Secure and Self-healing Control Centers of Critical Infrastructures using Intrusion Tolerance

Maryam Tanha[1], Fazirulhisyam Hashim[2], and Shamala Subramaniam[3]
(Corresponding author: Maryam Tanha)
Department of Computer Science, University of Victoria[1]
Victoria, BC, Canada, V8P 5C2
(Email: tanha@uvic.ca)
Department of Computer & Communication Systems Engineering, Faculty of Engineering, Universiti Putra Malaysia[2]
43400 UPM Serdang, Selangor, Malaysia
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia[3]
43400 UPM Serdang, Selangor, Malaysia
(Received Nov. 14, 2014; revised and accepted Jan. 16 & Mar. 6, 2015)

Abstract

Nowadays, critical infrastructures are highly integrated with state-of-the-art information and communication technologies to enhance their efficiency. Due to far-reaching societal and economic impacts caused by failure or malfunction of critical infrastructures, cyber security and self-healing capability are among their salient features. A new security paradigm referred to as intrusion tolerance is envisaged to complement the existing security solutions (i.e., intrusion prevention and detection), as well as to provide availability and self-healing capabilities, particularly for the control centers as the key components of critical infrastructures. However, intrusion tolerance techniques are associated with substantial cost. In this paper, we propose an intrusion tolerant system architecture which incorporates distinctive features, namely dynamic redundancy level, and hybrid and hierarchical rejuvenation mechanism. The acquired results from security analysis of the proposed architecture show improvements compared to two established architectures. Also, analysis of the incurred cost demonstrates the cost-effectiveness of the proposed architecture.

Keywords: Control center, critical infrastructure, intrusion tolerance, self-healing

1 Introduction

In recent decades, the growing dependence of critical infrastructures on Information and Communication Technology (ICT) and open standards has raised serious concerns about security issues. Critical infrastructures are complex physical and cyber-based systems and assets that lay the foundations for a modern society, and their secure and dependable operation is of utmost importance for national security and economy. Smart grid (as the modern power grid), public health system, and transport system are examples of critical infrastructures. Cyber systems serve as the backbone of critical infrastructures, thus cyber security incidents may not only affect the cyber domain but also potentially impact their dependent physical systems [37]. The cyber-physical dependencies, large-scale operation, heterogeneity and complexity along with sophisticated and novel attacks pose grave and new threats to the mission critical applications in critical infrastructures.

Using open standard software and protocols have opened avenues for attackers to pose dire threats to different sections of critical infrastructures' communication system, particularly SCADA and control systems. Some of the recent high-profile attacks such as Stuxnet worm [16,28] and FLAME [29] have been mainly targeted at control systems of critical infrastructures and crucial organizations. Moreover, the security objectives of critical infrastructures differ from the ICT security goals in their order of significance. Availability, continuity of service and safety are the main security priorities in critical infrastructures.

On top of all the mentioned issues, the widespread and socio-economic impacts of malfunction or failure of critical infrastructures resulting from accidental or malicious events mandate more automatic and robust security solutions [5, 40]. These security approaches can be associated with self-healing capabilities of critical infrastructures. Self-healing responses to malicious acts of sabotage and natural calamities is one the essential features of critical infrastructures. Self-healing is defined as the attribute of a system to be able to recognize abnormal operation (the disturbances may result from security intrusions) and subsequently making proper adjustments to

restore to normal conditions [11]. Control centers are considered as the brain of critical infrastructures. They are in charge of data analysis and decision making. For instance, in smart grid as a critical infrastructure [2], based on the assembled data, the control centers make appropriate adjustments to power supply to satisfy demand as well as spot and respond to the defects or failures by sending control commands to field devices. Figure 1 illustrates a control center which supervises other sections in a critical infrastructure. This figure also depicts some of the key components of the control center such as SCADA servers and historian databases.

SCADA systems (as the key components of control centers) play a pivotal role in the proper operation of critical infrastructures, any malfunction or failure of these systems and their underlying software systems may result in widespread and devastating effects on industry, economy and people's daily life.

Therefore, the correct functioning of SCADA systems in exigent security circumstances is of paramount importance. Two of the dire threats to SCADA are Denial of Service (DoS) and unauthorized access/integrity breach [23]. These threats will result in the unreliability of the control signals from the monitoring system as well as the collected data gathered from different sections of critical infrastructures that are used for decision making or other purposes.

In addition, thanks to the time-criticality of the communication and control in some critical infrastructures such as smart grid, a delay of a few seconds (following from an availability attack) may lead to irreparable harm to the national economy and security [18].

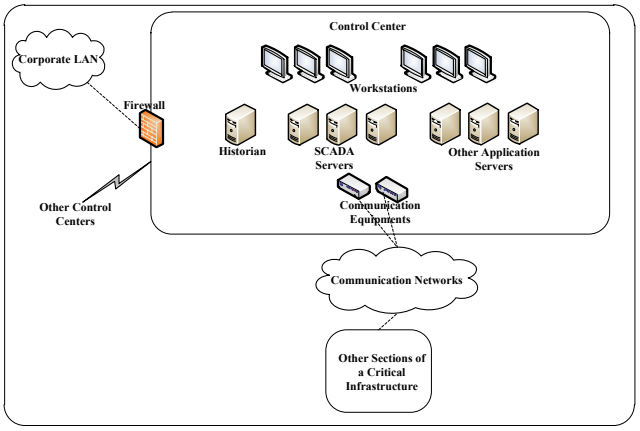

Figure 1: Control center in critical infrastructures

The aforementioned security concerns serve as contributing factors to change our mind set about the level of security that can be achieved through conventional security approaches (i.e., prevention and detection [15]) especially for critical infrastructures.

To satisfy the mentioned security requirements, a promising mechanism called intrusion tolerance has come to existence and it has received considerable attention in recent years [3–5, 7, 20, 22, 26, 27, 31, 38, 40, 47]. Intrusion tolerance is concerned with the fact that it is always probable for a system to be vulnerable to security compromise as well as for some attacks to be launched successfully on a system [40]. In spite of these assumptions, intrusion tolerance mechanisms ensure that the system prolongs its normal activities (or acts in a degraded mode providing only essential services) even when it is under attack or partially compromised. Thus, rather than preventing intrusions from happening in the system, they are tolerated by adopting and triggering appropriate mechanisms such as redundancy, diversity, rejuvenation, and so on.

In this paper, we propose an Intrusion Tolerant System (ITS) architecture to enhance the availability and self-healing capabilities of critical infrastructures while decreasing the associated cost with intrusion tolerance techniques.

The main contributions of our research can be summarized as follows:

- We highlight the importance of intrusion tolerance approach which raises the possibilities for enhancing the security of crucial components in critical infrastructures, particularly control centers and Supervisory Control and Data Acquisition (SCADA) systems.
- An ITS architecture is proposed to enhance the level of security in control centers of critical infrastructures.
- To provide the availability and self-healing capabilities required by critical infrastructures, special focus is placed on redundancy and rejuvenation as two intrusion tolerance techniques. Also, we propose dynamic redundancy level and hybrid and hierarchical recovery algorithms to alleviate the substantial cost associated with these techniques.

The paper is organized as follows. Section 2 provides a detailed analysis of intrusion tolerance as a promising security solution for critical infrastructures. Moreover, the most commonly used intrusion tolerance techniques are presented and a comparison is made between some of existing ITS architectures. In Section 3, a detailed discussion on the proposed intrusion tolerant architecture (its modules and embedded algorithms) for control centers of critical infrastructures is presented. The security and cost analysis of the proposed ITS architecture is provided in Section 4. Finally, Section 5 draws the conclusion. It should be noted that the terms recovery and rejuvenation are used interchangeably throughout this paper.

2 Intrusion Tolerance for Critical Infrastructures

Intrusion tolerance is commonly referred to as the third generation of security technologies [14] which provides complementary features to conventional security mechanisms, i.e., prevention and detection. It shows enormous

potential to be adopted and deployed in critical infrastructures' control centers in which the correct service and availability is of great importance. The impacts of availability violation in critical infrastructures are substantial and affect the physical world. The possible consequences of service disruption in critical infrastructures range from financial loss to human loss. As an instance in the context of smart grid, a compromised server in control center may result in sending misleading data to the field device. This attack affects the availability (i.e., not allowing unauthorized access and providing correct service). The viable consequences may be equipment damage (if control commands that are sent to the field device lead to overload conditions), blackouts or safety issues (if a line is energized while linemen are in the field servicing the line). To increase the availability of critical infrastructures' control centers the self-healing capabilities are essential. Recovery mechanisms enable the self-healing feature for critical infrastructures, thus in this paper we placed especial focus on rejuvenation mechanisms in order to enhance the availability.

Intrusion tolerance and its paradigms (e.g., replication and recovery) enable secure and normal operation of the control centers of critical infrastructures, even when the system is being attacked or partially compromised. The primary goal of intrusion tolerance is to tolerate malicious events and sustained attacks as well as masking, removing or recovering from intrusions. Thus, intrusion tolerance measures avert security failures and aid to maintain the availability of the system. Moreover, intrusion tolerance places emphasis on the impact of the attack rather than the cause of it [41].

During the last decade, various research have been conducted on intrusion tolerance and multiple intrusion tolerant architectures with specific features and applications have been proposed. The Willow architecture [17], COCA [48], DIT [39], MAFTIA [34], SITAR [44], SCIT [3], Crutial [5], FOREVER [30] and Generic intrusion tolerant architecture for web servers [27] exemplify a number of the proposed ITS architectures. Some of these architectures are application-specific. For instance, the goal of COCA is to provide a secure and fault-tolerant Certification Authority (CA) while Crutial is a distributed firewall-like intrusion tolerant system for critical infrastructures protection such as power grid. But primarily, enhancing the security and availability of distributed services, Commercial Off The Shelf (COTS) servers and critical information systems have called for designing such architectures.

There are several intrusion tolerance techniques that are commonly used in intrusion tolerant systems. Some of the main techniques are as follows:

- Replication: Space redundancy or replication involves physical resource redundancy which is a key building block of many intrusion tolerant systems.
- Diversity: Replication suffers from the underlying problem of fate sharing for replicas [33, 43]. If an attacker discovers and exploits a vulnerability in one replica, it is highly likely that all replicas are susceptible to the same threat. Thus, diversity (usually in its most common form which is operating system diversity [10]) serves as a solution to alleviate this problem.
- Rejuvenation: Rejuvenation involves the restoration of a replica to a pristine state to eliminate the likely effects of intrusions or faults [43]. It can be triggered reactively following from intrusion detection or carried out proactively and periodically.
- Voting: Voting algorithms are employed to reach a consensus on the valid and final output of non-faulty replicated components in an ITS. Using Byzantine Fault Tolerance (BFT) agreement protocols or some criteria such as edit distance (e.g., hamming distance) and hash codes make the comparison feasible. Voting contributes to masking and tolerating intrusions [43].
- Secret Sharing: Secret sharing or threshold scheme is based on the idea of concealing a piece of information by splitting it into several shares and distributing among participants in a manner that specific subsets of the shares are required to rebuild the initial data [1, 6, 25]. This intrusion tolerance technique has been used in ITS architectures, e.g., COCA (a distributed certification authority) and its main purpose is providing confidentiality and integrity. Since in this paper our main goal is to provide availability and self-healing capabilities as the top security priorities for the critical infrastructures, we do not include secret sharing method in our proposed architecture.
- Proxy: Proxies serve as additional layers of defense between replicated servers and clients.

