www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2018, VOL. 64, NO. 2, PP. 107-114
Manuscript received September 30, 2017; revised April, 2018. DOI: 10.24425/119357

Proposal of a Secure Modbus RTU Communication with Adi Shamir's Secret Sharing Method

Éva Ádámkó, Gábor Jakabóczki, and Péter Tamás Szemes

***Abstract*—Drinking fresh water, turning the lights on, travelling by tram, calling our family, or getting a medical treatment are usual activities, but the underlying SCADA (Supervisory Control and Data Acquisition) systems like CIS (Critical Infrastructure Systems), ICS (Industrial Control Systems) or DCS (Distributed Control Systems) were always the target of many types of attacks, endangered the above mentioned simple activities. During the last decades because of the fast spread of the internet based services and the continuous technical development these systems become more vulnerable than ever. Full reconstruction and innovative changes in older SCADA systems has high cost, and it is not always rewarding. Communication protocols as Modbus (1979) serve as a main basis for SCADA systems, so security of Modbus has a major impact of the security of SCADA systems. Our paper raises and answers questions about the security of the Modbus RTU protocol. We focus on the serial Modbus protocol, because in that method we found many unsolved problems, like lack of authentication of the participants, lack of secure channel and so on. The aim of this paper to propose a secure communication alternative for Modbus RTU @ RS485 wire. The main advantage of the proposed method is the coexistence with traditional slaves and bus systems and only software update is necessary.**

***Keywords*—SCADA, Modbus RTU, secret sharing, secure communication**

I. Introduction

COMMUNICATION protocols as Modbus (1979) or Profinet (2000) serve as a main basis for SCADA systems. Modbus, as the 'de-facto' standard is used in 22% of industrial applications (as of 2010), therefor the known weaknesses of the standard pose a significant security risk. [1] The main design features of the Modbus protocol were reliability, speed and accessibility, but a very important feature was left out, security or to narrow it down, security through cryptographic methods. As it is well-known the Modbus protocol is a request-response protocol, the participating devices are in master-slave relationship with each other. Only the master can initiate requests, the slave serves as a server. The master broadcasts these messages on the physical layer and the addressed slave answers, but neither the slave, nor the master can authenticate themselves in a satisfying manner.

There are several different applications of the Modbus protocol. We focus on the Modbus RTU, where the security on the physical layer is as important as the network security in the case of Modbus TCP. Opposed to the Modbus RTU the Modbus TCP/IP is the object of numerous studies and has number of tested methods.

Authors are with University of Debrecen, Faculty of Engineering, Hungary (e-mail: adamko.eva@eng.unideb.hu, jakabockigabor@gmail.com, szemespeter@eng.unideb.hu).

The goal of this article is to examine networks based on the Modbus RTU standard with the attack tree method, then present a security protocol that counteracts these vulnerabilities. In the second section we give a brief introduction to SCADA systems and some recent attacks, the third section is about the Modbus RTU protocol and its revealed vulnerabilities. In the fourth section we present the developed secure protocol. Section five stands for the practical implementation of the developed protocol, and in section six we briefly mention some related solutions. Then in the last section we present our conclusions of the research.

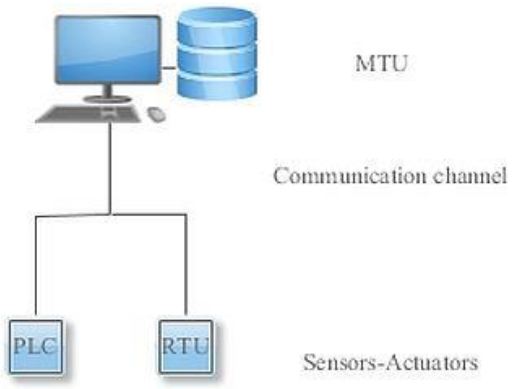

Fig. 1. Structure of SCADA system

II. SCADA system

SCADA stands for Supervisory Control and Data Acquisition, SCADA system is capable of gathering, tracking, analyzing real time data and monitoring or controlling sensors and actuators. It is widespread in industrial application, like electric power generation, water distribution, waste control, transportation and so on. SCADA systems typically consist of a central host computer - which usually marked as an MTU (Master Terminal Unit) - and field devices - Remote Telemetry Units (RTU) or Programmable Logic Controllers (PLC) -, like sensors or actuators. MTU is an input-output device controlled by a human operator through a Human Machine Interface (HMI), which is a collection of software components and databases. Field devices can measure or set values of parameters in a system depending on its function. PLC and RTU are almost the same, difference between them only that the RTU has more communication interfaces while PLC cannot just measure or set variables but can process and control them. MTU and

RTUs/PLCs can be connected through different communication systems, like radio signals, cables, wireless and so on. In these communication systems different communication protocols can help solving the problem of message exchange. The most commonly used industrial communication protocols are the following: Modbus, Profinet, DNP3 (Distributed Network Protocol), IEC 60780 (International Electrotechnical Commission), OPC (OLE for Process Control) or Ethernet/IP. A typical SCADA structure can be seen on Figure 1.

