Keywords: industrial system, ICS, PLC, SCADA, DCS, cryptography, industrial control systems, cyber attack, digital signatures, encryption, security ICs, deepcover

TUTORIAL 5522

Industrial Systems Need the Added Protection of Security ICs

By: Christophe Tremlet, Security Segment Manager
Nov 27, 2012

Abstract: As industrial control systems (ICSs) have become increasingly connected and use more off-the-shelf components, new vulnerabilities to cyber attacks have emerged. This tutorial looks at three types of ICSs: programmable logic controllers (PLCs), supervisory control and data acquisition (SCADA) systems, and distributed control systems (DCSs), and then discusses security issues and remedies. This document also explains the benefits and limitations of two cryptographic solutions (digital signatures and encryption) and elaborates on the reasons for using security ICs in an ICS to support cryptography.

A similar version of this article appeared on *EE Times*, November 7, 2012.

Introduction

Until now, most industrial control systems (ICSs) were designed mainly for high reliability, safety, and the maximum uptime. While the industry has been focusing for decades on fulfilling these requirements, digital security was historically almost not considered at all. In the 1990s some governmental agencies started examining cyber security for critical infrastructures such as electrical power distribution. These efforts were still confidential. Now with the emergence of Stuxnet and the numerous publications it has triggered, cyber attacks against industrial control and automation systems are a key concern to all stakeholders.

Industrial Control Systems Are at Risk

We admit from the outset that a discussion of all ICS structures susceptible to a security attack would deserve an entire article. It is difficult indeed to limit our discussion to a few important ICSs, but that is what we must do. So we will consider three different types of an ICS.

1. **Programmable logic controllers** (PLCs), which are widely used for manufacturing process automation or control of subsystems. PLCs are often connected as part of a wider infrastructure.
2. **Supervisory control and data acquisition** (SCADA) systems, which monitor and control geographically distributed, critical infrastructures such as water distribution or electrical power distribution systems.
3. **Distributed control systems** (DCSs), which control industrial processes such as chemical manufacturing and power generation. They generally comprise several automated subsystems.

Implementations of these systems can, and will, differ greatly depending on the ultimate industrial application.

Some systems will be physically concentrated, limited to a well-defined manufacturing facility, or spread over a very wide geographical area. Nonetheless, all these systems operate under specific performance expectations or, for our discussion, constraints. For instance, a SCADA system must have an extremely high level of uptime, ideally "five 9s" or "six 9s" ("five 9s" means 99.999% availability, which is equivalent to about 5 minutes of downtime per year). For other industrial control and automation systems, the most critical performance constraint might be extremely fast reaction times.

So ICSs can be similar, yet vastly different. Added to this, we see two trends today. ICSs are becoming more and more connected; and they are using more standard, off-the-shelf components like workstations that rely on standard software such as the Windows® operating system or communication over IP. These trends create new vulnerabilities to cyber attacks.

IT Technologies in an Industrial Environment—Not Always a Perfect Fit

ICSs are also now incorporating technologies used in IT, but the ICSs must still operate within their specific performance objectives or constraints. There is an important, immediate consequence of this technology convergence: some threats to standard IT components now also apply to ICSs. However, because of their different performance objectives and working environments, the security remedies used in the IT world are not necessarily applicable to ICSs.

Before we go into a deep dive for the specifics, let's take a simple example. A SCADA system is monitoring the pressure of cooling water in an industrial installation and is expected to raise an alarm when a loss of pressure is detected. In this potential emergency situation, we *want* the operator to take immediate action.

Consider now an operator's response in a classic IT infrastructure. After some minutes of inactivity the IT workstation has probably locked itself; the operator must type his password to login and, usually after three unsuccessful attempts, the workstation would lock again. Now the IT operator needs to contact an administrator to get the password reset. Time is passing. A similar, reiterative procedure would be devastating in an industrial setting. With an ICS in this emergency, we want an operator to act immediately; any hesitation is a critical loss of time. This is an example of a very standard IT procedure that is *not* applicable, and is even detrimental, to an ICS.

Threats to Industrial Control Systems

One cannot rely on the custom architecture of ICSs to protect them from electronic threats. Historically, cyber attacks were not considered serious threats to control and automation systems because ICS networks were either isolated or did not follow IT standards. We can make three important observations about this.