3 Proposed ITS Architecture

Typical intrusion tolerant systems have single primary focus. For instance, Scalable Intrusion Tolerant Architecture for Distributed Services (SITAR) is detection triggered, and Self Cleansing Intrusion Tolerance (SCIT) is recovery based. Some ITS architectures (e.g., Crutial) apply a hybrid rejuvenation approach (i.e., both proactive and reactive recovery) that enhances the level of security, but the complexity and cost of redundancy and recovery increases enormously. Based on our feasibility studies, using adaptive redundancy as well as a hybrid and hierarchical rejuvenation approach assists in reducing the incurred cost. Also, the specific requirements of the critical infrastructures' control centers (e.g., self-healing capabilities, delay sensitivity) underscore the need for a new ITS architecture that suits these systems.

By securing the software systems that manage the sub systems of the control centers, we would be able to mitigate the consequences of cyber security incidents that

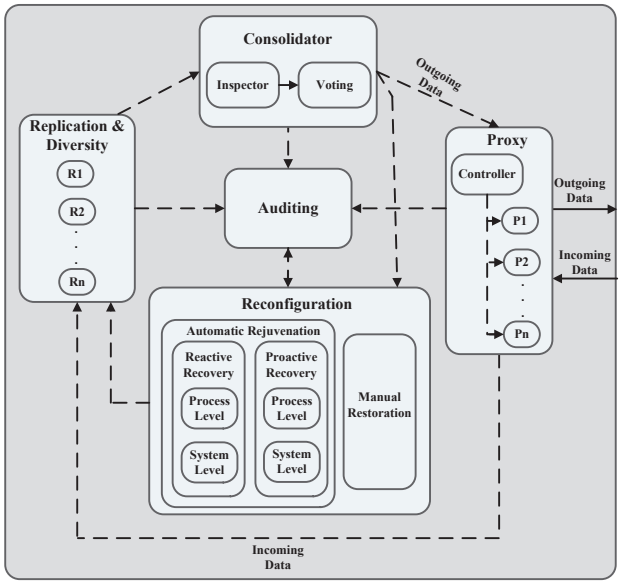

Figure 2: The proposed ITS architecture

affect the physical domain (e.g., blackouts in smart grid) and subsequently, enhancing the cyber-physical security of critical infrastructures' control centers. Our proposed ITS encompasses a rich blend of a wide spectrum of different intrusion tolerance techniques. As illustrated in Figure 2, the proposed system comprises five modules, namely replication & diversity module, auditing module, consolidator module, reconfiguration module and proxy module. The role and working principles of the aforementioned modules are elucidated in the following sections. In general, our proposed system is a security architecture that can be hosted by a dedicated server (including auditing module, consolidator module, reconfiguration module and proxy module) that manages a number of physical/virtual replicas (in replication & diversity module). These replicas run one or more critical applications in control centers as well as agents of the proposed ITS.

It should be noted that in the proposed ITS architecture the emphasis is placed on offering algorithms for automatic and hierarchical rejuvenation as well as managing replication and rejuvenation mechanisms cost-effectively. This is due to the importance of availability and self-healing capabilities for the critical infrastructure along with addressing the issue of substantial cost incurred by intrusion tolerance techniques. In essence, recovery-oriented computing is a vital aspect of a self-healing system [11] such as critical infrastructure. Intrusion tolerance techniques such as redundancy and rejuvenation contribute towards provisioning availability and self-healing characteristics. Moreover, to avoid the proposed ITS from being compromised by the intruders, it is assumed that all the components' tasks and their communications are performed in a trusted platform. Proxy module also helps to enhance the security of the ITS.

3.1 Replication & Diversity Module

Replication in ITSs is usually integrated with Byzantine Fault Tolerance (BFT) algorithms in which the number of replicas is required to be $3f+1$ to tolerate f faulty replicas. As a result, a fault/intrusion tolerant distributed system is obtained which is enabled to tolerate f Byzantine (i.e., arbitrary) faults. The aforementioned arbitrary faults model accidental faults or malicious attacks and intrusions. Specifically, the key idea of BFT algorithms is to enable a system to automatically continue correct operation despite the fact that some of its components show arbitrary, probably malicious behavior. BFT algorithms have already been adopted to design intrusion tolerant services such as network file systems, cooperative backup, large scale storage and certification authorities [42].

The replication & diversity module consists of a number of replicas for a critical entity (usually a physical server running crucial applications such as Master Terminal Unit (MTU) or historian databases as shown in Figure 1) in the control centers of the critical infrastructures. In addition, with regard to different levels of security needed in different points of the critical infrastructures, this module can be modified accordingly. The replicas can be physically distributed (i.e., in different machines) for application such as automatic grid separation in emergency states. As another example in the context of smart grid, the replication module can be utilized in substations with replicas as virtual machines running in the same host. Although the system does not tolerate physical faults, it may provide adequate protection for substations in smart grid. Similar approach has been used in [5].

In this module, the number of replicas is assumed to be $2f+1+k_{max}$ (to tolerate f faulty/compromised replicas provided that there are trusted components) and the value of f and k_{max} ($f, k_{max} \geq 1$) are indicated in the deployment time. A similar approach also used to design a distributed firewall-like protection device named Crutial Information Switch (CIS) in [5]. k_{max} denotes the maximum possible number of concurrent recoveries. The reason why the value of k_{max} is added to the number of replicas will be discussed in the reconfiguration module section.

As mentioned before, diversity decreases the possibility of being vulnerable to the same intrusion for different replicas. In the proposed ITS, all replicas have operating system diversity to decrease the probability of sharing the same vulnerabilities. Operating system is considered a vital element of each replica on account of hosting the SCADA system and other critical components in control centers. Subsequently, any misconfiguration or vulnerability in it may bring down the SCADA system and causes the adversaries achieve breakthroughs [23]. Hence, operating system diversity proves an appropriate approach for applying diversity to replicas. However, the number of existing and tailored operating systems are limited, thus the diversity level is confined to this number. To have

an effective system using both redundancy and diversity, the redundancy level and diversity level are expected be equal. More specifically, if the total number of replicas is assumed to be $2f+1+k_{max}$, the ideal degree of diversity should be the same. By using a different operating system in each replica, it is less probable that the replicas suffer from similar vulnerabilities. If the diversity degree is less than redundancy degree, at least two replicas will have the same operating system and consequently will experience the same fate in the event of intrusions. In contrast, if we assume the diversity degree more than redundancy degree, part of the diversity level is useless since it is not applied to any replica.

3.2 Consolidator Module

In this module, the outputs are inspected and then consolidated into one. More specifically, this module aims at examining the responses/outputs of the replicas to identify possible infected/compromised ones. It is composed of the following sub modules.

- Inspector: Acceptance testing [45] as an intrusion tolerance technique is entailed in the inspector module. It involves application-specific checks with regard to the security policy to ensure the sanity of outgoing data (e.g, control commands that are sent to substations in critical infrastructure) from the replicas. Any symptom of security compromise detected by it will trigger the reactive recovery sub module in the reconfiguration module. In contrast with SITAR, acceptance testing is only performed on the outgoing data in our proposed ITS architecture. This would result in decreasing the delay imposed by the proposed ITS for processing the incoming data in critical systems such as smart grid control systems that are delay sensitive. The incoming data must satisfy the time requirements otherwise it is not useful. Also, some preliminary check on the incoming data can be performed by proxies. In contrast, due to the crucial importance of the outgoing data which are usually the control commands in critical systems their sanity and correctness should be tested before letting them leave the system.

- Voting: This sub module is intended for masking the impacts of intrusions as well as ensuring the integrity of replicas outputs. Based on a voting algorithm, it seeks for the correct output by comparing the redundant outputs from the active replicas that passed the inspector. In this way, it will arrive at a consensus on the final desired output to be passed to the proxy module. This output can be a command or information from the control centers destined for a device or component in critical infrastructures.

3.3 Reconfiguration Module

Reconfiguration module consists of two sub modules namely, automatic rejuvenation and manual restoration. When the proposed ITS is able to mask an intrusion, it uses the automatic rejuvenation sub module, otherwise it takes advantage of restoration which involves human intervention. Manual restoration happens when for instance the system is targeted by DoS attacks and only capable of provisioning the essential services. The sub modules descriptions are provided in the following sections.

Automatic rejuvenation: Automatic rejuvenation mostly addresses the required self-healing capabilities of critical infrastructures. In the event of detecting an abnormal behavior of a replica or periodically, it triggers a recovery for the respective replica. Also, automatic rejuvenation enables the concurrent rejuvenation of at most k ($1 \leq k \leq k_{max}$) replicas out of $2f+1+k_{max}$ (total number of replicas). The assumption for the total number of replicas eliminates the impact of compromised replicas (at most f) and recovery on the availability of the system. It should be noted that k has a fixed value in Crutial (usually $k=1$) whereas the value of k is dynamic in our proposed ITS architecture.

In this module, a hybrid rejuvenation approach, i.e., proactive and reactive recovery, has been used to address the shortcomings of the two aforementioned rejuvenation approaches. Reactive recovery mainly relies on the underlying intrusion detection methods and subsequently is subjected to the same drawbacks such as inability to detect unknown attacks and false positives. In contrast, proactive recovery can compensate for dormant or undetected intrusions. By assuming an asynchronous distributed system model and proactive recovery and it is not possible to guarantee that recoveries are performed within known time bounds. Thus, we have used a hybrid distributed system model that uses some trusted components to ensure that replicas are always rejuvenated in accordance to predefined time bounds.

A hybrid rejuvenation mechanism will enhance the performance of the system through decreasing the possible duration of time a compromised replica may disrupt the normal operation of the system [31].

To come up with a cost-effective and hybrid rejuvenation mechanism, we were inspired by a hierarchical reactive recovery method that has been proposed recently in [14]. This model eliminates the need for complete recovery when the system is partly compromised. The merits of this model can be considered as reduced total recovery time, improved flexibility and dependability.

In the proposed ITS architecture, reactive recovery can be triggered externally and at the system level by the consolidator module or internally (within a replica) in a hierarchical fashion (including process level recovery and system level recovery). Proactive recovery is performed periodically by choosing an active replica based on smallest rejuvenation time stamp in a hierarchical manner. It

is triggered by the proactive recovery sub module of the reconfiguration module. The details of the proposed rejuvenation algorithm are provided in Section 3.6.

