Cryptographic authentication on the communication from an 8051 based development board over UDP

Bogdan Groza [1] , Pal-Stefan Murvay [2] , Ioan Silea [1] , Tiberiu Ionica [1]
Politehnica University of Timisoara, Faculty of Automatics and Computers, Romania [1]
Continental Corporation, Infotainment and Interpersonal Department, Romania [2]
bogdan.groza, ioan.silea, tiberiu.ionica @aut.upt.ro
stefan.murvay @continental-corporation.com

Abstract

Implementing cryptography on devices with low computational power is a necessity as they became involved in communications over public networks. Even more, these devices became ubiquitous and are used in a large area of applications, from home-office systems to industrial control systems. We deal with the design and implementation of a cryptographic protocol that can be used to assure the authenticity of the information broadcasted over UDP from an 8051 based system-on-a-chip to a large number of receivers. The protocol that we use is similar to the well known TESLA protocol that was used in sensor networks, and by using such a protocol information can be broadcasted to a large number of receivers without requiring secret shared keys or expensive public key operations. Some implementation details and experimental results are presented as well, and they show that implementing a cryptographic authentication protocol is feasible even in a constrained environment as offered by the 8051 microcontroller.

1. Introduction

System-on-a-chip (SoC) are ubiquitous devices that are used in a large variety of applications from home-office to industrial systems. The main constraint, with respect to cryptography, is the low computational power that these devices have. Still, these devices may be involved in tasks which put precious information at a risk. The most obvious scenario is the case when information travels over public networks such as the Internet. In such a situation, information is exposed, and may be subject to forgeries produced by some adversaries.

In order to eliminate such a risk, the only solution is the use of cryptography. Implementing cryptography, as already stated, raises problems from the computational point of view, since it is somewhat unsuitable to implement many cryptographic primitives such as for example public key primitives. Also, implementing a cryptographic protocol on a SoC is not as trivial as implementing on a high level language where the programmer has access to a large variety of classes that can be used for cryptographic functions and communication purposes. Here only a reduced library with a limited set of functions is available.

The 8051F124 development board that we work on is a device that fits well in the aforementioned category. From the security point of view, in the scenario that we deal with, we are first interested in assuring the authenticity of information. This means to give a guarantee that information originates from a particular entity and was not altered during the transmission by potential adversaries. Also authenticity refers to assuring the freshness of information since replayed information can not be considered authentic as the original sender may not be present at the given time. In order to assure the authenticity of some information the use of Message Authentication Codes (MAC) or Digital Signatures is required. However, in a constrained environment the use of digital signatures is not possible as they are public key primitives which require more computational power; in fact they can be from hundred to thousands times more computational intensive than MAC codes. Therefore the use of MAC codes is the only alternative. Unfortunately, this alternative comes with a limitation as well, and this limitation is the fact that MAC codes require a secret shared key between the sender and each receiver. For example, a scenario with n receivers will require n distinct shared keys, and more, a distinct MAC code must be computed by the sender for each receiver – this obviously leads to an inefficient protocol. The limitations go even further since such a solution is not

easily scalable as every new receiver that joins the stage must share a new key with the sender and this complicates things further as a key exchange protocol must be run between the sender and each receiver. After all, it is obvious that the use of MAC codes with secret shared keys is an inefficient solution for our purpose. Fortunately, a good solution exists. The solution consists in the use of elements from a one-way chain, i.e. an array generated by the successive composition of a one-way function, as keys for MAC codes and time synchronization – solution proposed by Perrig et al. [11]. The first proposals for such a protocol can be found in [1], [11], [12]. Such a solution feats well to the scenario that we address, therefore we will use a protocol based on one-way chains and time synchronization to assure the authenticity of the information that is broadcasted from our low computational power device.

The paper is organized as follows. In section 2 the application setting is presented and several details about the board are given, this description is necessary in order to understand the constraints of our environment. In section 3 the cryptographic protocol is explained. Section 4 holds some implementation details and experimental results. In section 5 are the conclusions of our paper.