III. MODBUS

A. Modbus RTU

Modbus is an open access communication protocol designed by the Modicon and released in 1979, it become a 'de-facto' standard in industrial applications for today. Modbus versions like Modbus RTU, Modbus ASCII and Modbus TCP/IP give solution for communication on data link and application layer of ISO/OSI model. The place in the ISO/OSI model of the protocol can be seen on Figure 2. Modbus is a master-slaves, request-response protocol where only the master can initiate a request, the slaves only able to response to it.

Modbus TCP/IP is the type of Modbus which located on the application layer of the ISO/OSI model, it provides client server communication between devices, the client role is provided by the Master of the serial bus and the Slaves nodes act as servers. Numerous studies examine the security of this protocol, and offer different solutions. However older latency systems based on Modbus RTU/ASCII should revise the level of the security of the system too.

Modbus RTU is located on the second layer of the ISO/OSI model, it communicates on a serial line usually on RS485 or RS232. It is a master-slaves protocol, where only one master and at most 247 slaves can be part of one network. Only the master can initiate a request as it above mentioned, depending on the type of the request the master is waiting for a single response or many responses. Request can be sent in unicast or broadcast mode, it requires that each slave have a unique address. The master starts only one transaction at a time. The slaves never transmit data without receiving a request from the master node and never communicate with each other. A Modbus RTU message consists of frames sending continuously, a frame contains four different fields, first is the address of the slave, second is the function code - it specifies the slave what kind of action to perform -, third one is the data, and last is the error checking code, it uses a Cyclical Redundancy Checking (CRC) method. The CRC is the only feature which supposed to provide the data integrity. On the serial bus between two frames should be at least a 3.5 characters long delay. Figure 3. shows the structure of the Modbus frame, and on Figure 4. a message exchange can be seen in unicast mode.

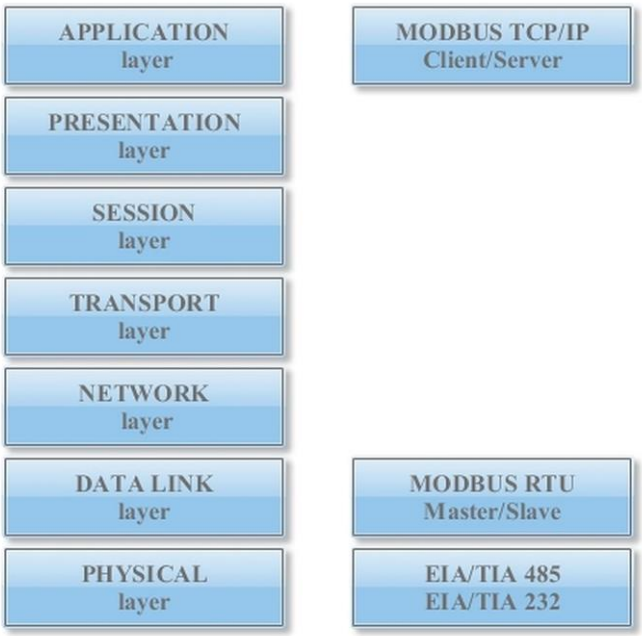

Fig. 2. Modbus protocol in ISO/OSI model

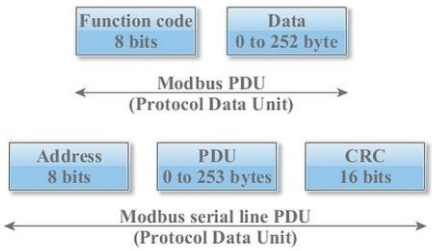

Fig. .3 Modbus RTU Frame

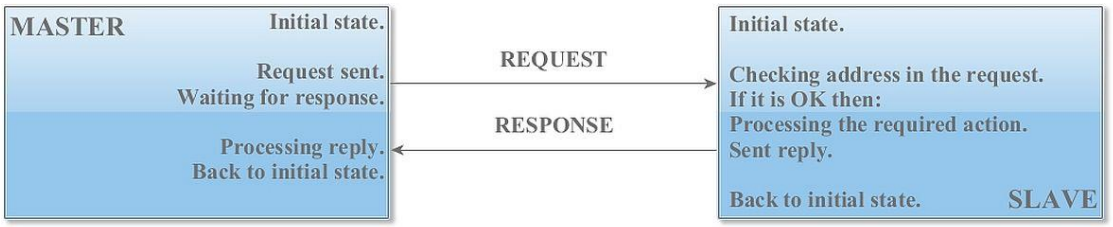

Fig. 4. Modbus RTU unicast communication

B. *Modbus vulnerabilities*

As it mentioned before during the design process the main features of the Modbus protocol were reliability, speed, and accessibility, but security, authentication and data integrity were left out. Our research addressed to find the vulnerabilities of the Modbus protocol, the following liabilities revealed from the literature:

- Lack of Confidentiality. [2]
- Lack of Integrity.
- Lack of Authentication.
- Sensibility for the Man in the middle (MITM) attack. [3]
- Sensibility for the Denial of Service Attack (DoS). [4]
- Interception [5]
  - Slave Reconnaissance
  - Modbus Network Scanning etc.
- Interruption
  - Remote Restart
  - Baseline Response Replay etc.
- Modification
  - Diagnostic Register Attack
  - Broadcast Message Spoofing etc.
- Fabrication
  - Direct Slave Control etc.