Manual restoration: This sub module is triggered when the intrusion (whether detected or not) is non-maskable (e.g., more than f replicas have been compromised). This may cause the system to be in graceful degradation mode, stopped functioning mode or complete failure mode all of which require human intervention and corrective measures to return to the normal working state.

3.4 Auditing Module

This module maintains audit logs for all modules. The logs would be useful for security administrator to monitor and analyze the operation of the system.

3.5 Proxy Module

The Proxy module is placed on the boundary of the ITS architecture where the data comes in or goes out. The proxy module shields the internal structure of the ITS from attackers as well as acting as a load balancer.

The incoming data go through the proxy module as the first layer of defense. This data is then forwarded to the replication & diversity module to be dealt with. Moreover, the control commands from the SCADA system (outgoing data in Figure 2) pass the proxy to reach the devices or other components of critical infrastructures.

Proxy module is composed of several proxies located in different virtual machines that have diversity in their operating systems and are managed by a controller. Proxies can have three modes, namely online, offline, and cleansing. The number of online proxies can be one or more based on the decision of the controller. Depending on a defined exposure time for proxies and a round-robin algorithm, the controller deals with the rotation and changing turn between proxies [3]. When the exposure time requirement for a proxy is met, it will go through the rejuvenation process (or cleansing process) and will be in cleansing mode. Then, its mode will be altered to offline mode and it will be ready to be chosen by the controller to go online.

3.6 Cooperative Operation of Replication and Reconfiguration Modules

In the proposed ITS architecture, we mainly focus on two intrusion tolerance techniques, namely redundancy and rejuvenation. The cooperative operation of replication and rejuvenation module provides availability and self-healing features in a cost-effective manner. This would assist in enhancing the level of security while reducing the cost.

Hybrid and hierarchical rejuvenation algorithm: While in [14] the hierarchical recovery is performed at three levels and only applies to reactive recovery, our proposed algorithm employs a hierarchical recovery strategy at two levels (i.e., process level and system level) for both proactive and reactive rejuvenations. It should be noted that a process is defined as an instance of a computer program. A process can be a system process such as a background process for logging and monitoring or an application process such as Internet Explorer or any application that is running on a SCADA server or other application servers in a control center in critical infrastructure. Algorithm 1 shows the proposed hybrid and hierarchical rejuvenation mechanism which is provided by the cooperation of replication & diversity and reconfiguration modules. Process manager is a module executed in each active replica to handle the process level recovery. At the deployment time, critical processes in the replicas are identified. With regard to this, there are two sets of processes, namely active set (includes running processes) and standby set. To differentiate between reactive process level recovery and proactive process level recovery, the process manager includes two components, namely PLRR (Process Level Reactive Recovery) and PLPR (Process Level Proactive Recovery).

PLRR (Line 1 in Algorithm 1) acts as a type of host-based IDS which features self-healing capabilities. Based on a timeout period, it examines the pool of active processes. In the event of finding any suspected process, PLRR will obtain the relevant checkpoint, kills the process and activates its peer from the standby set (if there is any) otherwise the system level reactive recovery (SLRRTriggered denotes a system level reactive recovery) may be performed.

PLPR (Line 15 in Algorithm 1) deals with process level proactive recovery. After a timeout period which is set in the proactive recovery sub module of the reconfiguration module, a proactive rejuvenation signal is sent to the replica with the least rejuvenation time stamp. In this case, if there are standby processes available for all the critical active processes, the PLPR will replace each active process with its peer from the standby set in a similar way to PLRR. However, if the aforementioned condition is not satisfied, the proactive recovery (SLPRTriggered indicates a system level proactive recovery) may be carried out at system level for the respective replica.

The process level recovery is time-saving compared to system level recovery as well as it is more secure since it mainly involves internal information and communication exchange in a machine. Moreover, it does not require the replica to go offline for performing the recovery and causes less overhead on the replica.

Dynamic redundancy level algorithm: The total number of replicas represents the redundancy or diversity level (we assume that they have the same value). The redundancy level is a linear and increasing function of f and k. However, increasing f will have more impact on the

Algorithm 1 Hierarchical recovery

```
1: Begin
2: if PLRR − timeout then
3:    Detection&Polling();
4:    for all the suspected processes (j) in replica i do
5:       if Process[j].StandbyAvailable() then
6:          Process[j].ObtainCheckpoint(Suspect);
7:          Process[j].Kill(Suspect);
8:          Process[j].ActivateStandby();
9:          Reset respective recovery timer
10:      else
11:         Replica[i].SLRRTriggered = True;
12:         Exit the for loop
13:      end if
14:   end for
15: end if
16: if PLPR − timeout then
17:   Polling();
18:   for all the processes (j) in replica i do
19:      if Process[j].StandbyAvailable() then
20:         Process[j].ObtainCheckpoint(Suspect);
21:         Process[j].Kill(Suspect);
22:         Process[j].ActivateStandby();
23:         Reset respective recovery timer
24:      else
25:         Replica[i].SLPRTriggered = True;
26:         Exit the for loop
27:      end if
28:   end for
29: end if
30: End
```

increase of the number of replicas compared to k. This means that increasing f incurs more resource cost. Having less impact on the resource cost, increasing the value of k would lead to increase in the cost of recovery which may be more preferable to the resource cost increase following from increasing the value of f.

Having a dynamic redundancy level is regarded as an effective way to cut the resource cost and recovery cost to a certain extent. An adaptive redundancy level algorithm has been offered in [27] based on attack and alert severity. It provides a triplex regime for critical applications, i.e., three web servers out of N web servers process each client request. This regime can be dynamically increased provided that the attack rate increased. After a specific period of time with no compromise detection, the regime will be decreased. Three algorithms for dynamic redundancy design have been presented in [24] for NAN (Neighbourhood Area Network) gateways in critical infrastructure which considered availability threshold and cost minimization. Our proposed dynamic redundancy algorithm is integrated with recovery mechanism and handled by the automatic rejuvenation module. Considering the possible maximum redundancy/diversity level ($MaxRL = MaxDL = 2f + 1 + k_{max}$) as an upper bound for the total number of replicas, the offered adaptive redundancy level algorithm involves increasing the value of k (Line 7 in Algorithm 2) based on the number of system level recovery signals sent to the reconfiguration module. If the number of ongoing rejuvenations is equal to k and $RL < MaxRL$ (RL denotes redundancy level and it is equal to $2f + 1 + k$ which is the number of active or under recovery replicas), then a new incoming system level rejuvenation signal would result in increasing the number of k by one, otherwise (if $k = k_{max}, i.e, RL = MaxRL$) the recovery signal is ignored by the automatic rejuvenation module. After a time-out period, if the number of ongoing rejuvenations is less than k, this value is reduced by 1. We can also conclude that the number of standby replicas is calculated by subtracting the value of (RL) from the value of ($MaxRL$) in each moment. Algorithm 2 illustrates the adaptive redundancy algorithm.

Algorithm 2 Dynamic redundancy level

```
1: Begin
2: k = 1;
3: RL = 4;
4: MaxRL = 6;
5: while true do
6:    if SLRRTriggered || SLPRTriggered then
7:       if OngoingRejs = k && RL < MaxRL then
8:          k++;
9:          RL++;
10:         OngoingRejs++;
11:         Reset respective recovery timer
12:      else
13:         if OngoingRejs < k then
14:            OngoingRejs++;
15:         end if
16:      end if
17:   end if
18:   if RL − timeout then
19:      if OngoingRejs < k then
20:         k−−;
21:         RL−−;
22:      end if
23:   end if
24: end while
25: End
```

3.7 A Case Study: Trojan Horse Attack on Smart Grid Control Centers

For the benefits of readers we describe the working principle of the proposed ITS architecture by an attack scenario in the context of smart grid. Suppose a possible intrusion scenario in which an attacker (an outsider or a malicious insider) has bypassed prevention or even detection mechanisms and has gained access to the SCADA system in smart grid (this attack can be considered as privilege escalation). Subsequently, he/she tries to infect one or more replicas of a critical component. Figure 3

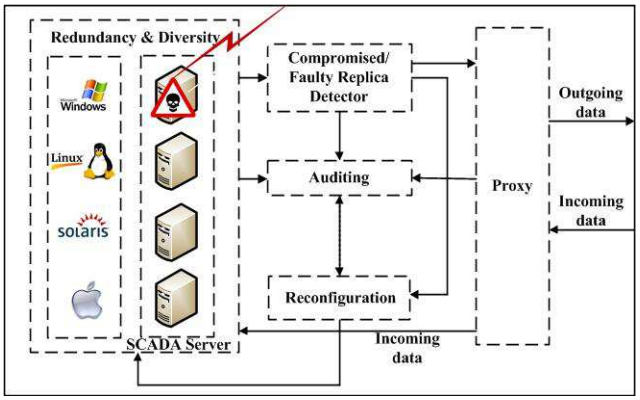

Figure 3: Case study: Trojan horse

depicts a SCADA server that has been compromised by a Trojan horse. It is less probable that more than one replica be infected by this attack since there is diversity in the operating systems of the replicas (all of them are not vulnerable to the same type of attack). We can consider more than one type of attack (not only Trojan horse) on the system but due to the diverse replicas it is highly unlikely that more than one replica affected by the same type of attack. As long as the current redundancy level is less than or equal to the maximum allowed by the ITS, there can be faulty replicas that have been intruded even by more than one type of attack (whether unknown attack or known attack). It is possible that the adversary causes the replica become malfunctioned by running a Trojan and changing some system files which may result in sending inappropriate control commands (in case of automatic operation). However, the command must first pass the consolidator. It is highly probable that the compromised replica(s) being recognized (due to the fact that the replicas have different operating systems, all of them may not be infected by the same attack targeted at a special type of vulnerability, and thus the generated responses would be different) by the inspector (using detection capabilities) and the infected replicas would undergo reactive recovery. Another possibility is that the infected outputs may be masked by the voting module. In addition, process manager running in each replica may detect the infection and trigger the process level rejuvenation. Even if the intrusion tolerance mechanisms fail to detect the intrusion, it is possible that the attack's impact is masked through proactive recovery (whether at process level or system level). During the performance of attack masking measures by the proposed ITS, if the number of required concurrent rejuvenations becomes more than current RL ($RL < MaxRL$), the redundancy level will be increased adaptively (at most up to $MaxRL$).

3.8 Comparison of Existing ITS Architectures and the Proposed ITS Architecture

The intrusion tolerance techniques can be utilized to analyze and compare different intrusion tolerant architectures. Some representatives of existing ITS architectures have been compared by conducting qualitative analyses in [22,26]. Table 1 depicts such analysis with emphasis on the paradigms of intrusion tolerance employed in several ITSs. The spectrum of architectures have distinct features. In this paper, we conduct a comparative analysis to enable a clear reflection of their respective attributes. Moreover, our provided comparison encompasses a higher volume of ITSs.