2. The addressed scenario

2.1. Application setting

A brief description of the scenario that we address follows, the application setting is suggested in figure 1. We are interested in a scenario where information from a low computational power device must be broadcasted to a large number of receivers. More particular, we used an 8051 based development board, which broadcasts the temperature acquired from a local sensor to a large number of receivers. The board has a temperature sensor embedded and an extension board for Ethernet communication. Such a scenario is commonly present in industrial control systems where measurements from processes are taken and sent to the controllers. The relevance of cryptographic security in this area is widely acknowledged as several recent papers show [3], [4]. Also such a scenario is wlog and can be extended to other applications as well. The information that is broadcasted must be guaranteed to be authentic by the use of MAC codes and time synchronization between the participants. Therefore, this scenario implies the presence of a low computational power device, a number of receivers, and a time synchronization server. Several details about the low computational power device are in the next section; all other participants are standard computers.

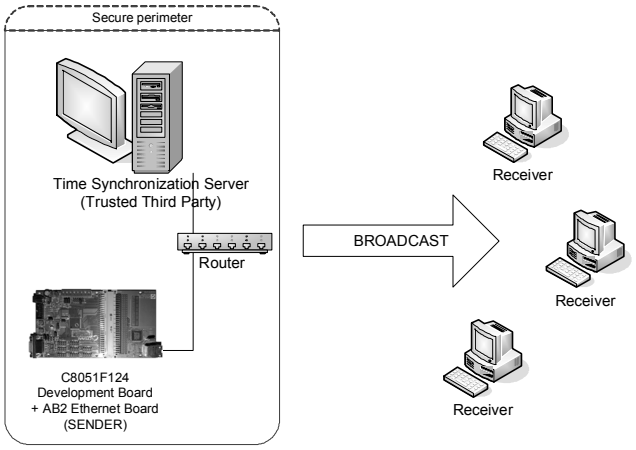

Figure 1. Application setting.

2.2. Description of the development board

The authentication protocol was implemented on a C8051F124 development board [15] from Cygnal Integrated Product Inc., now owned by Silicon Laboratories [16]. For fulfilling the communication requirements, the board was connected to an AB2-extension Ethernet development board that uses a CS8900A low cost Ethernet controller. This ensemble is presented in figure 2.

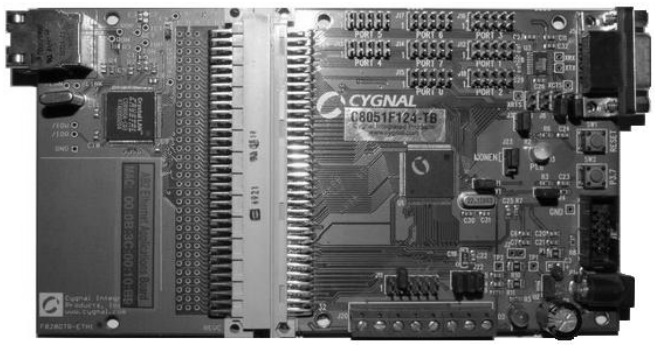

Figure 2. C8051F124 Development Board connected to an AB2 Ethernet Board

The 8051 microcontroller found on the board has an internal oscillator that comes calibrated to obtain a 24.5 MHz frequency. By using the integrated Phase-Locked Loop (PLL), the output frequency is currently set to 49

MHz. The available memory on the board consists in 256 bytes internal RAM and 64kbytes external RAM for data storage and 64KB of flash memory for storing the application code.

Temperature readings are made using the microcontroller's built-in temperature sensor which is internally connected to an Analog to Digital Converter (ADC). The ADC works on 12 bits and the output is the temperature value in Celsius degrees, see [18] for more details.

The development board has five timers. Two 16-bit timers are used by the AB2 extension board and another three 16-bit auto-reload and capture timers are available for use by the application. In particular we have used one timer for the temperature readings and one timer was programmed to measure the local time for synchronization. Using one timer for synchronization purposes was needed as the board doesn't have any clock to keep track of the local time, for example a real time clock.

The AB2 development board uses a CS8900A low-cost Ethernet LAN controller. This controller can be used in embedded applications that communicate over UDP or TCP sockets. More complex applications such as HTTP or FTP servers can be done as well. In particular, in our application we programmed it for UDP communication, as this is the best way to achieve a broadcast communication.

3. The cryptographic protocol

The use of one-way chains has proven to be a very effective mechanism in assuring authenticity of information broadcasted in low computational power environments. This is because by the use of elements from one-way chains as keys for MAC codes in conjunction with time synchronization removes the need for secret shared keys, which a MAC will require otherwise. The first proposal is in [11], [12] and afterwards other papers extended this idea or proposed related solutions [1], [6], [7], [8]. In one of our previous papers we have already pointed out the relevance of a one-way chain based authentication protocol for an industrial control system [8]. For the current scenario we will use a protocol based on one-way chains and time-synchronization similar to the TESLA protocol [11].