We summarized the revealed weaknesses of the Modbus RTU protocol with the simple and manageable method of attack tree [2,6]. In our previous papers, we focused on algorithmic security of the protocol, but not on the physical level, so the attack tree does not contain the analysis of the physical security of networks or methods to gain access to them, like through social engineering. The final attack tree is presented on Figure 5.

With this technique, the system can be examined from several points of view. Our first goal was to determine the purpose of the malicious attacker. As we can see many deficiencies can be found because the lack of the two most crucial components of security properties confidentiality and integrity. On Figure 4., we can see the attacker targets four weaknesses of the protocol, these are the root elements of the attack tree. First is the interception of data, which includes for example the channel monitoring, then the interruption of the communication - it can be caused by for example a DoS attack -, third one is the modification of messages and the last one is the fabrication of data. Our second goal was to uncover how can a malicious attacker achieve these targets, due to the related research all the root elements can be reached by MITM attack, like planting gateways or compromised slave or master devices into the network.

1) *Interception*

In the tree, the first root is the interception of data. During interception, the attacker captures messages sent over the network, thus acquiring information about the parameters and operation of the system, so the confidentiality of the data and system is lost in this attack.

2) *Interruption*

With the second root, the attacker's goal is to degrade the effectiveness of the system e.g. economic efficiency, or similar parameters or to manipulate them, or just simply overload the system to shut it down. Compromising the normal operation of the system cause the damage of data or system integrity.

3) *Modification*

The third root is for modification, which means that an unauthorized individual modifies the messages sent over the channel, resulting the damage of data or system integrity.

4) *Fabrication*

Last root is for fabrication, during fabrication a malicious attacker releases itself as an authorized user and send fabricated message to the participants of the system. Many properties of security lost through this attack, like data integrity, authentication, or confidentiality.

IV. SECURE PROTOCOL

A. *Problem formulation*

In the previous section we showed that the main problems of the Modbus RTU protocol are the lack of security parameters as confidentiality, integrity, and authentication of master and slaves. The latter causes that a slave device accepts request and performs tasks any time if the destination address is its own address in the Modbus PDU, and the master does not verify the origin of the replies it receives. So, the problems to be solved are to prevent the messages from eavesdropping, to precisely authenticate the slaves and the master of the network and to provide data integrity. The authentication cannot be done by trusted third parties or organizations, for not every system is connected (or can be connected) to these organizations, so security protocols based on digital certificates are excluded. In our solution we take advantage of the fact that the Modbus RTU based communication not always uses the full of its standardized message length and the relative low update frequency of the messages. In most cases the MTUs and field devices have an Advanced Encryption Standard (AES) engine and encrypt the standard request and response before sending it. Many similar data put through the system in normal operation, - like usual requests from the master, or sensor data, which has long alteration times -, and because the physical layer is exposed to attacks, the attacker can obtain large magnitude of data, shortening the time needed to compute the shared secret key, which can be then used to read the slaves responses or alter the master requests. Also, the data encryption is not enough alone to solve all the earlier revealed security problems of the protocol.

TABLE I
NOMINATIONS 1.

Nomination	Explanation
M	Master.
S_i	Ith Slave.
$NumSlavesone$	Number of the slaves in the network.
$RandomP(x)$	A random polynomial over $GF(p)$.
(x_i, y_i)	Ith point of $RandomP(x)$ polynomial.
a_i	Ith coefficient of $RandomP(x)$ polynomial.
n	Degree of $RandomP(x)$
p	Large prime number.
$prime()$	Prime generator function.
$skey()$	Secret key generator function.
$encrypt()$	Function for encryption.
$decrypt()$	Function for decryption.
$LagrangeIP()$	Function constructing secret by Lagrange interpolation.
K_S	Secret key of the master.
C_i	Challenge value of ith slave.
P_{ij}	Share.
$Address_i$	Address of the ith slave.

