The Crutial Way of Critical Infrastructure Protection

Critical infrastructures are essentially physical processes controlled by networked computers. They're usually as vulnerable as any other interconnected computer system, but their failure has a high socioeconomic impact. The critical utility infrastructural resilience (Crutial) information switch is an intrusion-tolerant and self-healing device designed to protect these critical infrastructures.

Alysson Neves Bessani, Paulo Sousa, Miguel Correia, Nuno Ferreira Neves, and Paulo Verissimo
University of Lisbon

Today's Internet is a web of sophisticated devices, fibers, and electromagnetic signals, interconnecting computational and electronic systems all over the world and making the global society and economy move. However, tips of this web are extending their reach to physical infrastructures such as power, water, gas, oil, and transportation. Therefore, the threat is no longer against systems with a limited economic value, such as home banking or online shopping, but against the infrastructures that support modern life.

A system's operational risk is proportional to the level of threat and degree of vulnerability (risk = threat × vulnerability). Critical infrastructures face a conspicuously high risk. Possible threats include extortion,[1] terrorism, and government-sponsored attacks. Unfortunately, these infrastructures' vulnerability levels are high. Critical infrastructures are physical or mechanical processes controlled electronically by systems, usually called supervisory control and data acquisition (SCADA) or process control systems (PCSs), composed of computers interconnected by networks. These control systems used to be based on proprietary solutions, which provided some, albeit a rather weak, form of security, essentially by obscurity. However, critical infrastructure companies are now using standard hardware and software. Controllers are industrial PCs running off-the-shelf operating systems (such as Windows or Linux), the networks are often wired (or even wireless) Ethernet, and control and supervision protocols are normally encapsulated on top of UDP-TCP/IP. The common defense, "the hackers don't know our systems," is no longer true. Furthermore, the technologies being used—especially software—often aren't even best of breed, but older versions plagued with vulnerabilities. (See the "Trends in critical infrastructure protection" sidebar.)

We describe work done in the context of the European project, Critical Utility Infrastructural Resilience (Crutial), with a particular emphasis on a novel architecture and protocols that preserve legacy systems and on a new device that provides incremental protection, assuring different levels of resilience to different parts of the infrastructure, according to their criticality.

The Crutial architecture

One of the cruxes of the critical infrastructure protection problem is the security of the interconnections among infrastructure providers, regulators, operators, and others. To tackle this problem, we need an architecture that lets us model and reason about this reality.

In Crutial, the entire infrastructure architecture is represented as a wide area network (WAN) of local area networks (LANs). Typically, a critical information infrastructure is formed by facilities, such as power-transformation substations or corporate offices, modeled as collections of LANs, which are linked by a wide-area network, such as dedicated phone lines or the Internet, and modeled as a WAN.[2]

This architecture lets us define realms with different trustworthiness levels. The problem we need to solve is how to defend realms from one another—that is, a LAN from another LAN or from the WAN. However, there's virtually no restriction to the granularity level of a realm, which can be a single host. Conse-

Trends in critical infrastructure protection

Critical infrastructure protection is harder to address than information and communication technology (ICT) protection because of these infrastructures' interconnection complexity, which can lead to various kinds of problems. Consider the power grid, in which geographically dispersed production sites distribute power through different voltage level stations (from higher to lower voltage) until energy eventually flows into our houses. Both the production and distribution sites are typically controlled by supervisory control and data-acquisition (SCADA) systems, which are remotely connected to supervision centers and to the corporate networks (intranets) of the companies managing the infrastructures. The intranets are linked to the Internet to facilitate, for example, communication with power regulators and end clients. These links create a path for external attackers. Operators access SCADA systems remotely for maintenance operations, and sometimes equipment suppliers keep links to the systems through modems. From the viewpoint of keeping the system working, this is all a good idea, but from a security perspective, this looks like a recipe for serious problems.[1] This conclusion is far from speculation: a few years ago, an Internet worm entered a nuclear plant's supervision systems through a supplier's modem, which almost caused a disaster.[2]