As seen in the previous section, the application needs a time synchronization server, which acts as a trusted third party (TTP), for assuring time synchronization and distributing the initialization key of the sender to the receivers. However, when a TTP is involved there are two types of trust in which the TTP can be engaged: unconditional trust and functional trust. A TTP is called unconditionally trusted if it has access to the secret and private keys of users. A TTP is called functionally trusted if all that we request is to behave honest, and the TTP does not have access to private or secret information.

Of course our scenario requires for the time synchronization server to be functionally trusted, since the server does not have access to the secret keys used for the MAC codes, but indeed, if the time synchronization server is corrupted and gives false time values then one can easily forge the packets. For example, if the time given by the time synchronization server is behind the correct time from the sender then an attacker can forge the packets as keys may be already disclosed for packets that also verify the security condition from the perspective of a receiver with wrong time synchronization. Therefore we rely on the fact that the time synchronization server behaves honestly, i.e. is functionally trusted.

The description of the protocol now follows. The structure of the broadcasted packet is the following:

$$Sender \rightarrow Receivers : P_i = \left\{ \theta_i, MAC_{k_{i+1}}\left(\theta_i\right), k_i \right\}, i = \overline{1,n}$$

These packets are broadcasted at time $i \cdot \delta$ where δ is the key disclosure interval, see the experimental results form section 4 for the exact values of δ. Here θ_i is the value of the temperature acquired by the temperature sensor at time $i \cdot \delta$ and $MAC_{k_{i+1}}\left(\theta_i\right)$ is a MAC computed on these value with key k_{i+1} which will be disclosed in the forthcoming packet. The value of k_i is computed as $k_i = Hash^{n-i}\left(k_{gen}\right)$, where n is the length of the chain, *Hash* is a hash function and k_{gen} is a random value from which the chain is generated.

All receivers must follow a time synchronization procedure with the time server, which consists in the following standard steps (a similar procedure was used in [7], [11], [12] and other papers to which the reader is referenced for further details):

$$Receiver \rightarrow TimeServer : \ Nonce_R$$

$$TimeServer \rightarrow Receiver : \left(\begin{array}{c} Nonce_R, CurrentTime, DisclosureDelay, \\ StartTime, Duration, k_0 \end{array} \right)_{SigRS}$$

The receivers must follow this procedure every time the chain is exhausted, this happens at time *CurrentTime*+*Duration*. For future work, if the board proves to be good enough to handle public key operations, such as the one from [10] we plan to move this entire procedure on the board and possibly renounce on the time server.

Also, the sender, which is the microcontroller, must follow an initialization stage on the time server. The main purpose of the initialization on the time server is to announce the time when the broadcast starts, the key disclosure period and the initialization key for the one-way chain, i.e. $k_0 = Hash^n\left(k_{gen}\right)$. In the addressed scenario we had considered that the communication line between the board and the synchronization server is secure, as they are part of a secure perimeter. Therefore a simple announcement to the time synchronization server is enough. However, the problem is that the board doesn't have in its development library any function that can be used to generate random numbers. Implementing a function for generating random numbers is future work for us as in this context generating random numbers on the board can be avoided. However, implementing such a function does not seem to be a trivial task, as there are not many sources of entropy on the board. More, information can not be stored at runtime since the board does not have non-volatile memory, except for the flash that stores the program and which is dangerous to erase since erasing it too frequent would lead to memory inconsistencies. In the absence of non-volatile memory, the use of counter mode for an encryption function is a problem because when the board goes off-line we can not store the value of the counter and when power-up the same numbers will be generated by the board. In these conditions we must help the board with values from the time server. Fortunately, this can be done as they are part of the secure perimeter. Therefore the device will use random nonces received from the time server in order to generate a one-way chain. However if the random value from the time server is simply used to compute the one-way chain, then what we get is unconditional trust for the time server, since the time server is now in possession of the secret key used by the board. Therefore, in order to remove this shortcoming, we will use a random nonce from the time server just to derive the key that is directly used in the construction of the one-way chain. The following are the initialization steps between the board and the time server:

$$Sender \rightarrow TimeServer:\ LocalTime$$
$$TimeServer \rightarrow Sender:\ Nonce$$
$$Sender \rightarrow TimeServer:\ k_0, StartTime$$

Now the value of k_{gen} can be easily computed as $k_{gen} = MAC_{k_{secret}}\left(Nonce\right)$. Here k_{secret} is a secret random value written on the flash of the board. In this way a time server, if providing fresh random nonces, will not know the secret keys used by the sender – therefore the time server is only functionally trusted.

