Chapter 6

DESIGN AND IMPLEMENTATION OF A SECURE MODBUS PROTOCOL

Igor Nai Fovino, Andrea Carcano, Marcelo Masera and Alberto Trombetta

Abstract The interconnectivity of modern and legacy supervisory control and data acquisition (SCADA) systems with corporate networks and the Internet has significantly increased the threats to critical infrastructure assets. Meanwhile, traditional IT security solutions such as firewalls, intrusion detection systems and antivirus software are relatively ineffective against attacks that specifically target vulnerabilities in SCADA protocols. This paper describes a secure version of the Modbus SCADA protocol that incorporates integrity, authentication, non-repudiation and anti-replay mechanisms. Experimental results using a power plant testbed indicate that the augmented protocol provides good security functionality without significant overhead.

Keywords: SCADA systems, Modbus, secure protocol

1. Introduction

Information and communications technology (ICT) systems are prone to vulnerabilities that can be exploited by malicious software and agents. Modern critical infrastructure assets (e.g., power plants, refineries and water supply systems) use ICT systems to provide reliable services and offer new features. Many maintenance and management operations at these installations involve the use of supervisory control and data acquisition (SCADA) systems, and are conducted remotely using public networks, including the Internet. While the automation and interconnectivity contribute to increased efficiency and reduced costs, they expose critical installations to new threats.

Several studies (see, e.g., [7, 15]) have discussed the threats to critical infrastructure assets. According to Carcano, *et al.* [4], critical infrastructures are exposed to serious *ad hoc* attacks that can interfere with – or even seize control of – process control networks at industrial installations. When one considers

the criticality of the activities performed by a process control network (e.g., gas turbine operation or refinery control), an attack could have devastating consequences to the installation itself as well as other infrastructures due to cascading effects.

The use of traditional ICT security techniques (e.g., firewalls, intrusion detection systems and antivirus software) are effective at dealing with vulnerabilities in corporate networks [15]. However, they do not address attacks that specifically target process control networks. A major concern is the intrinsic weakness of communication protocols used in the SCADA systems that monitor and control field devices in critical infrastructure installations.

SCADA protocols such as Modbus, DNP3 and PROFIBUS were designed decades ago for serial communications between SCADA devices (masters and slaves). Because of network isolation and low threat levels, security features such as authentication, integrity and confidentiality were not considered in SCADA protocol design and implementation. However, with the advent of the Internet era, SCADA vendors began to port SCADA protocols over TCP/IP, offering flexible, economical solutions that also provided interoperability with legacy SCADA implementations. As a result, SCADA networks are highly vulnerable to attacks that would be considered obsolete in the ICT context. For example, as Carcano, *et al.* [4] have demonstrated, the lack of authentication, integrity and non-repudiation mechanisms in SCADA protocols makes it possible to create *ad hoc* viruses that compromise master devices and cause them to send potentially destructive messages to sensors and actuators.

This paper describes the design and implementation of a "secure" Modbus protocol that satisfies the basic security requirements of modern ICT protocols. Experiments with the new protocol demonstrate that it is feasible to augment existing SCADA protocols with security mechanisms without incurring significant real-time performance penalties.

2. Related Work

Most critical infrastructure components adopt network architectures that are tailor-made to the specific systems being operated. These systems also use dedicated SCADA architectures and protocols whose vulnerabilities and attack patterns are different from traditional ICT systems and networks.

Creery and Byres [6] present a detailed analysis of the threats affecting a power plant. In particular, they categorize the devices used in the plant and discuss intrinsic vulnerabilities in the devices and how they relate to the overall power plant architecture. Chandia, *et al.* [5] describe several strategies for securing SCADA networks; their strategies are designed to reduce overhead and to accommodate legacy SCADA systems.

