Secure Communication Protocol: Application to Large Number of Distributed Sensors

Fadi Obeid
Université Bretagne Loire
Laboratoire Lab-STICC
UMR CNRS 6285
ENSTA Bretagne, Brest
Email: fadi.obeid@ensta-bretagne.org

Philippe Dhaussy
Université Bretagne Loire
Laboratoire Lab-STICC
UMR CNRS 6285
ENSTA Bretagne, Brest
Email: philippe.dhaussy@ensta-bretagne.fr

***Abstract*—Supervisory control and data acquisition (SCADA) systems control many of our critical industrial infrastructures. Currently, most SCADA systems are considered insecure due to their lack of security measures. The increased number of connections between SCADA systems along with other factors, caused an augmentation in the threats and attacks on SCADA. Many solutions were proposed to secure SCADA communications. However, any undertaken security measure while communicating unencrypted messages would not be robust in case of attacks. The numerous embedded devices used in SCADA make it hard to consider classic cryptography, although, some SCADA systems already implemented such algorithms. In this paper, we present a new approach to secure SCADA communications by using dynamically modified signals instead of plain or encrypted messages. Our solution can be implemented in low cost electrical chips (used on sensors, switches, etc.) or as a software (used on servers, etc.). The Reconfigurable Information Transmitter Agent (RITA) protocol that we present can also be used to secure any type of communication that respects the protocol's constraints. The solution, while still lacking the necessary analysis to ensure its security level, is promising. This approach is planned to be implemented on a large number of distributed underwater sensors.**

***Keywords:* Information Security, Secure Communication, Cryptography, SCADA**

I. Introduction

SCADA systems [3], [2] monitor and control infrastructures to facilitate and improve their usage. Our application model consists of a large number distributed sensors, which is a specific part of a SCADA model. SCADA systems are considered secure by isolation and have little security measures. For example, the most important security measure in a nuclear facility is the military guards defending it.

While these measures were acceptable in the past, recently there was a high increase in the attacks on SCADA. For instance, the security by isolation is only effective if we suppose that we can trust all the individuals that has access to the physical space of all the SCADA entities. This is a very optimistic consideration, and it becomes illogical in the case of water pipes, power lines, and traffic lights.

SCADA has also seen an augmented need for outside connectivity to facilitate the monitoring and the control over the infrastructures. The new security requirements render the security by isolation ineffective, which means that new measures need to be taken. Another reason for SCADA systems being unsecured is that most companies that rely on SCADA systems do not consider securing these systems because of the expected high costs.

Most SCADA nodes are embedded devices with low computational and power provisions. Additionally, in many cases SCADA communication needs to respect a hard real-time constraint. The consequence of these 2 reasons is a lack in cryptography support making any security protocol based on cryptography unsatisfying.

Our proposal, the Reconfigurable Information Transmitter Agent (RITA) protocol, is to replace cryptography with an approach that is expected to have a satisfying security level with a very low cost (power, time, and physical 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.

We plan to implement this approach on a use case of a large number of distributed sensors. We are currently working on multiple simulations and experimentations. Some are used to improve the protocol's performance, others are used to find additional information about the protocol's security. We also have simulations directly related to the use case.

This article is organized as follows: We start by presenting a background of our work in section II. In section III, we present the principles of our protocol. In section IV, we compare the robustness of our protocol with the robustness of other similar-looking protocols such as the one-time pad. In section V we present some experimentations with the protocol. We demonstrate an overview of our use case in section VI and explain how we apply our approach in this case. Finally, we summarize our work and present future plans in section VII.

* Partially supported by Research Grant from Brest Metropolis as well as the French Government Defense Procurement and Technology Agency - Information Department (DGA-MI).

II. Background

Many studies [8], [7], [14] were conducted to address the security problems in SCADA and/or give some theoretical solutions and guides for improving SCADA security. Other studies [13], [5] aim to enhance the security level and fortify the provided services along with the managed critical data. SCADA-specific security solutions and SCADA-specific IDS are proposed in [6] and [15] respectively.