On a more positive side, critical infrastructure companies and governments have been showing a high level of concern about this state of affairs, and have been pushing several security measures. However, a careful reading of the legion of guidelines being produced (for example, US National Institute of Standards and Technology specification 800-82[3]) shows a common trend that we can summarize in a couple of ideas: critical infrastructure protection is a network security problem that should be handled using secure communication protocols and perimeter protection through firewalls. These mechanisms are a first level of protection with undeniable utility and importance. However, they put critical infrastructure protection at the level of ICT security, which is inadequate because of these infrastructures' criticality and societal relevance. Moreover, firewalls are known to be vulnerable to intrusions and denial-of-service attacks. The US National Vulnerability Database reported 9 vulnerabilities that might lead to intrusions in commercial firewalls in 2005, 15 in 2006, and again 15 in 2007. The total number of vulnerabilities reported for these three years is 79. We clearly need more than classical prevention security mechanisms such as firewalls.

However, when building new solutions, developers must recognize that critical infrastructures feature numerous legacy subsystems and noncomputational components, such as controllers, sensors, and actuators, which can't be modified for operational reasons or others, for several years to come. In addition, a company's main concern is keeping its critical infrastructure working at the expected level of service. Security mechanisms that stand in the way of operation are unacceptable.

References

1. F. Garrone et al., "Analysis of New Control Applications," Project Deliverable D2, CRUTIAL EC project IST-2004-27513, Jan. 2007.
2. D. Geer, "Security of critical control systems sparks concern," *IEEE Computer*, Jan. 2006, pp. 20-23.
3. K. Stouffer, J. Falco, and K. Kent, "Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security," *Recommendations of the Nat'l Inst. of Standards and Technology*, initial public draft, special publication 800-82, US Nat'l Inst. of Standards and Technology, 2006.

quently, this architecture supports solutions both with outsider threats—protecting a facility from the Internet—and insider threats—protecting a highly critical host from other hosts in the same physical facility by locating them in different LANs.

A *Crutial information switch* protects each LAN (see Figure 1). A CIS provides two basic services:

- The *protection service* ensures that the incoming and outgoing traffic from the LAN satisfies the security policy of the organization managing the LAN.
- The *communication service* supports secure communication between the CIS and, ultimately, between LANs. It provides secure channels and multicast between CISs.

In this article, we focus on the protection service.

A CIS is a distributed protection device with the following main characteristics:

- It resembles a *distributed firewall*[3] because organizations can deploy it not only at the network border but also inside the network to protect critical equipment.
- It uses a *rich access-control model* that considers the involvement of different organizations and lets the access-control rules depend on context information.[4]
- It's *intrusion-tolerant*[5]—that is, it operates correctly even if an attacker intrudes in some of its components. Its objective is to withstand a high degree of hostility from the environment.

Researchers have proposed other intrusion-tolerant services (a brief survey is available elsewhere[6]); however, the CIS design differs from those services in two essential ways. First, a firewall-like component has to be transparent to the protocols that pass through it, so it cannot modify these protocols to obtain intrusion tolerance. Second, internal CIS intrusion-tolerance mechanisms must be transparent to recipient nodes, because such nodes can't protect themselves from malicious messages forwarded by faulty CIS components not satisfying the security policy.

