Experimenting with the secure control of a robot over TCP/IP

Bogdan Groza, Toma-Leonida Dragomir
Politehnica University of Timisoara, Faculty of Automatics and Computers, Romania,
bogdan.groza@aut.upt.ro, toma.dragomir@aut.upt.ro

Abstract

An authentication protocol, based on cryptographic techniques, is implemented and used in the communication necessary for the control of a mobile robot over a public network. The robot is connected via an 802.11 wireless network to a local computer; however the control of the robot is done from a remote host over TCP/IP and in this way the information involved in the control scenario may be exposed to several security risks. The application fits in the context of a remote controlled system and the interest in using cryptographic techniques in this area has drastically increased in the last years. Instead of using standardized solutions, such as the SSL, we use as a new approach an authentication protocol based on one-way chains. The advantage of this approach is that only simple cryptographic primitives, such as hash functions and message authentication codes, are needed. Experimental results are presented, and the results show that it is feasible to use such a protocol since transfer rates and computational overhead are kept at the desired level for the control scenario. An analysis of the performance of the protocol based on the line utilization rate is done. Also, we give a partial solution for the treatment of communication delays.

1. Introduction

As pointed out by many recent papers the use of cryptography in the field of control systems is a major challenge, as these systems need to communicate over public networks where information is exposed to adversaries [5], [6]. The difficulty in using cryptographic techniques in control systems is twofold, first from the requirements over the equipments and second from the involvement in the dynamics and accuracy of the control system itself. Therefore, the first problem that must be solved is that the use of cryptography requires computational power or communication resources that may not be available. For this purpose different protocols were proposed, such as for example [20] which can be used to assure cryptographic security on the communication line between Supervisory Control and Data Aquisition (SCADA) equipments. As for the second kind of problems, the issue that must be solved is that communication over the public networks, or over any unreliable network, can introduce communication delays, or even uncertainties regarding the arrival of commands and responses. For this purpose several control techniques were developed that can deal with such kind of uncertainties, an example is in [13].

Our interest is the first type of concern, namely the development of efficient cryptographic protocols, which require low computational power. We avoid the use of standardized solutions, such as the TLS or SSL as we are not interested in an encrypted communication line to assure the confidentiality of the information and instead we are interested in assuring the authenticity of information. It is commonly acknowledged that in industrial control systems authenticity is much more important than confidentiality as information can not be used as long as there is no guarantee over its source and freshness. For this purpose we propose and use a class of authentication protocols based on one-way chains which significantly differs from the SSL paradigm. The merit of this approach is first as an experiment from which we can draw certain conclusions on the efficiency of such protocols. And second, the use of such a protocol does not require an asymmetric encryption function, as the SSL. Therefore this approach can be used where asymmetric encryption has to be avoided and only simple one-way functions are affordable.

This paper extends our previous result from [10]. In addition to [10] we will also make an analysis of the protocol performance based in line utilization rate. More, we will give a potential solution for the treatment of communication delays which are caused by the unreliability of the network.

The paper is organized as follows. In section 2 we describe the application setting, and in section 3 a solution for dealing with delays is outlined. The cryptographic protocol is presented in section 4. Implementation details are in section 5, while in section 6 we give some experimental results. Section 7 holds a performance analysis for the protocol while section 8 holds the conclusions of our paper.

2. Application setting

An X80 robot connected to a local computer via a WiFi 802.11 communication link is used. Several relevant technical details about the robot are resumed in what follows; the manufacturer website can be found at [21] for more details on this device.

The robot stands on two wheels with 18 cm diameter, each of them connected to a 12V DC-motor that can be controlled independently. The built-in commands allow three types of control for the two DC motors: open loop Pulse-Width Modulation (PWM), closed loop position control and closed loop velocity control. The regulators for the wheels are of proportional–integral–derivative type (PID), the values for the PID parameters, i.e. the k_P, k_I, k_D values, can be set by the use of built-in commands. We have used for the PID the values that are also used in the demo application given by the producer.

The robot is equipped with the following type of sensors: ultrasonic sensors, infrared range sensors, human detection sensors, temperature sensors. For our application we have used only the three ultrasonic sensors from the front of the robot. Also, the robot has a video camera which provides images at a resolution of 352x288 pixels; the producer indicates a rate of at most 4 fps for the webcam (in our application we acquired new images from the robot at a rate of 1 fps). The camera is attached to a mobile head which can be moved vertically and horizontally by a servo-motor. Other devices are attached to the robot, such as a microphone and a speaker; further details can be found in the technical documentation from [21].

As depicted in figure 1 the robot is connected to a local computer, which plays the role of the local controller, via a wireless router. This computer also plays the role of a server and accepts a connection from a remote host. The communication between the robot and the application from the local computer is done via a software gateway, provided with the X80 installation kit. The producer indicates that commands can be sent to the robot over this wireless link at rates exceeding 10 Hz. The robot has a web-interface which can be easily used to configure the robot. Since the wireless connection between the robot and the local computer supports WEP security, we are not interested in assuring the security on this side. What we are interested is to assure the security in the communication between a remote computer and the local computer to which the robot is connected. For this purpose we build a client application which we run on a notebook in order to connect over TCP/IP to the server and send commands to the robot. The client application plays the role of a remote controller. In such a scenario the use of cryptography is needed since packets between the client and the server travel over public networks and can be easily intercepted and modified by malicious adversaries. Details on the client and server applications are given in section 4.