As it can be seen in Table 1, replication and diversity are the techniques adopted by almost all of the ITSs. The adaptive redundancy that has been employed in the ITS for web servers and our proposed architecture contributes towards reducing the redundancy cost. Although design diversity (e.g., using different operating systems) is the dominant type of diversity used by the ITSs, FOREVER and Crutial can employ time diversity (i.e., rejuvenation introduces diversity). Some ITSs such as FOREVER, Willow and Crutial apply a hybrid recovery method whereas others like SCIT only use proactive recovery. To the best of our knowledge, none of the existing ITSs utilizes the hierarchical recovery strategy introduced in [14]. However, as mentioned earlier, our proposed ITS architecture uses a hybrid and hierarchical recovery approach which decreases the recovery cost. One of the indirection techniques that is widely preferred is the use of proxies as the mediator between the COTS servers and the outside network. Intrusion detection methods whether anomaly-based or signature-based are very common among the ITSs. Byzantine agreement algorithms and secret sharing are the other intrusion tolerant mechanisms that have been implemented in some of the architectures. Among the ITSs shown in Table 1, SCIT and SITAR have drawn more attention in published intrusion tolerance research and investigated with regard to their performance [9,19,21,38]. In addition, interested readers can find useful hints about implementation of the proposed architecture in existing architectures as stated in [3, 5, 44, 46]. For instance, the SCIT has used Virtual Box as the virtual machine monitor and setup four Ubuntu 9.04 server edition on each machine. This approach may be applied to the replicated servers in our proposed architecture.

4 Performance Evaluation

This section is divided into two parts namely, security analysis and cost analysis. Security analysis using a semi Markov model shows the effectiveness of the hybrid rejuvenation approach to enhance the security whereas cost analysis demonstrates the cost-effectiveness of the proposed algorithms.

4.1 Security Analysis

Security quantification of the proposed ITS architecture is needed for assessing the outcome of the desired perfor-

mance measures as well as performance comparison with other architectures. To achieve this goal, a state-space model is developed that incorporates an attacker's behavior along with the system's response to an attack or intrusion [9, 12, 19, 21, 38]. State transition diagrams assist in the evaluation of the transitions impacted by the inter-domain dependencies in the cyber-physical systems. They describe how the attacker's actions cause transitions to failure states [32]. The main advantage of state transition models is the ability to provide a fine-granular system description which includes the dynamic behavior of system [13]. Moreover, these models are tailored to model immense and complex systems such as the critical infrastructures. In this paper we utilize Semi-Markov Process (SMP) which is a generalization of both continuous and discrete time Markov chains which allows arbitrary state holding time distribution functions, probably relying on both the current state and on the state to be visited afterwards [8]. We place focus on the evaluation of the sensitivity of the steady-state availability and Mean Time To Security Failure (MTTSF) as performance measures to variations in model parameters (i.e., p_{I} and h_{G}). The same approach has been used in [19,21,38]. A comparison between the proposed ITS and two of the existing ITSs, namely SITAR and SCIT has been made using these parameters. The aforementioned architectures have been chosen for comparison with our proposed ITS firstly, due to their main focus on improving availability which is the top security priority in critical infrastructures and secondly due to their popularity compared to other existing ITS architectures. The analytical evaluation has been carried out using MATLAB simulator.

4.1.1 System Model

The derived state transition diagram for the proposed ITS is shown in Figure 4. Considering both reactive and proactive intrusion tolerance measures (particularly in terms of the hybrid rejuvenation mechanism as the case in our proposed ITS and also in Crutial) differentiates this state transition diagram from its peer in [19]. Thus, it can serve as a generic model for analyzing the behavior of various ITS architectures.This state transition diagram incorporates different security related states of the ITS and their respective interrelationships. Table 2 presents these security states and their corresponding descriptions.

The system changes from one state to the other during its functional lifespan following from normal usage, abuse, maintenance and corrective measures, failures, and so on. Therefore, the behavior of the system is portrayed as the transitions between the states and each transition corresponds to a specific event. Since the interval between the transition from one state to the other (i.e., state holding time or inter event time) is inclined to be random, its underlying process is defined as a stochastic process [13]. In our system, this process is associated with arbitrary probability distributions, thus, it can be modeled using an SMP.

Table 2: Different states of the system and their respective descriptions

State	Description
G	Good
V	Vulnerable
I	Intruded
DMC	Detected Masked Compromised
UMC	Undetected Masked Compromised
UNC	Undetected Not masked Compromised
DNC	Detected Not masked Compromised
GD	Graceful Degradation
FS	Fail-secure
F	Failed

An SMP can be studied by finding the embedded discrete time Markov chain that requires two sets of parameters [12, 19]:

1) Mean sojourn time (i.e., state holding time) for each state;

2) The transition probabilities between different states.

With respect to Figure 4, the Discrete Time Semi Markov Model (DTSMM) possesses a discrete state space X_s= {G, V, I, UMC, DMC, DNC, UNC, FS, F} for which h_i indicates the mean sojourn time in state $i \in X_s$ and p_{ij} represents the transition probabilities between states i and j $(i, j \in X_s)$.

4.1.2 Availability Formulation and Analysis

We analyze the sensitivity of the availability with respect to two parameters, including the probability of intrusion (p_{I}) and the mean time to resist becoming vulnerable to intrusions (h_{G}) [35, 36]. The steady-state availability A is defined as the probability that the system is in one of normal functioning states. One approach to determine the availability is to pinpoint what the unavailable states (i.e., states FS, F and UNC) are. Thus, the steady-state availability A can be formulated as,

$$A = 1 - (\pi_{\text{UNC}} + \pi_{\text{FS}} + \pi_{\text{F}}) \tag{1}$$

where π_i, $i \in \{UNC, FS, F\}$ denotes the steady-state probability of being in state i for the SMP, that can be computed as,

$$\pi_i = \frac{\nu_i h_i}{\sum \nu_j h_j}, \quad i, j \in X_s \tag{2}$$

where h_i indicates the mean state holding time in state i and ν_i denotes the embedded Discrete Time Markov chain (DTMC) steady-state probability in state i. We can derive ν_is from the following two equations,

$$\nu = \nu \cdot P \tag{3}$$

$$\sum_i \nu_i = 1, \quad i \in X_s \tag{4}$$

Table 1: Comparative analysis of intrusion tolerant architectures (Y: Yes, N: No, O: Optional)

	COCA	DIT	Willow	SITAR	SCIT	MAFTIA	Crutial	FOREVER	ITS for web servers	Proposed architecture
Replication	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Diversity	N	Y	Y	Y	O	Y	Y	Y	Y	Y
Proactive Recovery	Y	Y	Y	N	Y	N	Y	Y	Y	Y
Reactive Recovery	N	Y	Y	Y	N	Y	Y	Y	Y	Y
Hierarchical Recovery	N	N	N	N	N	N	N	N	N	Y
Voting/BFT Agreement	Y	Y	N	Y	N	Y	Y	Y	N	Y
Proxy	N	Y	N	Y	N	N	N	N	Y	Y
Intrusion Detection Capabilities	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
Secret Sharing	Y	N	N	N	N	Y	N	N	N	N

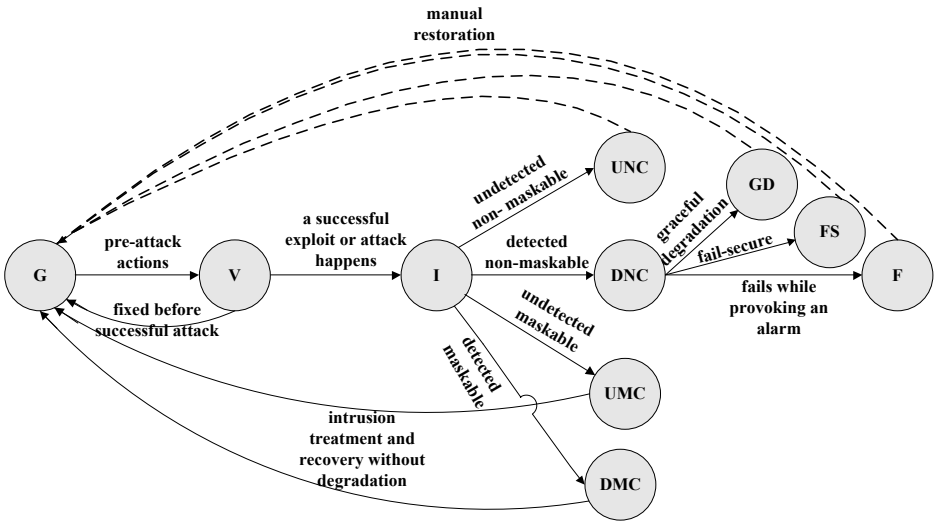

Figure 4: State transition diagram for the proposed ITS

where the P is the transition probability matrix of the corresponding DTMC for the proposed ITS,

$$P = \begin{array}{c} \\ G \\ V \\ I \\ DMC \\ UNC \\ UMC \\ DNC \\ FS \\ GD \\ F \end{array} \begin{array}{c} \begin{array}{cccccccccc} G & V & I & DMC & UNC & UMC & DNC & FS & GD & F \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1-p_{\rm I} & 0 & p_{\rm I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{\rm DM} & p_{\rm UN} & p_{\rm UM} & p_{\rm DN} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_{\rm FS} & p_{\rm GD} & p_{\rm F} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

In this paper, the mean state holding times h_i for all the states of DTMC have been assumed to have the same values as [19] except for the state UMC which is a new state (corresponding to proactive recovery) for our proposed ITS.

