RITA SECURE COMMUNICATION PROTOCOL: APPLICATION TO SCADA

Fadi Obeid[1] and Philippe Dhaussy[2]

[1]PhD. at Ensta Bretagne, Brest, France
fadi.obeid@ensta-bretagne.org
[2]HDr. at Ensta Bretagne, Brest, France
philippe.dhaussy@ensta-bretagne.fr

ABSTRACT

Supervisory control and data acquisition (SCADA) systems have their own constrains and specifications. These systems control many of our critical industrial infrastructures, yet they are hardly secured. The biggest problem in securing these systems is the lack of cryptography support especially that most SCADA systems work in real-time which is not compatible with most cryptography algorithms. Additionally, a SCADA network may include a huge amount of embedded devices with little computational powers which adds to the cost of any security improvement. In this paper we present a new approach that would secure SCADA communications by coding information without the need of the complex cryptography algorithms. The reconfigurable information transmitter agent (RITA) protocol that we present does not need the already installed devices to be modified nor replaced, it only needs to add costless electrical chips to these devices. This approach can also be used to secure any type of communication that respects the protocol's constraints.

KEYWORDS

Information Security, Network and Communication Security, SCADA Networks, Cryptography

1. INTRODUCTION

Information Security (*InfoSec)* is the act of protecting a set of information against unauthorized entities. A complete protection means that the information can be created, manipulated, and red by authorized entities only. The measurements to ensure InfoSec depend a lot on the state of the information, whether the information is stocked in a data base, being processed, or being communicated between entities. In this article we are interested in the measurements taken to secure the communication of information.

Securing a communication is needed when the channel being used for communication is considered insecure as in the case of Internet communication, wireless communication and others. In most cases, InfoSec relies on cryptography algorithms to ensure a secure communication. The complexity of a cryptographic algorithm requires additional power, time and space. Although good cryptography algorithms exist, few are considered secured today. An additional problem with cryptographic algorithms is that many were considered to be secure until attacks and analysis proved they are not, which means that what is considered secure today may be insecure tomorrow.

SCADA systems are considered secured by isolation, still, they can be attacked from the inside. Also, due to the augmented connectivity to the outside, it is wise to consider effective security measurements before being able to have authorized outside access.

While security protocols are implemented in many systems, most of SCADA systems are still unsecured. Most companies that rely on SCADA systems do not consider securing these systems because of the expected high costs. This high cost is the consequence of cryptography use which also breaks the real-time constraint of SCADA systems.

Our proposal is to replace cryptography with a measurement that is expected to have a satisfying security level with a very low cost (power, time, and space). Our approach does not need for the already installed system to be replaced nor upgraded which means that the SCADA system would be available during the shift from unsecured to secured.

In this article, we will present the concept of our proposal while unfolding the first and most basic version of our protocol.

2. SCADA

SCADA systems can be found in modern industrial facilities such as water pipes, power plants, oil refineries, chemical factories and nuclear facilities. These systems use coded signals over communicating channels to monitor and control numerous devices on multiple and distant sites.

Unlike standard networks, most of the SCADA nodes are special purpose embedded computing devices with limited capacities such as remote terminal units (RTUs) and programmable logic controllers (PLCs). These nodes exchange data (exp. temperature is *x*, water level is *y*, etc.) and commands (exp. turn off water) between each others and with the supervisory system. The supervisory system can also build statistics about the system and how it is being used based on the received data.

Figure 1 presents a general SCADA network and its communication to a second SCADA, a local network, and the Internet.

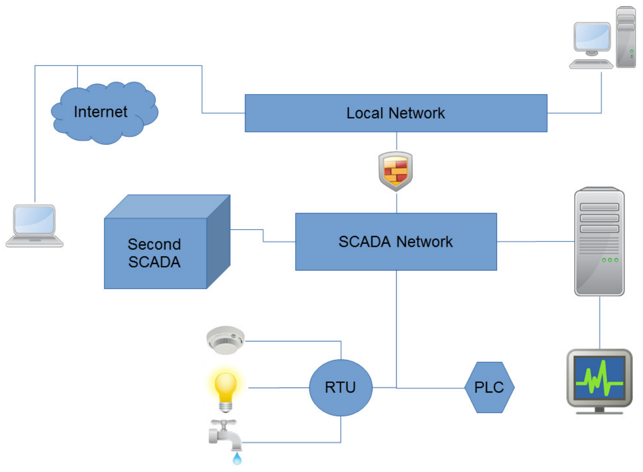

Figure 1. General SCADA network presentation

In addition to using special purpose embedded computing devices, other SCADA properties also affect their security as well, we are mainly interested by the following [1]:

- Non-stop availability: Devices are required to work non-stop for years, exp. traffic lights.
- Geolocation: Nodes can be very sparse and geographically extensive, exp. water pipelines.
- Hard conditions: Nodes may exist in hard physical conditions, exp. chemical factories.
- Performance: Devices must have a hard real-time constraint, exp. microchips industries.

