1, ..., CIS_n\}$. These replicas are deployed in the intersection between the WAN and the LAN in such a way that all data crossing the boundaries of one of these networks must pass through the CIS. The hybrid system model encompasses two parts [45]: the *payload* and the *wormhole*.

Payload. *Asynchronous system* with $n \geq 2f + 1$ replicas in which at most f can be subject to *Byzantine failures*. If a replica does not fail during the execution of the CIS it is said to be *correct*, otherwise it is said to be *faulty*. Every CIS replica has a local clock that is assumed only to make progress. These clocks are not synchronized. We assume *fault independence* for the replicas, i.e., the probability of a replica be compromised is independent of another replica failure. This

assumption is substantiated by the diversity mechanisms employed in the CIS replicas (different OS, passwords, and internal firewall), and its coverage can be made as high as desired, through additional kinds of diversity [38]. Finally, we assume also that the station computers can not be compromised[4].

Wormhole. *Asynchronous secure tamper proof subsystem* $W = \{W_1, ..., W_n\}$ in which at most $f_c \leq f$ local wormholes can *fail by crash*. We assume that when a local wormhole W_i crashes, the corresponding payload replica CIS_i crashes together. Each local wormhole stores two symmetric keys: K_W – shared between the wormholes and used for vote authentication; and K_{LAN} – shared between the station computers of the LAN and the local wormholes, such that station computers only accept messages authenticated with this key (the IPSec key).

Network. The assumptions underlying LAN and WAN communication are as follows. We consider that the messages arriving at CIS replicas both from the WAN and the LAN have *unreliable fair multicast* semantics, a trivial extension of the commonly assumed *fair links* abstraction [2, 34] to multicast: if a message is multicasted infinitely many times it will be received by all its correct receivers infinitely many times. The two primitives offered by this service are: *U-multicast*(G,m), to multicast a message m to the group G, and *U-receive*(G,m), to receive m that was multicast to G, where G can be either WAN or LAN. This is substantiated in practice by the traffic replication devices. We assume that *all* communication between replicas and other machines from the WAN and the LAN are based on these primitives. Additionally, all CIS replicas communicate through point-to-point reliable channels for voting approved messages. These channels can be implemented on the protected LAN or on a separate network (that can be a *Virtual LAN* configured on the LAN or WAN switches acting as traffic replication devices – see Figure 2).

Cryptography. Our protocols use a *collision-resistant hash function* H, which receives an arbitrarily long input and produces a fixed-length output in such a way that it is infeasible to find two

[4]It is the trusted network that we aim to protect, exactly in the sense of preventing it from being compromised.

messages with the same hash. Additionally, the HMACs used in IPSEC are assumed to inherit the collision resistance property from the hash functions in which they are based [32], i.e., that it is infeasible to find two messages that for a key K have the same MAC. A message m is signed with a key K by concatenating m with a MAC of m produced with K. We use m_σ to represent a message m signed with some key K, i.e., $m_\sigma = m.MAC(m,K)$.

3.3 Service Properties

Before defining the service properties offered by the CIS, let us define the concept of legal message: a message is said to be *legal* if it is in accordance with the current deployed policy P. A message not in accordance with P is said to be *illegal*. Moreover, a message is said to be *processed* by the destination machine if its content is delivered to the application layer (e.g., the SCADA system). The basic properties offered by the CIS are the following:

- *Validity*: A legal message received by at least one correct CIS replica is forwarded to its destination machine;
- *Integrity*: An illegal message is never processed by its destination machine.

Notice that these two properties are sufficiently weak to be satisfied by a system with unreliable fair multicast communication and strong enough to ensure that only legal messages will be processed at LAN hosts.

3.4 Message Processing Algorithm

This section presents the main algorithm executed by the CIS to process messages incoming from the WAN to the protected LAN. The same algorithm is used to handle messages coming from the opposite direction (possibly with a more relaxed policy).

The policy verification is made in a *policy engine* accessed through the function *PolEng_verify*. We assume that all aspects of policy verification are encapsulated inside this component, which acts as an oracle that says if a message is legal or not.

Wormhole interface. The interactions between a replica CIS_i and its local wormhole W_i are made through a well defined interface that offers three services, invoked through the operations described in Table 1.

Operation	Return Type	Description
$W\_create\_vote(m)$	byte array	returns a MAC of message $\langle i, m \rangle$ produced with the wormhole shared key K_W
$W\_sign(m, C_m)$	byte array	returns a MAC of message m produced with the shared key K_{LAN}, if C_m contains at least $f+1$ votes (returned by $W\_create\_vote(m)$) from different replicas
$W\_verify(m_\sigma)$	boolean	returns *true* if σ is a MAC of m produced with K_{LAN}, and *false* otherwise

Table 1: Wormhole services specification.

The Algorithm. The CIS replicas execute Algorithm 1 for processing incoming messages. The algorithm is composed by three code blocks (WAN message reception, LAN message reception, and message retransmission) and all these blocks can be executed by different threads. We assume the existence of synchronization mechanisms that manage the concurrent access of the threads to the shared sets (e.g., execution of lines 1 and 2 is atomic). For readability, we chose to not include such mechanisms explicitly in the algorithm since they are not required for algorithm correctness.

T_{vote} is the single configuration parameter of the payload protocol and it defines the expected time required to receive, vote and sign a legal message. Additionally, the algorithm uses three variables: *Voting*, the set of messages being voted; *Pending*, the set of messages received and approved by the replica that were already signed by the wormhole but not yet forwarded to the station computer; and *TooEarly*, the set of correctly signed messages forwarded by some other replica but not yet received (from the WAN) by the replica.