Our protocol is based on the concept of not using the same secret information (such as a key) twice. This is considered in abstract to be unbreakable since an attacker would not have useful information about the encryption.

The only cryptosystem that is considered perfectly secure is the Vernam cipher, also called the one-time pad. Gilbert Vernam patented this invention in the USA in July 1919 [12]. A variation of the one-time pad was patented a few years later in Germany by Siemens and Halske [1]. The one-time pad is based on a list of shared keys that can only be used once. These keys should be randomly generated and never reused.

The main requirement of the one-time pad is the ability to share a list of random keys.

Using a truly random key (K), the one-time pad can generate a cipher-text (C). The simplest implementation of the one-time pad is $C = K \oplus M$ where M is the plain-text. Since an intruder can only know C the function becomes mathematically unsolvable (1 equation with 2 unknowns).

The one-time pad was exploited on various occasions, especially during and after world war two. Although, in all documented exploits, the properties of the one-time pad were not respected. Either the key list was reused, or multiply produced, or the generated values were not truly random.

In 1949, Claude Shanon proved that the one-time pad is unbreakable and that any unbreakable system must have the same properties as the one-time pad [11].

When implemented, the lists of shared keys need to be updated constantly which can be problematic. To the best of our knowledge, other than the one-time pad, there are no other cryptosystems that were considered unbreakable.

We also use dynamic substitution and dynamic substitution boxes which are already used in many secure algorithms. The idea is to replace the static s-box with a dynamic s-box ensuring additional security. [4] demonstrate a dynamic substitution model.

III. Principles of the Protocol

Our protocol consists of two entities sharing and maintaining a secret. We call these entities security boxes, and once they share a secret they become twins. In comparison with the one-time pad, the secret boxes represent the agents and the shared secret represents the lists of secret keys.

A. Initial Concept

The shared secret is based on 2 tables of randomly pre-initialized values, along with an updating algorithm. The algorithm uses simple binary operations (such as XOR, addition, etc.) To guarantee a high performance in physical implementations. The operations of our algorithm take place after sending/receiving a message and not before. This means that messages are sent and processed with almost no security related latency.

If we want to secure the communication between 2 devices A and B, we proceed as follows: The twins SA and SB are placed each on a device, allowing these devices to communicate with each others. A security box can be an embedded device, a computer program, a smart phone application, etc. Any communication between devices would go through both security boxes. The first security box (sender) matches the communicated message with a value from the secret table. The second security box (receiver) would reveal the actual message by matching the value in its own version of the table. After sending or receiving a value, each security box would update its copy of the secret table based on the value it sent or received.

Figure 1 demonstrates an example of how this works. We have 2 devices A and B communicating with each others using the twins SA and SB. Each security box uses T_s to send messages and T_r to receive them. T_s in SA is equal to T_r in SB and T_r in SA is equal to T_s in SB. If we don't use different tables, asynchronous communication would not be possible (exp. case in which both boxes send a message at the same time) Note that values in the same table should never be equal.

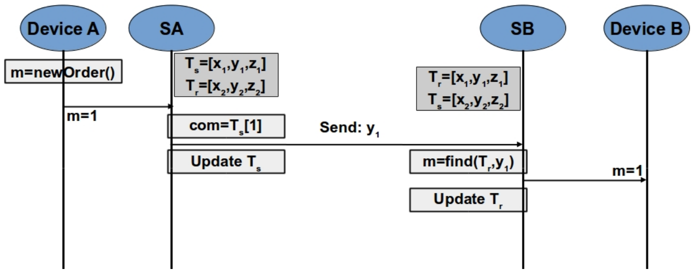

Figure 1. RITA Protocol Example

Initially, T_s in SA is equal to T_s in SB and T_r in SA is equal T_r in SB. Since the algorithm would always conduct the same operations on both sides during updates, table T_i in SA would be equal to T_i in SB if both has done the same number of updates.