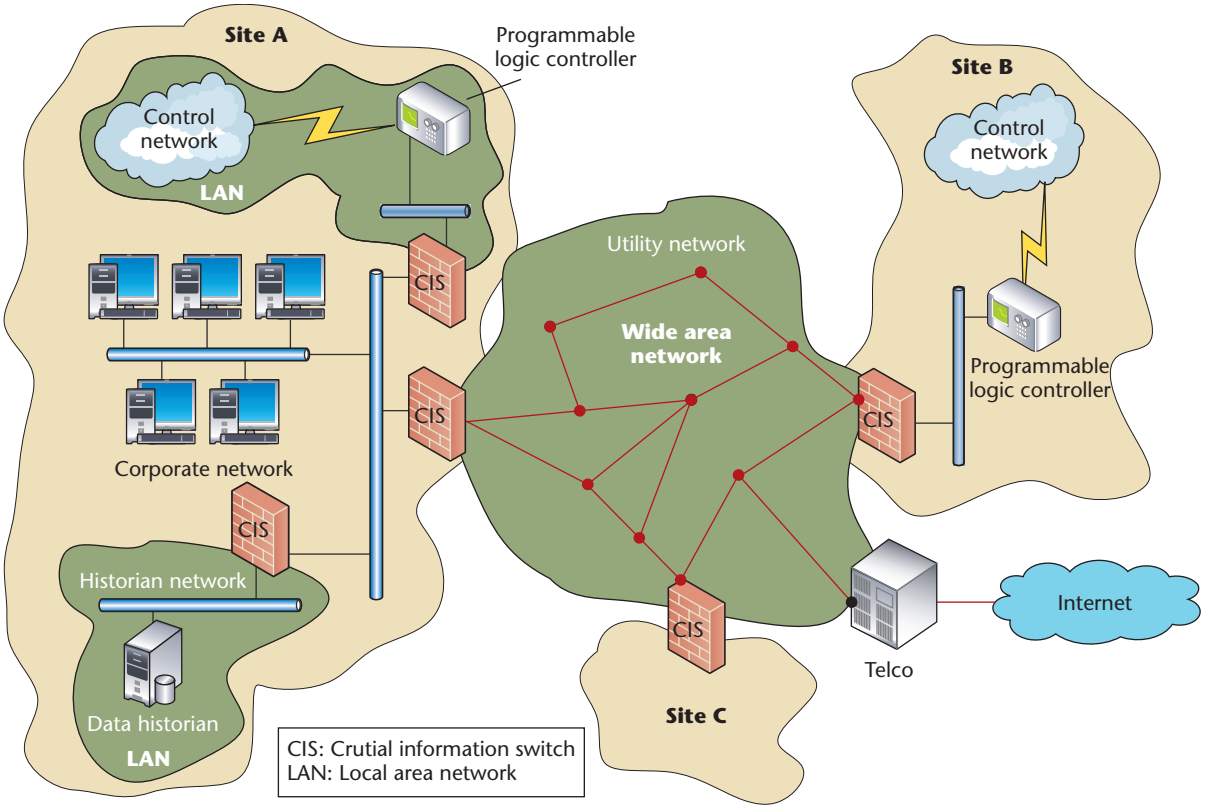

Figure 1. Crutial architecture. Crutial information switches (CISs) connect various sites' LANs to the WAN and check that all incoming and outgoing traffic meets the security policies of the host organization.

The CIS protection service

From a functional viewpoint, the CIS protection service works like a firewall. It captures packets passing through it, checks whether they satisfy the security policy being enforced, and forwards the approved packets, discarding those that don't satisfy the policy. However, several of the CIS's characteristics make it a unique protection device.

Distributed policies

A CIS can be used redundantly, enforcing the same policies at different points of the network. An extreme case on 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 perimeters[3] and is especially useful for critical information infrastructures, given their complexity and criticality, with many routes into the control network that can't be easily closed (for example, Internet, dial-up modems, virtual private networks, and wireless access points).

Application-level semantics

Critical infrastructures have many legacy components that were designed without security in mind, and thus don't use security mechanisms such as access control and cryptography. Because these security mechanisms aren't part of the SCADA/PCS protocols and systems, we must deploy protection mechanisms (such as access control) at some point between the infrastructure and the hosts accessing it. The CIS must inspect and evaluate the messages considering application-level semantics because the application (infrastructure) doesn't verify it.

Rich access-control model

In addition to the capacity to inspect application-level messages, the CIS must support a rich access-control policy that accounts for the multiorganizational nature of the critical infrastructures as well as their different operational states. Consider, for example, the power grid. Several companies are involved in generating, transmitting, distributing, and regulating energy, and several of these parties can execute operations in the infrastructure. Moreover, almost all operations are based on a classical state model of the grid.[7] In each state, operators must take specific actions (for example, actions defined in a defense plan to avoid or recover from a power outage) and many of these actions aren't allowed in other states (for example, they usually can't

separate a generator automatically when the grid is in normal state). These two complex facets of access control in critical infrastructures require more elaborate models than basic mandatory, discretionary, or role-based access control. To deal with this, the Crutial architecture adopts a more elaborate model: PolyOrbac (organization-based access control).[4] It lets us specify security policies containing permissions, prohibitions, obligations, and recommendations, taking into account the subject's role, organization members, the desired action, the action's target object, and the context in which the action is executed. For example, "In context 'emergency,' operators from company C can execute maintenance operations on device D."