Many security problems are caused by the properties mentioned above: The use of special purpose devices with limited input/output choices causes communicated messages to be easily predictable. Cryptography is hardly supported since performance would be dramatically reduced. Renewing and updating devices would be very expensive, this is caused by the geolocation of the devices and the availability constraint. The geolocation constraint also allows possible unexpected links to the outside reducing system security. And finally, the geolocation and hard conditions of devices discourage temper resistance. Any approach to secure SCADA systems should consider those properties and specifications.

SCADA properties are not the only aspects causing security problems, many of the choices (made by SCADA manufacturers and users) that characterize current SCADA systems also reduce SCADA security [1]:

- Using open standards which grant attackers more knowledge of the system.
- Using COTS (commercial off-the-shelf) hardware and software (which lacks of security).
- Using fail-safe constraints increases safety while decreasing security.
- Protocols vulnerabilities, whether conceptual or caused by implementation errors.

3. RELATED WORK

Many efforts were put to secure SCADA communication [2,3,4,5]. In the technical and research world, these efforts led to solutions which would insure a high level of security in SCADA. In practice, the proposed solutions are expensive and their requirements are not met in the SCADA networks. Since none of these solutions is proven to be perfect, no one would take the risk and pay the elevated price, which is in some cases changing their whole SCADA system. The imperfection of these solutions would mean constant updates and upgrades that SCADA managers would not risk. For these reasons, we think that any SCADA security solution that desires to pass from the research world to practice should have perfection properties. Also a good security solution for SCADA systems would not need to replace the already installed materials.

The only known perfectly secure cryptosystem is the Vernam cipher, also called the one-time pad. Gilbert Vernam patented this invention in the USA in July 1919 [6]. A few years later, a variation of the one-time pad was patented in Germany by Siemens and Halske [7]. The one-time pad is based on a list of shared keys that can only be used once. If implemented in SCADA systems, the list of shared keys need to be updated constantly which is very time consuming in most cases (exp. water pipes).

In 1949, Claude Shannon proved that the one-time pad is indeed unbreakable and that any unbreakable system must have the same essential characteristics as the one-time pad [8]. The most essential characteristic of the one time-pad is the use of a different key for each encryption.

To the best of our knowledge, there are no other cryptosystems that were considered unbreakable since the one-time pad.

4. PROTOCOL

In this section we will explain the principles of the protocol and its characteristics. We will also demonstrate the communication schema, and finally we show some details of the needed security measurements for the protocol to work effectively.

4.1. Principles of the Protocol

The most important principle of our protocol is the secret sharing between two entities that we call security boxes. These security boxes are considered twins, our security relies on the possibility for these twins to share and maintain a secret which is similar to a symmetric key in the case of cryptographic algorithms.

The shared secret is a table of randomly pre-initialized values, along with a secret algorithm with predefined random operations. To make sure the performance is almost intact, the chosen operations are simple binary compositions (exp. binary XOR) and substitutions. Also, most of the operations take place after sending/receiving a message and not before. That way, the message is sent and processed with almost no security related latency.

The 2-security boxes are initialized together before being placed each on the entry/outlet of any type of device that needs its communication secured. A security box can be an embedded device or an algorithm implemented on another already existent device such as a computer program, a smart phone application, etc. Any communication between the 2 devices would go through both security boxes, the first security box translates the communicated message to a matching secret message and the second security box would reveal the match for the received secret. After sending or receiving a secret message, a security box would change the secret table.

Figure 2 represents a synchronous version of the communication using our security protocol. The twins *Agent A* and *Agent B* communicate synchronous messages. In our latest version of the protocol asynchronous communication is possible. Each time a security box sends or receives a message it changes the used value in the table using function *f*. Function *f*'s output depends not only on the input but on the current state of the security box (ex. whole table, number of communicated messages, etc.). This dependency exists for security reasons so outputs would not be redundant or have a pattern.