In figure 2 a view over the application as a control system is presented. The main purpose of the application is to control the movements of the robot between some target points. The control is based on the information that is received from the environment via the above mentioned sensors. For the tests that were done in section 5 the control was done manually from the application interface by letting the robot to perform some basic movements, however any discrete control algorithm can be implemented as well. The objective of the paper was the development of the secure communication protocol and not of the control algorithm who can be further implemented in an abstract function from the source code.

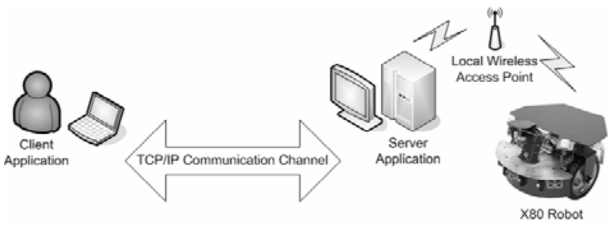

Figure 1. Application setting

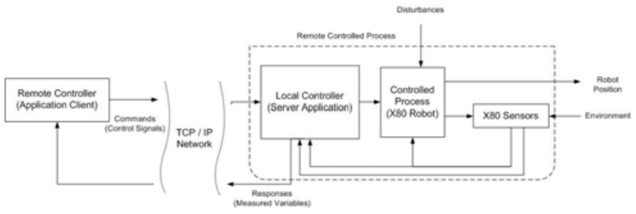
Disturbances
Remote Controlled Process
Remote Controller (Application Client)
Commands (Control Signals)
TCP / IP Network
Local Controller (Server Application)
Controlled Process (X80 Robot)
X80 Sensors
Robot Position
Environment
Responses (Measured Variables)

Figure 2. Application setting as a control system

3. Dealing with communication delays

An important issues in the proposed control scenario are the communication delays introduced by the communication over TCP/IP. Here we give a partial treatment on this issue.

The structure depicted in figure 3 corresponds to a remote control system where the delays of the commands and responses are depicted as $e^{-\tau s}$ and $e^{-\tau' s}$ respectively. These delays may be different, first because they can have different causes and second because the size of the command and responses may be different as well. These delays make the correspondence $u'(t) = u(t-\tau)$ and $y'(t) = y(t-\tau')$.

Let $\tau_1 = \min\{\tau, \tau'\}$ and $\tau_2 = \max\{\tau, \tau'\}$. We can observe that when the communication starts, for $t < \tau_1$ there is no communication between the two sides, while for $t \in [\tau_1, \tau_2]$ there is only a partial communication, either some commands, either some responses were received. Only for $t > \tau_2$ there is a complete full-duplex communication, both commands and responses are received, with the particular delays.

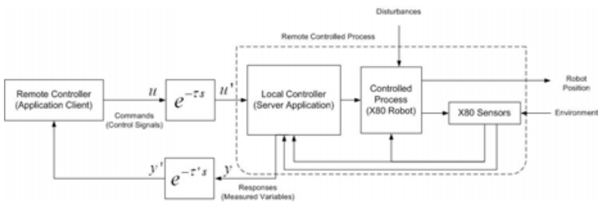
Disturbances
Remote Controlled Process
Remote Controller (Application Client)
u
$e^{-\tau s}$
u'
Commands (Control Signals)
Local Controller (Server Application)
Controlled Process (X80 Robot)
X80 Sensors
Robot Position
Environment
y'
$e^{-\tau' s}$
y
Responses (Measured Variables)

Figure 3. Application setting as a control system in the presence of delays

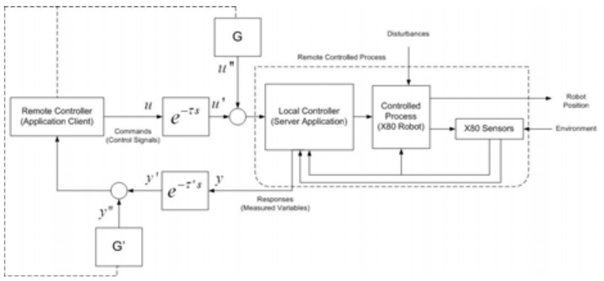
G
u''
Disturbances
Remote Controlled Process
Remote Controller (Application Client)
u
$e^{-\tau s}$
u'
Commands (Control Signals)
Local Controller (Server Application)
Controlled Process (X80 Robot)
X80 Sensors
Robot Position
Environment
y'
$e^{-\tau' s}$
y
Responses (Measured Variables)
y''
G'

Figure 4. Application setting as a control system with signal generators for dealing with delays

In order to deal with the inactivity periods $[0, \tau_1]$ and $[\tau_1, \tau_2]$ we can use the structure depicted in figure 4 in which *G* and *G'* are two auxiliary generators to compensate delays. In this case u'' and respectively y'' will be the commands and responses in the case when there is no signal. The *G* and *G'* generators may have multiple purposes: to initialize the remote controller in a predictive state for the process, to set the controlled process in a particular point, to predict the state of the controller etc. The dotted line in figure 4 denotes that the generators and the remote controller must be synchronized in some way and subsequently they have to work synchronously.