Other researchers have focused on securing SCADA communication protocols. For example, Majdalawieh, *et al.* [13] present an extension of the DNP3 protocol (DNPSec) that attempts to address some of the well-known security problems of master-slave control protocols such as device authentication, message integrity and message non-repudiation. A similar approach has been

adopted by Heo, *et al.* [8]. On the other hand, Mander, *et al.* [14] present a proxy filtering solution that attempts to identify and mitigate anomalous control traffic. The BACnet protocol [2] implements several security features; however, its authentication mechanism is vulnerable to man-in-the-middle attacks, parallel interleaving attacks and replay attacks [9]. Wright, *et al.* [19] present a low latency encryption protocol for SCADA link protection based on CRC. This protocol is very effective for serial SCADA communications; however, no updates related to this research effort have been released since 2006.

3. SCADA Systems

This section discusses the main concepts related to information assurance and SCADA security.

First, we clarify the concepts of "threat," "vulnerability" and "attack." As defined in [11], a "threat" is the potential for a violation of security; it exists when there is a circumstance, capability, action or event that could breach security and cause harm. A "vulnerability" is a weakness in the architecture, design or implementation of an application or service [1, 3]. An "attack" occurs when a threat agent exploits a system by targeting one or more vulnerabilities.

SCADA systems are widely used to control process systems in industrial plants. They rely on sensors to gather data and actuators to perform control actions. A SCADA system typically involves the following actors/components:

- **Operator:** A human operator monitors the SCADA system and performs supervisory control functions over plant operations.

- **Human-Machine Interface (HMI):** This system presents process data to the human operator and enables the operator to control the process. The SCADA system gathers information from PLCs and other controllers over a network using dedicated application layer protocols. An HMI can also be connected to a database, which records trends, diagnostic data and management information (scheduled maintenance procedures, logistic information, etc.).

- **Master Terminal Unit (MTU):** This master device gathers data from remote PLCs and actuators, presents the data to the operator via the HMI and transmits control signals. It contains the high-level control logic for the system.

- **Remote Terminal Unit (RTU):** This device acts as a slave in the master/slave architecture. It sends control signals to the device under control, acquires data from devices, receives commands from the MTU and transmits the data gathered to the MTU. An RTU could be a PLC.

Securing SCADA systems is an important problem (see, e.g., [15]). However, while the majority of research efforts have concentrated on addressing traditional ICT system vulnerabilities, we focus our efforts on the SCADA

communication protocols that are used by MTUs to send commands and receive data from RTUs. Several SCADA protocols, e.g., Modbus, DNP3 and PROFIBUS, have been developed for industrial control applications. We focus on Modbus, the predominant protocol in the oil and gas sector. The security flaws of Modbus are well established (see, e.g., [10]).

4. Modbus Protocol

Modbus is an application layer protocol that provides client/server communications between devices connected on different buses or networks. Modbus communications are of two types: (i) query/response (communications between a master and a slave), or (ii) broadcast (a master sends a command to all the slaves). A Modbus transaction comprises a single query or response frame, or a single broadcast frame. A Modbus frame message contains the address of the intended receiver, the command the receiver must execute and the data needed to execute the command. Modbus TCP basically embeds a Modbus frame into a TCP frame [16]. The Modbus protocol defines several function codes, each of which corresponds to a specific command. Example function codes are:

- **Read Coils (0x01):** This function code is used to read the status of the coils in a remote device. The request specifies the starting address (address of the first coil) and the number of coils. The coils in the response message are packed as one coil per bit in the data field. Status is indicated as 1 = ON and 0 = OFF.

- **Write Single Coil (0x05):** This function code is used to write a single output in a remote device to ON or OFF. The requested ON/OFF state is specified by a constant in the request data field. A value of `0xFF00` requests that the output be ON; `0x0000` requests that it be OFF. All other values are illegal and do not affect the output.

- **Write Multiple Coils (0x0F):** This function code is used to force each coil in a sequence of coils in a remote device to the status of ON or OFF. The normal response returns the function code, starting address and quantity of coils forced.