Intrusion tolerance

The security level of systems currently connected to the Internet is inadequate for the infrastructures we're concerned with. To improve the CIS's security and dependability, we designed it to be intrusion tolerant, or Byzantine fault tolerant. It's replicated in n machines and follows its specification as long as at most f of these machines are attacked and their behavior corrupted. This approach works if each system replica is sufficiently diverse to ensure that a similar attack won't affect more than f machines simultaneously.

Self-healing

To enhance CIS resilience, we use self-healing mechanisms through proactive and reactive recovery. Proactive recovery is useful to periodically rejuvenate the replicas, guaranteeing perpetual correct operation as long as no more than f replicas become faulty in a given recovery period.[8] Reactive recovery is a complement of proactive recovery that lets the CIS deal with compromised replicas more quickly when attackers make conspicuous actions (for example, a denial-of-service attack).[9]

A hierarchy of CIS designs

Give the various criticality levels of critical infrastructure equipment and the cost of using a replicated device, it's worth defining a hierarchy of CIS designs that are incrementally more resilient.

Nonintrusion-tolerant CIS

Figure 2 depicts the design of a nonintrusion-tolerant CIS. The nonintrusion-tolerant CIS is the cheapest design because it requires only one machine, just as a classical firewall. Nevertheless, it offers better protection than normal firewalls by using application-level policy enforcement and a rich access-control model. These characteristics reduce the probability of an attacker constructing a well-crafted message and deceiving the CIS to let it through. Although misleading the CIS is more difficult, an attacker might find a way to compromise the machine in which the CIS is running (for example, by exploiting a operating system vulnerability) and take control of CIS operation. Attackers might need days, weeks, or even months to find a vulnerability and deploy such an attack, but from experience we know that with a reasonable probability they'll eventually succeed in their quest.

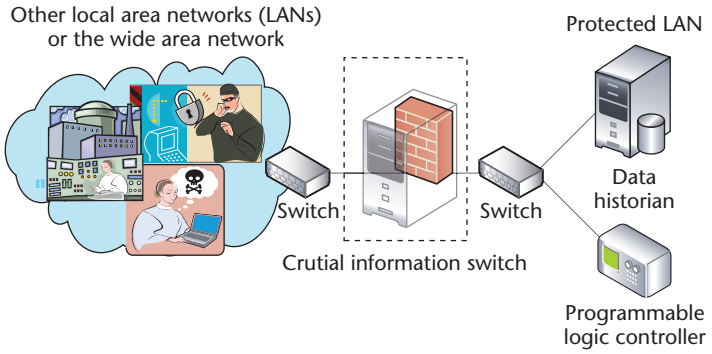

Figure 2. Nonintrusion-tolerant CIS design. This design uses application-level policy enforcement and a rich access-control model on a single machine to reduce the chances of an attacker deceiving the CIS with a well-crafted message and gaining access to the system. However, given enough time, an attacker could compromise the CIS machine and access the protected network.

Intrusion-tolerant CIS

To build a CIS that still works even if a malicious attacker intrudes upon some of its components, we apply the intrusion tolerance paradigm.[5] To understand the intrusion-tolerant CIS's design rationale, consider the problem of implementing a replicated firewall between a nontrusted WAN (or LAN) and the LAN we want to protect. Further assume that we wish to ensure that only the correct messages (according to the deployed policy) go from the nontrusted side, through the CIS, to the computers and devices in the protected LAN. A first difficulty to address is that all n replicas must receive traffic (instead of only one as in a normal firewall) so every replica can participate in the decisions. Second, up to f replicas can be faulty and behave maliciously toward the other replicas and the receiver computers (for example, the SCADA controllers).

To solve the first problem, we use some device (for example, an Ethernet hub) to broadcast the traffic to all replicas. These replicas verify that the messages comply with the security policy and approve messages if and only if at least $f + 1$ replicas vote in favor. This guarantees that at least one correct replica thinks that the message should go through. The CIS then transmits an approved message to the destination using a distinguished replica, the *leader*, so no unnecessary traffic multiplication occurs inside the LAN.

