Journal of Applied Computing Research, 1(2):95-103
July-December 2011

Performance Evaluation and Comparison of Algorithms for Elliptic Curve Cryptography with El-Gamal based on MIRACL and RELIC Libraries

Daniel Fernando Pigatto, Natássya Barlate Floro da Silva, Kalinka Regina Lucas Jaquie Castelo Branco

Universidade de São Paulo. Av. Trabalhador São-carlense, 400, 13566590, São Carlos, SP , Brasil.
pigatto@icmc.usp.br, kalinka@icmc.usp.br, ays@grad.icmc.usp.br

Abstract: This paper presents a comparison between two implementations of the elliptic curve cryptography (ECC) algorithm based on El-Gamal. Each implementation uses different cryptographic libraries: MIRACL and RELIC. The application of this algorithm meets the common needs of most critical embedded systems, focusing on unmanned aerial and ground vehicles (UAVs and UGVs) and addressing the particularities of each scenario, such as limitations of computing resources and energy supply. One of the challenges is to study and evaluate the impact of the use of each library via a performance evaluation based on statistical techniques. The results have presented the best solution and the influences of each algorithm on response times for each experiment, varying the sizes of keys used by the algorithms.

Keywords: Cryptographic Libraries, ECC, Embedded Systems, MIRACL, Performance Evaluation, RELIC.

Introduction

Security in computer networks has been a topic of intense research due to the increasing use of computers in recent decades and their interconnection networks of all kinds and sizes, often via Internet. With all these possibilities of exchanging information and accessing data stored in databases scattered around the world, organizations and even home users have viewed the need to define mechanisms to ensure that their information is properly protected.

A new type of computer system, best known as embedded system, has drawn the attention of security analysts due to the criticality of the information exchanged by most of these devices. The use of embedded systems has become increasingly common in homes, businesses and natural phenomena monitoring. Among them, there is another class of embedded systems, i.e. critical embedded systems, which involves environmental monitoring, military and agricultural systems. These systems require greater care in relation to the data collected and exchanged between two devices or control bases, or the pilotage of aerial and ground vehicles considered autonomous and unmanned.

Regarding the critical embedded systems, security becomes an even more desired requirement, since the exchange of information among these devices and, often, with their bases of control, is constant. Security aims to provide an appropriate approach to each specific scenario and secure the system against malicious entities that can deliberately access information or modify the operation of these devices, also considering their resource constraints.

The article is organized as follows. Section II presents works related to this paper. Sections III, IV and V are dedicated to a bibliographic review of concepts related to Critical Embedded Systems, Security and Security of Critical Embedded Systems, respectively; Section VI describes the elliptic curve algorithm and our implemented algorithms; Section VII explains the performed experiments and presents the results; finally, section VIII presents the conclusions.

Related Works

Most researches on security have proposed an associative way, integrating public and private key algorithms, and providing confidentiality, authenticity, integrity and performance to their cryptographic applications. However, the performance of elliptic curve algorithms has shown to be feasible for application to not only basic tasks of public key encryption, but also data encryption, an action usually conducted by private key algorithms. The paper (Jena *et al.*, 2009) proposes an efficient cryptosystem to encrypt long messages. The difference is in the mathematical operations performed by the algorithm, which significantly reduce the complexity of operations, enabling its use in systems with constraints, as embedded systems.

The research (Peng and Fang, 2010) compares implementations of public key algorithms on smart cards. The selected algorithms are RSA (Rivest, Shamir and Adleman) and ECC (Elliptic Curve Cryptography). Only the latter was implemented and run on an Intel 8051. The time needed for the key generation was 5.2 seconds, encryption 21.3 seconds, and decryption 17.1 seconds. These values, according to the authors, are low and viable for the architecture addressed.

Some modifications to encryption systems that traditionally use public key algorithms may provide some advantages. In the case of embedded systems, the limitations of memory storage and memory cache are factors that should be considered in the execution of cryptographic algorithms. The ECC can be applied to scenarios with those types of limitations because, as the key size used is considerably smaller than that of RSA, the resource consumption will be correspondingly reduced. In paper (Habib *et al.*, 2009) the technique of replacing RSA by ECC was applied to reduce the resource consumptions of a specific security scheme for WiMAX and improve its performance.