To create the twins relation between SA and SB, we have multiple choices: SA and SB can be initialized together with the same algorithm and same secret table. SA and SB can have a common default architecture, then, using a key (used only once), they can each generate the common table and algorithm. In both cases, the algorithm can either be specific to the twins, or a general algorithm configured upon initialization.

We notice, before sending the message, SA only have to read an item in the table using the index. Also, after receiving the coded message, the signal is revealed by searching the index of a value in a table. This has a very good impact on the communication speed.

B. Improved Security

To improve the security of our approach, each table has 2 rows, one for communication uses (leaks information) only and one for updating only (fully secret). Each value in the communication row has a matching value in the updating row. which means that if we send $T_s[0][i]$ at some point, we use $T_s[1][i]$ as the base element to update T_s. When the table is being updated, all the values in the updating row play a role in the production of a new table (communication row and updating row). Since an attacker can only see $T_s[0][i]$, and he cannot see any value of the updating row, the input of the updating algorithm is totally obscure to him.

If additional security is needed, it would have a small effect on performance, but still be done. We can use 2 additional tables, each has 2 values. The first table S is used to scramble messages being sent and the second table R is used to unscramble received messages. The first value in each of these tables is the value used for scrambling and unscrambling and the second is used to update the table itself. S in SA is initially equal to R in SB and R in SA is initially equal to S in SB.

Finally, we use dynamic substitution and permutation boxes in the updating algorithm. This is important to create confusion and diffusion, which means that the output of the updating algorithm does not leak information about its input. In this case, the output is the table of both communication values and secret values, while the input is the row containing the secret values only. Later, when another message is sent, an attacker can see a small part of the output which is a communication value. When used correctly, this would make it harder for an attacker to analyze the protocol, if not impossible (we cannot be sure about this without additional verifications).

C. Usability

The protocol is efficient when using small tables, this means that we need to have a very limited vocabulary of the messages being sent.

The strong points of the protocol are present when applied on a limited vocabulary in m2m communications. However, the RITA protocol is not limited to small size messages, limited signals, nor m2m communications.

In other cases, we can use multiple boxes for the same message, and base our tables on binary codes. This has not been tested yet, it is however expected to be less effective in terms of cost than classic cryptography.

IV. Robustness

The one-time pad uses a key only once, reusing a key is a vulnerability, this is due to the possibility to attack the protocol by detecting the periodic reuse of the key. the key in RITA is the combination between the last state of the table (which is secret) and the initial algorithm (whether known or not, configurable or not, etc.). The last state would depend on all the previously sent messages, more precisely, on the corresponding secret for each message. It also depends on the initial table which is a secret.

For the moment, we cannot be sure whether it is important for the updating algorithm to be secret or not. If so, it can be done by generating different combinations of the operations in the algorithm itself, which would be initialized when the security boxes are initialized. This adds to the cost, especially in the case of a direct physical implementation of the approach.

Whether the algorithm is known or not, its input is always secret. If the attacker does not know the initial table, and all the subsequent updates to the table, he cannot know its current state. If we compare this to the one-time pad, the table may go back to an already visited state at some point and send the same message, this is equivalent to using the same key. However, the reuse of the key (which is the secret table) in the case of RITA is not periodic, this is due to the randomness of the information being sent. Also, the attacker cannot analyze the reuse of the key due to the confusion and diffusion.

Consider the following case: at some point, we want to send m_j and the current state of the table is s_k, at another point of time, the table state happens to be s_k as well, and the signal to send is m_j. Both cases would be using the same key which breaks the property of the one-time pad that makes it perfect. However, from an attacker's point of view, he can never know at which state this happens, and the dynamic of the data being communicated prevents this from happening periodically.