This treatment solves only the problems related to initialization, further, different delays can appear as well on the communication channel and the same generators can be used to take action in the case when there are some time-outs. In the absence of control signals, commonly used values for this purpose in the G generator are "zero control" and "last available command".

4. The cryptographic protocol

One-way chain based authentication protocols were initially proposed by Lamport [14] in order to authenticate a user to a remote system while avoiding the weaknesses of password based authentication. However, the practical use of Lamport's scheme in the S-Key system proposed in [12] resulted in an insecure system which has several weaknesses [15]. Later, one-way chains were used to assure the authenticity of information that is broadcasted to large number of receivers by using elements of one-way chains as keys for Message Authentication Codes (MAC) [17], [16], [1]. The solution proposed by Perrig et al. [17] has the great merit that MAC codes can be used for sending information to multiple receivers although the same authentication key is used. This is due to the use of time synchronization since otherwise MAC codes require a distinct secret shared key between the sender and each receiver which leads to an inefficient protocol due to the large number of keys. The same could be achieved by the use of digital signatures; however digital signatures can be from hundreds to thousand times more computational intensive then a MAC. Therefore, due to its computational efficiency, this protocol was also used in constrained environments with low computational power and communication abilities such as wireless sensor networks [16]. Also an analogous solution was proposed in [1] which avoids the use of time synchronization by requiring a response from the client. As pointed out in [11] this solution can be relevant in the context of a control system, due to the nature of such a scenario which is essentially based on a feed-back between the controller and the controlled process. In what follows we will study the practical implementation of such a protocol.

More motivation on the use of this class of protocols may be useful. The most important thing is that at the core of such protocols only a simple one-way function can be used; in our case a hash function. This is significantly different from the SSL paradigm, which uses the hybrid encryption paradigm (the use of a public key to encrypt a secret key, that is later used for the encryption of the messages) and requires the use of an asymmetric encryption mechanism. Therefore the proposed protocol may be used in the absence of such an encryption mechanism. Also, as pointed out by the TESLA protocol [17], simple MAC codes can be used to authenticate information for a large number of receivers by using the same key; therefore a solution based on one-way chains is largely scalable. However, in this paper we will not use a protocol based on time-synchronization, and all that we use is a protocol based on challenge response. This is first because we do not need a large number of receivers, and all that we need is a one-to-one communication, and second because it is expected that the time-synchronization based protocol while largely scalable, will have a fixed send-receive rate. In contrast, the challenge-response based protocol will give flexible rates and an increase in performance. Therefore, we leave the implementation of the protocol based on time synchronization as future work.

The structure of the communication sessions for the protocol is as follows:

Session i

$A \rightarrow B:\ c_i, MAC_{k_{A,i+1}}(c_i), k_{A,i}$

$B \rightarrow A: r_i, MAC_{k_{B,i+1}}(r_i), k_{B,i}$

Here *A* and *B* are the communication participants. *A* plays the role of the controller, in our application it is also the role of the client which can command the robot remotely, while *B* plays the role of the controlled process, which in our application is also the server to which the robot is connected. The messages exchanged are $c_{A,i}$, $r_{B,i}$ which represent the command and the response respectively, and *MAC* is a message authentication code. The keys for each entity are denoted by $k_{entity,i}$, here $entity \in \{A,B\}$, and is computed as $k_{entity,i} = Hash^{n-i}(k_{entity,0})$ where $k_{entity,0}$ is some secret random value generated by each entity, *Hash* is a hash function and *n* is the number of communication sessions which must be chosen in advance, details on the protocol can be found in [9], [11]. It is easy to observe that the keys form a hash chain. For the efficient computation of such a chain several optimization techniques were proposed [4], [7], [18]. However, we did not use them in our application as they are intended for constrained environments and the computation and storage of the entire chain was not a problem on the computers that we used.

We note that the only attack that an adversary can launch on this protocol is to delay packets, for this purpose the server application will halt the robot if no authentic packet is received after a delay of 1 second. Even when new commands are not sent from the controller to the controlled process, the application still communicates over the previously described protocol, by sending blank command packets; this is needed also to update the information that is received from robot sensors on the remote controller's side.

We now proceed by giving details on the commands and responses structure from our application. The command message has the following structure: the first byte indicates the command code; each command has a unique identification number which corresponds to the number of the built-in command from the documentation of the robot [21]. A second byte follows which gives the response code; we used only the value 128, which indicates in our application that a response as described in what follows is needed. Another 14 bytes are appended which represent the values for *cmd1, cmd2, cmd3, cmd4, cmd5, cmd6, time* - this follows the general structure for a command that is sent to the robot according to the documentation of the robot.