The algorithm begins when a replica receives a message coming from the WAN (lines 1-14). If the received message is not in the *TooEarly* set and it is legal, all correct replicas will approve it (line 4) and then invoke the *approve* function (which will be explained later) to vote and sign this message (line 6). While the message is being voted and signed, it is stored in the *Voting* set (lines 5 and 7). Then, the message is inserted in the *Pending* set (line 8) and it remains in this set until

Algorithm 1 CIS payload (replica CIS_i).

```
{Parameters}
integer T_vote {Expected time to vote a message}
{Variables}
set Voting = ∅ {Messages being voted}
set Pending = ∅ {Not yet forwarded messages}
set TooEarly = ∅ {Messages forwarded before their arrival}
{Code for WAN message reception and processing}
upon U-receive(WAN, m)
 1: if m_σ ∈ TooEarly then
 2:     TooEarly ← TooEarly \ {m_σ}
 3: else
 4:     if PolEng_verify(m) then
 5:         Voting ← Voting ∪ {m}
 6:         m_σ ← approve(m)
 7:         Voting ← Voting \ {m}
 8:         Pending ← Pending ∪ {m_σ}
 9:         waitRandom()
10:         if m_σ ∈ Pending then
11:             U-multicast(LAN, m_σ)
12:         end if
13:     end if
14: end if
{Code for LAN message reception and processing}
upon U-receive(LAN, m_σ)
15: if m_σ ∈ Pending then
16:     Pending ← Pending \ {m_σ}
17: else if W_verify(m_σ) then
18:     TooEarly ← TooEarly ∪ {m_σ}
19: end if
function approve(m)
20: vote_i ← W_create_vote(m)
21: ∀CIS_j ∈ CIS, send(j, ⟨VOTE, H(m), vote_i⟩)
22: C_m ← ∅
23: repeat
24:     wait until receive(j, ⟨VOTE, H(m), vote_j⟩)
25:     C_m ← C_m ∪ {vote_j}
26:     σ ← W_sign(m, C_m)
27: until σ ≠ ⊥
28: return m_σ
{Periodic task for message retransmission}
for each T_vote that Voting ≠ ∅
29: ∀m ∈ Voting : U-multicast(WAN, m)
```

it is received from the LAN (lines 15-16)[5]. Finally, the replica waits for a random time interval (function *waitRandom* – line 9) and if the message is still in the *Pending* set, it is forwarded to the LAN (lines 10-11). The random waiting is implemented to avoid that all replicas forward the accepted message.

The algorithm contains several controls to deal with message losses, replica failures, and abnormal message ordering. The first of these controls deals with message omissions on the WAN: when a replica receives and approves a message m, it stores it in the *Voting* set (line 5) before starting the vote and sign procedure. The message is removed from this set only when it is signed (line 7). For each T_{vote} time units that *Voting* is not empty, its content is multicasted to the other CISs (line 29). This ensures that the message being voted will eventually be received by other correct CIS replicas (due to the fairness assumption) and will then be voted. It is possible that some replica forwards a correctly signed message to the LAN without some other replicas having received it from the WAN. In order to deal with these "early messages" on the LAN and optimize CIS performance,

[5]Recall that when a replica forwards a message to the LAN, it goes not only to the station computers but also to all CIS replicas.

we use the *TooEarly* set. When a not-pending legal message is received in the LAN, it will be stored in this set (lines 17-18) and stay there until it is received from the WAN (lines 1-2). As it will be shown in the correctness proofs presented in Section 3.6, the *Pending* and *TooEarly* sets are not necessary to satisfy the CIS properties defined in Section 3.3. These sets are used with two goals: to reduce message duplication in the LAN (*Pending* set + *waitRandom* function), and to optimize CIS performance by avoiding policy verification and message approval when a message was already previously signed and forwarded by some other replica (*TooEarly* set). Moreover, given that messages arriving from the WAN and the LAN have unreliable semantics, these sets need to be periodically reset in order to avoid messages staying there forever.

The most important part of the algorithm is the *approve* function (lines 20-28), which comprises the steps executed to vote for and sign a legal message. The function begins with the replica calling the wormhole to build a vote for the message (line 20) and sending this vote to all other replicas (line 21). Each replica then waits to receive votes from other replicas until it manages to get a sufficient number of valid votes to make the wormhole produce a signature for the approved message (lines 23-27).

3.5 Wormhole Algorithm

The implementation of the three services provided by the wormhole is presented in Algorithm 2. The replica id is stored inside the tamper proof memory of the local wormhole together with two symmetric keys that are used to authenticate different messages: the key K_W is used to authenticate vote messages that can be later verified by other wormholes; and the key K_{LAN} is used to sign approved messages to be forwarded to the protected LAN. These keys are used to authenticate messages using MACs.

The service $W\_create\_vote(m)$ uses the key K_W to generate the MAC for $\langle i, m \rangle$ (line 1). Since the id of the replica i cannot be modified by the payload and the key is secretly stored inside the wormhole, it is impossible for a malicious payload to impersonate other replicas in the voting

Algorithm 2 Wormhole services (local wormhole W_i).

{Parameters}
integer i {Replica id – for vote generation}
key K_W {Wormholes key – for vote authentication}
key K_{LAN} {Service key – for message authentication}
{Services}
service $W\_create\_vote(m)$
1: **return** $\mathsf{MAC}(\langle i, m\rangle, K_W)$

service $W\_sign(m, C_m)$
2: **if** $|\{v \in C_m : \exists j \text{ s.t. } v = \mathsf{MAC}(\langle j, m\rangle, K_W)\}| \geq f+1$ **then**
3: **return** $\mathsf{MAC}(m, K_{LAN})$
4: **else**
5: **return** $\perp$
6: **end if**
service $W\_verify(m_\sigma)$
7: **if** $\mathsf{MAC}(m, K_{LAN}) = \sigma$ **then return** *true* **else return** *false*

phase.