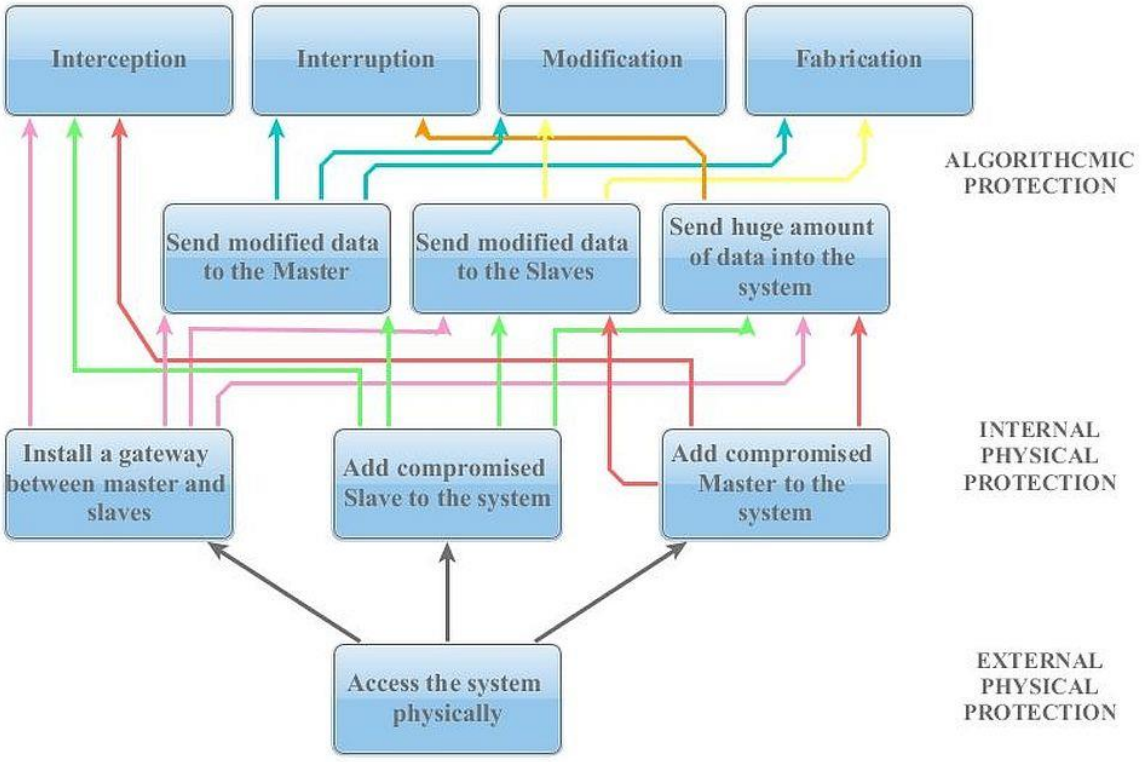

Fig. 5. Modbus RTU attack tree

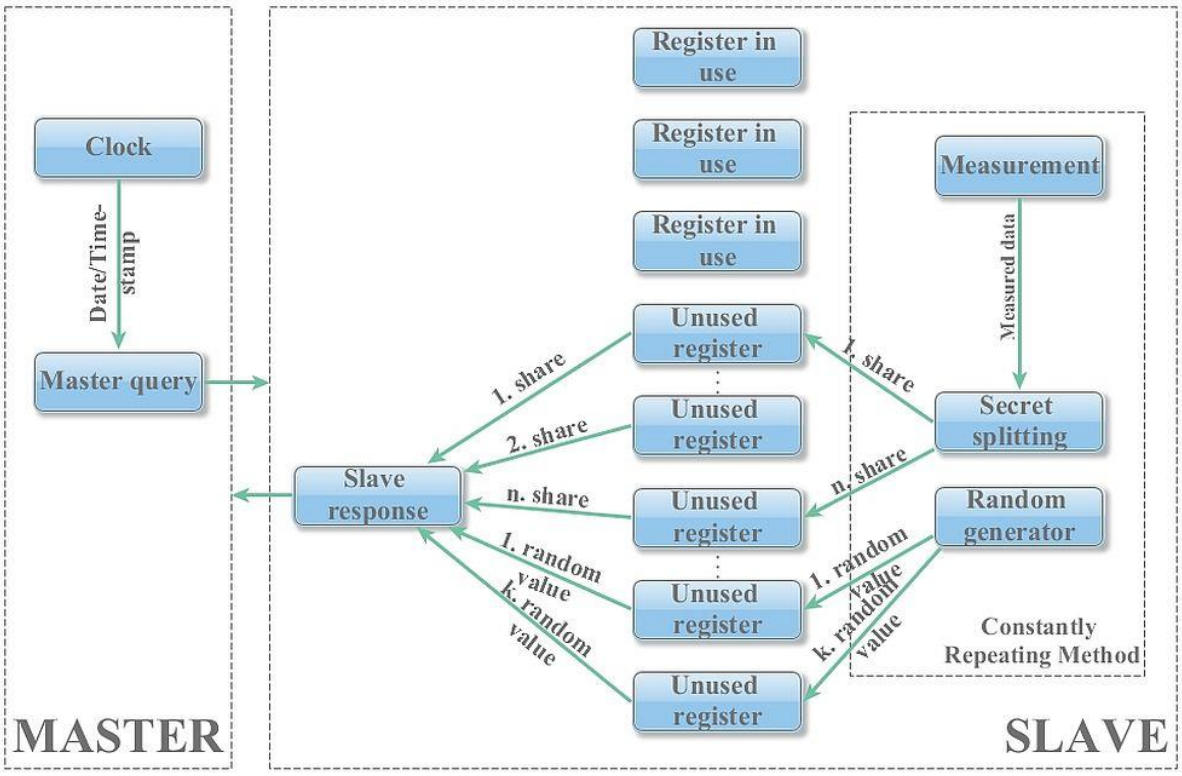

Fig. 6. Secure Modbus RTU Sensor slave side

TABLE II
NOMINATIONS 2.