4.2. Characteristics

The security boxes transform a plain message to a secret message/code and vice versa to ensure that messages in the channel are unintelligible to any eavesdropper that may be analysing channel communication. Only intelligible messages are accepted by the security boxes.

The security boxes can be used as a middle-ware between 2 devices, the translation from plain to secret message and vice versa can be different from one side to the other to make sure both devices send and receive messages they understand.

The security boxes can also be used as a middle-ware between the devices and the communication channel, no matter what form of message the device produces. The security box transforms a produced message into a secret message that respects the protocol used in the communication channel without the need to change or adjust the secured device itself. The only modifications are the ones we do to the security box itself.

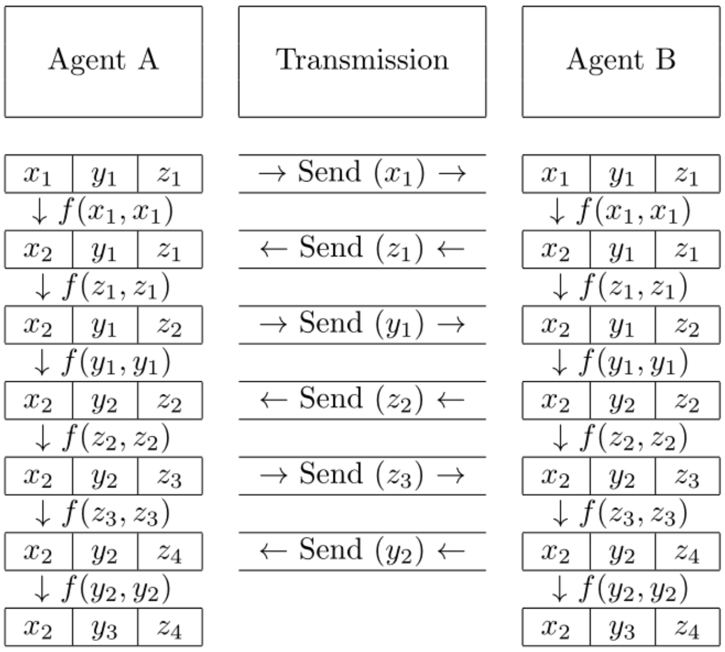

Figure 2. Simple Synchronous Method

A very important aspect of the security boxes is that they do not require any changes from the devices being secured. It is up to the security box to adjust itself depending on the secured devices. To do so, the security device can either be a general box which would require configuration depending on the secured devices. This would increase the flexibility of the box while reducing its performance and security level. The second method is to have boxes specially designed depending on the requirements of the secured devices. Although the second method provides less flexibility, it insures maximum performance and security. While the second method seems extreme with its need to recreate a security box depending on the requirements, it is feasible since the part that would change in the security box is very small and easy to modify.

4.3. Communication Schema

Figure 3 describes the communication schema using our protocol. *A* and *B* are communicating in a synchronous fashion. A creates a clear message *clm* and sends it to *A.mySecurityBox* which codes it into a matching coded message *com* and forwards it to the communication channel before updating the secret table using the contents of com and other variables. *B.mySecurityBox* is waiting for this message, it receives *com*, generates *clm* and sends it to *B* before updating its own version of the secret table using the same variables and operations. Finally *B* would answer by creating a new *clm* (response) and sending it to *A* in the same fashion.