Critical Embedded Systems

Devices with specific functions, such as those present in a microwave oven, are called embedded systems. There exist several definitions of embedded systems: Netrino (2011) defines them as a combination of hardware and software, and maybe a few mechanical parts, designed to perform a specific function. In some cases, they are part of a system or product.

These systems can be found in many everyday situations. Vahid and Givargis (2002) have listed some applications including portable electronics (cell phones, pagers, digital cameras, calculators and PDAs), appliances (microwave ovens, thermostats, security systems, washing machines and lighting systems), office automation (fax machines, printers and scanners), business equipment (cash registers, alarm systems and card readers) and automobiles (electronic fuel injection, antilock brakes and active suspension).

The embedded computing differs from traditional computing in many ways. Functionality is a common goal, but when working with embedded systems there are other factors such as resource constraints in processing, memory and energy that must be considered.

The embedded systems must often provide sophisticated features, such as (Wolf, 2008):

- **Complex algorithms**: The operations performed by microprocessors may be very sophisticated. An example is the microprocessor that controls an automobile engine, which must perform complicated filtering functions to optimize the performance of the car, minimizing pollution and fuel use;
- **User Interface**: Microprocessors are often used to control complex user's interfaces, which may include multiple menus and many options. Navigation map on GPS (Global Positioning System) is a good example of a sophisticated interface.

When working with embedded systems, the operations must frequently return to meet specific demands (Wolf, 2008):

- **Real time**: Many embedded systems have to operate in real time, i.e. if the answer is not delivered in a very short interval of time, the system may have serious problems. In some cases, the existence of failures in fulfilling the task may even threaten lives. In other cases, delays may not represent security problems, but they may cause the dissatisfaction of customers - communication delays in printing, for instance, may result in shuffled pages;
- **Multirate**: Not only must transactions be completed within deadlines, but many embedded systems have multiple activi-

ties being developed at the same time. They can simultaneously control some operations performed at low rates of execution and others that work at high rates. Multimedia applications are examples of Multirate behaviour. The audio and video of a multimedia stream run at very different rates, but should remain well synchronized. Failures in these operations by any of the parts (audio or video) may jeopardize the entire presentation.

In addition to these aforementioned factors that must be taken into account when designing embedded systems, there are concerns about their inherent costs. Wolf (2008) cites the following:

- **Manufacturing cost**: The total cost of building a system is very important in many cases. The manufacturing cost is determined by factors including the type of microprocessor used, the amount of memory required and the input/output devices needed;
- **Power and energy**: The energy consumption directly affects the cost of hardware, since a system may require a greater supply of energy. The power consumption affects the battery lifespan, which is important in many applications, as well as consumption by dissipation, which can be important even in desktop applications.

According to Craigen (2005), another concern must be taken into consideration:

- **Development cost (software)**: The software associated with an embedded system typically costs US$15 to US$30 per line of code, reaching values above US$100. When working with critical applications, the cost per line of code is approximately US$1,000, which can become very high even for a small application that has, for example, 5,000 lines of code.

Examples of critical embedded systems are UGVs (Unmanned Ground Vehicles), which are devices used mainly in military, environmental and agricultural applications to automate processes.

In most cases, general-purpose programmers do not concern about the performance of their applications since they can use available resources without limitations and need their programs to run "fast enough", not "as fast as possible".

In embedded systems, on the other hand, performance is a clear goal of every developer, i.e., developed programs must consider specific short response times. At the heart of embedded computing there is real-time computing, which is the science and art of programming respecting specific times (Wolf, 2008). The program receives input data and has a deadline for the completion of the necessary computational operations. If the program does not produce the required result on time, it is not useful, even if the eventual output produced is correct.

Security

According to Stapko (2008), computer security is the action of protecting personal or confidential information and/or computational resources from individuals or organizations that could deliberately destroy or use such information for malicious purposes. Some properties must be guaranteed for a full and effective implementation of security in computer systems (Bishop, 2004; Kurose and Ross, 2006; Stallings, 2008):

- **Confidentiality**: It is the concealment of information or resources, protecting them from unauthorized disclosure. A practical example can be observed in educational institutions, where the access to information is restricted to classes of users. There is also a concern to keep each student's personal information secret. Confidentiality is therefore to ensure that only the sender and the intended recipient will be able to understand the message content. If an intruder intercepts the message, he/she must not be able to extract information from the ciphertext.
- **Authentication**: It is the guarantee that the reporting entity of the communication is the one it claims to be. Both sender and recipient must confirm the identity of other parties. When communication occurs between humans personally, this problem is easily solved by visual recognition. The problem exists when communication does not allow parties to see each other (the case of computer systems).

