Security and availability on embedded systems

N. Burger & Y. Langeron & R. Cogranne & P. Lallement
University of Technology of Troyes, Troyes, France

ABSTRACT: With the fast-paced development of the Internet of Things and its applications within the emerging field of Industry 4.0 - decentralizing decisions by remotely monitoring data and automata - the issues of security and reliability of the whole communication pipeline between the connected devices taking part in this smart industry become crucial. In such context of embedded systems, microcontrollers are widely preferred over microprocessors as they are cheaper, smaller and less energy consuming. Unfortunately, the implementation of security features on microcontrollers, such as signing and ciphering functions, can largely reduce the availability of embedded systems because these functions are energy consuming and computationally complex. Thus, a trade-off has to be found between the prescribed level of availability and security. It is important to note that such a trade-off greatly depends on how the embedded systems will be used, how they are supposed to communicate between each other and if a central node with high computing resources is available. For instance, a common architecture typically consist of several embedded systems communicating up and down with a unique server. Indeed, this architecture is used in several areas where a monitor must supervise and treat data, which is the reason why this setup is chosen. The present paper aims at proposing a method to reach the right trade-off between security and availability, depending on the available resources. However, this problem is difficult to address because of the complexity to measure the security or the availability of a system. Solutions featuring those characteristics and a generic approach are presented to find the most suitable trade-off, in the use case of Industry 4.0.

1 INTRODUCTION

Embedded systems communicate between each others since the beginning of computer engineering. Today, these objects communicate on a network larger day after day : the Internet. For several years now the expression ”the Internet of Things” is used. However, the expression ”embedded systems” regroups many things : Smartphones, automobile electronics, sensors, and miniaturized computers like Raspberry... These objects have different functionalities, hence different computational power. For example, a communicating sensor embedded in a mechanical piece has physical constraints. Components such as micro-controller, battery, antenna, etc will be impacted by these constraints. If a Smartphone, because of these constraints, is less powerful than a computer, a communicating sensor will be much less powerful than a sensor. These constraints alone can define connected objects and their issues (Agrawal & Das 2011).

The eco-system of embedded systems changed drastically since their emergence. Embedded systems were *connected* to a physical interface, that was possible to physically secure. But with the more and more numerous communicating objects - with a server (for supervision purposes) or with other objects - physical security to access an object is not sufficient anymore (Sagstetter, Lukasiewycz, Steinhorst, Wolf, Bouard, Harris, Jha, Peyrin, Poschmann, & Chakraborty 2013).

Whether the communication is radio-based, wired, or other, it is necessary to secure data exchanges. Security is a word that brings together many concepts. These concepts can be split in three categories:

- Confidentiality: The data are exchanged without anyone being able to understand it.
- Integrity: A modification of the exchanged data is detectable.
- Authentication: The data are signed by an emitter. The emitter can not repudiate the message. Another entity can not impersonate the emitter.

These three categories cause several needs that embedded systems must meet. Regardless of the system, the tasks answering these needs will take time

and energy. Moreover, embedded systems will have different options to answer these needs, depending on their capabilities.

Comparing all the possible embedded systems outweighs the scope of this paper. That is why this study will focus on a specific restraint embedded system based on a small, procedural, real-time micro-controller with a transceiver. Possible applications, for a micro-controller like this, can be mass production of communicating sensors for the industry 4.0 or home automation.

2 PROBLEM AND CONTEXT

2.1 *Security definitions*

Authentication, integrity and confidentiality are based on mathematically complex functions. From an emitter point of view, authentication can be answered by a signature algorithm (S), integrity can be answered by a hash algorithm (H), and confidentiality can be answered by an encryption algorithm (E). From a receiver perspective, authentication can be answered by a verification algorithm (V), integrity can be answered by the same hash algorithm (H), and confidentiality can be answered by a decryption algorithm (D).

These functions take some time to execute :

- T_s: the time taken by S to sign a message
- T_h: the time taken by H to hash a message
- T_e: the time taken by E to encrypt a message
- T_v: the time taken by V to verify a signature
- T_d: the time taken by D to decrypt a message

Moreover, security also influences autonomy because these functions use computational power, hence energy, and reduce the autonomy of an embedded system. A balance must be found between security and energy consumption, reliability and real time constraints (Jiang, Pop, & Jiang 2017). Nevertheless this study will focus on the availability of the receiver from a real-time situation perspective. Autonomy, influenced by the energy consumption of the cryptographic functions, is discarded in this paper.

2.2 *Embedded systems secured communications*

Let's take two embedded systems as in figure 1: an emitter and a receiver.