Nomination	Explanation
$\xrightarrow{secure}$	Secure channel.
$\xrightarrow{public}$	Public channel.
SC	"Send challenge" function code.
SI	"Shares inside" function code.
FC	Function code.
TS	Time stamp.
$EDataRequest$	Encrypted data part of a request.
$DataReq$	Data part of a request.
$EDataResponse$	Encrypted data part of a response.
$DataRes$	Data part of a response.

B. Initialization

We recommend that the initial steps of the secure protocol be taken when the network is built or for latency systems at the time of regular revisions. First message exchange should be taken on a cryptographically secure channel. Both master and slave side require a random generator and a prime number generator. Generated prime and random numbers should be large enough, and all calculation of the protocol should be taken in a finite field. Through the initialization part we construct a random polynomial over a finite field $GF(p)$, where the following restrictions needed for the proper working of the protocol:

- Order of the field should at least be $NumSlaves^2 + NumSlaves + 1$.
- Generator prime of the finite field p should be large enough.
- The coefficients (a_i) of the random polynomial are randomly chosen of a uniform distribution over $[0, p)$
- The $y_i$values of the random polynomial are computed $mod\ p$ for distinct $x_i$values chosen from $[0, p)$
- The size of y_i values of the random polynomial should be equal size to p.

Secret key and challenge values generated in the initialization part will authenticate the slaves to the master, and vice versa, and these keys then planted in the devices so it is only accessible to the specific instruments. In this paper we assume that the AES engine is present in both the slave and the master devices, thus we recommend at least a 128 bit long secret key. [7] (In the example network described in section five we only generate a fairly strong secret key, and use this one for each device, due to limited time.) Initialization steps are as follows, nomination can be seen in Table I and II:

1: M and S_i synchronize the time
2: M generates a large prime number:
$p = prime()$
3: M generates a random private key:
$K_S = skey()$
4: M constructs a random polynomial over $GF(p)$:
$RandomP(x) = a_0 + a_1\, x + a_2\, x^2 \ldots + a_n x^n$
where: $n > NumSlaves$ and
$RandomP(0) = a_0 = K_S$
5: M selects $NumSlaves$ amount of points (x_i, y_i) on the polynomial, where:
$x_i \neq x_j$ for $i, j = 1 \ldots NumSlaves$
6: M sends unicast request consisting the shares to the slaves, and waits for challenges:
$M \xrightarrow{secure} S_i : (x_i, y_i)$
request: $[i, SC, (x_i, y_i), CRC]$
7: S_i receives the share:
$S_i \xleftarrow{secure} M : (x_i, y_i)$
8: S_i generates a challenge:
$C_i = prime()$
9: S_i sends the challenge to the master:
$S_i \xrightarrow{public} M : C_i \oplus y_i$
response: $[i, SC, C_i \oplus y_i\ , CRC]$
10: M receives the challenge:
$M \xleftarrow{public} S_i : C_i \oplus y_i$
11: M calculates C_i :
$C_i = y_i \oplus (C_i \oplus y_i)$
12: M sends unicast requests consisting of extra shares to the slaves:
$M \xrightarrow{public} S_i : (P_{i1} || P_{i2} || \ldots || P_{iNumSlaves}) \oplus C_i$
request: $[i, SI, (P_{i1} || P_{i2} || \ldots || P_{iNumSlaves}), CRC]$
$P_{ik} \neq P_{il}\ for\ any\ k \neq l$
$k, l = 1 \ldots NumSlaves^2$
13: S_i receives the extra shares:
$S_i \xleftarrow{public} M : (P_{i1} || P_{i2} || \ldots || P_{iNumSlaves}) \oplus C_i$
14: S_i reconstructs K_S from the n pieces of imprints it already has:
$K_S = LagrangeIP((x_i, y_i), P_{i1}, P_{i2}, \ldots P_{iNumSlaves})$
15: M creates an address table about the slavess:
$(Address_i, (x_i, y_i), C_i)$

C. Communication

A usual SCADA network contains two types of field devices if we consider the aim of the instruments, actuator and sensor type. Actuator in general means turning energy into motion, but in this paper term actuator is used in the meaning of an RTU/PLC which is able to modify a property of a SCADA system (e.g. setting the temperature or turning the lights off). Sensors are the same as commonly known, these are devices which, detects and measures some property of the system, input can be light, heat, motion, moisture, pressure, etc. We created two types of the communication method depending on the type of the field devices to use the benefits of unused registers in RTUs/PLCs, in case of sensors where the measured data is stored in the device, on the contrary of the actuators which never store any data.