Most SCADA protocols in use today were designed decades ago, when the technological infrastructure and threat landscape were quite different from how they are today. For example, Modbus was originally published in 1979 for a multidrop network with a master/slave architecture. Because Modbus networks were isolated and free from security threats, key aspects such as integrity, authentication and non-repudiation were not taken into consideration in the design of the protocol. The next section discusses Modbus vulnerabilities and how the vulnerabilities could be exploited by an attacker.

5. Modbus Vulnerabilities

The transportation of Modbus messages using TCP introduces new levels of complexity with regard to managing the reliable delivery of control packets in a process control environment with strong real-time constraints. In addition, it provides attackers with new avenues to target industrial systems.

Modbus TCP lacks mechanisms for protecting confidentiality and for verifying the integrity of messages sent between a master and slaves (i.e., it is not possible to discover if the original message contents have been modified by an attacker). Modbus TCP does not authenticate the master and slaves (i.e., a compromised device could claim to be the master and send commands to the slaves). Moreover, the protocol does not incorporate any anti-repudiation or anti-replay mechanisms.

The security limitations of Modbus can be exploited by attackers to wreak havoc on industrial control systems. Some key attacks are:

- **Unauthorized Command Execution:** The lack of authentication of the master and slaves means that an attacker can send forged Modbus messages to a pool of slaves. In order to execute this attack, the attacker must be able to access the network that hosts the SCADA servers or the field network that hosts the slaves. Carcano, *et al.* [4] show that the attack can be launched by creating malware that infects the network and causes malicious messages to be sent automatically to the slaves.
- **Modbus Denial-of-Service Attacks:** An example attack involves impersonating the master and sending meaningless messages to RTUs that cause them to expend processing resources.
- **Man-in-the-Middle Attacks:** The lack of integrity checks enables an attacker who has access to the production network to modify legitimate messages or fabricate messages and send them to slave devices.
- **Replay Attacks:** The lack of security mechanisms enables an attacker to reuse legitimate Modbus messages sent to or from slave devices.

Firewalls and intrusion/anomaly detection systems can defend against *ad hoc* exploits that target Modbus vulnerabilities. However, it is always possible to circumvent these security controls. The best way to address the security threats is to solve them at their origin – by attempting to "repair" the security holes in the Modbus protocol. But such a solution is difficult to implement because it requires significant changes to the control system architecture and configuration. Instead, we adopt a practical approach in which a small number of security mechanisms are introduced into the protocol to protect against the attacks described above.

6. Secure Modbus Protocol

A communications protocol is generally considered to be "secure" if it satisfies traditional security requirements such as confidentiality, integrity and non-

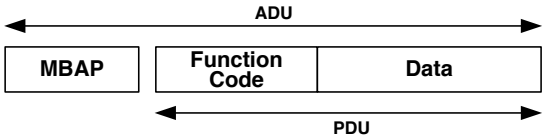

Figure 1. Modbus application data unit.

repudiation [3]. In other words, a "secure" Modbus protocol should guarantee that:

- No unauthorized entity is allowed to access the content of a message.
- No unauthorized entity is allowed to modify the content of a message.
- No entity is allowed to impersonate another entity.
- No entity is allowed to negate a performed action.
- No entity is allowed to reuse a captured message to perform an unauthorized action.

In this work, we do not consider the confidentiality requirement for Modbus messages for two reasons. First, enforcing confidentiality does not mitigate any of the attack scenarios presented above. Second, confidentiality is generally implemented using encryption, which is expensive and introduces considerable overhead that can impact real-time performance.

The original Modbus Serial protocol defines a simple protocol data unit (PDU), which is independent of the underlying communication layer (Figure 1). The mapping of Modbus messages to specific buses or networks introduces additional fields in an application data unit (ADU).