Finally, by using Equations (1)-(4), the steady-state availability ($A_{\rm P}$) of our proposed ITS is computed as,

$$A_{\rm p} = 1 - \frac{h_{\rm UNC}p_{\rm I}p_{\rm UN} + h_{\rm F}p_{\rm I}p_{\rm DN}p_{\rm F} + h_{\rm FS}p_{\rm I}p_{\rm DN}p_{\rm FS}}{h_{\rm G} + h_{\rm V} + p_{\rm I}(h_{\rm I} + h_{\rm DMC}p_{\rm DM} + h_{\rm UNC}p_{\rm UN} + h_{\rm UMC}p_{\rm UM} + h_{\rm DNC}p_{\rm DN} + h_{\rm GD}p_{\rm DN}p_{\rm GD} + h_{\rm FS}p_{\rm DN}p_{\rm FS} + h_{\rm F}p_{\rm DN}p_{\rm F})} \quad (5)$$

In a similar manner, the steady-state availability for SITAR ($A_{\rm SITAR}$) and SCIT ($A_{\rm SCIT}$) are derived as,

$$A_{\rm SITAR} = 1 - \frac{h_{\rm UNC}p_{\rm I}p_{\rm UN+}h_{F}p_{\rm I}p_{\rm DN}p_{\rm F} + h_{\rm FS}p_{\rm I}p_{\rm DN}p_{\rm FS}}{h_{\rm G} + h_{\rm V} + p_{\rm I}(h_{\rm I} + h_{\rm DMC}p_{\rm DM} + h_{\rm UNC}p_{\rm UN} + h_{\rm DNC}p_{\rm DN} + h_{\rm GD}p_{\rm DN}p_{\rm GD} + h_{\rm FS}p_{\rm DN}p_{\rm FS} + h_{\rm F}p_{\rm DN}p_{\rm F})} \quad (6)$$

$$A_{\rm SCIT} = 1 - \frac{h_{\rm F}p_{\rm I}p_{\rm F}}{h_{\rm G} + h_{\rm V} + p_{\rm I}(h_{\rm I} + h_{\rm UMC}p_{\rm UM} + h_{\rm F}p_{\rm F})} \quad (7)$$

It should be pointed out that some of the transition probabilities may have different values or even may not be applicable for all three ITSs. This follows from the fact that the three ITSs do not possess the same state space (DTSMM's state space for SITAR does not include state UMC whereas SCIT does not contain the states DMC, DNC, UNC, GD and FS).

In Figure 5, the availability for SCIT falls sharply when the probability of intrusion increases compared to the other two ITSs. This is due to the fact that SCIT lacks detection capabilities and only uses periodic rejuvenation. SCIT may alleviate the impacts of attacks on the system, but it does not identify the type of attack (if it is a known attack) to deal with it more appropriately (e.g., triggering recovery when an attack is detected). However, its advantage is dealing with unknown attacks. Considering Figure 5, availability performance of the proposed ITS

shows 0.6% and 36% improvement compared to SITAR and SCIT respectively. Figure 6 shows the positive impact of increasing the time that the system is in the good state on the availability (i.e., the availability increases as the h_{G} rises). For larger values of h_{G}, there is a slight difference in availability performance of the three ITS. In this figure, availability performance of the proposed ITS presents 0.3% and 9% improvement compared to SITAR and SCIT respectively. This is mostly due the use of the hybrid and hierarchical recovery approach in the proposed ITS. While proactive recovery (reflected in state UMC in Figure 4) deals with dormant faults in the system or unknown attacks against the system, reactive recovery (reflected in state DMC in Figure 4) eliminates the effects of known attacks on the system.

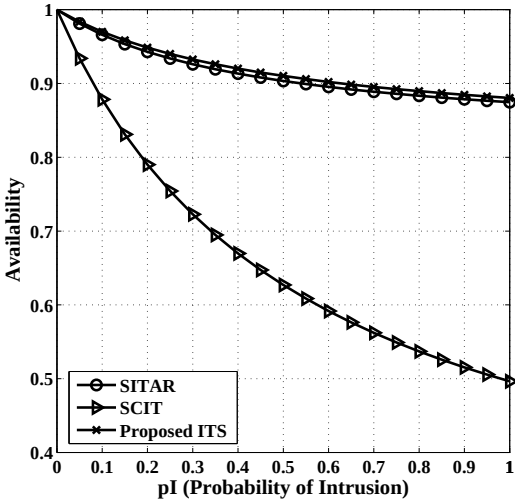

Figure 5: Impact of probability of intrusion on availability

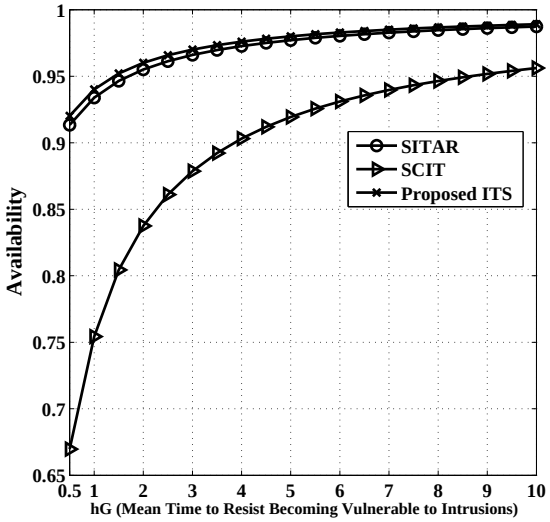

Figure 6: Impact of state holding time in state G on availability

4.1.3 MTTSF Formulation and Analysis

MTTSF is defined as the mean elapsed time for the system to reach one of the security-compromised states (also called absorbing states), provided that the system begins in state G [19]. In the context of critical infrastructure, it can demonstrate the resiliency and robustness of the proposed ITS for control centers' critical components such as SCADA and application servers. A secure and robust ITS is expected to have a high MTTSF when facing intrusions. Using a similar approach to availability analysis, we analyze the MTTSF with regard to p_{I} and h_{G} parameters. We take advantage of an SMP with absorbing and transient states. In the state transition diagram shown in Figure 4, the set of states $X_a = \{\text{UNC, GD, FS, F}\}$ are considered as the absorbing states (i.e., the probability of moving out of these states is zero). These states indicate the security compromised states. The rest of the states are called transient states and denoted by $X_t = \{\text{G, V, I, UMC, DMC, DNC}\}$. The transition probability Matrix M exhibits the transition probabilities between the transient states (i.e., Q) and the states originating from transient states to absorbing states (i.e., C) in an organized form.

$$M = \begin{pmatrix} Q & | & C \\ -- & | & -- \\ 0 & | & I \end{pmatrix}$$

Matrixes Q and C are as follows:

$$Q = \begin{array}{c} \\ G \\ V \\ I \\ DMC \\ UMC \\ DNC \end{array} \begin{array}{c} \begin{array}{cccccc} G & V & I & DMC & UMC & DNC \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1-p_{\text{I}} & 0 & p_{\text{I}} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{\text{DM}} & p_{\text{UM}} & p_{\text{DN}} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$C = \begin{array}{c} \\ G \\ V \\ I \\ DMC \\ UMC \\ DNC \end{array} \begin{array}{c} \begin{array}{cccc} UNC & FS & GD & F \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ p_{\text{UN}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & p_{\text{FS}} & p_{\text{GD}} & p_{\text{F}} \end{pmatrix} \end{array}$$

Then we can compute the MTTSF by the following formula [19],

$$MTTSF = \sum_{i \in X_t} V_i h_i \tag{8}$$

where V_i indicates the average number of times the transient state i has been visited before the DTMC arrives at one of the absorbing states and h_i indicates the mean state holding time in state i.

Let q_i be the probability of start in state i (here, it is assumed that the DTMC starts in state G) and q_{ji} be the transition probability from the transient state j to the transient state i. So, the V_is can be computed through solving the system of equations,

$$V_i = q_i + \sum_j V_j q_{ji}, \quad i, j \in X_t \tag{9}$$

Finally, we use Equation (8) to calculate the MTTSF for the proposed ITS as,

$$M_{\rm P} = \frac{h_{\rm G}p_{\rm I}^{-1} + h_{\rm V}p_{\rm I}^{-1} + h_{\rm I} + h_{\rm DMC}p_{\rm DM} + h_{\rm UMC}p_{\rm UM} + h_{\rm DNC}p_{\rm DN}}{1 - p_{\rm DM} - p_{\rm UM}} \tag{10}$$

Using the same approach, we derive the expression for SITAR [19] and SCIT as follows,

$$M_{\rm SITAR} = \frac{h_{\rm G}p_{\rm I}^{-1} + h_{\rm V}p_{\rm I}^{-1} + h_{\rm I} + h_{\rm DMC}p_{\rm DM} + h_{\rm DNC}p_{\rm DN}}{1 - p_{\rm DM}}$$

$$M_{\rm SCIT} = \frac{h_{\rm G}p_{\rm I}^{-1} + h_{\rm V}p_{\rm I}^{-1} + h_{\rm I} + h_{\rm UMC}p_{\rm UM}}{1 - p_{\rm UM}}$$

As illustrated in Figure 7, MTTSF has a reciprocal relationship with the probability of intrusion, i.e., it decreases as the probability of intrusion rises. The proposed ITS architecture shows improved MTTSF with regard to $p_{\rm I}$ (17% compared to SITAR and 2% compared to SCIT) since it has more security features (e.g., proactive and reactive recovery) and thus more system states (corresponding to tolerance measures) when dealing with intrusions.

As shown in Figure 8, MTTSF ascends when the system spends more time in state G. In this graph, the proactive rejuvenation in SCIT seems to have more effects on the MTTSF when increasing the $h_{\rm G}$ in comparison with the reactive rejuvenation in SITAR. The acquired results show that the stability of our proposed ITS is better than the others. The improvement in MTTSF performance is 16% and 0.8% compared to SITAR and SCIT respectively. The acquired results for MTTSF also prove the security enhancement of the proposed architecture compared with the other two systems. As mentioned in availability analysis, the masking capabilities ($p_{\rm M}$) of the proposed ITS have been improved.

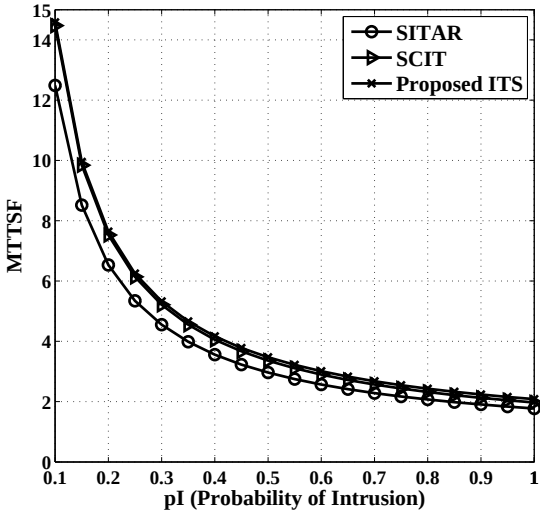

Figure 7: Impact of probability of intrusion on MTTSF

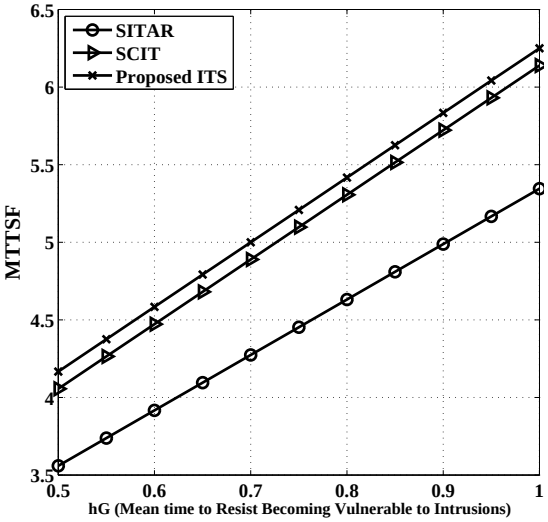

Figure 8: Impact of state holding time in state G on MTTSF

4.2 Cost Analysis

One of the downside of using intrusion tolerance is the substantial cost incurred by the underlying techniques such as redundancy and rejuvenation. Thus, the aforementioned techniques should be used meticulously. The proposed algorithms provide assistance for using redundancy and rejuvenation in an efficient manner to decrease the incurred cost. Since there is a trade-off between cost and security, the proposed algorithms make an effort to maintain an accepted level of security while reducing the associated cost. We considered two types of cost (in terms of overhead) as follows:

- Rejuvenation cost: The rejuvenation cost is represented by the number of system level rejuvenations.
- Redundancy cost: The redundancy level can be considered as a performance metric that can represent the incurred redundancy cost. It should be noted that the redundancy level is influenced by both proactive and reactive rejuvenation mechanisms since they affect the number of concurrent rejuvenations.