As for the responses from the server which hosts the robot, each response packet includes the following information: the values of the 3 sonar sensors and the 2 encoders from the wheels, each of these values has 2 bytes, and the value of the last image acquired from the camera on the robot head, which consists in 76086 bytes. A first byte in the response message indicates the type of the response, this byte corresponds to the response requested in the command, and for the moment only responses with this structure were used. A second byte is reserved for future use, just for symmetry with the structure of the command message. This leads to a size of 76092 bytes for each response value.

The following is the detailed structure of the messages from the authentication protocol:

Session i

$A \rightarrow B:$

$c_i = (cmdcode_i, cmd_{1-6}, time),\ MAC_{k_{A,i+1}}(c_i), k_{A,i}$

$B \rightarrow A:$

$r_i = (sonar_{1-3}, encoder_{1-2}, image)\ MAC_{k_{B,i+1}}(r_i), k_{B,i}$

It is easy to observe that the protocol introduces a delay of 1 session for the authentication, this means that the values received in session i can be checked for authenticity only in session $i+1$, when the corresponding key of the MAC is received. This disadvantage must be accepted since this is the only way to avoid the use o a secret key between the two participants. The alternative solution, which uses time synchronization as in [16], [17], introduces as well an authentication delay equal to the key disclosure period.

It should be also stated, that such a protocol requires an initialization stage in which the values of $k_{A,0}$ and $k_{B,0}$ are securely exchanged between the two entities – these values are not confidential, however each entity must be ensured that the values originate from the respective communication participant and that they are new.

Basically, any key exchange protocol can be used for this purpose; in particular we have used a digital signature. We underline that indeed this signature is a public key primitive; however digital signatures can also be computed by using symmetric functions. Digital signatures that are built on symmetric primitives are also called one-time signatures, their use in practice is limited mostly because they do not offer the same flexibility as number theoretic based signatures such as RSA or DSA. Still, one can implement a digital signature based on simple one-way functions, therefore we underline that this protocol can be based entirely on one-way functions. Also as future work we intend to use such signatures on some low computational power microcontrollers for the implementation of a similar protocol.

5. Implementation details

A client-server application was developed. The client application can be used to connect over TCP/IP to the server application hosted on the same computer to which the robot is linked via the application gateway offered by the producer. The client is able to command the robot remotely, by sending commands (the basic movements are implemented: forward, backward and turns to left or right). Also, the images that are collected from the robot are sent to the client. We underline that all this information is send with the authentication protocol described in section 3. Therefore al information is authentic and packets cannot be corrupted in transit by adversaries. The control flow used in the application is depicted in figure 5. In order to prevent time-outs, a Redundant Command Generator and a Redundant Response Generator are used to issue a command or a response.

We choose C# as the environment to implement our application. The robot SDK available from the producer was intended to be used in VC++ or Visual Basic 6. We avoid the use of VB 6.0 since it is out of date and also we avoided the use of VC++ since it leads to more work in the implementation. Instead, we choose to implement the application in C#.

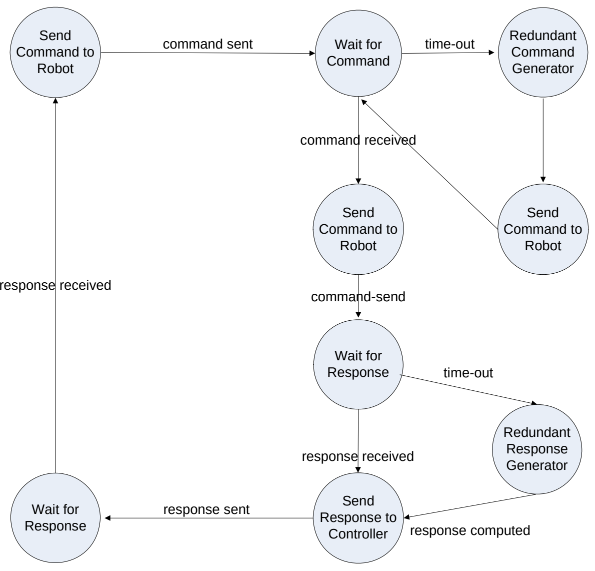

Figure 5. Control flow for the protocol

Using the ActiveX control offered by the producer in C# is fairly easy, however the control crashed several times when sensor readings are done, therefore the use of try/catch structures was needed. Rather late we found that there is a different software package that can be used to communicate with the X80 robot hosted at [22]. This seems to give better results than the one from the producer and although we didn't use it here we plan to use it in some forthcoming applications for potential improvements.

As for the cryptographic primitives involved, we have used all the hash functions and message authentication codes available in .NET in order to achieve comparative results: RIPEMD, MD5, SHA1, SHA256, SHA384, SHA512. The following classes were used: *MD5CryptoServiceProvider*, *RIPEMD160Managed*, *SHA1Managed*, *SHA256Managed*, *SHA384Managed*, *SHA512Managed*, *HMACMD5*, *HMACRIPEMD160*, *HMACSHA1*, *HMACSHA256*, *HMACSHA384*, *HMACSHA512*.