For the case when the communication line is not secure between the board and the time server, the use of MAC codes and nonces to ensure the information is fresh is the only alternative. The use of just one secret shared key between the board and the time synchronization server should not represent a problem. Again, the problem that must be solved in this context is the generation of fresh nonces on the sender's side.

4. Implementation and results

4.1. The cryptographic primitives involved

For the development of the application two distinct environments were used. This is because the synchronization server and the receivers were standard computers where a high level language could be use, while the board had to be programmed in a lower level language. In particular the source code for the board was written in the ANSI C programming language from the Cygnal IDE that comes with the board. As a high level language, we used the .NET and in particular the C# language in which the time server and the receiver applications were written.

For implementing the previously described authentication protocol on the microcontroller, several cryptographic primitives were needed. We used the MD5 [13] hash function and the HMAC [9] message authentication code (based also on MD5). Both algorithms were implemented based on the Request for Comments (RFC) implementations available at [9], [13]. The code was optimized by removing the parts used for testing so that it will consume as little memory as possible to leave enough memory space for storing a one-way chain of a reasonable length.

The one-way key chain was generated using the MD5 one way hash function. The number of elements from the one-way key chain that can be stored is restricted by the data memory left after implementing the application – 3780 bytes. Therefore, if the entire chain has to be stored in memory only a chain with a length of 220 elements can be used. The efficient computation and storage of hash chains is a known problem that was addressed by several papers [2], [5], [14]. The main solution is to compute the entire chain and store only some values for the efficient subsequent computation, this gives an efficient time-memory tradeoff.

As for the cryptography in the .NET applications we used the classes available in the framework, more concrete the *System.Security.Cryptography* namespace. For computing MD5 hashes the

MD5CryptoServiceProvider class was used and for HMAC calculations we used the *HMACMD5* class. The digital signatures on the messages sent from the time server to the receivers, and their verifications, were done using the RSA signature from the *RSACryptoServiceProvider* class.

4.2. Implementing the communication

The Ethernet communication from the board was done using UDP sockets. The TCP/IP library [17] that comes with the AB2 development board can be configured by the developer to meet the specific needs of the application. Therefore, we configured it to enable the use of UDP sockets. Two UDP sockets are used. One is for the communication with the Time Server in the initialization task, and one is for broadcasting the messages to receivers. The TCP/IP library makes the communication easy by the use of the following functions: *mn_open*, *mn_send*, *mn_receive* and *mn_close* - see the documentation for further details [17].

UDP and TCP/IP communication in the .NET applications was implemented by using the *Socket* class from the *System.Net.Sockets* namespace.

4.3. The protocol and its performance

The application must be evaluated both in terms of computational time for the cryptographic primitives involved and communication speed when the protocol is running.

Table 1 shows the computational time of several cryptographic functions implemented on the 8051F124 development board. The computational time was measured by running each test 1000 times and computing the mean value. Inputs of different lengths were tested for each cryptographic primitive as shown in table 1.

Time estimates were done using one of the five internal timers available on the microcontroller. The timer was configured as a counter enabling us to get the current time value before the call to the function and after the function call has ended. The key used for the HMAC calls was the same as its input.

In the broadcast application the MD5 hash function was used. Indeed this function is known to be weak and collisions can be found. However, breaking the protocol requires inverting this function which is not yet feasible; therefore we have used it in the experiments. Of course, for future use we plan to use more secure hashes such as SHA-256 or other variants from the SHA-2 family. Computational times for SHA1 and RC5 are left only for comparison purposes as these functions were not used in our application.

Cryptographic primitive	**Input length** (bytes)	**Runtime** (10^{-3}seconds)
MD5	0	1.91
	26	1.95
	62	3.64
	80	3.63
SHA-1	0	4.33
	26	4.37
	62	8.46
	80	8.44
HMAC-MD5	0	7.13
	26	7.36
	62	9.30
	80	12.58
HMAC-SHA-1	0	16.74
	26	16.97
	62	21.30
	80	29.39
RC5	4	5.18

Table 1. Computational time for the cryptographic primitives on the 8051 based development board.