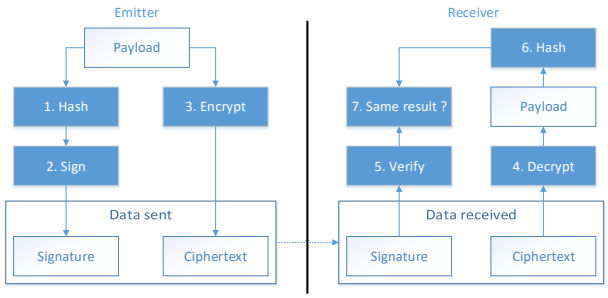

Figure 1: RSA signature

The emitter must:

- 1. Hash its message
- 2. Sign the hash of its message
- 3. Encrypt its message

The computing time for the emitter will be

$$T_{em} = T_h + T_s + T_e \quad (1)$$

And the receiver must :

- 4. Decrypt the message
- 5. Hash the message decrypted
- 6. and 7. Verify the signature and compare with the has previously calculated.

The computing time for the receiver will be

$$T_r = T_d + T_h + T_v \quad (2)$$

The signature (emitter side) and decryption (receiver side) have to be done with a private key. The receiver must be able to communicate with any emitter, and because of memory limitations, it is not possible to store the keys of all the emitters on the receiver memory.

A private key, k_{1d}, is used on the receiver for decryption : only the receiver can decrypt the messages from the emitter. And these messages were encrypted with the public key k_{1e} on the emitter.

Another private key, k_{2s} is used on the emitter for signing : only the emitter can sign its own messages. These signatures will be verified by the receiver with the public key k_{2v}. Two couples of keys are then needed : $K_1 = (k_{1e}, k_{1d})$ and $K_2 = (k_{2s}, k_{2v})$.

The receiver potentially manages many emitters. For example, a receiver in a nuclear power plant receives messages from hundreds of emitters. Once the messages received, they must be proceeded quickly so the data are supervised in real-time.

When the receiver receives too many messages at the same time, it will lead to message losses because the receiver will be busy to decrypt and verify the messages.

This study offers some methods to approach and quantify these *too many messages* and *at the same time* variables.

3 APPROACH AND METHODOLOGY

3.1 *Security level and algorithms*

First, some variables need to be fixed : the security level chosen is the most secured in the sense it manages the confidentiality, the integrity and the authentication of the data.

Not all messages in real life need to be so secured, but for the purpose of this article, the problem is simplified to focus on the workload of the receiver. Indeed, the receiver will be busy a certain amount of time to check these three security parameters. First to calculation of this duration is first needed.

RSA-1024bits with SHA1 is chosen, mainly for the simplicity of the implementation, thanks to a C Microchip library. Each algorithm is set to its specific functions:

- S: RSA-1024bits sign
- H: SHA1
- E: RSA-1024bits encrypt
- V: RSA-1024bits verify
- D: RSA-1024bits decrypt

The RSA functions of encryption-decryption can be considered the same than the RSA signature-verification ones since the mathematical operations are the sames (Pawar & Ghumbre 2016). Indeed, in our case, the only difference is the key used for each operation:

- Encryption and decryption:

$$m^{k_{1e}} = c(modn) \tag{3}$$

$$c^{k_{1d}} = m(modn) \tag{4}$$

- Signature and verification:

$$m^{k_{2s}} = s(modn) \tag{5}$$

$$s^{k_{2v}} = m(modn) \tag{6}$$

The PKCS standard is used for the decryption and verification (on the receiver). So the function is mathematically optimized. Moreover, to balance the workload on the receiver, smaller exponent are used on its side, so the workload will be more important on the emitter for one ciphering or signature. Then, the workload per message is smaller on the receiver than the emitter, but the receiver manages several messages.

That is why equations 4 and 6 take a smaller amount of time than equations 3 and 5.

Using a smaller exponent as a private key on the receiver side for encryption may be a security issue : attackers can guess more easily the private key if they suppose it is a small exponent.

3.2 *Embedded systems definition*

The embedded system is an important variable on this problem. A modest micro-controller has been chosen to amplify the workload of both the receiver and emitter. Choosing a micro-controller means fixing the CPU speed, the Program Memory and the RAM. Each of these three variables can influence the security in some ways, because of memory limitation for example.

The measurements were conducted on a dsPIC30F3014 from Microchip with the specifications described in table 1.

Table 1: dsPIC30F3014 specifications

Parameter	Value
Architecture	16-bit
CPU Speed (MIPS)	30
Memory Type	Flash
Program Memory (KB)	24
RAM (KB)	2

An output of the micro-controller was set to 1 during the operation, so the amount of time spent by the micro-controller to do each operation can be checked.

3.3 *Variables*

Let's note T_r the total duration needed for the receiver to verify the security (see 2), T_{em} the duration between two emissions of the same emitter, and N the number of emitters.