Through both type of communication, it is recommended to encrypt only the data segment of the messages, to evade known plain text attacks.

1) Communication with actuator slaves

Communication between master and a slave requires synchronized time and a shared secret key, which is provided by the initialization part of the protocol. Now every slave knows the secret key of the master and the master knows a unique challenge value from every slave, these two make possible that all the participants are capable of authenticate each other.

Communication description can be seen below:

1: M initiates a request:
Encrypts the TS and data part of the request with K_S:
$EDataRequest = encrypt(K_S, DataReq||TS)$
2: M sends the request to the slave:
$M \xrightarrow{public} S_i : EDataRequest$
request: $[i, FC, EDataRequest, CRC]$
3: S_i receives the request:
$S_i \xleftarrow{public} M : EDataRequest$
4: S_i makes the checks:
5: ***if*** $i = Address_i$ ***then***
S_i decrypts the data part:
$DataReq = decrypt(K_S, EDataRequest||TS)$
if TS fresh enough ***then***
if $DataReq$ is correct ***then***
S_i performs the task
else
S_i goes back to initial mode
end if
else
S_i goes back to initial mode
end if
else
S_i goes back to initial mode
end if
6: S_i encrypts the data part of the response and the $(x_i \oplus y_i)$ with K_S:
$EDataResponse = encrypt(K_S, DataRes||(x_i \oplus y_i))$
7: S_i sends the response to the master:
$S_i \xrightarrow{public} M : EDataResponse$
response: $[i, FC, EDataResponse, CRC]$
8: M receives the response:
9: $M \xleftarrow{public} S_i : EDataResponse$
10: M decrypts the response:
$DataRes\ ||(x_i \oplus y_i) = decrypt(K_S, EDataResponse)$
11: M makes the checks:
if $x_i \oplus (x_i \oplus y_i) = y_i$ ***then***
if $DataRes$ is correct ***then***
M processes the response
else
M goes back to initial mode
end if
else
M goes back to initial mode
end if

2) Communication with sensor slaves

Communication method is basically the same between the sensor type devices and master, like between master and actuator type field instruments. Only the data part of the messages contains different information, till the method we presented in the previous section used a simple controlling or measured data as a data part this one uses a more complex data, because in order to extend the level of data integrity we increased the entropy of the system. Every other step is the same as above.

The sensor device on Figure is end node of a data acquisition network. In predefined intervals the slave measures, then saves the gathered data in some registers used in the communication, then requested by the master to send this data. In our solution, we modified the Modbus RTU protocol, so the slave, after recording the fresh measurements, fills up the unused registers with data generated from the measurements by functions of the secret key and the challenge values. The number and the position of the secret parts defined by the secret key and the challenges too. All of these 'imprints' of data is needed to recreate the original data – opposed to the traditional secret sharing method – because the creation of these imprints only makes the access more difficult to the secret. The more registers are needed to decrypt the secret, the more grows the entropy of the system, which in turn makes more difficult to an attacker to gather relevant data. Keep in mind that if the number of imprints are the same as the number of unused registers the decrypting of the encrypted data become trivial, but the message cannot be consisted only of these imprints, so after the secret sharing the free registers filled up with randomized data seemingly similar to the imprints. Nominations can be seen in Table III. Details can be seen on Figure 6 and below:

TABLE III
NOMINATIONS 3.

Nomination	Explanation
$MeasuredData$	Measured data by the slave.
$numberOfShares$	Number of secret shares.
$numShareReg_i$	Position of the ith register that contains share inside.
$numRandomReg_i$	Position of the ith register that contains random value inside.
$numOfUUReg$	Number of unused, empty registers.
$SRandomP(x)$	A random polynomial over $GF(p)$.
(xV_i)	Ith x value of $SRandomP(x)$polynomial.
cf_i	Ith coefficient of $SRandomP(x)$ polynomial.
$contentReg_i$	Content of the ith register.

1: S_i measures a data : $MesuredData$
2: S_i makes the following calculations:
$numberOfShares = K_S \bmod C_i$
3: S_i calculates the position of the registers to put shares inside:
for $m = 0$ ***to*** $m < numberOfShare$ ***do***
$numShareReg_m = (numberOfShares + m) \bmod (numOfUUReg + 1)$
$cf_m = random(\ )$
$xV_m = random(\ )$
end for
4: S_i calculates the position of the registers to put random values inside:
for $m = numberOfShares + 1$ ***to*** $m < numOfUUReg$ ***do***
$numRandomReg_m = (numberOfShares + m) \bmod (numOfUUReg + 1)$
end for
5: S_i constructs the random polynomial:
$SRandomP(xV) = MeasuredData + cf_1 xV + cf_2 xV^2 \ldots + cf_1 xV^{numberOfShares}$
6: S_i puts share into the right register
for $m = 0$ ***to*** $m < numOfShare$ ***do***
$contentReg_{numShareReg_m} = (xV_m||\ SRandomP(xV_m)) \oplus C_i||\ 00\ldots0$
end for
7: S_i puts random values into the right register
8: ***for*** $m = numOfShares + 1$ ***to*** $m < numOfUUReg$ ***do***
$contentReg_{numRandomReg_m} = (random(\ ) \oplus C_i|00 \ldots 0$
end for
9: S_i constructs the data part of the response:
$contentReg_1\ ||contentReg_2||contentReg_{numOfUUReg}$