The protocol insures the security properties as follows:

Confidentiality: If an attacker C intercepts the message m_i corresponding to signal x, sent from SA to SB, C does not know the meaning of this message, therefore the property is respected. If in the future, if C intercepts the same message m_i, this time m_i can correspond to x, but can also be a different signal, the attacker has know clue which is the case.

Authenticity: Messages between twins have very limited choices, for instance, if we have a table of 3 messages each encoded in 32 bits, at any point of time, having a correct message has a chance of $3/2^{32}$. This means that when a message is received by a security box, and it happens to be correct/readable, we can be sure it is authentic.

Authenticity: If a message is forged, it has a probability of success of $n/|G_b| = n/(2^b)$ where n is the number of different possible signals 3 in the example above, and b is the default length used for the coded messages (32 for example). $3/|G_{32}| = 3/2^{32} \approx 7 * 10^{-10}$ is about fifth the probability of gaining the jackpot Mega Millions multi-state lottery in the United States.

Integrity: For the same reason, modifying a message, or the order of messages, or replaying a message, also has a chance of $3/2^{32}$ of using a correct message. Also, even if the attacker considers already sent messages, each time he forges/modifies/resends a message, he would have the same chance of success.

Availability: An attacker C can interrupt the communication between SA and SB, however, he cannot replace the interrupted messages. If SA and SB use a time constraint based on a continuous checkup messages when no other data needs to be exchanged, any interruption would be detected and cause an alert.

V. Experimentations

In order to study and improve the algorithm, we created different computer based simulations with 2 or more devices communicating using the security boxes. Although this is not ideal for performance measures, our experimentations helped us improve the algorithm from its original form to the security level it currently has.

We also experimented with different configurations of the protocol, for example, the differences between updating the whole table after each message or updating only the column of the communicated value.

In some cases, we realized some simple pattern search on the table states, and the values being sent. Even in cases where the size of messages was very small (8 bits) and the original messages from the devices were static, there was no simple patterns in the table states nor the coded messages being sent. We also noticed, that recently sent values may reappear, or not. This is important because it means that an attacker cannot improve his guess based on the last sent messages.

In figure 2 we have a device that generates messages as fast as possible (no wait between messages). These messages are then treated by a security box, we have two cases, one in which only the sent value from the table is updated after sending, and the second is where all values are updated. As expected, updating the whole table costs more time than updating only one value, however additional information can be useful in this area.

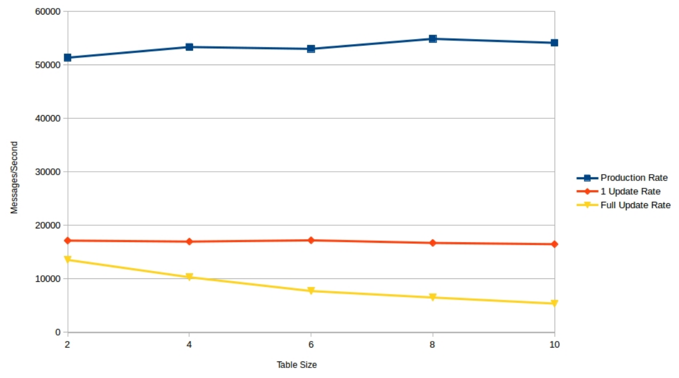

Figure 2. Simulation With no Wait Between Messages

In figure 3 we simulate different latencies in messages production. In our case, if we send less than 20000 messages per second (wait more than 0.05 ms between messages), using the protocol with one variable update does not affect the performance of the communication. Also, if we send less than 6666 messages per second (wait more than 0.00015 ms between messages), having full table updates does not affect the performance of the communication. This is rather efficient, while the protocol can still be modified to improve its efficiency. Also, a physical implementation is supposed to have even better results.