Other duration, like time to send data to a central server, time to read some input on the receiver, time to read the message from the antenna can influence the availability of the system. However, these functions were voluntarily omitted to simplify the problem and to expose how security specifically influences the availability (Jiang, Guo, Ma, & Sang 2012).

From these data, the number of messages a receiver can treat without being overload can be determined, depending of the security (more specifically the time used for it) :

$$N = \lfloor T_{em}/T_r \rfloor \tag{7}$$

4 RESULTS AND DISCUSSIONS

4.1 *Measurements*

In the figure 2, it is shown that with a specific T_r and T_{em}, the receiver is totally busy. Graphically, it is visible that after 10 emitters, the receiver will miss some messages.

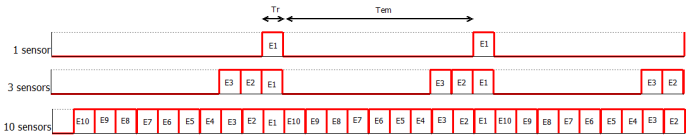

Figure 2: Receiver process time occupation

The assumption that messages follow each other was done to simplify the problem. Real life emitter can send messages at the same time, overlapping each other time frame. A CSMA/CA-like can be implemented by introducing an alea between each messages but it would eventually return to this simple case of receiver overload.

The execution times of the different operations are referred in table 2 and were given by an oscilloscope.

Table 2: Operations and execution times.

Operation	Execution time
RSA 1024 signature + SHA1	158.4 ms
RSA 1024 verification + SHA1	6.8 ms
RSA 1024 encryption	159.2 ms
RSA 1024 decryption	6.8 ms

The total duration of unavailability for the receiver when receiving a message is

$$T_r = 6.8 + 6.8 = 13.6ms$$

.

The minimum time frame for an emitter between two message emissions is

$$T_{em} = 158.4 + 159.2 = 317.6ms$$

.

If emitters never sleep and continuously send messages to the receiver, a message will be received each 317.6 ms. It can now be determined that the number of emitter a receiver can manage in optimal scenario where messages are sent one after another is:

$$N = \lfloor T_{em}/T_r \rfloor = \lfloor 317.6/13.6 \rfloor = 23 \quad (8)$$

Thus, 23 emitters can be associated to a receiver.

4.2 *Comparisons and improvements*

This result can be adapted to other scenario. For example, time measurements with RSA-2048bits will probably take more time for both emitters and receivers and thus, change the number of emitters N.

Totally different encryption/decryption and signing/verifying algorithms can be used and compared on this embedded system. For example, a combination of AES and RSA will probably give a different N.

This comparison tool can be used to compare embedded systems instead of algorithms. For example, a measure done with AES128-CCM (symmetric solution of encryption + integrity + authentication) on a CC2538SF53 micro-controller supporting of Hardware acceleration for cryptography gave us a computing time of 210μs (encryption) and 120 μs (decryption).

Symmetric algorithms, however, give a hard time for key management. This paper doesn't take into account this complexity, but some engineers could use the asymmetric algorithms to exchange a symmetric key.

This means that several type of messages will then be exchange: some encrypted with RSA, some with AES for example. That will imply different duration for these messages but it can be easily implemented on this method.

5 CONCLUSIONS

These results can be used to determine how many sensors can be attributed to a receiver on a specific area. Methods such as a header containing the ID of the sensor can help the receiver to know if it must verify the message or ignore it and thus, save time.
If an industry needs to deploy 30 sensors for instance, in the case studied previously, two receivers will be used.

Moreover, this method can be used to compare micro-controllers, and it can help embedded systems designers to better choose their components, regarding their needs.

More research can add other constraints on the equation, like the impact of security on the battery of the emitter (receivers are supposed to be plugged on a power source). This way, embedded systems designers will be able to choose their level of security depending on the real time and autonomy requirements.

REFERENCES

Agrawal, S. & M. L. Das (2011, December). Internet of Things - A paradigm shift of future Internet applica-

tions. In *Nirma University International Conference on Engineering*, pp. 1–7.

Jiang, W., Z. Guo, Y. Ma, & N. Sang (2012, June). Research on Cryptographic Algorithms for Embedded Real-time Systems: A Perspective of Measurement-based Analysis. In *2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems*, pp. 1495–1501.

Jiang, W., P. Pop, & K. Jiang (2017, July). Design optimization for security- and safety-critical distributed real-time applications. *Microprocessors and Microsystems 52*(Supplement C), 401–415.

Pawar, A. B. & S. Ghumbre (2016, December). A survey on IoT applications, security challenges and counter measures. In *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, pp. 294–299.

Sagstetter, F., M. Lukasiewycz, S. Steinhorst, M. Wolf, A. Bouard, W. R. Harris, S. Jha, T. Peyrin, A. Poschmann, & S. Chakraborty (2013, March). Security challenges in automotive hardware/software architecture design. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 458–463.