$W\_sign(m, C_m)$ calculates the MAC σ of m using the shared key K_{LAN} if and only if the replica payload presents a certificate set C_m containing at least $f+1$ valid votes produced by different replicas wormholes (lines 2-3). If there is no such number of valid votes, the wormhole returns the error value $\perp$ (line 5).

Service $W\_verify(m_\sigma)$ receives as input a message m allegedly signed with K_{LAN} and returns *true* if the MAC for m produced using K_{LAN} corresponds to σ, and *false* otherwise (line 7).

3.6 Correctness

This section proves that the CIS message processing protocol satisfies the service properties defined in Section 3.3. In these proofs, all line numbers refer to the Algorithm 1, unless specified otherwise.

Lemma 1 *If $f+1$ correct CIS replicas call the approve function for the same message, this message will be signed.*

Proof: When a replica i calls $approve(m)$, a vote for this message, denoted by $\langle i, m\rangle_{K_W}$, is produced in the corresponding wormhole (line 20). This vote will be sent to all replicas (line 21), and since channels are reliable, all replicas will receive and store it in a certificate set C_m (lines 24-25). If $f+1$ correct replicas call $approve(m)$, these $f+1$ replicas will receive and store $f+1$ valid votes in C_m. As described in lines 2-3 of Algorithm 2, when C_m contains at least $f+1$ valid votes for a certain message, a valid MAC for this message will be produced using the service key K_{LAN}. This

MAC is the signature of the message. ■

Lemma 2 *If a legal message is received by some correct CIS replica and the message was not previously forwarded by some other CIS replica, it will eventually be signed.*

Proof: When a legal message m is received by a correct CIS replica and m was not previously forwarded by some other CIS replica ($m \notin TooEarly$), it will be verified by the policy engine (line 4), stored in the *Voting* set (line 5) and then the function $approve(m)$ is called (line 6). According to the algorithm, if a replica stays more than T_{vote} blocked at line 6, the retransmission of line 29 will be triggered periodically (at each T_{vote} time units) until $Voting = \varnothing$ (which happens only if the replica executes line 7, after the wormhole signing the message). Since we assume that the unreliable multicast is fair (see Section 3.2), if a replica retransmits m many times, eventually all correct replicas will receive it. Given that $n \geq 2f + 1$, there will be always $f + 1$ correct replicas that will receive m and call $approve(m)$. And the Lemma 1 states that if $f + 1$ correct replicas call this function, m will be signed. ■

Now we present theorems for the Validity and Integrity properties.

Theorem 1 (Validity) *A legal message received by at least one correct CIS replica is forwarded to its destination.*

Proof: By Lemma 2, we know that a message m received by some correct replica will be signed (or else it was already forwarded). Since a message is signed only if $f + 1$ processes approved it, we know that at least one correct replica i that signed m will store it on the *Pending* set. The message will stay in this set until it is received in the LAN, i.e., until it is forwarded (lines 15-16). If the message is not forwarded by some other replica within the random waiting period of replica i, then replica i will forward it (lines 9-11). ■

Theorem 2 (Integrity) *An illegal message is never processed by its destination machine.*

Proof: Recall that *(i.)* in our system model, the destination machine can not be corrupted and it only processes messages signed with key K_{LAN}; *(ii.)* the key K_{LAN} is stored in the tamper proof

local wormholes; and *(iii.)* message m is only signed with this key in the presence of a certificate set containing at least $f+1$ valid and signed votes for m from different processes. Given these three facts, it is almost direct to see that illegal messages ("approved" by at most f processes) can not be signed (due to *(ii.)* and *(iii.)*) and, would not be processed by their destination machines. ■

4 Design Options

In this section we present some design options for the CIS protection service just described. Since these options make the CIS more expensive, we decided not to include them in the main design, but to present these in a separate section.

4.1 Dealing with DoS Attacks from Malicious Replicas

A malicious CIS replica can flood the WAN and LAN networks with illegal messages aiming to delay the forwarding of legal messages. This kind of attack can degrade the performance of the CIS as a whole. More generically, intrusion-tolerant designs are typically vulnerable to DoS attacks given that they compromise the network fairness/reliability commonly assumed in Byzantine fault-tolerant algorithms (e.g., [13, 50, 51]).

Considering the CIS architecture described before, a practical way to deal with this problem is to integrate an intrusion detection system in the traffic replication devices in order to monitor the networks connecting the CIS replicas and issue alarms when some replica behaves maliciously. The response to these alarms could be done by a human operator or by some kind of automatic fail-safe system that could shutdown malicious replicas and notify the system administrator. There are available in the market some switches with integrated intrusion detection systems, e.g., Cisco Catalyst 6500 and Nortel Ethernet Routing Switch 8600.

An alternative solution is to build a distributed secure wormhole, in which all replicas' local wormholes are connected by a secure network. With this control network, the whole voting protocol can be securely executed by the wormholes (in a crash-failure model), preventing malicious

replicas from disturbing it. Moreover, with the local wormholes connected through a secure network, if a malicious replica floods the LAN with invalid messages, other correct CIS replicas can notify the distributed wormhole about this behavior. When the malicious replica's local wormhole knows that at least $f+1$ other replicas suspect that its payload is faulty, it can shutdown the machine or recover it. For more information on this kind of design see [42].

4.2 Supporting Stateful Firewalls

The CIS design presented in Section 3 applies to stateless firewalls, which is the most common type of firewall used. However, some applications require stateful security policies, in which traffic is approved or denied taking into consideration the messages approved/denied in the past (e.g., some data message is only forwarded if some connection message was sent before).