V. ATTEMPT OF PRACTICAL IMPLEMENTATION

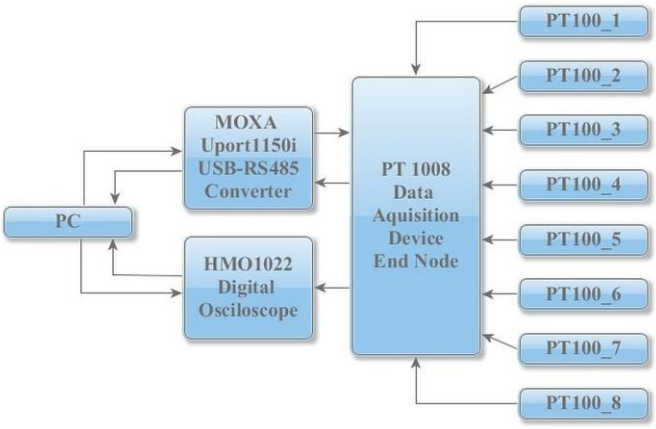

Fig. 7. Architecture of the test system

A. *Introducing the system and the method of measurement*

The first attempt of implementing the secure Modbus RTU protocol was on a device developed in the Department of Electrical Engineering and Mechatronics at the University of Debrecen. It is a data acquisition and control device which has several types of peripherals. The microcontroller, driving the device is an Atmel ATxmega 128a type. The firmware written by students of the department previously was made in Basic language. For the limited development time, the secure protocol was implemented in this Basic firmware. The data to be encrypted was supplied by eight PTC thermoresistors, because the thermic data is a frequent feature of SCADA communication networks. These thermic sensors measured the temperature of the MSc laboratory of the department, where the system was installed. The master of the network was a personal computer, with a simple Modbus client software running, and logging the gathered data to a comma separated text file. In this test, we only examined the applicability of the protocols on the slave's side, for in our opinion it is the most critical point in the protocols development to be implementable on a relatively low performance device (such as the used data acquisition device). The physical layer of the network was RS485 2-wire bus, the end node connected with the master via a Moxa Serial-to-USB converter device. Figure 7. shows the structure of the example network.

B. *Results of the practical application*

During our test, it has become clear that though the time needed for the firmware to complete one cycle (accounted to the larger entropy and encryption of the message) is grown by 500 %, the overall performance of the device is not impaired in any mean, all of its functionality remained the same, thus the protocol can be implemented in lower performance end devices:

- The time needed for the slave to run its program once (initialization not included, just the measurement and encryption) is grown to 500%.
- The time of the program cycle, in which the response is sent to the master is grown by 23%.
- Beside the above experiences, the devices performance is not impaired in any manner.
- The protocol does not affect the communication, over the testing period, the number of dropped messages was 1:15 000.

C. *State of art*

During our research we found out a few types of solutions to improve the security of the Modbus RTU protocol. There are papers which, provide only data integrity with the help of retransmitter devices, they are able to detect and correct errors. Other publication can detect the intrusion with a model-based system. [8,9] The most common solutions are the bump in the wire devices, many types are accessible on the market like Yasir (Yet Another SecurIty Retrofit), SEL-3021-2 (Serial Encrypting Transceiver) or AGA SCM (American Gas Association SCADA Cryptographic Module). These devices provide authentication, integrity, and confidentiality. [10,11,12]

CONCLUSION

Although our study is in no means near its end, the results thus are far more promising. The initial goal, to create a security measure, which can be applied over multiple types of industrial communication protocols, without modifying these protocols in any manner seems to be reachable in the near future. The initial tests conducted on typical SCADA end devices show that the proposed algorithm can be used on lower performance devices too, and we think that this protocol can work in systems that are based on protocols same as the Modbus RTU like Profinet and so on.

Over the next period of our study, we would like to extend our research to other communication protocols as well as to prove our solution right with methods like Applied-Pi calculus, or discover its vulnerabilities or flaws of our reasoning. In the meantime, we seek the opportunity to test our solution in a larger, industrial system, to eliminate its applicability's weaknesses.