- **Integrity and non-repudiation**: Integrity refers to the reliability of data or resources; it is the assurance that neither improper nor unauthorized changes have been made during communication (i.e., modification, insertion, deletion or repetition). The receiver can also verify that the message has come from a particular sender, action that consists in a protection of denial known as non-repudiation.
- **Availability and access control**: With the increasing amount of denial of service attacks, network security has become extremely important. Availability refers to the ability to access information and services when necessary, i.e., a system will be available to offer services according to the system design whenever users request them. Ensuring access to legitimate users of the system involves the concept of access control, which aims to ensure that entities that request access to resources can only do so if they have the appropriate access rights and perform their access in a well-defined way.

One of the most common ways to implement security in a computer system is known as cryptography. In short, cryptography can be explained as a set of methods and techniques to encrypt data by using an encryption algorithm parameterized by a key, converting an original text, called plaintext, into an unreadable text, called ciphertext. It is then possible for the receiver to decrypt this ciphertext, that is, to perform the reverse process and retrieve the original information (Moreno *et al.*, 2005; Tanenbaum, 2003).

Typically, new algorithms are opened to the community as they are developed and confidentiality of information is ensured by the key, which must be kept secret and offered only to relevant entities. The key size in this case is very important, since it determines the encryption level (Tanenbaum, 2003). Furthermore, based on the type of key, one can classify the cryptography as symmetric key or public-key.

Both types of encryption have been widely used to develop secure systems. Although asymmetric algorithms (public key) require 100 to 1000 times more processing than symmetric algorithms (private key), there are situations and applications in which asymmetric encryption has advantages and is convenient for use (Coulouris *et al.*, 2001).

Security of Critical Embedded Systems

The implementation of cryptographic algorithms in computational systems has added an additional level of execution, increasing the response time during the execution of tasks. Security and performance are antagonistic concepts. When the security level of a cryptographic solution is increased, the performance is lost and vice versa, implying the need for a balance between them in accordance with the final application.

When working with critical autonomous vehicles, the role of security is vital to ensure that these devices can function properly without suffering interventions from malicious entities that may divert them from their routes and/or pre-defined areas of activity. Therefore, some common scenarios in which this type of vehicle is used are considered to propose a security solution that ensures its smooth operation.

An important factor is the delineation of an area in which the vehicle shall operate and move within the pre-set limits. A system must monitor the vehicle's position through its location coordinates and the area imposed for their actions. When the vehicle exceeds the limits and does not return within a certain period of time, usually too short, commands must be sent back to avoid the loss of information or harm of the vehicle if unforeseen obstacles show up.

This communication should use authentication mechanisms to ensure that the commands are being sent from an authorized control base to perform that action (or even a controller responsible for monitoring the actions of the vehicle). This is a way to prevent external entities from deliberately sending commands to change the direction of the device for their own profit or to cause damage to the devices.

In addition to this authentication, in some cases, it is necessary to ensure the confidentiality and integrity of information sent in real time to a control base or other vehicles involved in a particular operation. This implies the necessity of using symmetric and asymmetric algorithms for a secure exchange of information.

An algorithm that has been the focus of research in critical embedded systems is the ECC (Elliptic Curve Cryptography). It works with considerably smaller key sizes and, consequently, performs calculations that demand fewer computational resources. Nev-

ertheless, the level of security offered by the algorithm with small keys is equivalent to the use of other public key algorithms with larger key sizes.

Elliptic Curve Cryptography (ECC) and Developed Algorithms

The three accepted encryption schemes are based on three mathematical problems (Jena *et al.*, 2009): Integer factorization problem (IFP), Discrete logarithm problem (DLP), and Elliptic curve discrete logarithm problem (ECDLP). The latter offers a higher level of security, since it operates with smaller key sizes in comparison to RSA and DSA (Digital Signature Algorithm). To achieve an appropriate level of security, RSA and DSA should use 1024-bit key size based on the time needed to break these figures, while the ECC needs to operate with 160-bit keys.