We note that the computation of SHA1 with the use of *SHA1CryptoServiceProvider* is significantly slower than for *SHA1Managed*. Experimental results regarding the computational performance of these primitives can be found in the following section. The communication was implemented over the standard TCP sockets available in the *System.Net.Sockets* namespace.

6. Experimental results

Some experimental results are mandatory in establishing the communication and computational performance of the protocol. First, some results on the cryptographic primitives involved are needed. The results from tables 1 and 2 show the computational time, expressed in seconds, for hash functions and message authentication codes. The computational time is estimated by computing the function for 10^6 times and then computing the arithmetic mean (in every iteration the new input of the function is the previous output).

Hash Function	CPU Intel T2300@1.66Ghz	CPU Intel E6750@2.66Ghz
MD5	$9.37\times10^{-6}s$	$5.15\times10^{-6}s$
RIPEMD160	$2.81\times10^{-6}s$	$1.56\times10^{-6}s$
SHA1	$2.03\times10^{-6}s$	$1.40\times10^{-6}s$
SHA-256	$3.28\times10^{-6}s$	$1.87\times10^{-6}s$
SHA-384	$9.53\times10^{-6}s$	$4.21\times10^{-6}s$
SHA-512	$9.68\times10^{-6}s$	$4.37\times10^{-6}s$

Table 1. Computational time for some hash functions in .NET

H-MAC	Intel T2300@1.66Ghz	Intel E6750@2.66Ghz
MD5	$21.25\times10^{-6}s$	$11.56\times10^{-6}s$
RIPEMD160	$9.68\times10^{-6}s$	$5.15\times10^{-6}s$
SHA1	$22.18\times10^{-6}s$	$11.87\times10^{-6}s$
SHA-256	$10.78\times10^{-6}s$	$5.78\times10^{-6}s$
SHA-384	$35.78\times10^{-6}s$	$15.93\times10^{-6}s$
SHA-512	$35.93\times10^{-6}s$	$16.09\times10^{-6}s$

Table 2. Computational time for some MAC Codes in .NET

For the experimental results regarding the protocol the same hash function that is used for the computation of the session keys, i.e. the one-way chain, was also used for the computation of the HMAC. However, the application is flexible and allows the use of distinct functions for the computation of the key chain and the MAC.

In [6], [8] some terminology for evaluating the performance of communication over Internet for industrial systems is explained. These definitions, adopted by NIST (National Institute of Standards and Technology) and ODVA (Open DeviceNet Vendor Association), originate from [2], [3]. We will measure the communication performance by using the Round Trip Time (RTT), which is the time necessary to compute a command by the controller, send it to the controlled process and receive the desired response. In our application this implies the execution of the 10 steps that are suggested in figure 6. The use of RTT for measuring the performance of the protocol is needed as other metrics such as the response latency or the action latency from [8] will not be enough relevant for the efficiency of the performance of the protocol. In table 3 the average number of packets per second is given and also the average closed loop latency resulted from the previous value (the values are taken for the first 1 minute of run).

Hash Function for keys and MAC	Output Length (in bits)	Packets/Second (Average Value)	Round Trip Time
MD5	*128*	*64*	*0.016 s*
RIPEMD160	*160*	*56*	*0.017 s*
SHA1	*160*	*61*	*0.016 s*
SHA-256	*256*	*56*	*0.017 s*
SHA-384	*384*	*52*	*0.019 s*
SHA-512	*512*	*50*	*0.020 s*

Table 3. Communication statistics for different hash functions and MAC codes

For example in the worst case the closed loop latency is at 0.02 seconds, this is for the SHA512 cryptographic function. We also note that the minimum and maximum number of packets sent over each second can vary a lot, and therefore we considered just the average values for the entire run-time. These results were achieved in a LAN, but the application can be tested as well on any other network that supports TCP/IP communication.