REFERENCES

[1] Ádámkó, Éva., Jakabóczki, Gábor. „Security analysis of Modbus RTU." Proceedings of the Conference on Problem-based Learning in Engineering Education. 2015. 5-11.
[2] Byres, E. J., Franz, M., & Miller, D. (2004, December). The use of attack trees in assessing vulnerabilities in SCADA systems. In Proceedings of the international infrastructure survivability workshop.
[3] Nardone, R., Rodríguez, R. J., & Marrone, S. (2016, December). Formal security assessment of Modbus protocol. In Internet Technology and Secured Transactions (ICITST), 2016 11th International Conference for (pp. 142-147). IEEE.
[4] Chen, B., Pattanaik, N., Goulart, A., Butler-Purry, K. L., & Kundur, D. (2015, May). Implementing attacks for modbus/TCP protocol in a real-time cyber physical system test bed. In Communications Quality and Reliability (CQR), 2015 IEEE International Workshop Technical Committee on (pp. 1-6). IEEE.
[5] Huitsing, Peter, Chandia, Rodrigo, Papa, Mauricio & Shenoi, Sujeet (2008). Attack taxonomies for the Modbus protocols. International Journal of Critical Infrastructure Protection, 1, 37-44.
[6] Bruce, Schneier (1999). Attack trees. Dr Dobb's Journal, 24, .
[7] FIPS, PUB (2001). 197: Federal Information Processing Standards Publication 197. Announcing the ADVANCED ENCRYPTION STANDARD (AES)
[8] Yüksel, Ömer, Jerry den Hartog, and Sandro Etalle. "Reading between the fields: practical, effective intrusion detection for industrial control systems." Proceedings of the 31st Annual ACM Symposium on Applied Computing. ACM, 2016.

[9] Urrea, Claudio, Morales, Claudio & Mu\~noz, Rodrigo (2016). Design and implementation of an error detection and correction method compatible with MODBUS-RTU by means of systematic codes. Measurement, 91, 266-275.

[10] R. Solomakhin, Predictive YASIR: High Security with Lower Latency in Legacy SCADA, Technical Report TR2010-665, Department of Computer Science, Dartmouth College, Hanover, New Hampshire, 2010.

[11] Transceiver, Serial Encrypting. "SEL-3021 Serial Encrypting Transceiver

[12] Moore, Tyler, and Sujeet Shenoi, eds. Critical Infrastructure Protection IV: Fourth Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection, ICCIP 2010, Washington, DC, USA, March 15-17, 2010, Revised Selected Papers. Vol. 342. Springer Science & Business Media, 2010. Security Policy." (2005).

[13] Shamir, Adi (1979). How to share a secret. Communications of the ACM, 22, 612-613.

[14] Harn, Lien & Lin, Changlu (2010). Authenticated group key transfer protocol based on secret sharing. IEEE transactions on computers, 59, 842-846.

[15] Liu, Yining, Cheng, Chi, Gu, Tianlong, Jiang, Tao & Li, Xiangming (2016). A lightweight authenticated communication scheme for smart grid. IEEE Sensors Journal, 16, 836-842.

[16] Narayana, V Lakshman & Bharathi, CR (2017). IDENTITY BASED CRYPTOGRAPHY FOR MOBILE AD HOC NETWORKS. Journal of Theoretical and Applied Information Technology, 95, 1173.

[17] Goldenberg, Niv & Wool, Avishai (2013). Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. International Journal of Critical Infrastructure Protection, 6, 63-75.

[18] Urrea, Claudio, Morales, Claudio & Kern, John (2016). Implementation of error detection and correction in the Modbus-RTU serial protocol. International Journal of Critical Infrastructure Protection, 15, 27-37.

[19] Erez, Noam & Wool, Avishai (2015). Control variable classification, modeling and anomaly detection in Modbus/TCP SCADA systems. International Journal of Critical Infrastructure Protection, 10, 59-70.

[20] Shahzad, Aamir, Lee, Malrey, Lee, Young-Keun, Kim, Suntae, Xiong, Naixue, Choi, Jae-Young & Cho, Younghwa (2015). Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information. Symmetry, 7, 1176-1210.

[21] Fovino, Igor Nai, Carcano, Andrea, Masera, Marcelo & Trombetta, Alberto (2009). Design and Implementation of a Secure Modbus Protocol.. Critical Infrastructure Protection, 3, 83-96.

[22] Menezes, Alfred J, Van Oorschot, P & Vanstone, S (). Handbook of Applied Cryptography, C R CP ress, 1 996. Chapter, 5, 12.

[23] Modicon, I (1996). Modicon modbus protocol reference guide. North Andover, Massachusetts, , 28-29.

[24] Raiou, Costen (2016). Kaspersky Security Bulletin. Securelist, , 68-73.

[25] Communication network dependencies for ICS/SCADA Systems (2016). ,https://www.enisa.europa.eu/publications/ics-scada-dependencies

[26] Schneider Electrics, SCADA systems white paper (2012)

[27] Adrian Pauna, Konstantinos Moulinos, et.al. (2013). Can we learn from SCADA security incidents? https://www.enisa.europa.eu/publications/can-we-learn-from-scada-security-incidents.

[28] Karl Rauscher (2013). It’s Time to Write the Rules of Cyberwar. http://spectrum.ieee.org/telecom/security/its-time-to-write-the-rules-of-cyberwar.