To demonstrate the efficiency of the proposed algorithms, a cost analysis is conducted using OMNeT++ simulator. We consider six different simulation scenarios as follows:

- S_1: Only system level proactive rejuvenation (as is the case in SCIT)
- S_2: Hierarchical proactive rejuvenation
- S_3: Only system level reactive rejuvenation (it is used as a part of reconfiguration measures in SITAR)
- S_4: Hierarchical reactive rejuvenation
- S_5: Hybrid system level rejuvenation (as is the case in Crutial)
- S_6: Hybrid and hierarchical rejuvenation (our proposed algorithms)

In all scenarios, the adaptive redundancy level algorithm is used. The primary factor that differentiates one scenario from another is the type of employed rejuvenation mechanisms. Moreover, it is assumed that the system is under sustained attack (i.e., the incoming traffic to the ITS always includes attack traffic as well). S_6 is the scenario that completely shows the proposed algorithms. S_1 to S_6 serve as proofs of concept for the impact of using different recovery mechanisms on the incurred cost and security of the system.

Other assumptions and simulation parameters are as follows. The number of replicas (whether active or under recovery), denoted by RL, at each moment during simulation is $2f + 1 + k$ with the minimum value of 4 (i.e., $f, k = 1$) and maximum value of 6 (i.e., $MaxRL$).The traffic distribution follows Poisson process. The time between arrivals are exponentially distributed with rate 2ms. It is assumed that with current parameters, no failure happens (i.e., meaning the attackers are not able to compromise more than f replicas in a way that the infected replicas cannot be detected or the impact of the intrusion cannot be masked). So, the system will always be available and operates correctly. Moreover, the possible results will be calculated as the average over 10 runs except for the S_1 and S_2 in which no randomness is used and therefore repeating the experiment will not affect the results. The default parameters used in our simulation are tabulated in Table 3.

Table 3: Simulation assumptions

Parameter	value
Number of runs	10
f	1
k	1,2,3
$MaxRL$ ($MaxDL$)	6
Redundancy level (RL)	4-6
Number of active replicas at the beginning of the simulation	4
Simulation time	21600s
Number of critical processes in each replica	5
Number of backups for each process in per replica	3
Replica processing time	exponential(0.003s)
Process level reactive rejuvenation period	300s
Process level proactive rejuvenation period	240s
System level rejuvenation time	600s

4.2.1 Simulation Results and Discussion

This section demonstrates the efficacy of the proposed algorithms using different scenarios.

Rejuvenation cost: Figure 9 illustrates the number of performed system level rejuvenations in distinctive scenarios respectively. It is shown that the total number of carried out system level rejuvenations have been decreased in S_6 (in which our proposed hybrid and hierarchical recovery is used). Thus, the rejuvenation cost is reduced while the security and resiliency enhanced using a hybrid and hierarchical rejuvenation approach. Although in S_2 the number and time duration of recoveries have minimum value, the level of security and tolerance is not satisfactory because the system only uses hierarchical proactive recovery and it does not possess detection capability.

But as a virtue, S_2 shows the effectiveness of hierarchical proactive recovery in terms of rejuvenation cost reduction compared to S_1. This is also applicable to S_4 which demonstrates the desired effect of hierarchical reactive recovery compared to S_3 in which only system level reactive recovery is employed. Nevertheless, S_4 still suffers the problem of not being able to handle unknown and novel attacks as well as false positives. Also, S_5 (which uses hybrid system level rejuvenation) shows reduced rejuvenation overhead compared to S_1 and S_3 in which proactive and reactive system level recovery have been employed respectively.With regard to the aforementioned issues, we can draw the conclusion that the limitations of the first five scenarios are alleviated in S_6.

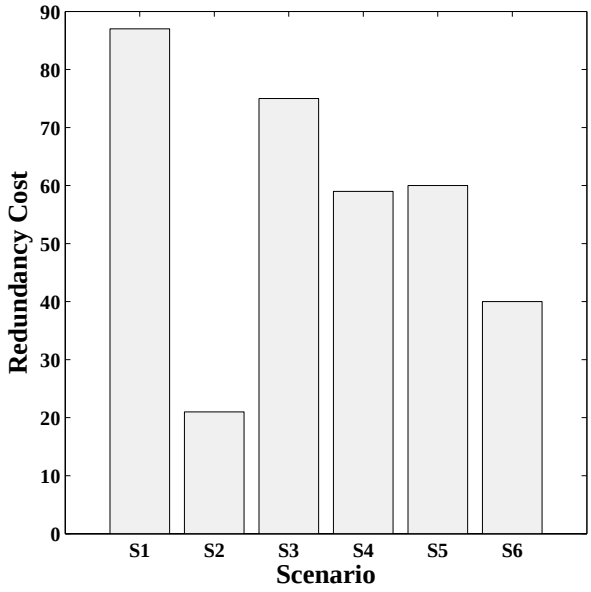

Figure 9: Rejuvenation cost in terms of the number of performed system level rejuvenations in each scenario

Figure 10 illustrates a detailed view of the provided results in Figure 9. In fact, the total number of rejuvenations in each scenario is gained from calculating the average values of these parameters for each replica over 10 simulation runs and then summing the averages (except for S_1 and S_2 as mentioned earlier). Furthermore, the redundancy level is reflected in Figure 10 by showing zero number of recoveries and zero rejuvenation time for the sixth replica in S_1 and S_2, i.e., the redundancy level is 5 ($RL = 5$) in these scenarios. This is mostly due to the inability of detection in these scenarios that would result in less number of performed concurrent rejuvenations and consequently less changes in the in the redundancy level.

Process level and system level rejuvenations: Figure 11 displays a view of the average number of performed

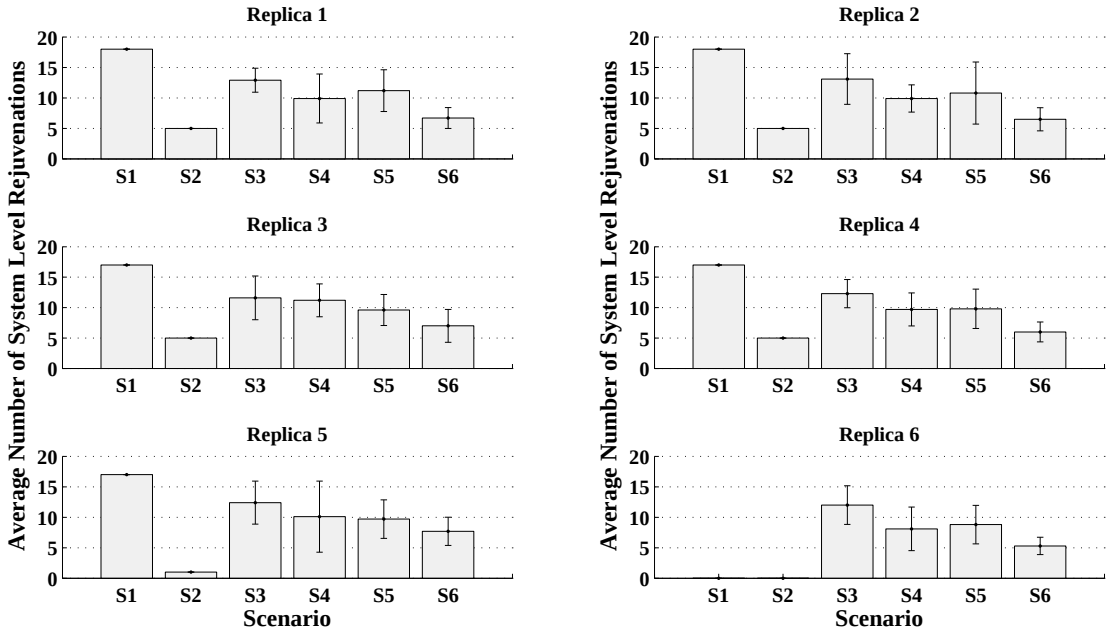

Figure 10: Average number of performed system level rejuvenations per replica for each scenario

process level rejuvenations and the average number of system level rejuvenations in scenarios that features hierarchical recovery, i.e., S_2, S_4 and S_6. In all three scenarios, most of the rejuvenations are carried out at process level. This shows the impact of hierarchical recovery on the reduction of recovery overhead resulting from system level rejuvenations (in S_1, S_3 and S_5 all the rejuvenations are done at system level). In S_6, the average number of process level rejuvenations has been increased compared to S_4 and consequently the average number of system level recoveries and rejuvenation overhead has been decreased. Although in S_2, the average number of system level rejuvenations is less than its peer in S_6, it does not satisfy the required level of security due to lack of detection abilities. Moreover, the average number of system level rejuvenations in S_2 depends on the frequency of triggering proactive recovery as well as the number of redundant processes related to each critical process in a replica. Assume there is a fixed number of redundant (backup) processes for each process. In this case, increasing the frequency of triggering proactive recoveries would result in the rise in the number of system level rejuvenations. In contrast, if the frequency of proactive recovery is constant, then increasing the number of backup processes assigned to each process, would reduce the number of system level recoveries (while increasing the number of process level recoveries). As it is shown in Table 3, the number of replicated processes is 4 for each crucial process in a replica. This means that it is expected to have on average of 4 process level rejuvenations per system level rejuvenation in S_2. This deterministic relation follows from the fact that S_2 only employs proactive recovery approach and it does not involve any randomness. Figure 11 confirms this matter for S_2. Although this relation is not deterministic for S_4 and S_6, with regard to Figure 11 we can roughly conclude that 2 and 3 process level recoveries are performed per system level recovery in S_4 and S_6 respectively. Obviously, S_6 shows less system level recovery cost in this case (more number of process level recoveries per system level recovery).