The CIS policy engine can provide statefull semantics as long as one ensures that all messages are verified in the same order in all CIS replicas. In other words, we need an agreement protocol to ensure that all messages are evaluated in total order. This protocol could require either f more replicas and some weak timing assumptions (e.g., [13]) or the same number of replicas and a wormhole [17, 16].

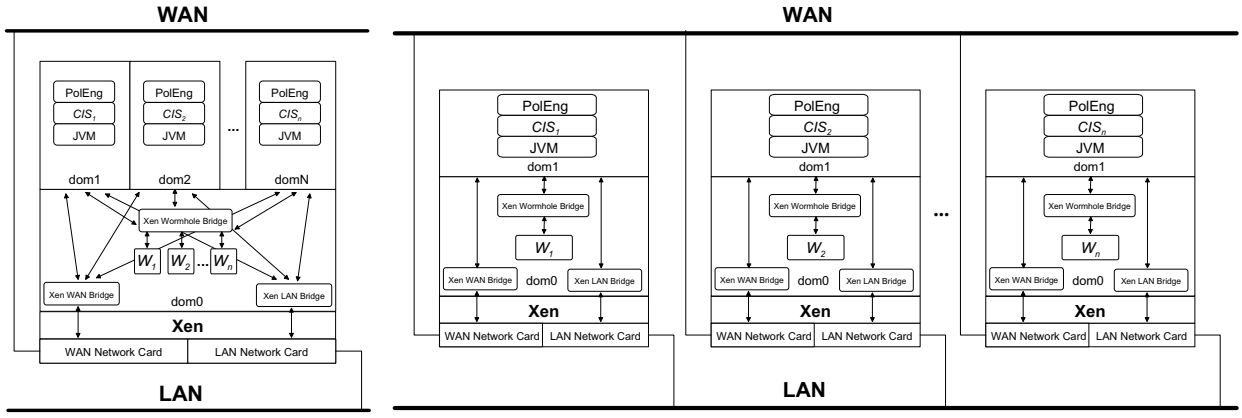
5 CIS Prototype

The intrusion-tolerant CIS can be deployed in several ways, depending on how critical is the LAN being protected and the budget available. For example, for very critical LANs, the CIS implementation must be based on different physical nodes, allowing the tolerance of both hardware and software faults. However, since price is always a major concern of power grid operators, a cheaper solution can be attained by resorting to virtualization. The various replicas are deployed in the same host, using virtual machines (VM) to isolate the different runtime environments, preventing intrusions from propagating from one replica to the others.

In order to evaluate the tradeoff of using physical or virtual replicas, we implemented two

prototype versions: one using physical replicas and another using virtual replicas running in a single physical host. The VM-based version has a special interest for two main reasons: first, to the best of our knowledge there are no previous experiments with replicated services deployed in virtual machines to obtain intrusion tolerance; second, a typical critical infrastructure has hundreds of station computers and other hosts to be protected, and a solution based on virtual machines can reduce the deployment cost[6].

The implementation of both versions of the prototype uses the XEN virtual machine monitor [4] with the Linux operating system. XEN is used for two different purposes: first, to isolate the wormhole from the payload by running these two system parts in different virtual machines; second, to deploy the different CIS replicas in the VM-based prototype version. The architectures of both versions of the prototype are presented on Figure 3.



(a) VM replicas.

(b) Physical replicas.

Figure 3: The architecture of CIS prototypes.

A XEN system has multiple layers, the lowest and most privileged of which is XEN itself. XEN may host multiple guest operating systems, every one executed within an isolated VM or, in XEN terminology, a *domain*. Domains are scheduled by XEN to make effective use of the available physical resources (e.g., CPUs). Each guest OS manages its own applications. The first domain, *dom0*, is created automatically when the system boots and has special management privileges.

[6]For example, the Italian power system contains about 1000 station computers [24], thus using three machines and two hubs to protect every one of these computers would require 3000 and 2000 extra machines and hubs, respectively.

Domain *dom0* builds other domains (*dom1, dom2, dom3, ...*) and manages their virtual devices.

CIS software. The two versions of the CIS run the same replica and wormhole software configured in different ways. Both replica and wormhole software are built as daemons with about, respectively, 3050 (CIS replica – excluding the policy engine) and 250 (wormhole) lines of Java code. In the current implementation, the policy engine is very simple, since it just checks if certain strings are present in the received packets. The CIS replica code makes use of the thread pool provided by the Java concurrency API. The HMAC algorithm used by the wormhole was based on the SHA-1 hash function and its default Java implementation (provided by Java Cryptography Extensions API). The waiting procedure (line 9 of Algorithm 1) was implemented in the following way: there is a maximum waiting time parameter that was divided by the number of replicas and each replica waits a random amount of time from its slice. To simplify the design and to avoid changes in the kernel, the CIS prototype operates at the UDP level, instead of IP level as most firewalls do. Therefore, there was no need to implement packet interception because packets are sent directly to the CIS. Moreover, authentication is not done at the IP level (as when using IPSEC), but in alternative the wormhole calculates the HMAC of the payload UDP packet. Notice that this type of authentication implies the same type of overhead of IPSEC authentication.

VM-based replication. Figure 3(a) presents the architecture of the VM-based prototype version. The n different local wormholes are deployed in domain $dom0$, and each of these wormholes receives requests and sends replies to their corresponding replicas. In the payload side, each replica i runs on a different domain dom_i. The replica payload is composed of the policy engine, the CIS protection service (CIS_i), and a Java Virtual Machine (JVM), given that both policy engine and protection service are implemented in Java. Given that the various payloads and wormhole run in different VMs, the communication between them is done through a virtual bridge. This bridge emulates a private pipe between each replica and its corresponding local wormhole. The two remaining bridges emulate the WAN and the LAN networks, and are configured to operate in broadcast mode.