1. Control and automation networks are increasingly connected to standard and open networks because they need to interface with other systems.
2. In industry, a proprietary protocol can easily be reverse engineered at a reasonable cost. The danger here is that any weaknesses in that protocol may never have been studied, understood, or remedied. It is dangerous to think that a proprietary protocol offers security just because its knowledge is not in the public domain. This assumption has been proven to be wrong and misguided; it is the "security through obscurity" concept.[1] This vulnerability of insecure protocols contrasts markedly with publicly known and trusted protocols for which threats and responses are well understood.
3. Networks are not the only path for cyber attacks. Consequently, an isolated infrastructure in an ICS might be vulnerable to other attack media such as USB keys or maintenance consoles.

Just like industrial networks not following documented and well-known protocols, so too the proprietary architecture of PLCs does not normally offer security protection. Stuxnet is unfortunately a very good example of this. Admittedly, Stuxnet only harmed proprietary software and PLCs with a specific configuration. This malware could not have been designed without an intimate understanding of the specific systems it was targeting. "System security should not depend on the secrecy of the implementation or its components."[2] Indeed, that approach makes the hardware (the PLC in this context) too vulnerable. So, again this cannot be stated too often: one cannot rely on the custom architecture of ICSs to act as a protection against electronic threats.

While ICSs have a different architecture from the typical IT infrastructure and fulfill different requirements, nonetheless, most of the threats to generic IT infrastructures can also affect ICSs. Unfortunately, the list of those threats is long and troubling: malware injection such as worms or viruses; software or hardware configuration changes; fake messages or orders from an attacker; identity theft; and unauthorized observation.

In summary, we face an extremely challenging situation. ICS security threats are similar to the ones harassing IT infrastructure, but all too often the specific requirements for ICS operation do not allow reuse of well-known IT security techniques!

Put the Highest Level of Protection *Inside* the ICS

With security being a pervasive concern for industrial and automation applications, protection and mitigation actions are being implemented. Until now, most of these defensive measures have included security procedures, environment and physical protection, and staff education. The ICS itself remains vulnerable. Before we leap to criticize the industrial community for these minimal precautions, we should recall that this is how security started in the traditional IT domain. This is really the first level of protection, a foundation that must be the start.

These traditional defensive tactics do not, in any way, provide the ultimate level of protection needed for an ICS. Procedures, even if audited on a regular basis, are never 100% followed; physical protection like locking doors can be bypassed and cannot be applied everywhere. Most important, defensive manual procedures do not cover attacks performed by highly skilled people with the time and budget to build the most sophisticated scenarios. Even worse, there are examples where bribery led ICS operators to bypass procedures.

No, the security answer is embedded. It is in the ICS hardware. The upper-level hierarchy of security protection techniques involves generic IT security techniques such as cryptography and hardware security (**Figure 1**).

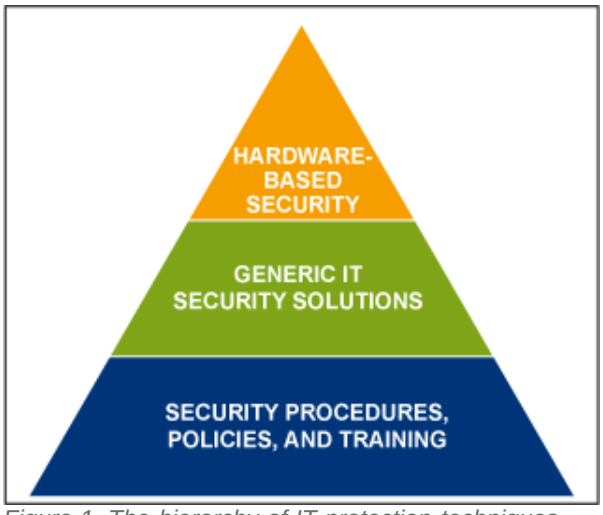

Figure 1. The hierarchy of IT protection techniques.

Generic IT security solutions are sometimes already there in the software. Some infrastructures are already

protected by firewalls; some secure protocols over IP such as TLS/SSL are also implemented. While again, all stages of the pyramid are necessary, we are now going to describe how hardware-based security brings the ultimate level of protection.

Protect the ICS with Embedded Cryptography

Generic IT policies cannot be systematically applied to the broad range of ICSs at work in industry. However, there is one technology used universally in the IT world that can be implemented: cryptography.