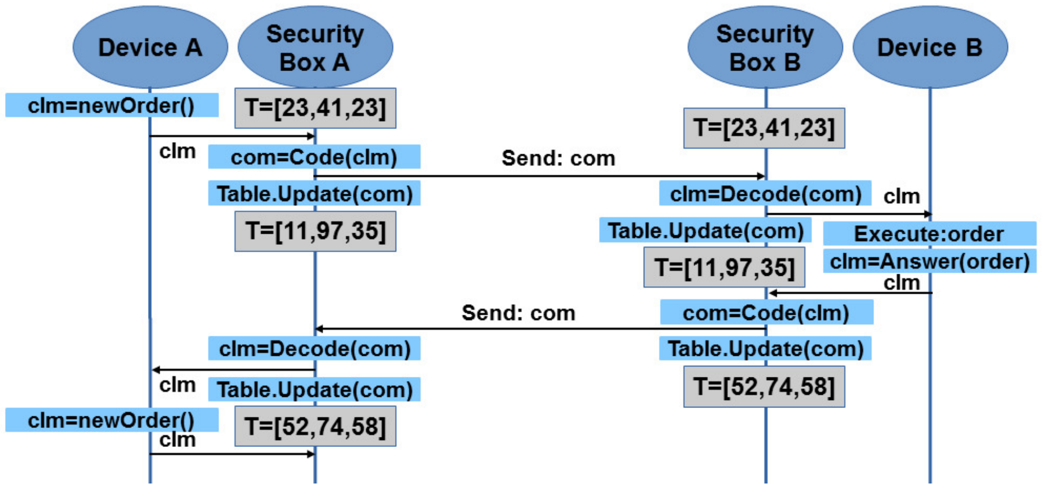

Figure 3. Communication Schema

4.4. Function Necessities and Security Measurements

The size of the secret table depends on the requirements of the secured devices and the different possible signals. For example, to control a lamp with *3* possible commands and *3* possible responses, we need a table of *3* values (for the most basic implementation of our mechanism).

The cryptosystem needs to insure confusion and diffusion [8]. To make sure that the minimum required confusion and diffusion is offered we use substitution boxes. It is also considered important that the transformation of the secret table is irreversible. This would reduce the possibilities of future analytical attacks.

It is clear that the values in the table should be different one from the other at any point of time or the same value would have multiple meanings which results in an ambiguous message.

A list of unacceptable/undesired values contains values considered to decrease the security of our system such as *zero* in addition to any value with Hamming weight equal to *1* or even *2*. these values are never used in the secret tables.

Getting a substitution box output should be normalized regarding time, power, etc. If we have a substitution box of the form *X = numberOfRows* and *Y = numberOfColumns*, and we are searching for the output *OUT(row=x,col=y)* then we should also search for a phantom output *OUT(row=X-x,col=Y-y)* without actually using it.
Any list/table search should consider the same approach. Looking for the value *x* in list *T* should have *phantom = T[l-i]* where *l* is the length of *T* and *i* is the index of *x* in *T*.

Although most phantom outputs are used after sending/receiving, they still affect the performance and power consumption of our approach. Therefore, phantom outputs should only be used when side channel attacks are considered as threats to the system.

5. DISCUSSIONS

In this section we will analyse the robustness of our approach and show an effective way of implementing it on already installed systems.

5.1. Robustness Analysis

Consider the following: *A* and *B* communicating using our protocol, they use a table of 3 values. *A* sends a message with value X_1 to *B* and replaces X_1 with X_2, X_1 is intercepted by an attacker. From the attacker's (let us call him *C*) point of view, X_1 has no signification other than a strange signal being sent from *A* to *B*, since *C* cannot understand the meaning of this signal then the confidentiality requirement is respected.

The integrity requirement is also respected since if *C* tries to change X_1, or to invent a message and send it to *B*, it has a negligible chance of succeeding. The success probability is actually $a/(2^b - c)$ where a is the number of possible values (*3* in our example), b is the number of used bits *32*, and c is the number of unacceptable values *1+32* if we only refuse values with hamming weight equal to *0* and *1*. $3 / (2^{32} – 33) = 6 * 10^{-10}$ which is *6* times lesser then the probability of gaining the jackpot Mega Millions multi-state lottery in the United States.

If *C* tries to redirect X_1 to *A*, A would not accept it since X_1 was replaced by X_2 and does not have any meaning to *A* any longer. The only thing *C* can do is to interrupt messages from *A* to *B* and *B* to *A*. Interrupted messages cannot be replaced, therefore, if *A* and *B* use a time constraint (the system knows something wrong if no message is received for *t* seconds) then both devices would know there is an undergoing attack or a connection problem.

5.2. Installation Method

Let us consider devices *A* and *B* are already functioning in our system, we wish to secure the communication between these 2 devices. If we do not want to break the communication between the devices we proceed as follows:

First we add the security box to *A* by switching the cables connecting *A* to the security box, and adding a cable between *A* and its security box. The security box would simply forward messages from *A* without securing them. Then we add the security box to *B*, once installed the security box would send a notification to *A*'s security box to start securing messages. After the secured communication is well established, unsecured messages would stop being accepted.

While a security box is being installed on a machine, this machine will not be able to send and receive messages only for the instance of cables switching.
In addition to the security boxes, we use port boxes (Figure 4). The current ports would simply route the messages to the correct channel. The ports may also be used to scramble the messages which would add to the system security while slightly affecting performance.