Besides the much smaller key size, ECC algorithm has specific advantages, such as only exponential-time attacks may be applied if the curve is carefully chosen. Even if factoring and multiplicative group discrete logarithms are broken, the ECDL can still be difficult to compute (Jena *et al.*, 2009). Establishing a comparison, the security level of an implementation of elliptic curves with 160-bit key is equivalent to RSA 1024-bit key size (Lenstra and Verheul, 2001).

Two algorithms were developed in this work. The method chosen for the implementation of ECC is El-Gamal, which combines the properties of the El-Gamal elliptic curve encryption method of exchanging messages. Its operation is given as follows: two users must share the same elliptic curve and a point P. Each one must choose a random number that acts as its private key, and multiply the known point, obtaining aP, which becomes its public key. At the beginning of the communication, the public key is transmitted to the other user, which has bP as public key and thus the private key b. For the exchange of messages the user needs to multiply his/her own private key by the public key of the other user, obtaining $b(aP)$, and then add this result to the message encoded in an M number. Therefore, the message will be $M + b\,(aP)$. When the message is delivered to the receiver, he/she will be able to decode it by multiplying his/her private key by the public key of the other user, $(a(bP))$, and subtracting the total content, $M + b\,(AP) - (bP) = M$.

In the implementation two libraries were used as tools to perform mathematical operations: MIRACL (Multiprecision Integer and Rational Arithmetic C/C++ Library) due to its application in other papers (Ramachandran *et al.*, 2007), and RELIC 0.2.3 (Library for Efficient Cryptography) (Relic, 2011). The MIRACL library produced by Shamus Software is proprietary, but free for educational use. It is intended to behave as a tool for developers of encryption systems and offers the necessary operations to handle large numbers and a full support for elliptic curves. RELIC, on the other hand, has been developed by researchers at UNICAMP in order to provide cryptographic tools based on flexibility and efficiency. It has implementations of arithmetic of large integers, arithmetic in binary and prime fields, elliptic curves over prime fields, among others.

The algorithm developed with the MIRACL library operates on blocks of fixed size of 18 characters and the algorithm that uses the RELIC library operates on blocks of 40 characters. Parameters that define the elliptic curve and the point used in common by the users are fixed. These definitions have been established according to some experiments that have proven their efficiency while operating with these block sizes. They have been performed by the authors as a simple comparison between possible block sizes.

The algorithms were developed in C language and run through three actions that must be informed as parameters: key creation, encryption and decryption. The latter two also require the names of input and output files. The implementations of the two algorithms have similar structures with four main functions responsible for creating keys, encrypting, decrypting and calling other functions. In addition, both algorithms have defined structures for the public and private keys.

The function responsible for creating the keys first creates a random integer as the private key and then multiplies a known point of the curve by that number to generate the public key. After being generated, keys are stored in two different files to be exchanged over the network, if necessary.

The function that encrypts a block of text starts transforming the message in a point on the curve. This is the main difference between the developed algorithms in this research. The MIRACL library maps a number of bytes, but it does not work when there is a NULL byte. It is therefore necessary to treat the block previ-

ously and indicate the places where this byte is present. After all of these actions, it is necessary to map the sequence of bytes in a number. From this number it is possible to find the point that is part of the curve where the x-axis is closer to that number.

With RELIC library, it is necessary to map the sequence of bytes, with no previous treatment for numbers. Through transactions between the number and the structure of a known point of the curve, the sequence of bytes is transformed into a point.

Using the messages mapped at points, the encryption is performed by multiplying the private key by the receiver's public key and add to the point of the message. In contrast, the function that decrypts the encrypted block subtracts the multiplication of the private key by the sender's public key, obtaining the message encoded at a point. After the reverse operation, it is possible to obtain the original sequence of bytes.

The function that calls other functions reads the parameters of the execution and operations are defined according to the action. When the action is to generate keys, an appropriate function is executed. However, for encryption and decryption, input and output operations must be performed. In encryption, it is necessary to read the bytes from the original file and write the encrypted blocks at points in an output file. In decryption, the program reads the points of the encrypted file and writes the byte sequences in the output file, restoring the original file.

In the MIRACL library scenario, functions to read and write points are already implemented, but in the case of RELIC it was necessary to create them. The structure of a point was studied to determine the basic data type of each component of the structure and encode functions of writing and reading for each structure.

Experiments and Results

The main application of the ECC algorithm in this work is specifically to the transmission of information between critical embedded devices in both air and ground vehicles (UAVs and UGVs), considering the particularities of each scenario and the concerns inherent in this type of system.