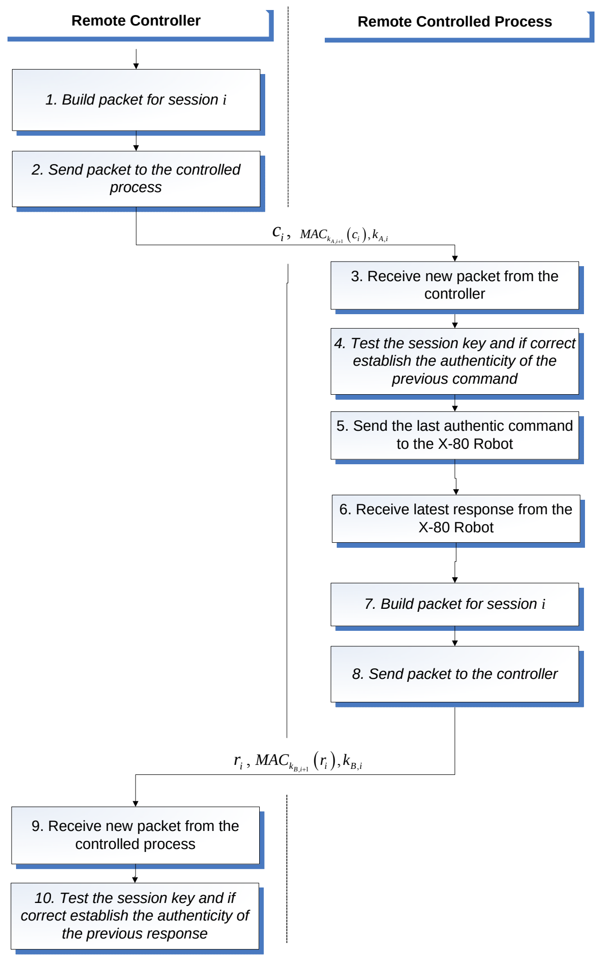
Remote Controller
Remote Controlled Process
1. Build packet for session i
2. Send packet to the controlled process
c_i, $MAC_{k_{A,i+1}}(c_i), k_{A,i}$
3. Receive new packet from the controller
4. Test the session key and if correct establish the authenticity of the previous command
5. Send the last authentic command to the X-80 Robot
6. Receive latest response from the X-80 Robot
7. Build packet for session i
8. Send packet to the controller
r_i, $MAC_{k_{B,i+1}}(r_i), k_{B,i}$
9. Receive new packet from the controlled process
10. Test the session key and if correct establish the authenticity of the previous response

Figure 6. Flowchart of the steps involved in one round trip (for session *i*)

The results from table 3, point out that it is the size of the hash functions and MAC that influences the communication performance. It is easy to observe that in table 1 the computational time for the SHA-256 function and the corresponding MAC is lower than for MD5 while in table 3 the best communication performance was achieved with the MD5 function due to its reduced output size. Therefore a reduced size for the output of the hash function is preferable, however MD5 is known for several weaknesses [19], and it is unlikely that in the future it will give a sufficient level of security. However, even for the use of the SHA-512 which gives the largest output, we still get an average value of 50 packets per second which is much more than the speed of the robot (for example the robot can get at most 4fps while we are sending an average of 50 fps). This finally shows that using cryptographic security is feasible for applications.

7. Performance analysis

A performance analysis for the protocol, based on the line utilization ratio, is useful in order to establish the performance of the protocol. First, the time required for one communication session, which consists in sending a command and receiving a response, is the following:

$$T_{session} = t_{ver}^{r} + t_{comp}^{c} + t_{send}^{c} + t_{prop}^{c} + t_{ver}^{c} + t_{comp}^{r} + t_{send}^{r} + t_{prop}^{r} \quad (1)$$

Here, t_{ver}^{r}, t_{ver}^{c} is the time to verify the response and the command, $t_{comp}^{c}, t_{comp}^{r}$ is the time to compute the command and the response, $t_{send}^{c}, t_{send}^{r}$ is the time to send the command and the response, while $t_{prop}^{c}, t_{prop}^{r}$ is the propagation time for the command and the response. As the protocol is symmetric on both sides, and time values in each pair are close, fortunately we can do some simplifications: $t_{ver} \simeq t_{ver}^{r} \simeq t_{ver}^{c}$, $t_{prop} \simeq t_{prop}^{c} \simeq t_{prop}^{r}$, $t_{comp} \simeq t_{comp}^{c} \simeq t_{comp}^{r}$, $t_{send} = t_{send}^{c} \approx t_{send}^{r}$. More, as the computation and the verification of a MAC code require almost the same amount of time we have: $t_{comp-ver} \simeq t_{ver}^{r} \simeq t_{ver}^{c} \simeq t_{comp}^{c} \simeq t_{comp}^{r}$. Therefore (1) leads to:

$$T_{session} = 4 \cdot t_{comp-ver} + 2 \cdot t_{send} + 2 \cdot t_{prop} \quad (2)$$

Now, from this entire time interval $T_{session}$ the line is utilized only for the fraction of time when the command and responses are actually sent, i.e.:

$$T_{send} = t_{send}^{c} + t_{send}^{r} \quad (3)$$

By using the same approximations as previously we get:

$$T_{send} \approx 2 \cdot t_{send} \quad (4)$$

Now the line utilization can be easily defined by using (2) and (4) as:

$$U = \frac{T_{send}}{T_{session}} = \frac{2 \cdot t_{send}}{4 \cdot t_{comp-ver} + 2 \cdot t_{send} + 2 \cdot t_{prop}}$$

$$= \frac{t_{send}}{2 \cdot t_{comp-ver} + t_{send} + t_{prop}} \quad (5)$$

By reducing with t_{send} we further get:

$$U = \frac{1}{1 + \frac{t_{prop}}{t_{send}} + \frac{2 \cdot t_{comp-ver}}{t_{send}}} \quad (6)$$

Practical examples with particular data can be useful; therefore we proceed with an analysis for three types of networks: Wide Area Network (WAN), Local Area Network (LAN) and Wireless Area Network (WLAN):