Cryptography answers most of the threats listed above. Still, it is not a magic wand and the approach cannot be as simple as, "I'll add crypto to my ICS and all of sudden it will be secure." Crypto algorithms and protocols are building blocks that should be implemented on a case-by-case basis after a thorough analysis of the threats to each subsystem. Restated simply, cryptography is a tool common to ICSs and IT infrastructure, but its implementation in an ICS must be tailored to the specific system. Within the broad range of cryptographic techniques, two are very important for an ICS: digital signature and encryption. We shall examine the merits of both techniques for an ICS.

Digital signature. These techniques are used to authenticate messages, orders, or pieces of software. Consider two examples for an ICS.

1. In a SCADA system, you want to trust information coming from a field sensor. You need to know that it comes from the actual sensor in its physical location, that it is not forged information sent by an attacker to disturb the system. When sent information is digitally signed by the sensor or sensor module, the receiving remote terminal unit (RTU) can verify that the message comes from a genuine and authorized sensor (**Figure 2**). This procedure also works in the other direction, i.e., in descending order to an actuator signed by its originator. The same approach can be used for communication among the RTU, PLCs, and the master SCADA system.

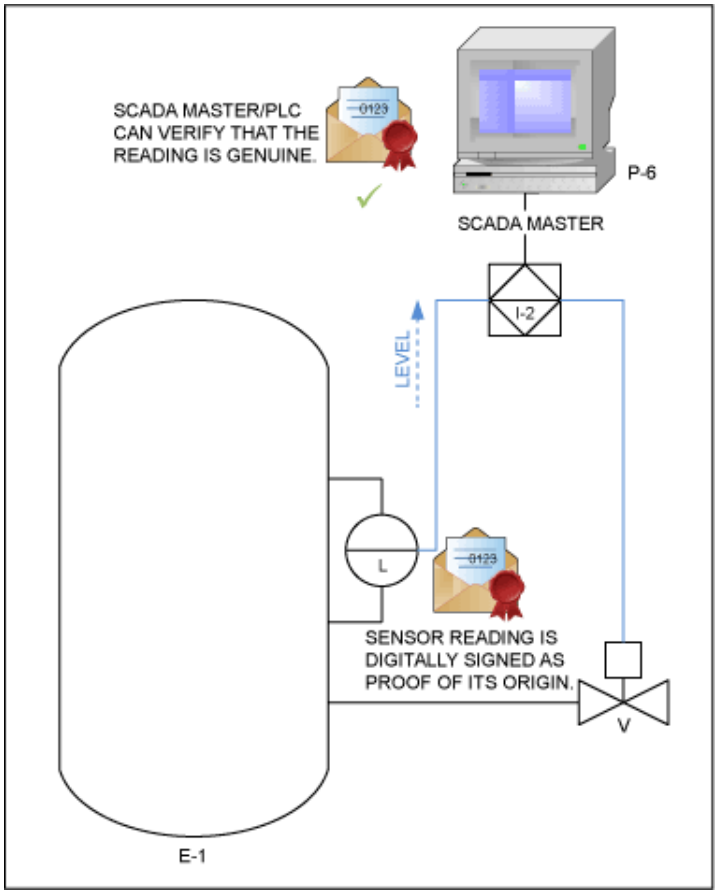

Figure 2. A digital signature is applied to a sensor reading.

2. Digital signatures also allow a system to check that a software package or software update comes from a trusted source. Let's use the Stuxnet example again. Malicious code was injected into PLCs and driving motors at an unexpected speed. This then caused the PLCs to run a wrong manufacturing process.[3] If, however, all software and software updates are digitally signed before being downloaded to the PLC, then the PLC can verify that the software is genuine before starting its execution.

This same digital-signature principle can protect a system against hardware configuration modifications. For example, attackers might modify sensor calibration data. Then the wrongly configured sensors would send erroneous information to their master systems, again disturbing the industrial processes. Particularly vulnerable to this type of attack are large-scale systems spread over a wide geographical area, like fluids distribution, for example, in which one cannot secure physical access to every sensor location. A digital-signature process solves this issue. When sending measurement information to a master, the sensor or sensor module can also send along its "signed" calibration data. Alternatively, the master can poll the sensors on a regular basis by sending a challenge and expecting a signed response, which includes a digest of the calibration or configuration data.