Key disclosure period (seconds)	**Packets sent**	**Authentic packets received**	**Loss Rate**
10	220	220	0%
1	220	220	0%
0.1	220	213	3.18%
0.08	220	204	7.27%
0.05	220	182	17.27%
0.04	220	174	20.9%
0.03	220	0	100%

Table 2. Communication statistics for the protocol.

Table 2 contains some values for the time interval and test results obtained with the previously described protocol. By using the values from Table 1 we can estimate that for the cryptographic functions that we used in the implementation, a broadcast speed of at most 138 messages per second can be reached. This was estimated by using the computational time required for HMAC-MD5. Choosing a key disclosure period that corresponds to a speed higher than this would lead to the inability of sending messages in the correct time period because computing the MAC would take longer than the key disclosure period. The message will be then considered to be not authentic by failing to verify the time constraint. However, due to other computations done in the application for building the message, like for instance the measurement of the temperature, and because of the network speed the actual broadcasting speed is much lower than 138

packets per second. Tests show that a speed of 10 to 20 packets per second is safe to use. This means that the time spent for cryptographic operations does not raise problems in the protocol performance.

5. Conclusions

An authentication protocol based on cryptographic techniques was developed. The protocol was efficiently used even in the constrained environment offered by the 8051 based development board. The experimental results show that a broadcast authentication protocol is efficient and a large number of receivers are able to receive information that can be checked for authenticity. The protocol implementation can be further improved, extended and used in other environments as well. One limitation of the protocol is at the size of the one-way chains, which is somewhat short due to the memory limitations of the device; a short chain will require frequent initializations. For this purpose improvements based on time-memory tradeoffs for the computation of the chain can be subject of our future work.

6. 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] D. Coppersmith and M. Jakobsson, "Almost Optimal Hash Sequence Traversal", *Proceedings of the Fifth International Conference on Financial Cryptography*, pp. 102-119, 2002.

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

[4] J. Falco, J. Gilsinn, K. Stouffer, "IT Security for Industrial Control Systems: Requirements Specification and Performance Testing", NDIA Homeland Security Symposium & Exhibition, 2004.

[5] M. Fischlin, "Fast Verification of Hash Chains", *The Cryptographers Track at the RSA Conference*, pp. 339-352, 2004.

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

[7] B. Groza, "Broadcast authentication protocol with time synchronization and quadratic residues chains", *Second International Conference on Availability, Reliability and Security*, pp. 550-557, IEEE Comp. Soc., 2007.

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

[9] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", Request for Comments: 2104, http://www.ietf.org/rfc/rfc2104.txt.

[10] A. Liu, P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks", TR-2007-36, North Carolina State University, Department of Computer Science, November 2007.

[11] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, "SPINS: Security Protocols for Sensor Network", Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM, 2001.

[12] A. Perrig, R. Canetti, J. D. Tygar, D. Song, "The TESLA Broadcast Authentication Protocol", In *CryptoBytes*, 5:2, Summer/Fall, pp. 2-13, 2002.

[13] R. Rivest, "The MD5 Message-Digest Algorithm", Request for Comments: 1321, http://www.ietf.org/rfc/rfc1321.txt.

[14] Y. Sella, "On the Computation-Storage Trade-offs of -Hash Chain Traversal", *Proceedings of the Seventh International Conference on Financial Cryptography*, pp. 270-285, 2003.

[15] Cygnal C8051F120/1/2/3/4/5/6/7 High-speed Mixed-Signal ISP FLASH MCU data sheet http://www.betatvcom.dn.ua/komplekt/pdf/digit/microcontroller/C8051F120.pdf

[16] Silicon Laboratories – High Performance, Analog Intensive, Mixed-Signal Integrated Circuits (ICs), www.silabs.com.

[17] Silicon Laboratories, "TCP/IP Library Programmer's Guide" - http://www.silabs.com/public/documents/tpub_doc/anote/Microcontrollers/Precision_Mixed-Signal/en/an237.pdf.

[18] Silicon Laboratories, "Using the On-chip Temperature Sensor",http://www.silabs.com/public/documents/tpub_doc/anote/Microcontrollers/Precision_Mixed-Signal/en/an103.pdf.