The Modbus TCP protocol introduces a dedicated Modbus application protocol (MBAP) header. The Slave Address field in a Modbus Serial message is replaced by a one-byte Unit Identifier in the MBAP Header. Also, the error checking field is removed and additional length information is stored in the MBAP header to enable the recipient to identify message boundaries when a message is split into multiple packets for transmission. All Modbus requests and responses are designed so that the recipient can verify that the complete message is received. This is accomplished by simply referring to the function code for function codes whose Modbus PDUs have fixed lengths. Request and response messages with function codes that can carry variable amounts of data incorporate a field containing the byte count.

The proposed Secure Modbus protocol is intended to satisfy the following security requirements:

- **Integrity:** The integrity of a Secure Modbus packet is guaranteed using a secure hashing function. The well-known SHA2 hash function is used to compute a secure digest of the packet, which is transmitted along with

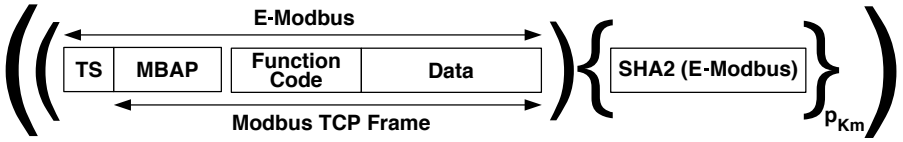

Figure 2. Secure Modbus application data unit.

the packet. The integrity of the received packet is verified by the receiver who computes the SHA2 value of the received packet and compares it with the received digest.

- **Authentication:** The integrity mechanism described above does not prevent an attacker from creating a malicious Modbus packet, computing its SHA2 digest, and sending the malicious packet and the digest to the receiver. To address this issue, the Secure Modbus protocol employs an RSA-based signature scheme [17]. Specifically, the originator of the Secure Modbus packet computes the SHA2 digest, signs the digest with its RSA private key, and sends the packet and the signed digest to the receiver. The receiver verifies the authenticity of the digest (and the packet) using the sender's public key. Thus, the receiver can ensure that the Secure Modbus packet was created by the purported sender and was not modified en route.

- **Non-Repudiation:** The RSA-based signature scheme also provides a non-repudiation mechanism – only the owner of the RSA private key could have sent the Secure Modbus packet.

- **Replay Protection:** The SHA2 hashing and RSA signature schemes do not prevent an attacker from re-using a "sniffed" Modbus packet signed by an authorized sender. Thus, the Secure Modbus protocol needs a mechanism that enables the receiver to discriminate between a "new packet" and a "used packet." This is accomplished by incorporating a time stamp (TS) in the Secure Modbus application data unit (Figure 2). The time stamp is used by the receiver in combination with an internal "time window" to check the "freshness" of the received packet. Our initial solution employed a simple two-byte sequence number and provided all Modbus devices with time windows of limited size to verify freshness. However, this solution was neither elegant nor completely secure. Consequently, our current implementation uses NTP time stamps that facilitate the evaluation of freshness with high precision. Of course, employing NTP time stamps requires an NTP server in the SCADA architecture to provide a reliable clock for all communicating devices.

The Secure Modbus protocol satisfies the minimum requirements of a "secure" protocol. However, it is just as important to ensure that the protocol can be implemented efficiently in real-world SCADA environments. Secure Modbus can be readily deployed in SCADA systems with adequate computing re-