Physical replication. Figure 3(b) presents the architecture of the CIS prototype that uses physical replicas. In this case virtual machines are used only to isolate the wormhole part from the payload part in each local physical replica. Therefore, local wormholes run in domain $dom0$ of each replica, whereas the policy engine, the CIS protection service, and the JVM, all run in domain $dom1$.

6 Experimental Evaluation

One of the worst possible types of attacks that the CIS can suffer is a DoS attack coming from the WAN. Such an attack may slow down CIS operation and affect the timeliness of SCADA/PCS supervision or control mechanisms that are normally deployed in critical infrastructures. Therefore, and following the reasoning presented in [10], this section presents an evaluation of the CIS performance (latency, throughput, and loss rate) when in the presence of a DoS attack.

We set up both versions of the prototype and connected them to two 100 Mbps switches representing the WAN and the LAN. A PC emulated the station computer and, in the WAN side, two PCs were deployed: a *good* sender trying to transmit legal traffic to the station computer, and a *malicious* sender causing a DoS attack. The VM-based CIS, the physical replicas, the good sender, the malicious sender, and the station computer executed each on a 2.8 GHz Pentium 4 PC with 2 GB RAM. Every machine run Linux 2.6.18, and XEN 3.0.3 was used to manage the virtual machines. The JVM was Sun JDK 1.5.11. The physical CIS replicas were configured with at most 100 processing threads and 1.5 GB of memory, while 33 threads and 512 MB of memory were used in the VM-based replication. In all experiments we consider $f = 1$ and, consequently, $n = 3$ (as ilustrated in Figure 3).

During the experiments, the malicious sender is constantly injecting illegal traffic at a certain rate. The IPERF tool[7] was employed to perform this task. The good sender transmits legal packets (of 1400 bytes) at a different rate. In the throughput and loss rate experiments, legal packets

[7] Available at `http://dast.nlanr.net/Projects/Iperf/`.

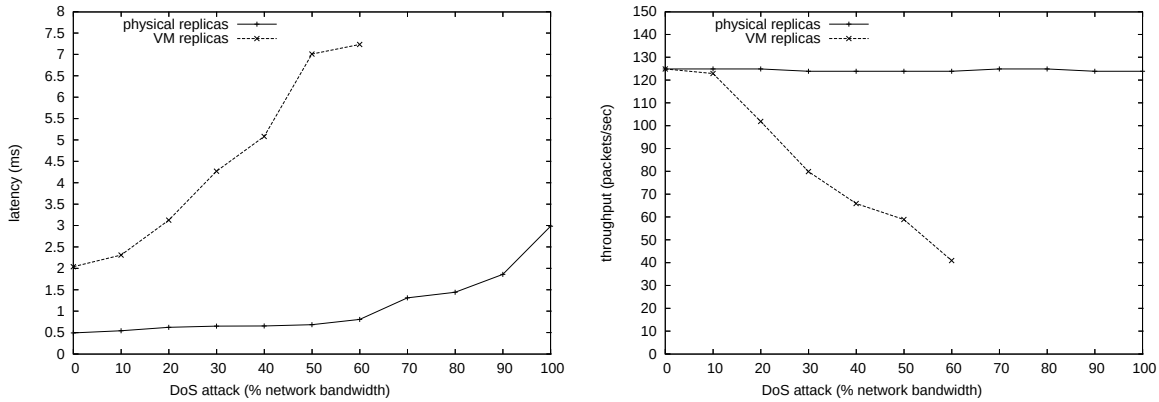
are sent at a rate of 124 packets/second and the obtained results correspond to the worst case values (lowest throughput and highest loss rate) that were observed. Notice that this is a very high traffic for a SCADA system, since it represents, for instance, 124 MMS commands being sent to a device per second. In the latency experiment, a legal packet is transmitted after the arrival of an acknowledgement (ACK) of the previous packet. The obtained results correspond to the average round-trip of 1000 legal packets (ACKs are not verified by the CIS), after excluding the 10% most distant from the average. The standard deviation value was roughly in the same order of magnitude of the average, due to the variability caused by the random time interval that each replica waits before forwarding a packet. Moreover, we observed that this mechanism was effective in reducing traffic multiplication inside the LAN. This effectiveness was higher when using physical replication than when using VM replication: less than 10% duplications in the former, and about 35% duplications in the latter.

Figure 4 presents the latency, throughput and loss rate of the two CIS prototype versions, when distinct levels of illegal traffic were being injected (by a DoS attack launched by the malicious sender) in the network[8]. Notice that the results for the two versions stop at distinct maximum levels of illegal traffic. This happens because after certain traffic levels, the systems become unstable, leading to high latencies and loss rates.

As expected, the results show that latency and message loss increase, while throughput decreases, when the amount of illegal traffic directed from the WAN to the LAN becomes larger. There are no pre-established acceptable limits for these parameters, except for the latency that in power systems is typically satisfactory if it remains lower than 200 milliseconds [24]. In the figure it is possible to observe that the latency was orders of magnitude smaller than this value and thus the CIS should not make great impact on the overall communication latency.

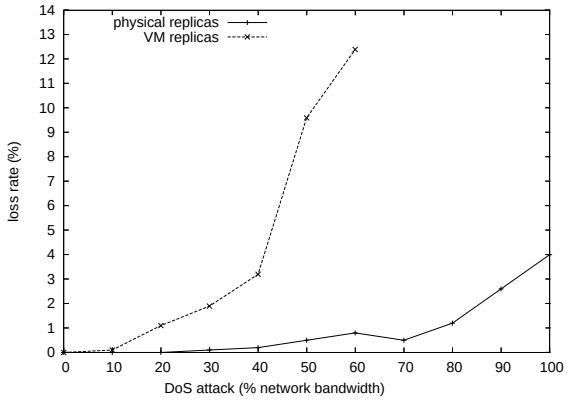
Comparing the maximum amount of illegal traffic that the two prototype versions can deal with, one can see that the physically replicated CIS is far more resilient to DoS attacks than the VM-based one. In every aspect (latency, throughput, and loss rate), physical replication withstands

[8]The values on the X axis correspond to the percentage of bandwidth that we configured the malicious sender to inject in the network.