Traditionally, intrusion-tolerant mechanisms ad-

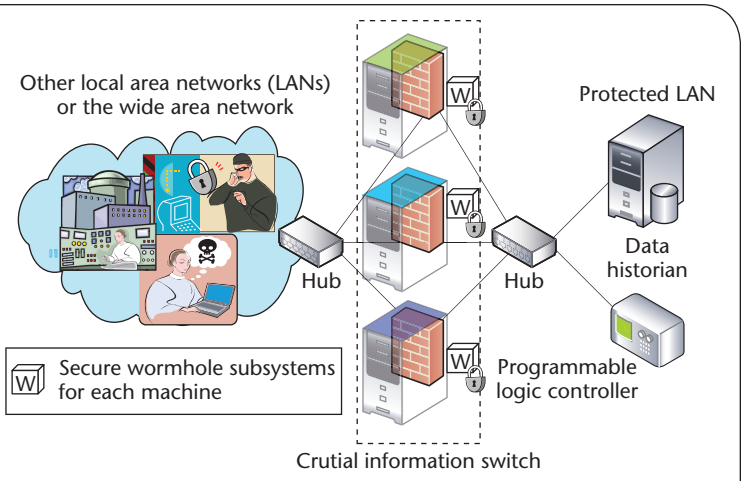

Figure 3. Intrusion-tolerant CIS design. This design offers application-level policy enforcement and a rich access-control model, and is resilient to a fixed number of intrusions. It exploits the IPSec Authentication Header protocol, using a secure wormhole subsystem on each of 2f + 1 machines to store the shared key. As with the nonintrusion-tolerant design, given enough time, an attacker could compromise the CIS machines and access the protected network.

dress the second problem using Byzantine masking protocols, which extract the correct result from the n replicas, despite f being malicious. Basically, the destination machine accepts only results (or messages) that are supported by $f + 1$ replicas. Because the CIS must send a result to the computers in the protected LAN, we must perform this consolidation either at the source or the destination. The simplest and most usual approach implements a front-end at the destination host that accepts a message if

- $f + 1$ different replicas send it,
- the message has a certificate showing that $f + 1$ replicas approve it, or
- the message has a signature generated by $f + 1$ replicas using threshold cryptography.[6]

This, however, would require changes to the end hosts, and would multiply the traffic in the protected LAN by n in certain cases (if every replica sent the message), which is undesirable.

Therefore, we should consolidate at the source and transmit only one—correct—message. What's innovative here is that we transport the source-consolidation mechanism to the protected LAN. Moreover, we must keep in mind that because a faulty replica (leader or not) has direct access to the LAN, it can send incorrect traffic to the protected computers, which can't distinguish between good and bad messages. This makes consolidation at the source a hard problem.

According to best-practice recommendations from expert organizations and governments, the standard protocols that will likely be generalized in SCADA/PCS systems will use IPSec to secure the communications. Consequently, we can assume that the IPSec Authentication Header (AH) protocol[10] will be available and exploit it in our solutions. Most current operating systems support this protocol and its transparent use from an application (for example, SCADA/PCS software) viewpoint. The basic idea is that the protected computers will only accept messages with a valid IPSec AH message authentication code (MAC), which can only be produced if $f + 1$ replicas approve the message.

However, IPSec AH MACs are generated using a shared key K (we assume that IPSec AH is used with manual key management) and a hash function, so using threshold cryptography is impossible (threshold cryptography can only be used in combination with public-key cryptography schemes). The shared key storage becomes an important vulnerability point in a highly resilient design. We need some secure component to store the shared key and produce MACs. This requirement calls for a secure component in an otherwise Byzantine-on-failure environment, which we call a *secure wormhole*.[11] We can deploy the wormhole as a set of local trustworthy components, one per replica, using technologies such as smart cards or cryptographic boards. Figure 3 depicts the design of an intrusion-tolerant CIS.

This CIS design requires $2f + 1$ machines to tolerate f intrusions. Thus, the configuration in Figure 3 can tolerate one intrusion. Such a design only makes sense if the different machines can't be attacked in the same way—that is, they must not share the same set of vulnerabilities. To achieve this goal, each CIS replica is deployed in a different operating system (for example, Linux, FreeBSD, and Solaris), which we configure to use different passwords and different services. The CIS uses the secret key stored in each replica's secure wormhole subsystem to produce a MAC for messages that at least $f + 1$ replicas approve. Given that the wormhole is secure, no malicious replica can force it to sign an unapproved message.