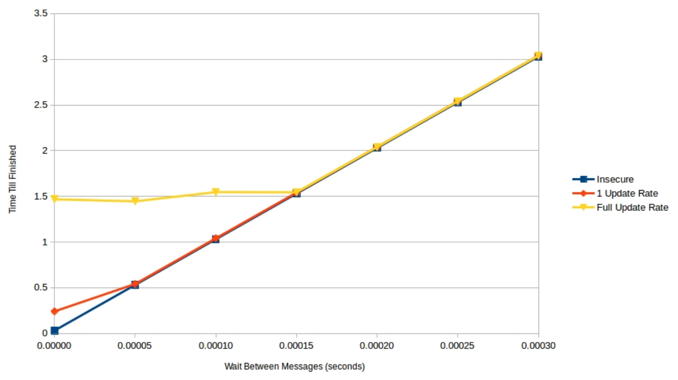

Figure 3. Simulation with Different Waiting Periods

VI. Use Case

Our use case consists of a large number distributed sensors which are attached to repeaters in under water smart cables. Figure 4 consists of an abstraction of how sensors are installed on the repeaters with a communication panel between each sensor and repeater. The communication is established as follows: The sensor S_i sends its measurements to the communication panel C_i. The communication panel prepares the message and sends it through the repeater R_i which forwards it through the cable until it reaches a host (A or B). The host forwards this message to the research center SC.

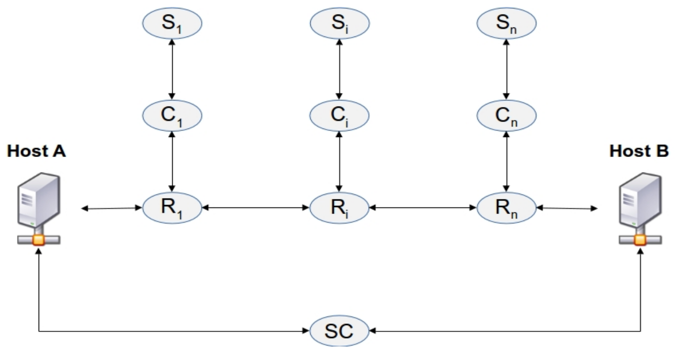

Figure 4. Abstract Model for distributed sensors

In some cases, it is possible to have data sent through only one host. Additionally, the communication between the hosts and the research center may have a different protocol than between the hosts and the communication panels.

We consider a worst case scenario where communication between the hosts and the research center is insecure. We also consider the communication between a host and a communication panel to be insecure. Our concept is to secure messages between the sensors and the communication panel. The application of the our security measures can be either on the sensor output or the panel input. However, we consider it safer to be on the panel input so it would not disrupt or affect the sensor measures. From the communication panel to the research center, the data is coded in a way it cannot be modified nor red by any middle entity.

The security properties are:

- Confidentiality: Messages from S_i to SC (and reverse) can only be read by SC and S_i.
- Authenticity: When SC receives a message from S_i, SC is sure that the message originates from the exact sensor and was not sent by another sensor, or forged by a third party. Additionally, when S_i receives a configuration message from SC, S_i is sure that the message originates correctly from SC.
- Integrity: Once a message is created by SC or S_i, it cannot be modified. The format itself can be modified depending on the protocols in the hosts and the communication panels, however the data and its meaning are untouchable, and if changed, the receiving party would detect that something is wrong.
- Availability: If at some point the communication is interrupted, both parties would know that.

The concept of the RITA protocol is not originally conducted for the purpose of this system. However, this system is a specific application of a SCADA model, and the RITA protocol matches the requirements and constraints of this system. Our work on this use case is still in the simulation phase, however, the current results are promising.

In the case of the distributed sensors, we can consider the limitations of the changing values to have a very limited vocabulary. For example, if we have a sensor that can change its value by 0.1 point each $1ms$, and send its state each $1ms$. Then, we can have a security box with tables of 3 values representing the following codes: $+0.1$, -0.1 and 0. The first information can be sent with a one time key, or without any security measures. From this point forward, we only send the changes, which highly reduces the vocabulary and the size of a message. The security box should be able to update the table in less than $1ms$, if not, messages would have to wait in the input pipeline of the security box. If the speed is high enough, this means that each message can be sent almost instantly, and, before the next message arrives at the security box, it would have already updated its table.