(a) Average Latency

(b) Maximum Throughput



(c) Maximum Loss Rate

Figure 4: Impact of increasing DoS attacks in latency, throughput and loss rate of the two versions of the CIS prototype.

even the more fierce DoS attacks with minor degradation. In the worst-case scenario, i.e., when the illegal traffic is being injected at 100 Mbps (100% of network utilization), the average latency is 3 milliseconds, the throughput is the maximum possible, and at most 4% of the packets are lost.

Although the VM-based CIS prototype is by comparison less resilient than when using physical replication, the displayed results show that it is a valuable alternative to obtain cheap intrusion-tolerance. VM replication resists DoS attacks using up to 60% of the network bandwidth, and in the worst case, the average latency is 7.5 milliseconds, the throughput is 40 packets per second (one third of the maximum possible throughput), and at most 12.5% of the packets are lost. VM replication has limitations because all replicas share the *same* physical machine and, consequently, the amount of available resources has to be divided. In particular, our experiments allowed us to

conclude that the amount of memory allocated for each VM is a critical factor in how much illegal traffic the CIS supports. Nevertheless, resisting 60 Mbps (60% of network utilization) of illegal traffic is a good achievement, specially if we take into account that it is very easy to limit this traffic at the WAN gateway. These results could be improved by taking advantage of the fact that the wormhole is centralized in the VM-based prototype (see Figure 3). A centralized wormhole can do the voting and signing procedures in one step, waiting for $f+1$ votes from distinct replicas and then signing the message. However, such design corresponds to a different algorithm and it is not included in this report. The interested reader can find the results of using this alternative design in [7].

We did an additional set of experiments in order to observe what would be the behavior of the CIS without replication, i.e., using a centralized CIS that simply receives packets and forwards them if they are legal. The idea was to measure the overhead of adding intrusion-tolerance to an application level firewall. Under the worst possible DoS attack, the centralized CIS presented the maximum possible throughput, no packet loss, and an average latency of 75 microseconds. Therefore, the physically replicated CIS presents a reasonable overhead in terms of latency and loss rate, which is mainly due to the extra traffic generated by the voting procedure.

Summary of the results. The results described in this section can be summarized in the following three points:

- The physically replicated CIS design is efficient given that it only needs 3 replicas to tolerate one Byzantine fault and it resists the worst possible DoS attack (full network utilization) with almost no degradation.
- VM-based replication is cost-effective because it uses 3 times less resources (replicas) and it is still able to resist up to 60 Mbps of illegal traffic with reasonable degradation.
- The random time interval that each replica waits before forwarding a packet is effective in reducing traffic multiplication inside the LAN: less than 10% duplications with physical replication, and about 35% duplications with VM replication.

In general, the results presented in this section are even more interesting if we consider that both versions of the CIS prototype were implemented at application level and not inside the kernel.

7 Related Work

The wormholes model is a generic hybrid distributed system model [45], so reusable as the asynchronous or synchronous model. In the past, two particular instances of the wormholes model, TCB [46] and TTCB [18], allowed to popularize the model. The CIS architecture proposed in this report uses a simpler wormhole instance that includes a set of services (described in Table 1) not present neither in TCB or TTCB. Moreover, the CIS wormhole is local and asynchronous, while both TCB and TTCB require timing assumptions and a control channel connecting the local wormholes.

Work on *intrusion-tolerant services* has been fundamentally done considering two replication models: Byzantine quorum systems [36] and state machine replication [13]. The main difference between these approaches is on the way they maintain state consistency among the replicas: quorum systems are based on intersections between different quorums (sets of servers) while state machine replication relies on total order multicast protocols to ensure that all replicas evolve in the same way. Neither of these approaches is used in the CIS algorithms. The replicated CIS forwards a message if it is accepted by $f+1$ replicas. Due to this particularity, CIS requires significantly less replicas ($n \geq 2f+1$) than the usual $n \geq 3f+1$ bound.

Some intrusion-tolerant services have been proposed in the past: file systems [13, 23, 37], certification authorities [40, 51], coordination systems [6] and DNS [12]. The only other research that we are aware of about intrusion-tolerant firewalls is the privacy firewall described by Yin et al. [50]. The privacy firewall was proposed to enforce confidentiality in Byzantine services. Since this objective differs from ours, the CIS design requires much less replicas ($n \geq 2f+1$ instead of $n \geq (f+1)^2$) and does not need expensive threshold signatures. More importantly, the privacy firewall requires modifications on both sides of the firewall, to be able to verify approved

messages, i.e., to verify if incoming messages are correctly signed. From the perspective of the service provided, the CIS is closer to normal firewalls than to the privacy firewall.

Regarding the *dependability of critical infrastructures*, several works discuss how to integrate security in these systems. Most of them either analyzes current security incidents (e.g., [11]) or focus on the use of traditional security mechanisms (like firewalls) to protect SCADA/PCS systems [9]. However, some researchers have argued that the critical infrastructures requirements are different from the ones in corporate networks, and, more importantly, it has been shown that most firewalls available today are unable to deal with these requirements, namely maintaining reasonable operation during a DoS attack [10]. The evaluation presented in Section 6 shows that the intrusion-tolerant protection proposed in this report is capable of meeting such a requirement, even with a low cost VM-based approach.