The structure of a secret message would contain the following information: *ID* of the sender (*portID_A*, *boxID_A*, and *deviceID_A*), ID of the receiver (*portID_B*, *boxID_B*, and *deviceID_B*), sequences of the message (*box_seq* and *device_seq*), and finally the coded data itself. Additional information can be added to the message if needed. Some information can also be reduced if it is found heavy for a system. For example, we can remove *boxID_A* since it is *boxID_B*'s twin and the only one who could have sent the readable message. In some cases, a box and a device can have the same ID and messages sequences which would also reduce the size of the secret message.

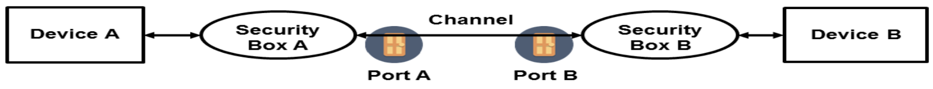

Figure 4. Using Port Boxes

6. General Structure

The connection between a device and a security box should not be exposed to other devices, entities, or the outside since it is unsecured. The connection between a port and a security box should be a single line or messages would be routed wrongly, messages passing this connection are already secured. The communication between ports can pass through any type of channels (wireless, Internet, etc.), messages are secured during this communication.

Our protocol can be used to support communication between multiple devices (Figure 5). Consider the following security boxes twins: *A* and *AA*, *B* and *BB*, *C* and *CC*. Devices *1* and *2* communicate with devices *4* and *5* through *A* and *AA*. Device *5* communicates also with device *3* using *2* twins: *B,BB* and *C,CC*. Device *3* and device *5* cannot communicate with each others.

If a message is sent from Device *3* to device *5* passes by the first twin *B,BB* and a second message passes by the second twin *C,CC*, there is a chance for the second message to arrive before the first one, therefore it is unsafe to have devices communicate with each others using multiple security boxes unless both devices *3* and *5* have the capacity of readjusting the order of received messages.

Finally, we have Port *1* communicating with ports *2* and *3*.

In conclusion we have the following:

- A security box can secure multiple applications/devices.
- A port can be connected to multiple security boxes and communicate with multiple ports.
- A device can communicate with multiple devices using the same security box.
- A device can communicate with a device using multiple security boxes if both devices have a measurement that keeps track of the correct messages order.

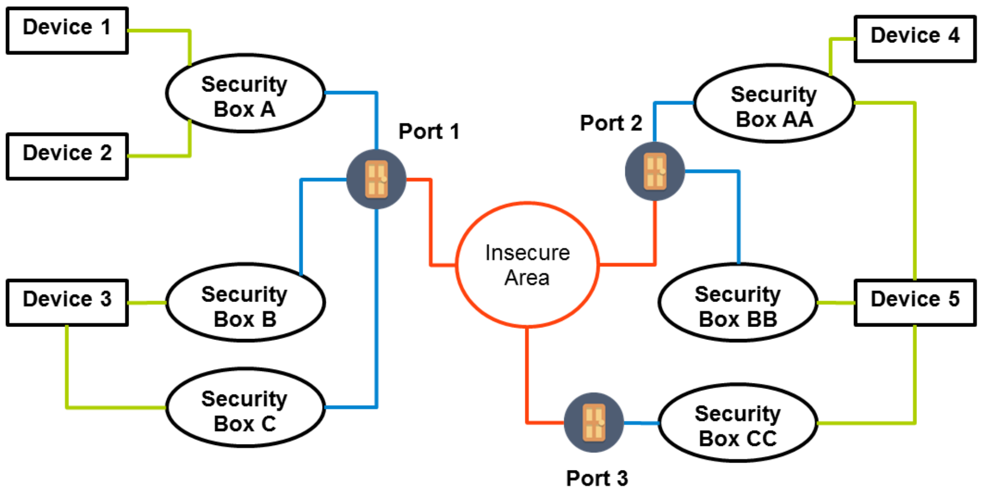

Figure 5. General Structure

7. Implementation

In this section we show an example of how the protocol can be used, and exhibit our simulation.

7.1. Example

Let us consider the following example that do not mention the use of port boxes (Figure 6).