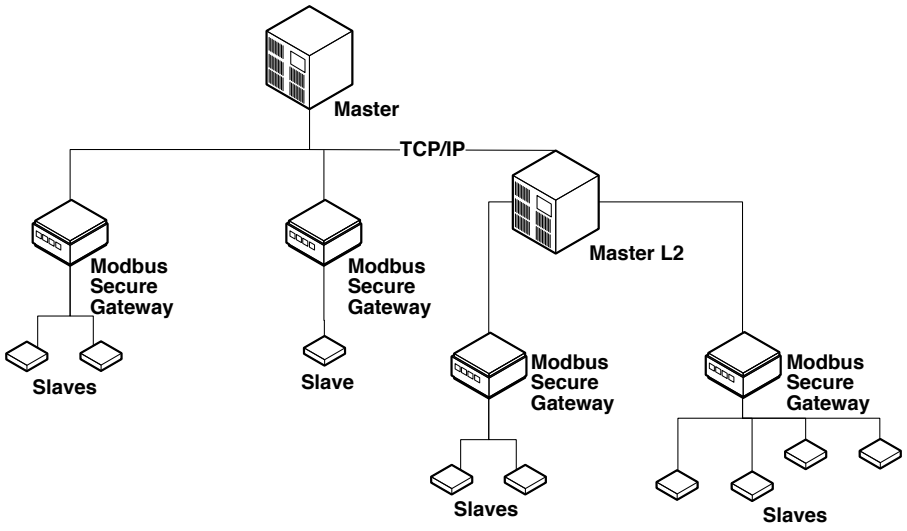

Figure 3. Modbus Secure Gateway.

sources, network bandwidth and modern, upgradeable slave devices. However, many critical infrastructure assets employ decades-old equipment; therefore, it is important to ensure that legacy systems can be retrofitted (at low cost) to support Secure Modbus.

We designed the Modbus Secure Gateway to facilitate the deployment of Secure Modbus in legacy SCADA environments. Figure 3 presents a schematic diagram of the Modbus Secure Gateway. It is a dedicated multi-homed gateway that hosts a TCP/IP interface connected to the process network and a set of point-to-point TCP or serial links connected to legacy slaves. The Modbus Secure Gateway operates as follows:

- When the Modbus Secure Gateway receives a packet on the process network interface:
  - It accepts only authenticated Secure Modbus TCP traffic from allowed masters.
  - It extracts the Modbus packet from the Secure Modbus packet.
  - It forwards the packet to the appropriate slave using the related point-to-point (serial or TCP) link.
- When the Modbus Secure Gateway receives a packet on one of the point-to-point links connected with a slave:
  - It creates a Secure Modbus packet containing the received original Modbus packet.
  - It signs the packet digest with the private key associated with the slave.

– It forwards the new packet to the appropriate master through its process network interface.

The Modbus Secure Gateway constitutes a single point of failure in the SCADA architecture. Therefore, it should be installed only when the "pure" Secure Modbus implementation is not feasible.

Next, we summarize the steps involved in sending and verifying a Secure Modbus request message:

- The master creates a valid Modbus request (M_{req}) with a time stamp and the serial slave address.
- The master computes the digest of the Modbus request, encrypts the digest with its private key (p_{K_m}) and sends the request along with the encrypted digest to a slave or to the Modbus Secure Gateway:

$$C = [TS|Modbus]\{SHA2(TS|Modbus)\}p_{K_m} \quad (1)$$

- The slave or the Modbus Secure Gateway verifies that the Modbus request is genuine using the master's public key (s_{K_m}):

$$M_{req} = \{C\}s_{K_m} \quad (2)$$

  Note that after verifying that the request is genuine, the Modbus Secure Gateway reads the unit identifier in the MBAP header and sends the Modbus request to the addressed slave.

Similar steps are involved when a slave sends a response to the master.

7. Secure Modbus Implementation

The basic communication layer between the operating system and a Secure Modbus device is implemented using sockets (Level 1). All Secure Modbus protocol communications send and receive data via sockets. The TCP/IP library only provides stream sockets using TCP and a connection-based communication service. Consequently, sockets are created using the `socket()` function, which returns a number that is used by the creator to access the socket.

Figure 4 presents the architecture of the Secure Modbus module that implements socket-based communications. The TCP/IP level manages the establishment and termination of connections, and the data flow in an established connection. The TCP Stream Builder sets up the connection parameters according to the following constraints:

- KEEP-ALIVE: Client-server applications use the KEEP-ALIVE time to detect inactivity in order to close a connection or to identify a communication problem. Using a short KEEP-ALIVE time can cause good connections to be dropped.