- Assume for the WAN a link distance of 10 km and the speed over optical fiber close to the speed of light $3 \times 10^8 m/s$. We get that propagation time is $t_{prop} = \frac{10 \times 10^3}{3 \times 10^8} s = 0.33 \times 10^{-4} s$. At data rates of 1 Gbps and for a packet of 32 Kb (we consider this as the average size for the command and response packets) we have $t_{send} = \frac{32 \times 10^3}{1 \times 10^9} s = 32 \times 10^{-6} s$. Now $\frac{t_{prop}}{t_{send}} \simeq 0.01 \times 10^2$, let $t_{comp-ver} = 10 \times 10^{-6} s$ and we have $\frac{2 \cdot t_{comp-ver}}{t_{send}} = 0.625$. It follows from (7) that $U \simeq 0.38$ and the line is only 38% of the time utilized for the actual communication, the rest of the time the line is free.
- Assume for the LAN a link distance of 100 m and the speed over copper media as $2 \times 10^8 m/s$. We get that propagation time is $t_{prop} = \frac{100}{2 \times 10^8} s = 0.5 \times 10^{-6} s$. At data rates of 100 Mbps and a packet of 32 Kb we have $t_{send} = \frac{32 \times 10^3}{100 \times 10^6} s = 0.32 \times 10^{-3} s$. Now $\frac{t_{prop}}{t_{send}} \simeq 1.56 \times 10^{-3}$, let $t_{comp-ver} = 10 \times 10^{-6} s$ and we have $\frac{2 \cdot t_{comp-ver}}{t_{send}} = 62.5 \times 10^{-3}$ and therefore it follows from (7) that $U \simeq 0.94$.
- Assume for the WLAN a link distance of 30 m for the and the speed close to the speed of light $3 \times 10^8 m/s$. We get that propagation time is $t_{prop} = \frac{30}{3 \times 10^8} s = 10^{-7} s$. At data rates of 10 Mbps and for a packet of 32 Kb (we consider this as the average size for the command and response packets) we have $t_{send} = \frac{32 \times 10^3}{10 \times 10^6} s = 3.2 \times 10^{-3} s$. Now $\frac{t_{prop}}{t_{send}} \simeq 31 \times 10^{-6}$, let $t_{comp-ver} = 10 \times 10^{-6} s$ then we have $\frac{2 \cdot t_{comp-ver}}{t_{send}} = 6.25 \times 10^{-3}$. Therefore it follows from (7) that $U \simeq 0.99$ and the line is 99% percents of time utilized.

Let us now assume that in general the time to compute a cryptographic function takes the average value of $10 \times 10^{-6} s$, this assumption is correct with respect to tables 1 and 2. As seen from the examples above the propagation time varies between $10^{-7} s$ to $0.33 \times 10^{-4} s$ while the data rate varies between 10 Mbps for a wireless network to 1 Gb for optical fiber which means that for a packet of 32 Kbits the transmission time is between $3.2 \times 10^{-3} s$ and $32 \times 10^{-6} s$. Let us consider that the propagation time $t_{prop} \in \left[10^{-7}, 0.33 \times 10^{-4}\right]$ and the transmission time $t_{send} \in \left[32 \times 10^{-6}, 3.2 \times 10^{-3}\right]$, the plot in figure 3 illustrates the variation of the line utilization under these variations.

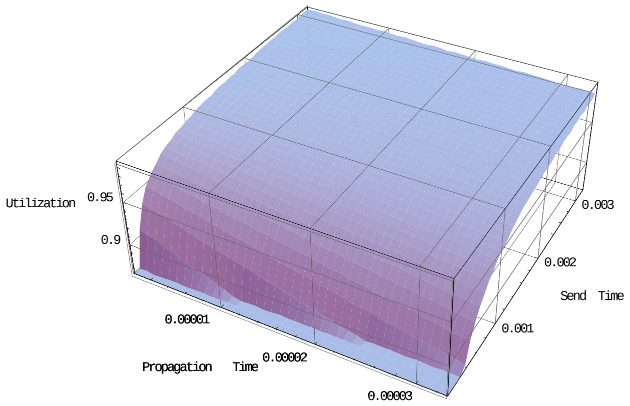

Figure 7. Variation of line utilization

The plot from figure 7 let us see that line utilization is low in the case when propagation time is high and send time is low, usually this happens on long communication lines. We can state the following two conclusions for the case when the line utilization is low. First, since the line is not utilized other processes can communicate over the same line, thus the protocol does not consume the entire bandwidth of the network. Second, larger packets can be sent on the line without significantly reducing the packet rates.

8. Conclusions

The use of cryptographic security in industrial control systems is an obvious demand. In this paper an authentication protocol based on cryptographic techniques for a remote controlled system was proposed and implemented. The experimental results from our application show that implementing cryptography is feasible, and leads to satisfactory transfer rates for the addressed scenario.