There were a project that investigated an advanced protection and communication infrastructure for critical infrastructures, TCIP (Trustworthy Cyber Infrastructure for the Power Grid). This project advocates the use of a publish-subscribe infrastructure to provide secure and QoS-enabled communications between different control centers in a power grid. Much of the work being developed in these projects is orthogonal to the work presented in this report and to the CRUTIAL project in general, since it is focused on QoS, real-time communication, and trust management [26].

Regarding the use of virtualisation technology to construct intrusion-tolerant services, the VM-FIT architecture proposed in [39] provides basic support for intrusion-tolerant replication of services. Similarly to our VM-based CIS prototype, VM-FIT uses virtualisation to provide a trusted component (i.e., a wormhole) on each machine. However, the goal of the VM-FIT wormhole is to intercept client-service interaction and to distribute requests to the replica group. Overall, the goal of the VM-FIT architecture is conceptually different from the goal of CIS. VM-FIT is a generic architecture that allows to transform any deterministic centralized service in an intrusion-tolerant replicated one, whereas CIS is a specific intrusion-tolerant replicated service. In the same line of VM-FIT, Chun et al. [15] propose a set of optimizations for a BFT state machine replication pro-

tocol to be used more efficiently in settings with several replicas are deployed as virtual machines on the same physical machine. These optimized protocols could be used to implement efficient stateful versions of the CIS based on VM replicas.

8 Conclusions

This report presented the CIS protection service, implemented by highly resilient distributed devices, aimed at protecting critical information infrastructures. The CIS has two distinguishing features: it is intrusion-tolerant, and it supports a rich access control model that takes into account application semantics. The design of this protection device solves the non-trivial problem of ensuring transparent intrusion-tolerant operation for standard protocols and components.

We implemented two versions of the CIS prototype, one using physical replicas, and another using VM replicas. A set of experiments were conducted in order to evaluate the behavior of both versions in face of DoS attacks. The results showed that physical replication offers higher performance than VM-based replication, but even the latter exhibited an overall good performance in terms of latency, throughput and packet loss rate.

CIS intrusion-tolerance mechanisms are *cheap* in four different ways: $2f + 1$ replicas suffice to tolerate f Byzantine servers whereas many existing intrusion-tolerant services require $3f + 1$ replicas; it does not require consensus (so, no timing assumptions are needed); the performance overhead (latency, throughput, and loss rate) of the prototype with physical replicas is minimal in comparison with the behavior of a non-replicated CIS, even in the presence of DoS attacks; and the VM-based prototype shows that the CIS can be deployed in a single physical machine while maintaining reasonable performance values. The VM-based prototype represents an additional result of this report given that, to the best of our knowledge, it is the first implementation of an intrusion-tolerant service in a single machine.

Acknowledgments. We thank Marcelo Pasin for his helpful comments.

References

[1] A. Abou El Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization Based Access Control. In *Proceedings of 4th IEEE International Workshop on Policies for Distributed Systems and Networks – Policy'03*, June 2003.

[2] M. Aguilera, W. Chen, and S. Toueg. On quiescent reliable communication. *SIAM Journal on Computing*, 29(6):2040–2073, 2000.

[3] Y. Amir, C. Danilov, D. Dolev, J. Kirsch, J. Lane, C. Nita-Rotaru, J. Olsen, and D. Zage. Scaling Byzantine fault-tolerant replication to wide area networks. In *Proceedings of the International Conference on Dependable Systems and Networks – DSN'06*, pages 105–114, June 2006.

[4] P. Barham, B. Dragovic, K. Fraiser, S. Hand, T. Harris, A. Ho, R. Neugebaurer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles - SOSP'03*, Oct. 2003.

[5] S. M. Bellovin. Distributed firewalls. *;login: The USENIX Magazine*, Nov. 1999.

[6] A. N. Bessani, E. Alchieri, M. Correia, and J. S. Fraga. DepSpace: a Byzantine fault-tolerant coordination service. In *Proceedings of the 3rd ACM European Systems Conference - EuroSys'08*, Mar. 2008.

[7] A. N. Bessani, P. Sousa, M. Correia, N. F. Neves, and P. Verissimo. Intrusion-tolerant protection for critical infrastructures. DI/FCUL TR 07–8, Department of Informatics, University of Lisbon, April 2007.

[8] G. Bracha. An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing - PODC'84*, pages 154–162, Aug. 1984.

[9] British Columbia Institute of Technology. NISCC good practice guide on firewall deployment for SCADA and process control networks. Technical report, National Infrastructure Security Coordination Centre, London, UK, Feb. 2005.

[10] E. Byres, D. Hoffman, and N. Kube. The special needs of SCADA/PCN firewalls: Architectures and test results. In *Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation*, 2005.

[11] E. Byres and J. Lowe. The myths and facts behind cyber security risks for industrial control systems. In *Proceedings of the VDE Kongress*, 2004.

[12] C. Cachin and A. Samar. Secure distributed DNS. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'04)*, pages 423–432, 2004.

[13] M. Castro and B. Liskov. Practical Byzantine fault-tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, 2002.

[14] M. Chérèc, D. Powell, P. Reynier, J.-L. Richier, and J. Voiron. Active replication in Delta-4. In *Proceedings of the 22th Symposium on Fault-Tolerant Computing - FTCS'92*, pages 28–37, July 1992.

[15] B.-G. Chun, P. Maniatis, and S. Shenker. Diverse replication for single-machine byzantine-fault tolerance. In *Proceedings of the USENIX Anual Technical Conference*, June 2008.

[16] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz. Attested append-only memory: making adversaries stick to their word. In *Proceedings of the 21st ACM Symposium on Operating systems principles*, Oct. 2007.

[17] M. Correia, N. F. Neves, and P. Verissimo. How to tolerate half less one Byzantine nodes in practical distributed systems. In *Proceedings of the 23rd IEEE Symposium on Reliable Distributed Systems - SRDS 2004*, pages 174–183, Oct. 2004.