In practical terms, the intrusion-tolerant CIS is more difficult to compromise than the nonintrusion-tolerant version because an attacker will need to find vulnerabilities and deploy attacks against $f + 1$ diverse replicas instead of just one. Compromising each replica might take days, weeks, or months, but attackers tend to be patient.

Intrusion-tolerant and self-healing CIS

Our last and most resilient CIS design combines intrusion tolerance with self-healing mechanisms to address the limitations we've described. Self-healing mechanisms use a proactive-reactive recovery service

that combines time-triggered periodic rejuvenations with event-triggered rejuvenations when they detect or suspect an attack.[9]

Proactive recoveries are triggered periodically in every replica even if they aren't compromised. The goal is to remove the effects of malicious attacks or faults even if the attacker remains dormant. Otherwise, if an attacker compromises a replica and makes a detectable action (for example, sending a message not signed with the shared key K), reactive recovery rejuvenates the compromised replica. Moreover, a (proactive or reactive) recovery's rejuvenation process doesn't simply restore replicas to known states because this would let an attacker exploit the same vulnerabilities as before. The rejuvenation process itself introduces some degree of diversity to restored replicas (changes the operating system, uses memory obfuscation techniques, changes passwords, and so on), so attackers will have to find other vulnerabilities to compromise a replica. Figure 4 depicts the design of an intrusion-tolerant CIS with self-healing mechanisms.

This CIS design requires $2f + k + 1$ machines to tolerate f intrusions per recovery period (dictated by the proactive recoveries' periodicity). The new parameter k represents the number of replicas that recover at the same time. Its value is typically one. If we omit this parameter from the calculation of the total number of required machines, the CIS could become unavailable during recoveries. Thus, the four-replica configuration in Figure 4 can tolerate an intrusion on one replica while another is being rejuvenated. The recovering period's length corresponds to the sum of each replica's recovery time. In our experiments, we recovered each replica in less than 2.5 minutes, which means that the recovering period was less than 10 minutes when using four replicas.[9] In this scenario, attackers would need to find vulnerabilities and deploy attacks against at least $f + 1 = 2$ replicas within 10 minutes, which seems difficult given that each recovery changes the set of vulnerabilities and the attackers must restart their work. Moreover, we can reduce a replica's recovery time to less than one minute using readily available technologies such as solid-state disks.

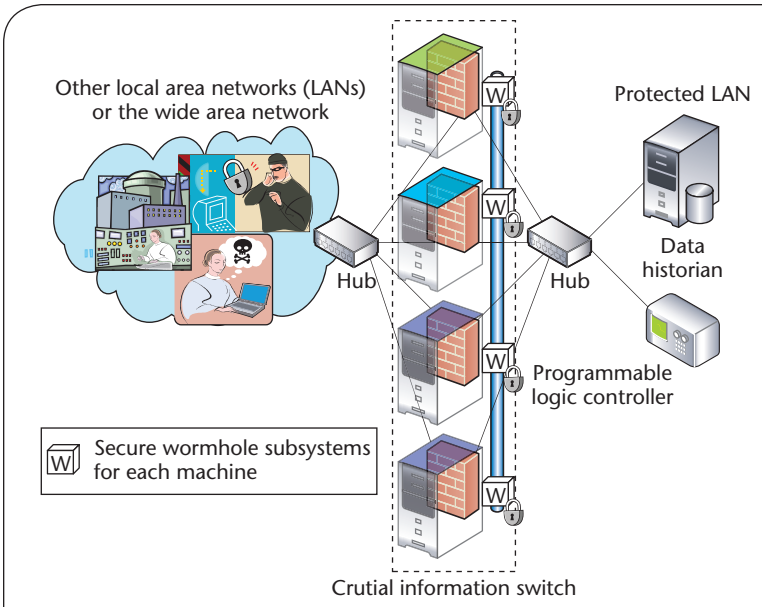

Figure 4. Intrusion-tolerant and self-healing CIS design. This design offers the same functionalities of the previous two designs, but adds failure-detection and rejuvenation mechanisms. A control network connects the secure wormhole subsystems in each replica machine. In this design, an attacker must compromise the machines within the recovery period and without being detected by the self-healing mechanism.

Deployment space

The intrusion-tolerant CIS (with or without self-healing) requires several replicas for effective deployment. However, because price is always a major concern, we can attain a much more cost-effective solution through virtualization.[12] We deploy the various replicas in the same host using virtual machines to isolate the different runtime environments, preventing intrusions from propagating from one replica to the others.