The experiments were set up according to rules to evaluate the performance of computing systems (Jain, 1991). A variety of terms, such as response variable, factors, and interaction levels is used during the stages of design and analysis of experiments. Response variable represents the result (output) of an experiment and is often the variable selected to measure the system's performance. Factors are the variables that affect the system response; levels are the values that a factor may assume; and interaction indicates the dependency between the factors evaluated (Jain, 1991).

The first steps were the definitions of a response variable to be evaluated, the amount of replication required for the experiments and the testing environment. The environment used to run the tests is a Pentium Dual-Core CPU T4300 2.10 GHz with 2 GB of RAM and Linux Ubuntu 11.04 operating system. To evaluate the efficiency of encryption and compare the results of the developed algorithms, the response variable selected was the average response time. The whole process performed in the experiments is the encryption and decryption of each input file, that is, the average response time refers to the sum of the average response times of each of these two steps. Each experiment was performed 15 times, ensuring a statistical validation since there was no large standard deviation among the results.

There are a few ways to accomplish the design of experiments. In this work we have

Table 1. Combinations of experiments.

Exp.	Library	Key Size	Message Size
1	MIRACL	160	50
2	MIRACL	160	100
3	MIRACL	256	50
4	MIRACL	256	100
5	RELIC	160	50
6	RELIC	160	100
7	RELIC	256	50
8	RELIC	256	100

used the full factorial design (Jain, 1991). In this type of planning, all combinations are used considering all factors and levels. Thus, it is possible to evaluate all factors, determine the effect of each factor on the experiments and verify the interactions between them. Table 1 shows the possible combinations of the experiments. The first factor is the library used, which has two levels: MIRACL and RELIC. The second factor is the size of the message, which may vary between 50 and 100 KB. Finally, the third factor assumed is the key size, also with two variations: 160-bit and 256-bit. Therefore, it is possible to generate eight different combinations for the experiments.

Figure 1 shows the comparison between the average response times achieved by each of the algorithms run in the first message size (smaller), considering the two key sizes. The algorithm based on MIRACL library has a considerably higher time than the one based on RELIC, in both cases. The times obtained are approximately 9 seconds (MIRACL) and 3.3 seconds (RELIC), in which the key size is 160-bit. When using 256-bit key size, the times are 20.9 seconds (MIRACL) and 10.6 seconds (RELIC).

Figure 2 shows the second comparison chart with the performance of algorithms to encrypt and decrypt the second message size (larger). There was a natural elevation in the response time due to increased data to be processed while maintaining the same characteristics of the previous comparison. The times obtained are approximately 18.1 seconds (MIRACL) and 6.6 seconds (RELIC), in which the key size is 160-bit. When using 256-bit key size, the times are 41.7 seconds (MIRACL) and 21.1 seconds (RELIC).

The charts show a better performance of the algorithm based on RELIC in both cases (Figures 1 and 2). We have calculated the percentage of influence of each factor of the performance evaluation, as well as the influence of the associated factors over the response time. The chart in Figure 3 shows that factor A (algorithm) exerted a 28% influence on the results, which is relevant to the comparison presented. Factor B (key length) exerted a greater influence on the response variable, with a total of 40%. Factor C (message size) influenced the results in 23%. The associated factors exerted small influences: BC influenced 4%, AC only 3%, AB only 2%, and ABC

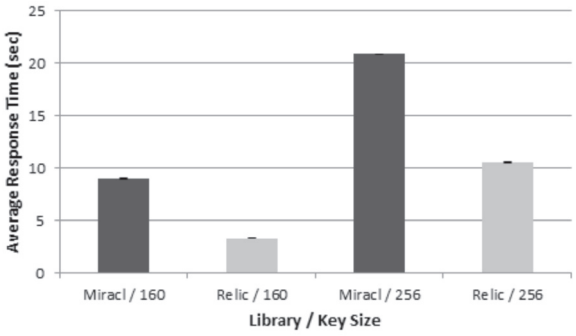

Figure 1. Comparison between MIRACL and RELIC libraries with the first message size (50 KB).