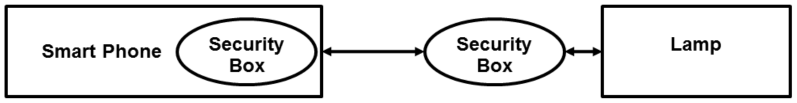

Figure 6. Lamp Control Example

We have a smart phone with many applications, one of which would control a lamp through a wireless communication channel. We need to make sure the communication between the lamp and the application is secured from any attack. The attacks may take advantage of the wireless insecurity or an insecure application installed on the smart phone. We create 2 security boxes: one would be an embedded device that is installed on the lamp's input/output, the second security box would be an algorithm on the smart phone and controls the input/output of the lamp control application.

Any communicated message between the lamp and the application would be translated into a secret message that has no meaning to any outsider (attacker) which guarantees confidentiality. Changing this message would result in a meaningless message which insures the integrity of the communication.

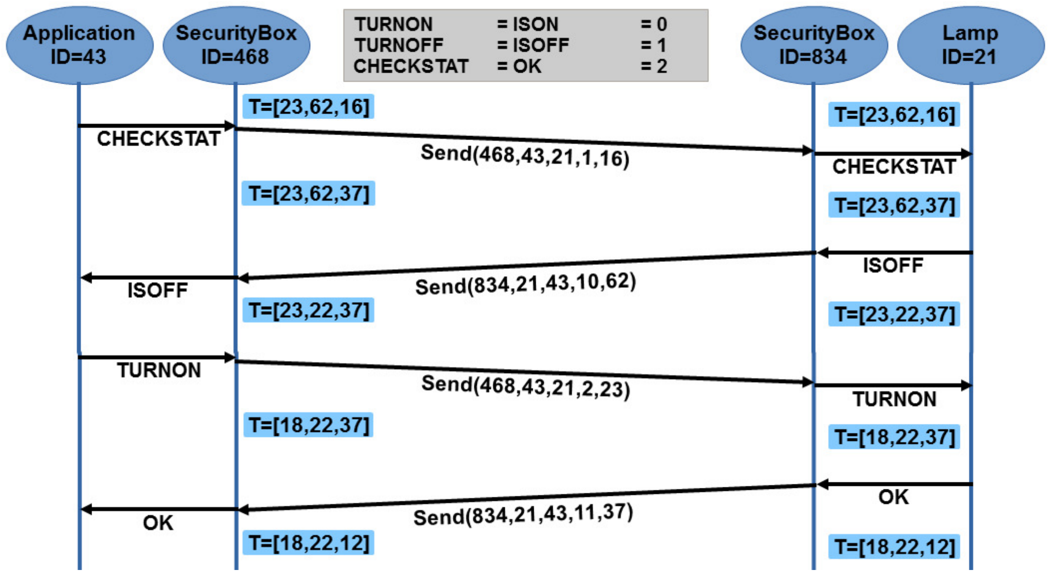

Figure 7. Simple Synchronous Method Example

In this example (Figure 7), we are using a table of *3* items, having *3* possible indexes means *3* different signals (information) sent and *3* different signals received. In the following we have: *TURNON = ISON = 0*, *TURNOFF = ISOFF = 1*, *STATUSCHECK = OK = 2*.

The first and most basic method consists on changing a value after using it. *Agent_A (ID=468)* and *Agent_B (ID=834)* start with a shared secret table *T* and a secret function *f*. *A (ID=43)* starts by sending a *CHECKSTAT* order, which is *index = 2*. Instead of actually sending *2*, *Agent_A* would send *T[2]* which is *16*. After sending *16*, *Agent_A* would change *T[2]* using *f* to obtain *37*. Since *37* replaced *16*, it would have the same index in the table *T*, and since the index is the actual indicator of the meaning of a message, then *37* would have the same meaning as *16* but with a different visible value. In other words, the next time *A* sends a *CHECKSTAT* order, or receives an *OK* response, the visible value in the message would be *37*.

7.2. Simulation

To test our protocol, we created a simulation of the lamp example using a python script on an Intel *i5* CPU *(2.60GHz*4)* with *4GB* of ram. We used the last version of the protocol which includes the following:

- All data are in 16-bits (table values, substitution-boxes values, coded messages, etc.)
- Asynchronous communication between twin security boxes.
- Many additional security measurements to make sure the security requirements are fulfilled.
- An improved version of the *f* function responsible for changing the secret table.
- Dynamic substitution boxes.
- An improved and secured use of messages sequencing.
- Possibility to send and receive unanticipated messages instead of expected signals only. This option reduces the performance.
- A resend option to be able to resend unreceived or erroneous messages.