A digital signature can also be used to authenticate hardware. This is useful, for example, when an RTU is connected to a SCADA network. Clearly, one would not want to have unknown (i.e., unauthenticated) hardware receiving and sending critical information but rather only genuine hardware used to build a critical ICS. Here is where a digital signature authenticates hardware and creates trust.

Encryption. This is also a useful, widely used technique to protect against the disclosure of information.

Manufacturing recipes are often precious assets as they can contain all know-how needed to manufacture products. By monitoring data from sensors or parameters of actuators, one could gain valuable information about a manufacturing process or even individual personal data. There are, for example, many publications about attackers getting to know users' behavior by tracking their electricity consumption through the smart grid. Data encryption over ICS networks would prevent such disclosure and would be applied as in a classic IT infrastructure.

Why Use Security ICs to Support Cryptography?

So far we discussed some applications of cryptography for ICS security. Cryptography is often implemented in software, so why would one use security ICs in an ICS? There are several reasons to do so. Security ICs actually offer several specific benefits: secure storage of keys; protection against key disclosure through side-channel attacks; simple implementation of bug-free cryptography; accelerated computation; the quality of random numbers; and trusted software through a secure boot.

Secure Storage of Keys

One concept is so important that we cannot overstate it: any security implementation must use standard crypto algorithms, e.g., AES, Triple DES for symmetric crypto, RSA, ECDSA for public key cryptography. And thus, within these crypto systems, the most valuable assets are the keys. When cryptography is implemented in software on a standard processor, cryptographic keys are stored in the general-purpose system memory and, thus, can be easily retrieved through malware, a debug port (JTAG), or a physical attack. Security ICs dramatically reduce these vulnerabilities.

- Secure microcontrollers integrate logical protection. Secure boot and the microcontroller's memory management unit (MMU) protect against malware injection. The JTAG port can be disabled.
- Security ICs integrate protection techniques such as metal shield, environmental sensors, and external mesh sensors against physical tampering. Moreover, their internal or external memory can be encrypted as well. The highest level of protection would be automatic destruction of keys if tampering is detected.

Protection Against Key Disclosure from Side-Channel Attacks

The power consumption of a microcontroller obviously depends on its activity. By monitoring the power consumption during crypto computation activity, one can retrieve the cryptographic keys. Other side-channel attacks are based on electromagnetic emission (EME).

The most advanced security ICs implement protection against side-channel attacks, making it impossible to retrieve keys through those side channels.

Implementation of Trusted, Bug-Free Cryptography

A common weakness found in systems implementing cryptography is a partial or "bogus" implementation of the algorithm. A faulty algorithm makes the ICS vulnerable and can lead to successful attacks a few months after the product is released. By using a security IC from the outset of system design, however, an ICS designer can be assured that the implementation of algorithms is *bug free*. That level of assurance can be further increased thanks to third-party evaluation or certification for a given standard.

Accelerated Computation

In ICSs, *response time* is often critical. For instance, one may expect a sensor's measurement to be sent to the RTU of a SCADA main controller within a given delay. With hardware cryptography engines built-in, security ICs offer better performance than software solutions. Moreover, our sensor's measurement can be digitally "signed" before it is sent, even if the sensor controller itself has limited computing performance. Security ICs can also

offload an application processor if the latter has limited computing resources.

In the tiniest, most constrained systems like sensor modules, the computing capability (typically an 8-bit microcontroller) to run a sophisticated math operation might not exist. In these situations, adding a secure IC is often the only option to have sufficient computing power for cryptography without redesigning the system.

Secure microcontrollers can even add a full security solution such as handling complete security protocols such as TLS/SSL.

Quality of Random Numbers

A common attack method against systems supposedly protected by cryptography is the so-called *replay attack*. This concept is fairly simple: an attacker records an encrypted or signed message (even though the attacker cannot decrypt it or understand it) and sends the recorded message a moment later. Our example of a digitally signed sensor measurement illustrates the problem here. Assume for the moment that water pressure in a remote pipe is normal. The sensor reports normal pressure in the water pipe and sends this message to the SCADA main controller. The attacker records this message. When the sensor later detects an abnormal pressure, the attacker now intervenes. The attacker replays the message recorded earlier and misleads the controller into believing that the system is in standard operating mode with normal pressure. Given what we know in our example situation, however, one would expect the SCADA system to report an alarm. The standard protection against such a replay attack is to introduce a random number into the transaction, and so prevent the re-use (replay) of a previous transaction.