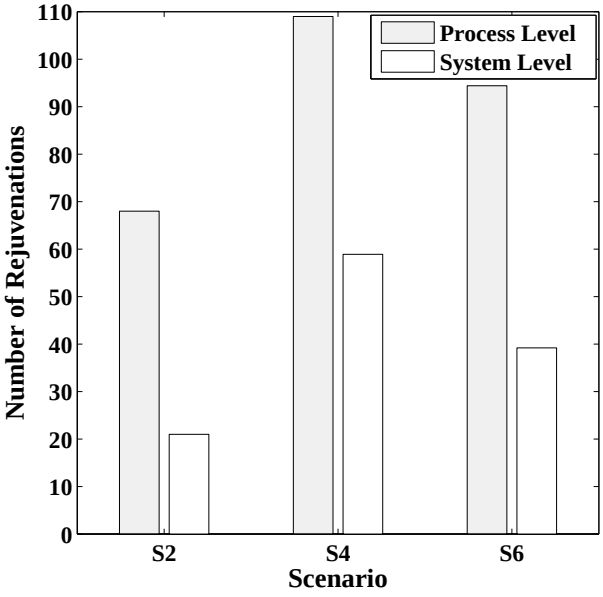

Figure 11: Average performed process level rejuvenations compared to system level rejuvenations

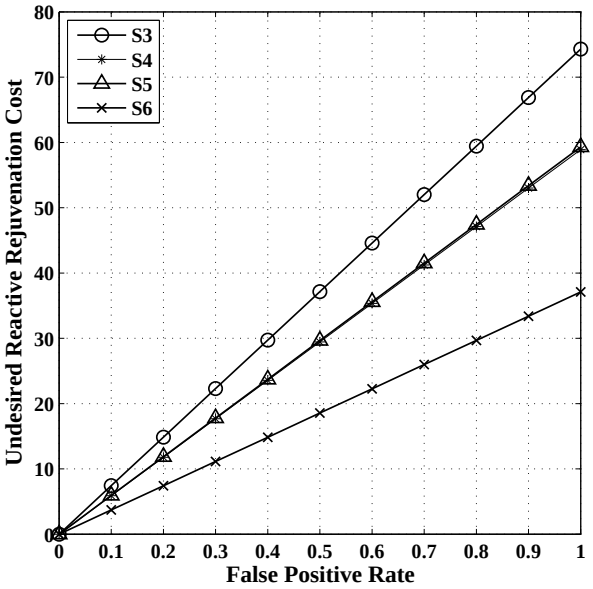

Figure 12: The impact of false positive rate on the undesired reactive recovery overhead

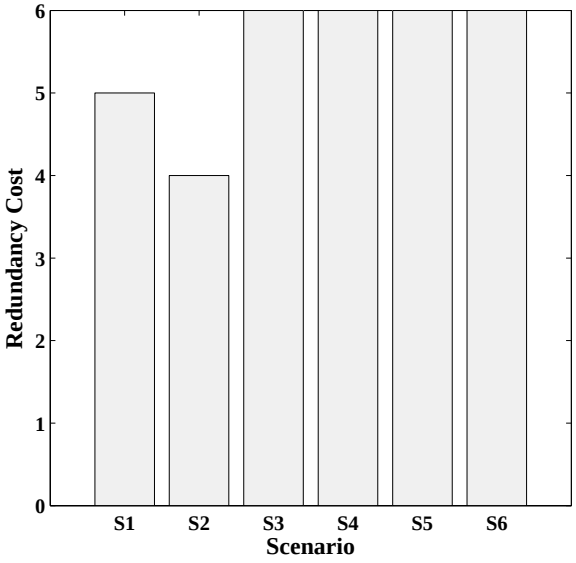

Figure 13: Average redundancy level at the end of simulation in each scenario

Undesired reactive rejuvenation cost: The false positive rate of the underlying detection mechanisms is an important factor that affects the reactive rejuvenation overhead. With respect to false positive and true positives, we can divide the reactive rejuvenation cost into two parts, namely desired reactive rejuvenation cost resulting from true positives and undesired reactive rejuvenation cost because of false positives. Therefore, as it is shown in Figure 12, the less the false positive rate is, the less the undesired rejuvenation cost would be in scenarios S_3-S_6. S_1 and S_2 are not shown in this figure since they only contain proactive recovery. S_4 and S_5 show almost the same behaviour since proactive rejuvenation in S_5 has minor impact on decreasing the number of system level reactive recoveries in this simulation; however, as mentioned earlier the security has been enhanced in S_5. Moreover, because the number of performed reactive recoveries in S_3-S_5 are more than S_6, the undesired reactive rejuvenation cost is increased sharply when the false positive rate rises. Thus, a fool-proof ITS is expected to have intrusion detection method with low false positive rate which subsequently result in less undesired reactive rejuvenation cost.

Redundancy cost: The average redundancy level at the end of simulation in different scenarios shows the effect of recovery mechanisms on the redundancy level (when the system is under attack). As shown in Figure 13, while the redundancy levels have the values of 5 and 4 for S_1 and S_2 respectively, the other four scenarios end up with level 6 which is the maximum value. Thus, in this simulation the effect of reactive recovery and hybrid recovery mechanisms on the redundancy level is more than the proactive recovery mechanism. When the system is armed with detection capabilities and reactive rejuvenation as well as proactive recovery, the possibility of increase in redundancy level is more (more attacks can be handled successful by the ITS). This follows from the fact that the ITS tries to adapt itself to the situation to maintain the desired availability. Thus, it increases the number of simultaneous rejuvenations which will have direct impact on the redundancy level.

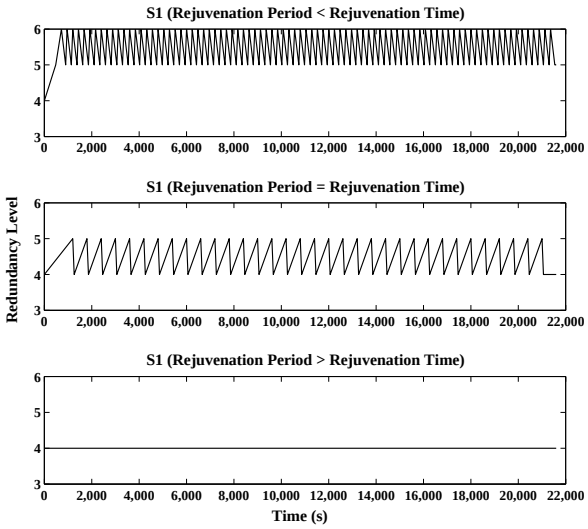

Figure 14: Proactive rejuvenation frequency impact on the redundancy level (S_1)

As a proof of concept, the impact of proactive recovery (S_1 and S2) on the redundancy level changes can be

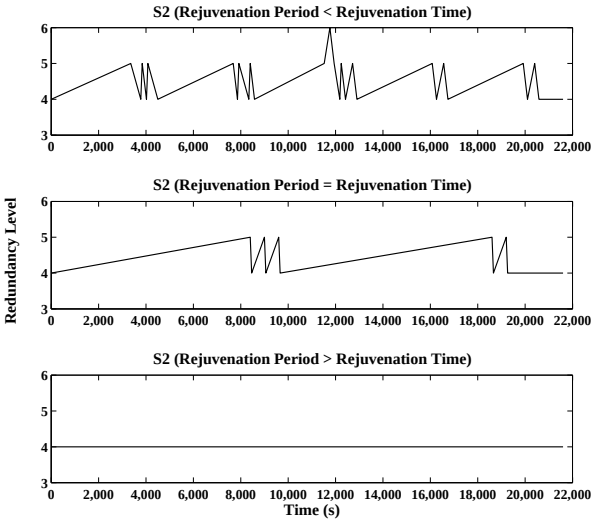

Figure 15: Proactive rejuvenation frequency impact on the redundancy level (S_2)

shown. As it has been depicted in Figures 14 and 15, the frequency of proactive rejuvenation has a direct impact on the redundancy level and subsequently on the number of concurrent system level rejuvenations. Three cases can be considered as follows:

1) Rejuvenation period < rejuvenation time: if this condition is met, then the probability of redundancy level changes is increased. This is due to the fact that most of the time the number of concurrent rejuvenations (i.e., k) is more than 1. That is, while one or more rejuvenations are in progress, another rejuvenation is triggered. In Figure 14, the number of redundancy level changes is more than its peer in Figure 15. This follows from the adoption of hierarchical proactive recovery in S_2 compared to S_1 that decreases the number of system level rejuvenations and the possibility of having more than one simultaneous rejuvenation. This also affects the maximum redundancy level during simulation time which is 6 for S_1 and 5 almost all the time for S_2. With regard to the aforementioned issues, the conclusion can be drawn that the redundancy cost has been declined in S_2.

2) Rejuvenation period = rejuvenation time: In this situation, the maximum redundancy level is 5 for both S_1 and S_2. Also, the shifts in redundancy level have been reduced for both scenarios. However, it is more tangible in S_2 which involves hierarchical rejuvenation. Obviously, S_2 shows less redundancy overhead compared to S_1. Concerning the incurred redundancy cost, this condition can be considered as a moderate situation. Nevertheless, if the time needed by an adversary to break into the system is less than the time between two successive recoveries (between two process level or system level recoveries or between a process level and a system level recovery) and the maximum redundancy level (i.e., 6) is reached, the required security level of the system will not be satisfied.

3) Rejuvenation period > rejuvenation time: Figures 14 and 15 illustrate no redundancy level change for S_1 and S_2 in this situation. So, the redundancy cost is minimum (the maximum redundancy level is 4 and the value of k is always 1). In spite of this advantage, as mentioned before, the security requirements may not be satisfied (the intrinsic trade-off between security and cost).

5 Conclusion

Cyber security of critical infrastructures is a hot research area due to the fact that these systems are tightly coupled with ICT. The security incidents in cyber domain may affect the physical world and may subsequently lead to nationwide and disastrous consequences such as cascaded failures in the whole system. The possible ramifications would be financial loss or even loss of lives. Thus, this paper provided an in-depth research on the significance of using intrusion tolerance as a security approach to improve the security of critical infrastructures' control systems. An ITS architecture was proposed to be adopted in control centers' critical components. Using different intrusion tolerance techniques such as replication and diversity (along with dynamic redundancy level), hybrid and hierarchical recovery made the proposed ITS outperform two of well-known architectures, namely SITAR and SCIT. To investigate the effectiveness of the aforementioned features, analytical modelling and cost analysis were conducted. The acquired results demonstrated decrease in the incurred cost by intrusion tolerance techniques enhancement while maintaining an appropriate level of security.

References

[1] N. Al Ebri, J. Baek, and C. Y. Yeun, "Study on secret sharing schemes (SSS) and their applications," in *International Conference for Internet Technology and Secured Transactions (ICITST'11)*, pp. 40–45, 2011.

[2] M. Badra and S. Zeadally, "An improved privacy solution for the smart grid," *International Journal of Network Security*, vol. 17, no. 4, pp. 1–8, 2015.