Figure 5 highlights two main dimensions in the design space. If the LAN protected by the CIS provides a critical service, the implementation should use different physical machines for each CIS replica, allowing tolerance to physical and software faults. If the protected service is part of an application that requires certain performance guarantees, without tolerance to unexpected delays, it will need at least $2f + k + 1$ replicas to ensure safe operation and availability even in case of f faults and k recovering replicas. If occasional delays aren't a problem, $2f + 1$ replicas suffice to maintain correct operation of the system.

Figure 5 presents the main four design options:

- *Virtual machines with few replicas.* In VMFR, we deploy $2f + 1$ replicas as virtual machines running in the same host. The system doesn't tolerate physical faults or bugs in the virtual machine monitor, but offers some level of protection if a virtual machine is subject to an intrusion.
- *Virtual machines with more replicas.* In VMMR, $2f + k + 1$ logical replicas run in different virtual machines. VMMR's tolerance to malicious behavior is the same as VMFR but here the system continues to verify and forward messages even with f faulty and k recovering replicas.
- *Few physical replicas.* FPR is similar to VMFR but has different physical replicas. It has some tolerance to accidental hardware and software faults as well as intrusions.

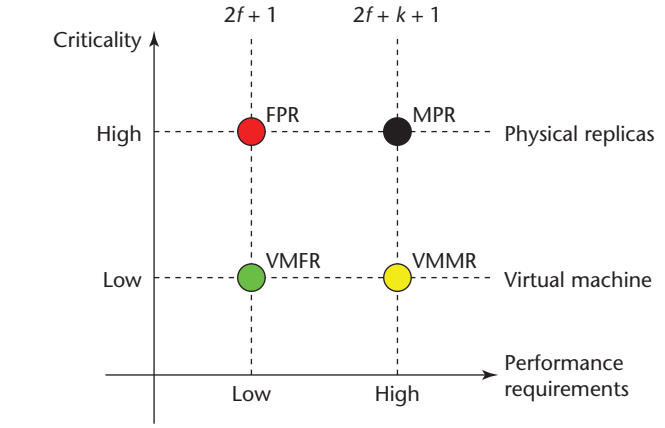

Figure 5. Deployment space for the intrusion-tolerant CIS. The two main dimensions are criticality (that is, the service requires tolerance to physical and software faults) and performance requirements (that is, the service requires certain performance guarantees). Implementations in this space can follow one of four main designs.

- *More physical replicas*. MPR is similar to FPR but has k more replicas so replicas can use self-healing without endangering service availability.

We can deploy these designs in different points of the critical infrastructure to protect different applications. For example, we can use an MPR deployment to protect a critical power grid defense system (for example, a system that performs automatic grid separation in emergency states[13]), whereas a VMMR implementation can adequately protect a substation data historian network.

Organizations can use the hierarchy of services we've presented to protect critical infrastructure facilities. Despite this research, however, work still remains to devise a complete dependable critical information infrastructure. The open problems we're currently addressing include making the communication infrastructure resilient to attacks (mostly denial of service) while avoiding the high costs of a dedicated network; and deploying, managing, and verifying the security policies enforced by the CIS. □

Acknowledgments

We warmly thank our Crutial project partners for the interesting discussions about the architecture presented here. The EC partially supported our work through project IST-2004-27513.