[18] M. Correia, P. Verissimo, and N. F. Neves. The design of a COTS real-time distributed security kernel. In *Proceedings of the Fourth European Dependable Computing Conference*, pages 234–252, Oct. 2002.

[19] W. S. Dantas, A. N. Bessani, and M. Correia. Not quickly, just in time: Improving the timeliness and reliability of control traffic in utility networks. In *Proceedings of the 5th Workshop on Hot Topics in System Dependability – HotDep'09*, June 2009.

[20] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures (extended abstract). In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO'92*, pages 457–469, Aug. 1992.

[21] D. Dzung, M. Naedele, T. P. V. Hoff, and M. Crevatin. Security for industrial communication systems. *Proceedings of the IEEE*, 93(6):1152–1177, 2005.

[22] L. H. Fink and K. Carlsen. Operating under stress and strain. *IEEE Spectrum*, Mar. 1978.

[23] J. Fraga and D. Powell. A fault- and intrusion-tolerant file system. In *Proceedings of the 3rd International Conference on Computer Security*, pages 203–218, 1985.

[24] F. Garrone (editor). CRUTIAL: Analysis of new control applications. Deliverable D2, EC project IST-2004-27513 (CRUTIAL), Jan. 2007.

[25] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson. 2006 CSI/FBI computer crime and security survey. Computer Security Institute, 2006.

[26] C. H. Hauser, D. E. Bakken, I. Dionysiou, K. H. Gjermundrød, V. S. Irava, J. Helkey, and A. Bose. Security, trust and QoS in next-generation control and communication for large power systems. *International Journal of Critical Infrastructures*, 2007. to appear.

[27] International Electrotechnical Commission. Inter control center protocol (ICCP). IEC 60870-6 TASE.2, 1997.

[28] International Electrotechnical Commission. Communication networks and systems in substations. IEC 61850, 2003.

[29] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen. Analysis of vulnerabilities in Internet firewalls. *Computers and Security*, 22(3):214–232, Apr. 2003.

[30] S. Kent. IP Authentication Header. RFC 4302 (Proposed Standard), Dec. 2005.

[31] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), Dec. 2005.

[32] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, Feb. 1997.

[33] B. Littlewood and L. Strigini. Redundancy and diversity in security. In *Proceedings of the 9th European Symposium on Research Computer Security - ESORICS 2004*, pages 423–438. Sept. 2004.

[34] N. A. Lynch. *Distributed Algorithms*. Morgan Kauffman, 1996.

[35] V. Madani and D. Novosel. Getting a grip on the grid. *IEEE Spectrum*, 42(12):42–47, Dec. 2005.

[36] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, Oct. 1998.

[37] M. A. Marsh and F. B. Schneider. CODEX: A robust and secure secret distribution system. *IEEE Transactions on Dependable and Secure Computing*, 1(1):34–47, Jan. 2004.

[38] R. R. Obelheiro, A. N. Bessani, L. C. Lung, and M. Correia. How practical are intrusion-tolerant distributed systems? DI-FCUL TR 06–15, Dep. of Informatics, Univ. of Lisbon, 2006.

[39] H. P. Reiser and R. Kapitza. Hypervisor-based efficient proactive recovery. In *Proceedings of the 26th IEEE Symposium on Reliable Distributed Systems - SRDS 2007*, Oct. 2007.

[40] M. K. Reiter, M. K. Franklin, J. B. Lacy, and R. N. Wright. The omega key management service. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 38–47, 1996.

[41] F. B. Schneider. Implementing fault-tolerant services using the state machine aproach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, Dec. 1990.

[42] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo. Highly available intrusion-tolerant services with proactive-reactive recovery. *IEEE Transactions on Parallel and Distributed Systems*, 2009. to appear.

[43] K. Stouffer, J. Falco, and K. Kent. Guide to Supervisory Control And Data Aquisition (SCADA) and industrial control systems security. Recommendations of the National Institute of Standards and Technology (NIST). Special Publication 800-82 (INITIAL PUBLIC DRAFT), Sept. 2006.

[44] M. van Eeten, E. Roe, P. Schulman, and M. de Bruijne. The enemy within: System complexity and organizational surprises. In M. Dunn and V. Mauer, editors, *Int. CIIP Handbook 2006*, volume II, pages 89–110. CSS, ETH Zurich, 2006.

[45] P. Verissimo. Travelling through wormholes: a new look at distributed systems models. *SIGACT News*, 37(1), 2006.

[46] P. Verissimo and A. Casimiro. The Timely Computing Base model and architecture. *IEEE Transactions on Computers*, 51(8):916–930, Aug. 2002.

[47] P. Verissimo, N. F. Neves, and M. Correia. CRUTIAL: The blueprint of a reference critical information infrastructure architecture. In *Proceedings of CRITIS'06 1st International Workshop on Critical Information Infrastructures Security*, Aug. 2006.

[48] P. Verissimo, N. F. Neves, and M. P. Correia. Intrusion-tolerant architectures: Concepts and design. In *Architecting Dependable Systems*, volume 2677 of *LNCS*. 2003.

[49] C. Wilson. Terrorist capabilities for cyber-attack. In M. Dunn and V. Mauer, editors, *International CIIP Handbook 2006*, volume II, pages 69–88. CSS, ETH Zurich, 2006.

[50] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin. Separating agreement form execution for Byzantine fault tolerant services. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles – SOSP'03*, pages 253–267, 2003.

[51] L. Zhou, F. Schneider, and R. Van Rennesse. COCA: A secure distributed online certification authority. *ACM Transactions on Computer Systems*, 20(4):329–368, Nov. 2002.