[3] A. K. Bangalore and A. K. Sood, "Securing Web servers using self cleansing intrusion tolerance (SCIT)," in *Second International Conference on Dependability (DEPEND'09)*, pp. 60–65, 2009.

[4] A. N. Bessani, "From Byzantine fault tolerance to intrusion tolerance (a position paper)," in *IEEE/IFIP*

41st International Conference on Dependable Systems and Networks Workshops (DSN-W'11), pp. 15–18, 2011.

[5] A. N. Bessani, P. Sousa, M. Correia, N. F. Neves, and P. Verissimo, "The crutial way of critical infrastructure protection," *IEEE Security & Privacy*, vol. 6, no. 6, pp. 44–51, 2008.

[6] T. Y. Chang, M. S. Hwang, W. P. Yang, "An improved multi-stage secret sharing scheme based on the factorization problem," *Information Technology and Control*, vol. 40, no. 3, pp. 246–251, 2011.

[7] Y. Deswarte and D. Powell, "Internet security: An intrusion-tolerance approach," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 432–441, 2006.

[8] S. Distefano, F. Longo, and K. S. Trivedi, "Investigating dynamic reliability and availability through state–space models," *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3701–3716, 2012.

[9] T. Dohi and T. Uemura, "An adaptive mode control algorithm of a scalable intrusion tolerant architecture," *Journal of Computer and System Sciences*, vol. 78, no. 6, pp. 1751–1774, 2012.

[10] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "Analysis of operating system diversity for intrusion tolerance," *Software: Practice and Experience*, 2013.

[11] D. Ghosh, R. Sharman, H. R. Rao, and S. Upadhyaya, "Self-healing systems survey and synthesis," *Decision Support Systems*, vol. 42, no. 4, pp. 2164–2185, 2007.

[12] C. Griffin, B. Madan, and T. Trivedi, "State space approach to security quantification," in *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*, vol. 2, pp. 83–88, 2005.

[13] B. E. Helvik, K. Sallhammar, and S. J. Knapskog, "8 Chapter - Integrated Dependability and Security Evaluation Using Game Theory and Markov Models," in *Information Assurance*, pp. 209–245, Morgan Kaufmann, Burlington, 2008.

[14] J. H. Huang and F.-F. Wang, "The strategy of proactive-reactive intrusion tolerance recovery based on hierarchical model," in *Web Information Systems and Mining*, vol. 6987 of *Lecture Notes in Computer Science*, pp. 283–293, 2011.

[15] P. Kabiri and A. A. Ghorbani, "Research on intrusion detection and response: A survey," *International Journal of Network Security*, vol. 1, no. 2, pp. 84–102, 2005.

[16] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *37th Annual Conference on IEEE Industrial Electronics Society (IECON'11)*, pp. 4490–4494, 2011.

[17] J. C. Knight, J. Hill, P. Varner, A. L. Wolf, D. Heimbigner, and P. Devanbu, "Willow system demonstration," in *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX'03)*, vol. 2, pp. 123–125, 2003.

[18] X. Li, X. Liang, R. Lu, X. Shen, X. Lin, and H. Zhu, "Securing smart grid: Cyber attacks, countermeasures, and challenges," *IEEE Communications Magazine*, vol. 50, no. 8, pp. 38–45, 2012.

[19] B. B. Madan, K. Goševa-Popstojanova, K. Vaidyanathan, and K. S. Trivedi, "A method for modeling and quantifying the security attributes of intrusion tolerant systems," *Performance Evaluation*, vol. 56, no. 1-4, pp. 167–186, 2004.

[20] A. Nagarajan and A. Sood, "SCIT and IDS architectures for reduced data ex-filtration," in *International Conference on Dependable Systems and Networks Workshops (DSN-W'10)*, pp. 164–169, 2010.

[21] Q. Nguyen and A. Sood, "Quantitative approach to tuning of a time-based intrusion-tolerant system architecture," in *3rd Workshop Recent Advances on Intrusion-Tolerant Systems (WRAITS'09)*, pp. 132–139, 2009.

[22] Q. L. Nguyen and A. Sood, "A comparison of intrusion-tolerant system architectures," *IEEE Security and Privacy*, vol. 9, no. 4, pp. 24–31, 2011.

[23] A. Nicholson, S. Webber, S. Dyer, T. Patel, and H. Janicke, "SCADA security in the light of cyber-warfare," *Computers & Security*, vol. 31, no. 4, pp. 418–436, 2012.

[24] D. Niyato, P. Wang, and E. Hossain, "Reliability analysis and redundancy design of smart grid wireless communications system for demand side management," *IEEE Wireless Communications*, vol. 19, no. 3, pp. 38–46, 2012.

[25] J. Pieprzyk and X. M. Zhang, "Ideal secret sharing schemes from permutations," *International Journal of Network Security*, vol. 2, no. 3, pp. 238–244, 2006.

[26] S. B. E. Raj and G. Varghese, "Analysis of intrusion-tolerant architectures for web servers," in *International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT'11)*, pp. 998–1003, 2011.

[27] A. Saidane, V. Nicomette, and Y. Deswarte, "The design of a generic intrusion-tolerant architecture for web servers," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 1, pp. 45–58, 2009.

[28] J. T. Seo and C. Lee, "The green defenders," *IEEE Power and Energy Magazine*, vol. 9, no. 1, pp. 82–90, 2011.

[29] sKyWIper Analysis Team, "sKyWlper (a.k.a. Flame a.k.a. Flamer): A complex malware for targeted attacks," tech. rep., Laboratory of Cryptography and System Security (CrySyS Lab), 2012.

[30] P. Sousa, A. N. Bessani, and R. R. Obelheiro, "The FOREVER service for fault/intrusion removal," in *Proceedings of the 2nd workshop on Recent advances on intrusiton-tolerant systems (WRAITS;08)*, pp. 1–6, New York, USA, 2008.

[31] P. Sousa, A. N. Bessani, M. Correia, and N. F. Neves, "Highly available intrusion-tolerant services with proactive-reactive recovery," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 4, pp. 452–465, 2010.

[32] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber physical system security for the electric power grid," *Proceedings of the IEEE*, vol. 100, pp. 210–224, Jan. 2012.

[33] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245–1265, 2010.

[34] R. Stroud, I. Welch, J. Warne, and P. Ryan, "A qualitative analysis of the intrusion-tolerance capabilities of the MAFTIA architecture," in *International Conference on Dependable Systems and Networks (DSN'04)*, pp. 453–461, 2004.

[35] M. Tanha and F. Hashim, "An intrusion tolerant system for improving availability in smart grid control centers," in *2012 18th IEEE International Conference on Networks (ICON'12)*, pp. 434–440, 2012.

[36] M. Tanha and F. Hashim, "Towards a secure and available smart grid using intrusion tolerance," in *Internet and Distributed Computing Systems*, Lecture Notes in Computer Science, pp. 188–201, 2012.

[37] C. W. Ten, G. Manimaran, and C. C. Liu, "Cybersecurity for critical infrastructures: Attack and defense modeling," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 40, no. 4, pp. 853–865, 2010.

[38] T. Uemura, T. Dohi, and N. Kaio, "Availability analysis of an intrusion tolerant distributed server system with preventive maintenance," *IEEE Transactions on Reliability*, vol. 59, no. 1, pp. 18–29, 2010.

[39] A. Valdes, M. Almgren, S. Cheung, Y. Deswarte, B. Dutertre, J. Levy, H. Saidi, V. Stavridou, and T. E. Uribe, "Dependable intrusion tolerance: technology demo," in *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX'03)*, vol. 2, pp. 128–130 vol.2, 2003.

[40] P. E. Verissimo, N. F. Neves, C. Cachin, J. Poritz, D. Powell, Y. Deswarte, R. Stroud, and I. Welch, "Intrusion-tolerant middleware: the road to automatic security," *IEEE Security & Privacy*, vol. 4, no. 4, pp. 54–62, 2006.

[41] P. Veríssimo, N. Neves, and M. Correia, "Intrusion-tolerant architectures: Concepts and design," in *Architecting Dependable Systems*, vol. 2677 of *Lecture Notes in Computer Science*, pp. 3–36, 2003.

[42] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Verissimo, "Efficient Byzantine fault-tolerance," *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 16–30, 2013.

[43] F. Wang, R. Uppalli, and C. Killian, "Analysis of techniques for building intrusion tolerant server systems," in *IEEE Military Communications Conference (MILCOM'03)*, vol. 2, pp. 729–734 Vol.2, 2003.

[44] F. Wang, F. Jou, F. Gong, C. Sargor, K. Goseva-Popstojanova, and K. Trivedi, "SITAR: a scalable intrusion-tolerant architecture for distributed services," in *Foundations of Intrusion Tolerant Systems [Organically Assured and Survivable Information Systems]*, pp. 359–367, 2003.

[45] R. Wang, F. Wang, and G. T. Byrd, "Design and implementation of acceptance monitor for building intrusion tolerant systems," *Software - Practice and Experience*, vol. 33, no. 14, pp. 1399–1417, 2003.

[46] Y. S. Wang and L. Wang, "Secure server switching system," in *Computer Engineering and Applications (ICCEA'10), 2010 Second International Conference on*, vol. 1, pp. 224–228, Mar. 2010.

[47] F. Zhao, M. Li, W. Qiang, H. Jin, D. Zou, and Q. Zhang, "Proactive recovery approach for intrusion tolerance with dynamic configuration of physical and virtual replicas," *Security and Communication Networks*, vol. 5, no. 10, pp. 1169–1180, 2012.

[48] L. Zhou, F. B. Schneider, and R. Van Renesse, "COCA: A secure distributed online certification authority," *ACM Transactions on Computer Systems,*, vol. 20, pp. 329–368, Nov. 2002.

Maryam Tanha is currently perusing her PhD in the Department of Computer Science, Faculty of Engineering, University of Victoria. She received her M.Sc. in Communication and Network Engineering from Universiti Putra Malaysia in 2013. Her research activities are focused on Software Defined Networking and survivability analysis, particularly for critical infrastructures. She is a member of IEEE.

Fazirulhisyam Hashim holds a M.Sc. degree from the Universiti Sains Malaysia and a Ph.D. degree from the University of Sydney, Australia. He is currently a researcher and senior lecturer at the Wireless and Photonic Network Research Center of Excellence (WiPNET) at the Universiti Putra Malaysia. His research interests include network security and QoS of next generation mobile networks, green communication systems, and wireless sensor networks. He is a member of IEEE.

Shamala K. Subramaniam received her B.Sc. degree in Computer Science from Universiti Putra Malaysia, in 1996, M.S. (UPM), in 1999 and PhD (UPM) in 2002. She is currently an Associate Professor in the Department of Communication Technology and Networking, Faculty of Computer Science, Universiti Putra Malaysia. Her research interests are computer networks, simulation and modelling, scheduling and real time systems.