The SC can use multiple security boxes, either physically implemented as for the security boxes, or computer simulated depending on its needs. In case the SC needs to communicate back with the sensors, we would need tables depending on the different possible configurations and commands.

VII. Conclusion

The RITA protocol is a new concept based on old techniques that were rarely used such as the one-time pad. We believe that given the right implementation area, the protocol can be very efficient, and even better than advanced cryptosystems.

The initial RITA protocol (without the advanced security schemes) was already published in NCS2016 [9]and ACO2016 [10], and legally patented.

Although many improvements were done since the original concept, the protocol is still in its early stages, and additional simulations and tests are required to improve it. An important test for the protocol would be its resistance to attacks and analyzes, especially checking for possible patterns (only simple patterns were tested for the moment).

We aim to have additional experimentations using multiple embedded devices simulating the sensors, the security boxes, and the communication panels. We are also working on other simulations not related to the distributed sensors project. We are exploiting the SCADA and IoT domains for usage of our protocol.

Finally, other analyses need to be done in order to insure the algorithm is secure. We need to verify whether the secrecy of the updating algorithm is important or not. If the algorithm is not secret, can an attacker, that has access to all the coded messages, be able to obtain some information about the secret table? or other messages? Also, can an attacker, with access to unlimited plain texts, analyze the device coding the messages to obtain information about its original (or current) secret table? These questions need to be studied carefully, such analyses take time in order to insure their completeness.

References

[1] Verfahren, vorrichtung und schaltungsanordnung zur nachrichtenuebermittlung in geheimschrift method, device and circuit for news about averaging in cipher, Mar. 10 1923. DE Patent 371,087.

[2] S. A. Boyer. *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009.

[3] A. Daneels and W. Salter. What is scada? 1999.

[4] M. A. El-Fotouh and K. Diepold. Dynamic substitution model. In *Information Assurance and Security, 2008. ISIAS'08. Fourth International Conference on*, pages 108–111. IEEE, 2008.

[5] I. N. Fovino, A. Coletta, A. Carcano, and M. Masera. Critical state-based filtering system for securing scada network protocols. *Industrial Electronics, IEEE Transactions on*, 59(10):3943–3950, 2012.

[6] I. N. Fovino, A. Coletta, and M. Masera. Taxonomy of security solutions for the scada sector. *Project ESCORTS Deliverable*, 2, 2010.

[7] V. M. Igure, S. A. Laughter, and R. D. Williams. Security issues in scada networks. *Computers & Security*, 25(7):498–506, 2006.

[8] R. L. Krutz. *Securing SCADA systems*. John Wiley & Sons, 2005.

[9] F. Obeid and P. Dhaussy. Rita secure communication protocol: Application to scada. In *8th International Conference on Network and Communications Security (NCS 2016)*. Editor(s): David Wyld et al, 2016.

[10] F. Obeid and P. Dhaussy. A secured data communication protocol. In *Conf. A Connected Ocean (ACO 2016), SeaTech Week, Brest*, october 2016.

[11] C. E. Shannon. Communication theory of secrecy systems*. *Bell system technical journal*, 28(4):656–715, 1949.

[12] G. Vernam. Secret signaling system, July 22 1919. US Patent 1,310,719.

[13] Y. Wang. sscada: securing scada infrastructure communications. *International Journal of Communication Networks and Distributed Systems*, 6(1):59–78, 2010.

[14] B. Zhu, A. Joseph, and S. Sastry. A taxonomy of cyber attacks on scada systems. In *Internet of things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing*, pages 380–388. IEEE, 2011.

[15] B. Zhu and S. Sastry. Scada-specific intrusion detection/prevention systems: a survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*, 2010.