The main objective of this paper was to establish the influence of the computational time and communication overhead induced by the use of cryptography on the speed of the commands and responses sent between the remote controller and the remote controlled process. As a conclusion on this, we remark that the computational time is not a problem on currently used computers and the communication overhead induced by the use of cryptography is in the order of several hundred bits per packet. As the simplest message authentication code requires at least 128 bits, and over long term, to increase security level, it is likely that 256 bits will be needed, we believe that such an overhead must be accepted. At least, for our scenario the use of message authentication codes of even 512 is acceptable.

The performance analysis from section 7 lets us conclude that over long distances with large propagation delays the line is not heavily utilized and other processes can communicate over the same line as well. Also, larger packets can be sent on the line without significantly reducing the packet rates.

Since the X-80 robot is a slow process, where time constraints are not a great issue, as future work the use of such an authentication protocol in a more restrictive environment, where time constraints are a serious issue, may be more interesting to address. The proposed protocol is generic and therefore it can be used in other control scenarios as well without major modifications.

Acknowledgements: This work was partially supported by national research grant PNCDI PN II - 940/2008.

9. References

[1] F. Bergadano, D. Cavagnino, B. Crispo, “Individual Authentication in Multiparty Communications”. Computer & Security, Elsevier Science, vol. 21 n. 8, 2002, pp.719-735.

[2] S. Bradner, “Benchmarking Terminology for Network Interconnection Devices”, RFC 1242, 1991.

[3] S. Bradner, J. McQuaid, “Benchmarking Methodology for Network In-terconnection Devices”, RFC 2544, 1999.

[4] D. Coppersmith and M. Jakobsson, “Almost Optimal Hash Sequence Traversal”, Proceedings of the Fifth International Conference on Finan-cial Cryptography, pp. 102-119, 2002.

[5] D. Dzung, M. Naedele, T.P. Hoff, M. Crevatin, “Security for Industrial Communication Systems”, Proceedings of the IEEE, vol. 93, no. 6, 2005.

[6] J. Falco, J. Gilsinn, K. Stouffer, “IT Security for Industrial Control Sys-tems: Requirements Specification and Performance Testing”, NDIA Homeland Security Symposium & Exhibition, 2004.

[7] M. Fischlin, “Fast Verification of Hash Chains”, The Cryptographers Track at the RSA Conference, pp. 339-352, 2004.

[8] J. Gilsinn, “Real-Time I/O Performance Metrics and Tests for Industrial Ethernet”, ISA Automation West, 2004.

[9] B. Groza, “Using one-way chains to provide message authentication without shared secrets”, Second International Workshop on Security, Pri-vacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2006), IEEE, 2006.

[10] Groza B., Dragomir T.L., Using a Cryptographic Authentication Protocol for the Secure Control of a Robot over TCP/IP, IEEE-TTTC International Conference on Automation, Quality & Testing, Robotics, AQTR 2008 (THETA 16).

[11] B. Groza, T.-L. Dragomir, "On the use of one-way chain based authenti-cation in secure control systems", Second International Conference on Availability, Reliability and Security, pp. 1214-1221, IEEE Comp. Soc., 2007.

[12] N. Haller, C. Metz, P. Nesser, M. Straw, “A One-Time Password Sys-tem”, RFC 2289, Bellcore, Kaman Sciences Corporation, Nesser and Nesser Consulting, 1998.

[13] O. C. Imer, S. Yuksel, T. Basar, “Optimal control of lti systems over unreliable communication links”, Automatica, (42), 2006.

[14] L. Lamport, “Password Authentication with Insecure Communication”, Communication of the ACM, 24, 770-772, 1981.

[15] C.J. Mitchell and L. Chen, “Comments on the S/KEY User Authentica-tion Scheme”, ACM Operating Systems Review, pp. 12-16, 1996.

[16] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, “SPINS: Security Protocols for Sensor Network”, Proceedings of Seventh Annual Interna-tional Conference on Mobile Computing and Networks MOBICOM, 2001.

[17] A. Perrig, R. Canetti, J. D. Tygar, D. Song, “The TESLA Broadcast Au-thentication Protocol”, In CryptoBytes, 5:2, Summer/Fall, pp. 2-13, 2002.

[18] Y. Sella, “On the Computation-Storage Trade-offs of -Hash Chain Tra-versal”, Proceedings of the Seventh International Conference on Finan-cial Cryptography, pp. 270-285, 2003.

[19] Xiaoyun Wang, Hongbo Yu, “How to Break MD5 and Other Hash Func-tions”, Advances in Cryptology - EUROCRYPT 2005, 24th Annual In-ternational Conference on the Theory and Applications of Cryptographic Techniques, pp. 19-35, 2005.

[20] A. K. Wright, J. A. Kinast, J. McCarty, “Low-Latency Cryptographic Protection for SCADA Communications”, Applied Cryptography and Network Security, Second International Conference, ACNS, pp. 263-277, 2004.

[21] Dr. Robot Inc., Developer and manufacturer of mobile robotics technol-ogy, http://www.drrobot.com/.

[22] X80 WiRobot, page maintained by T. Taylor http://sky.fit.qut.edu.au/%7Etaylort2/X80/.