During our simulations we were able to test different architectures with multiple devices, security boxes, and ports. Simulations were successful and showed no direct patterns on millions of communicated messages.

With all the security measurements implemented, we were able to communicate *4000* messages per second for each twins.

8. Conclusion

The SCADA community is looking for a security protocol that has a low cost and does not need constant upgrades. The protocol should also respect the constraints of SCADA such as the low computational power and the real-time environment.

Our security protocol can be a solution to SCADA security since on one side it has a good level of security based on simple operations and on the other side it does not require for the already installed system to change nor to stop working.

The most important security concerns (confidentiality, integrity, and availability) are taken into consideration. The simplicity of the used functions would outplay cryptographic algorithms in the matter of performance especially in the case of embedded devices.

No direct patterns were found, still, we have to continue analysing our protocol for indirect patterns.

We have already advanced in the design of the protocol to be able to communicate normal messages instead of a pre-set of values. We have also added a number of security measurements that consider potential attacks on the basic version of the protocol.

We may find uses of our protocol outside of the SCADA community if the requirements are met and the constraints are respected. For the moment, we focus our work on SCADA systems.

We are currently tuning our simulation to achieve the best possible performance.

We will soon implement our latest version of the protocol on a SCADA platform that we acquire. We have also considered collecting the communicated packages and diffuse them for white hat analysis.

We consider doing some security analysis to test the robustness of our protocol against specific attacks such as correlation and differential power analysis.

We are working with other partners that wish to implement our security protocol in their latest project which aims to put sensors on the repeaters of submarine communication lines to observe and forward information to a research centre.

ACKNOWLEDGEMENTS

The authors would like to thank research engineer Ciprian Teodorov for his encouragements, reviews, and valuable comments. The authors would also like to thank all colleagues that influenced this work.

FUNDING

Funded by the French Government Defence procurement and technology agency - Information department (DGA-MI) and the urban community “Brest Métropole” (BMO).

REFERENCES

[1] Igure, V.M., Laughter, S.A. and Williams, R.D., 2006. Security issues in SCADA networks. Computers & Security, 25(7), pp.498-506.

[2] Pollet, J., 2002, November. Developing a solid SCADA security strategy. In Sensors for Industry Conference, 2002. 2nd ISA/IEEE (pp. 148-156). IEEE.

[3] Bowen, C.L., Buennemeyer, T.K. and Thomas, R.W., 2005, June. Next generation SCADA security: best practices and client puzzles. In Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop (pp. 426-427). IEEE.

[4] Chandia, R., Gonzalez, J., Kilpatrick, T., Papa, M. and Shenoi, S., 2007, March. Security strategies for SCADA networks. In International Conference on Critical Infrastructure Protection (pp. 117-131). Springer US.

[5] Nicholson, A., Webber, S., Dyer, S., Patel, T. and Janicke, H., 2012. SCADA security in the light of Cyber-Warfare. Computers & Security, 31(4), pp.418-436.

[6] Vernam, G.S., 1919. Secret signaling system. U.S. Patent 1,310,719.

[7] Verfahren, 1923. Device and circuit for News About averaging in cipher. Google Patents DE 371,087. Available: https://www.google.com/patents/DE371087C?cl=en

[8] Shannon, C.E., 1949. Communication theory of secrecy systems. Bell system technical journal, 28(4), pp.656-715.

AUTHORS

Fadi Obeid is a PhD student at ENSTA Bretagne. He is currently researching security solutions for SCADA systems. He received his masters degree in information security and cryptology from the university of Limoges in 2013. He is mainly interested in side channel analysis, security properties verification using model checking, and unbreakable security protocols.

Philippe Dhaussy is a professor at CNRS Lab-STICC within ENSTA Bretagne. His expertise and his research interests include model-driven software engineering, formal validation for real time systems and embedded software design. He has an engineer degree in computer science from ISEN (French Institute of Electronics and Computer Science) in 1978 and received his PhD in 1994 at Telecom Bretagne (France) and his HDR in 2014. From 1980 to 1991, he had been software engineer and technical coordinator in consulting companies (Atlantide group), mainly in real-time system developments. He joined ENSTA-Bretagne in 1996, as professor. He has over 60 publications in the areas of software engineering and computer science. He has been co-supervisor for five PhD students, has been and is involved in several research projects as work package coordinator.