References

1. SANS Institute, "CIA Confirms Cyber Attack Caused Multi-City Power Outage," *SANS NewsBites*, vol. 10, no. 5, 2008; www.sans.org/newsletters/newsbites/newsbites.php?vol=10&issue=5.
2. P. Verissimo, N.F. Neves, and M. Correia, "The Crutial Reference Critical Information Infrastructure Architecture: A Blueprint," *Int'l Journal of System of Systems Eng.*, vol. 1, no. 1/2, 2008, pp. 78–95.
3. S.M. Bellovin, "Distributed Firewalls," *;login:*, Nov. 1999; www.usenix.org/publications/login/1999-11/features/firewalls.html.
4. A. Abou El Kalam et al., "Access Control for Collaborative Systems: A Web Services Based Approach," *Proc. IEEE Int'l Conf. Web Services*, 2007, pp. 1064–1071.
5. P. Verissimo et al., "Intrusion-Tolerant Middleware: The Road to Automatic Security," *IEEE Security & Privacy*, vol. 4, no. 4, 2006, pp. 54–62.
6. F.B. Schneider and L. Zhou, "Implementing Trustworthy Services Using Replicated State Machines," *IEEE Security & Privacy*, vol. 3, no. 5, 2005, pp. 34–43.
7. L.H. Fink and K. Carlsen, "Operating Under Stress and Strain," *IEEE Spectrum*, Mar. 1978, pp. 48–53.
8. P. Sousa, N.F. Neves, and P. Verissimo, "How Resilient Are Distributed f Fault/Intrusion-Tolerant Systems?" *Proc. 35th IEEE/IFIP Int'l Conf. Dependable Systems and Networks*, 2005, pp. 98–107.
9. P. Sousa et al., "Resilient Intrusion Tolerance through Proactive and Reactive Recovery," *Proc. 13th IEEE Pacific Rim Int'l Symp. Dependable Computing*, 2007, pp. 373–380.
10. S. Kent, "IP Authentication Header," IETF RFC 4302, Dec. 2005; www.ietf.org/rfc/rfc2402.txt.
11. P. Verissimo. "Traveling through Wormholes: A New Look at Distributed Systems Models," *ACM SIGACT News*, vol. 37, no. 1, 2006.
12. P. Barham et al., "Xen and the Art of Virtualization," *Proc. 19th ACM Symp. Operating Systems Principles*, 2003, pp. 164–177.
13. F. Garrone et al., "Analysis of New Control Applications," Project Deliverable D2, Crutial European Commission project IST-2004-27513, Jan. 2007.

Alysson Neves Bessani *is an assistant professor in the Department of Informatics at the University of Lisbon, Faculty of Sciences, and a member of the Large-Scale Informatics Systems (LaSIGE) research unit and the Navigators research team. His research interests include distributed algorithms, Byzantine fault tolerance, coordination, middleware, and systems architecture. Bessani has a PhD in electrical engineering from the Federal University of Santa Catarina, Brazil. Contact him at bessani@di.fc.ul.pt; www.di.fc.ul.pt/~bessani.*

Paulo Sousa *is an assistant professor in the Department of Informatics at the University of Lisbon, Faculty of Sciences, and a member of the LaSIGE research unit and the Navigators re-*

search team. His research interests include the construction of secure distributed systems, namely the design of dependable intrusion-tolerant architectures and protocols for distributed systems. Sousa has a PhD in computer science from the University of Lisbon Faculty of Sciences. Contact him at pjsousa@di.fc.ul.pt; http://lasige.di.fc.ul.pt/~pjsousa.

***Miguel Correia** is an assistant professor in the Department of Informatics at the University of Lisbon, Faculty of Sciences, adjunct faculty at the Carnegie Mellon Information Networking Institute, and a member of the LaSIGE research unit and the Navigators research team. His main research interests are intrusion tolerance, security, distributed systems, and distributed algorithms. Correia has a PhD in Computer Science from University of Lisbon Faculty of Sciences. Contact him at mpc@di.fc.ul.pt; www.di.fc.ul.pt/~mpc.*

***Nuno Ferreira Neves** is an assistant professor in the Department of Informatics at the University of Lisbon, Faculty of Sciences. His research interests are in parallel and distributed systems, in particular in the areas of security and fault-tolerance. He has a PhD in computer science from the University of Illinois at Urbana-Champaign. He is member of the editorial board of the International Journal of Critical Computer-Based Systems. Contact him at nuno@di.fc.ul.pt; www.di.fc.ul.pt/~nuno.*

***Paulo Veríssimo** is a professor in the Department of Informatics at the University of Lisbon, Faculty of Sciences, and Director of the LaSIGE research unit. He leads the Navigators research group and his research interests include architecture, middleware, and protocols for distributed, pervasive and embedded systems, as relating to real-time adaptability and fault/intrusion tolerance. Veríssimo has a PhD in electrotechnical and computer engineering from the Technical University of Lisbon. He's an associate editor of the Elsevier International Journal on Critical Infrastructure Protection and is a Fellow of the IEEE. Contact him at pjv@di.fc.ul.pt; www.di.fc.ul.pt/~pjv.*