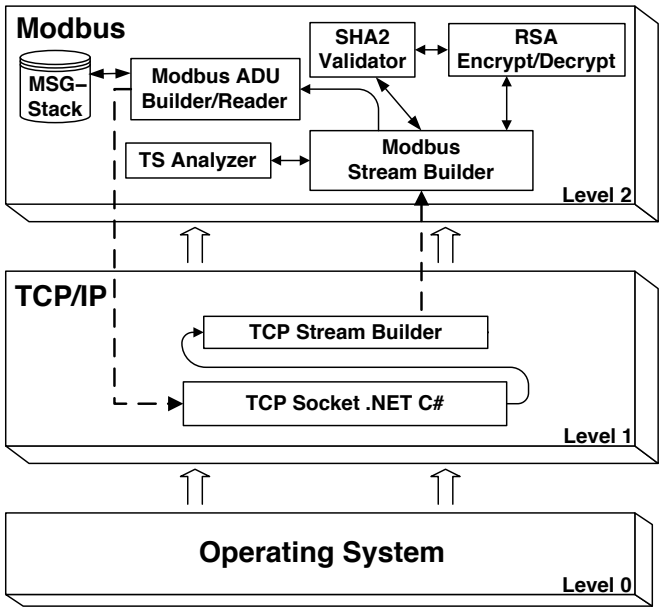

Figure 4. Secure Modbus module.

- TCP-NODELAY: The TCP-NODELAY parameter is used for real-time systems.

- TIME OUT CONNECTIONS: By default, a TCP connection is timed out after 75 seconds. The default value may be adjusted according to the real-time constraints imposed by the system.

The Secure Modbus module has four main components:

- **Modbus Stream Builder:** This component extracts the Secure Modbus packet contained in the TCP payload and sends it to the RSA Encryption/Decryption Unit that verifies the authenticity of the SHA2 digest. The Modbus Stream Builder then sends the digest to the SHA2 Validator to verify packet integrity. Finally, it sends the time stamp to the Time Stamp Analyzer to verify the freshness of the data. If all these conditions are satisfied, the Modbus Stream Builder sends the Modbus packet to the appropriate application.

- **RSA Encryption/Decryption Unit:** This unit uses the public key of the sender to verify the authenticity of the digest and the private key of the sender to sign the hash message.

- **SHA2 Validator:** This component calculates and validates the hash values of Secure Modbus request and response messages.

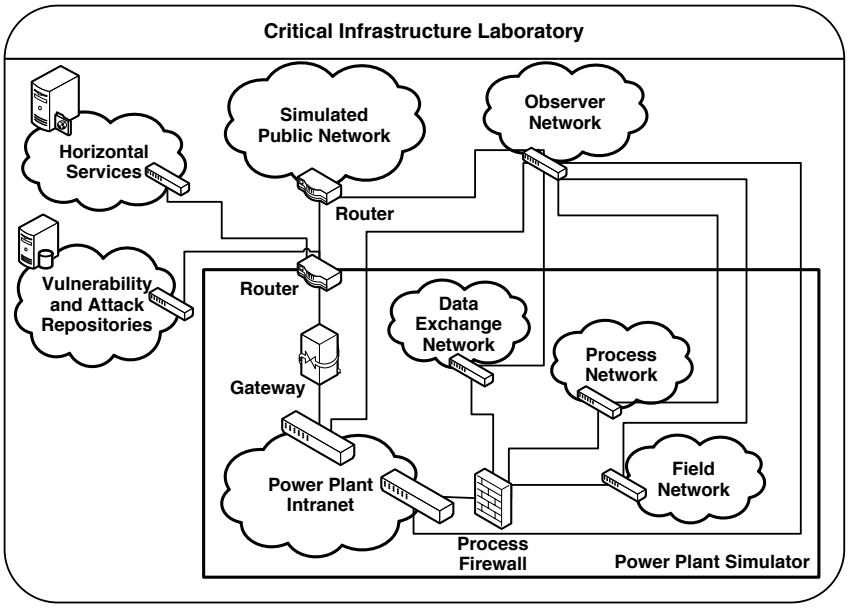

Figure 5. SCADA testbed.