Not all random number generators are of equal quality. There have been cases where a secret key has been retrieved in systems relying on poor-quality random number generators. This is the worst situation that one can envision. Now the cryptography becomes useless.[4] How does this happen? In a standard microcontroller the randomness of the number generated is not guaranteed. Compare this with security ICs, where the random number generator is designed meet challenging criteria in terms of entropy and is also tested with standardized methods.

Trusted Software Through Secure Boot

Unfortunately, Stuxnet is a brilliant demonstration of the importance of this topic. Systems operators and designers must ensure that all equipment upon which a SCADA or DCS system is built runs a well-identified, genuine piece of software. Secure boot and secure updates management are the ways to protect a device from malware or untrusted software injection. Both types of management are implemented in the newest state-of-the-art secure microcontrollers.

Security ICs Are Today's Ultimate Protection Devices

Today's security ICs integrate several functions designed to ensure the security of an ICS or any critical system 24/7. DeepCover® embedded security solutions cloak sensitive data under multiple layers of advanced physical security to provide the most secure key storage possible.

DeepCover authentication ICs, like the DS28E15, enable crypto-strong authentication, or rejection, of a subsystem to and from its master system. Employing the SHA-2 authentication protocol, they can authenticate the I/O expansion modules of a PLC. They can securely store and digitally sign the configuration and calibration data from a sensor module, thereby preventing it from being replaced by a fake device or having its key parameters changed by an attacker.

Security managers securely store secret keys. Additionally, the DS3645 can trigger a key destruction when a tamper is detected; the MAX36025 security manager supports AES authentication and encryption. DeepCover

security managers can be added to an existing microcontroller, which avoids porting software from a previous design.

Secure microcontrollers provide secure key storage, implement secure boot, enable software logical protection, and offer the most flexibility for implementing cryptography up to protocol levels such as PKCS #11. They also support network protocols and offload the main system processor from cryptographic operations. Devices like the MAXQ1050 are already providing this full range of services in smart meters. The new MAX32590 ARM926™-based microcontroller with its Linux® BSP is a single-chip solution for secure communication devices like gateways.

Conclusion

We have said a lot and perhaps you now ask, "So are we protected against cyber attacks because we are using security ICs?" The answer is not a simple "yes." Full system security requires a thorough identification of assets to be protected and an in-depth analysis of threats prior to any solution deployment. Effective security depends to a great extent on implementation of a number of cryptographic measures which bridge software and hardware.

Nonetheless, after a rigorous analysis, we see that security ICs definitely elevate the protection of ICSs to the highest level.

References

1. Scarfone, K., Jansen, W., and Tracy, M., **NIST Special Publication 800-123**, *Guide to General Server Security*, July 2008, http://csrc.nist.gov/publications/nistpubs/800-123/SP800-123.pdf.
2. Ibid.
3. For a Symantec report about Stuxnet, see Falliere, N., Murchu, L. O., and Chien, E., *W32.Stuxnet Dossier*, Version 1.4, February 2011, www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.
4. For a discussion of DSA requirements for random k value, see Lawson, N., "DSA requirements for random k value," root labs rdist, November 19, 2010, http://rdist.root.org/2010/11/19/dsa-requirements-for-random-k-value/.

Related Parts		
DS28E15	DeepCover Secure Authenticator with 1-Wire SHA-256 and 512-Bit User EEPROM	Free Samples
DS28E22	DeepCover Secure Authenticator with 1-Wire SHA-256 and 2Kb User EEPROM	Free Samples
DS28E25	DeepCover Secure Authenticator with 1-Wire SHA-256 and 4Kb User EEPROM	Free Samples
DS3645	DeepCover Security Manager with 4KB Secure Memory and Tamper Protection	Free Samples

MAX32590	DeepCover Secure Microcontroller with ARM926EJ-S Processor Core
MAX36025	DeepCover Security Manager for Tamper-Reactive Cryptographic-Node Control with AES Encryption
MAXQ1010	DeepCover Secure Microcontroller for Security Tokens with RTC and USB
MAXQ1050	DeepCover Secure Microcontroller with USB and Hardware Cryptography

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 5522: http://www.maximintegrated.com/an5522
TUTORIAL 5522, AN5522, AN 5522, APP5522, Appnote5522, Appnote 5522