- **Modbus ADU Builder/Reader:** This unit constructs and manages Secure Modbus application data units (ADUs). Also, it communicates with the SHA2 Validator and the RSA Encryption/Decryption Unit to authenticate packets.

- **Time Stamp Analyzer:** This component verifies the validity of time stamps using time windows or an NTP service.

The prototype was written in C# (MS.NET Framework version 2.0) under Microsoft Windows, and was then ported to a standard Linux environment (Ubuntu 10.0).

8. Experimental Results

The Secure Modbus protocol was tested using an experimental power plant testbed. Figure 5 shows the components of the SCADA testbed. The principal components are:

- **Field Network:** This network interconnects the sensors and actuators that interact with electromechanical devices in the power plant.

- **Process Network:** This network hosts all the SCADA systems. Plant operators use these systems to manage the power plant, send commands to sensors and actuators in the field network, and gather plant data and status information.

Table 1. Comparison of communication latency.

Modbus TCP		**Secure Modbus**	
Scan Rate	500 ms	Scan Rate	500 ms
Connection Time Out	1,200 ms	Connection Time Out	1,200 ms
Latency	26 ms	Latency	27 ms
Scan Rate	200 ms	Scan Rate	200 ms
Connection Time Out	500 ms	Connection Time Out	500 ms
Latency	29 ms	Latency	31 ms

- **Observer Network:** This is a network of sensors that gathers information about the system during the experiments.
- **Horizontal Services Network:** This network provides support features such as backup and disaster recovery.
- **Intranet:** This internal private network connects company PCs and servers. Some portions of the intranet are connected to the power plant via the process network.
- **Data Exchange Network:** This network hosts data exchange servers that receive data from the process network and make it available to operators who use the corporate intranet.

We conducted two experiments to evaluate the performance of the Secure Modbus protocol. The first experiment examined the latency resulting from the use of the SHA2 hashing and RSA-based signature schemes. The second examined the increased size of Secure Modbus packets for various function codes.

Table 1 compares the communication latency for Modbus TCP and Secure Modbus. The first set of results, corresponding to a master scan rate of 500 ms and a connection timeout of 1,200 ms, show a latency of 26 ms for Modbus and 27 ms for Secure Modbus – a negligible difference. A negligible latency difference of 2 ms (29 ms for Modbus TCP and 31 ms for Secure Modbus) is also observed for a master scan rate of 200 ms and a connection timeout of 500 ms.

Table 2 compares the size of Modbus TCP and Secure Modbus packets for four function codes. Secure Modbus packets are larger than the corresponding Modbus TCP packets. However, the increased size is not a significant issue even for SCADA networks with low bandwidth.

9. Conclusions

The Secure Modbus protocol offers key security features without introducing significant overhead that can impact real-time performance. The Modbus

Table 2. Comparison of packet size.

Function	Modbus TCP	Secure Modbus	Overhead
Write Coil (`0x05`)	11 bytes	43 bytes	291%
Write Register (`0x06`)	12 bytes	44 bytes	267%
Write Multiple Coils (`0x0F`)	260 bytes	292 bytes	12%
Write Multiple Registers (`0x10`)	260 bytes	292 bytes	12%

Secure Gateway facilitates the deployment of Secure Modbus in legacy SCADA environments. While the new protocol helps protect against several attacks, it does not address scenarios where an attacker seizes control of a master and sends malicious Modbus messages to slave devices, or where an attacker captures the master unit's private key and forges malicious Modbus messages that are signed with the stolen key. To address the first attack scenario, we are working on a dedicated filtering unit that will identify suspect Modbus messages. Our solution to the second scenario is to use a trusted computing platform to protect key rings. Our future research will also attempt to refine the signature scheme to improve real-time performance.

References

[1] O. Alhazmi, Y. Malaiya and I. Ray, Security vulnerabilities in software systems: A quantitative perspective, in *Data and Applications Security XIX*, S. Jajodia and D. Wijesekera (Eds.), Springer, Berlin-Heidelberg, pp. 281–294, 2005.

[2] American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE), BACnet, ASHRAE SSPC 135, Atlanta, Georgia (www.bacnet.org).

[3] M. Bishop, *Computer Security: Art and Science*, Addison-Wesley, Reading, Massachusetts, 2002.

[4] A. Carcano, I. Nai Fovino, M. Masera and A. Trombetta, SCADA malware: A proof of concept, presented at the *Third International Workshop on Critical Information Infrastructure Security*, 2008.

[5] R. Chandia, J. Gonzalez, T. Kilpatrick, M. Papa and S. Shenoi, Security strategies for SCADA networks, in *Critical Infrastructure Protection*, E. Goetz and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 117–131, 2007.

[6] A. Creery and E. Byres, Industrial cybersecurity for power system and SCADA networks – Be secure, *IEEE Industry Applications*, vol. 13(4), pp. 49–55, 2007.

[7] G. Dondossola, J. Szanto, M. Masera and I. Nai Fovino, Effects of intentional threats to power substation control systems, *International Journal of Critical Infrastructures*, vol. 4(1/2), pp. 129–143, 2008.

[8] J. Heo, C. Hong, S. Ju, Y. Lim, B. Lee and D. Hyun, A security mechanism for automation control in PLC-based networks, *Proceedings of the IEEE International Symposium on Power Line Communications and its Applications*, pp. 466–470, 2007.

[9] D. Holmberg, BACnet Wide Area Network Security Threat Assessment, NISTIR 7009, National Institute of Standards and Technology, Gaithersburg, Maryland, 2003.

[10] P. Huitsing, R. Chandia, M. Papa and S. Shenoi, Attack taxonomies for the Modbus protocols, *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 37–44, 2008.

[11] A. Jones and D. Ashenden, *Risk Management for Computer Security: Protecting Your Network and Information Assets*, Elsevier, Oxford, United Kingdom, 2005.

[12] R. Leszczyna, I. Nai Fovino and M. Masera, Simulating malware with MAlSim, *Computer Virology*, EICAR 2008 Extended Version, 2008.

[13] M. Majdalawieh, F. Parisi-Presicce and D. Wijesekera, DNPSec: Distributed Network Protocol Version 3 security framework, presented at the *Twenty-First Annual Computer Security Applications Conference (Technology Blitz Session)*, 2005.

[14] T. Mander, F. Nabhani, L. Wang and R. Cheung, Data object based security for DNP3 over TCP/IP for increased utility of commercial aspects of security, *Proceedings of the IEEE Power Engineering Society General Meeting*, pp. 1–8, 2007.

[15] M. Masera, I. Nai Fovino and R. Leszczyna, Security assessment of a turbogas power plant, in *Critical Infrastructure Protection II*, M. Papa and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 31–40, 2008.

[16] Modbus IDA, MODBUS Application Protocol Specification v1.1a, North Grafton, Massachusetts (www.modbus.org/specs.php), 2004.

[17] R. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, vol. 21(2), pp. 120–126, 1978.

[18] M. Wiener, H. Handschuh, P. Pallier, R. Rivest, E. Biham and L. Knudsen, Performance comparison of public-key cryptosystems, smartcard cryptocoprocessors for public-key cryptography, chaffing and winnowing: Confidentiality without encryption, DES, Triple-DES and AES, *CryptoBytes*, vol. 4(1), 1998.

[19] A. Wright, J. Kinast and J. McCarty, Low-latency cryptographic protection for SCADA communications, *Proceedings of the Second International Conference on Applied Security and Network Security*, pp. 263–277, 2004.