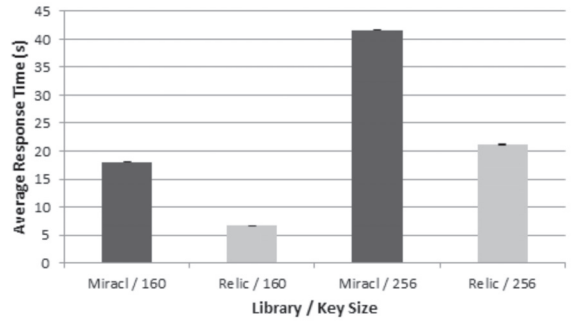

Figure 2. Comparison between MIRACL and RELIC libraries with the second message size (100 KB).

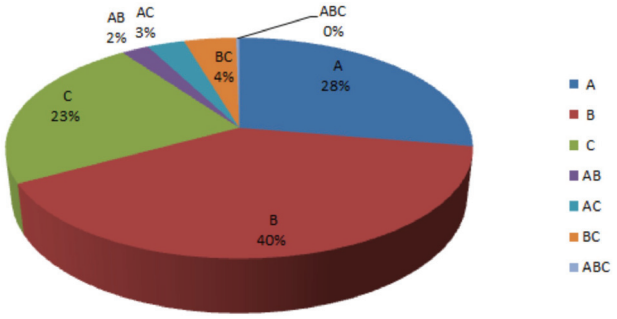

Figure 3. Influences of each factor on the response time (A - Algorithm; B - Key Size, C - Size of message).

associated exerted no influence.

These results have shown that the message size influences the outcome of the application due to the difference in the amount of data to be encrypted. However, the aim of this article was to show that the influence of the algorithm used is quite considerable. It is also important to point out that the adopted key size of 160-bit was recommended by NIST (Lenstra and Verheul, 2001). As the response time is crucial for critical embedded systems that often work with real-time tasks, RELIC is more suitable for the implementation of the ECC algorithm, considering the conditions of the environment used for the experiments. It was also possible to identify several variations when the experiments were conducted in an environment with similar characteristics to a critical embedded system.

It is important to notice that, initially, the time obtained may be classified as long. However, considering that the assumed key size is relatively large and critical embedded systems require the application of cryptography in most cases to send short commands, such as changing routes or missions, the performance presented has met the expectations, obtaining very short response times, which may be considered a solution for real-time systems, the focus of this work. It is also important to point out that these algorithms should be applied in association with symmetric key algorithms to significantly reduce the response times.

Conclusions

The scope of critical embedded systems requires the use of mechanisms that provide high levels of security, in most cases. A normal operation of the devices should be ensured, without compromising their end result. Thus, it becomes necessary to find solutions in cryptography that are computationally efficient and safe.

The state-of-the-art in critical embedded systems has shown some bets on the use of elliptic curve algorithms within the scope of these systems due to their advantages over other asymmetric algorithms. This study has presented a comparison between two implementations of the algorithm based on libraries found in other works. Accordingly, MIRACL library has showed an inferior performance than RELIC library. A performance evaluation was presented in order to compare the libraries and show the influence of each factor involved in the experiments.

For future works there is a need to perform these experiments in specific hardware whose limitations are similar to those of embedded systems. Moreover, in the unmanned air and ground vehicles scenario, these algorithms should be run and tested with movement, as most studies have focused on the evaluation of communication performance in which the elements are fixed.

References

BISHOP, M. 2004. *Introduction to Computer Security*. Upper Saddle River, Addison-Wesley Professional, 784 p.

COULORIS, G.; DOLLIMORE, J.; KINDBERG, T. 2001. *Distributed Systems Concepts and Design*, 3rd ed., Harlow, Addison-Wesley, 800 p.

CRAIGEN, D. 2005. Validation, Verification and Certification of Embedded Systems. Final Report of the NATO RTO Task Group IST-027/RTG-009. RTO-TR-IST-027 AC/323(IST-027) TP/31. Available at: http://www.rto.nato.int/pubs/rdp.asp?RDP=RTO-TR-IST-027. Accessed on: 02/10/2011.

HABIB, M.; MEHMOOD, T.; ULLAH, F.; IBRAHIM; M. 2009. Performance of WiMAX Security Algorithm (The Comparative Study of RSA

Encryption Algorithm with ECC Encryption Algorithm). *In:* INTERNATIONAL CONFERENCE ON COMPUTER TECHNOLOGY AND DEVELOPMENT, 9, Washington, 2009. *Proceedings...* Washington, ICCTD, p. 108-112.

http://dx.doi.org/10.1109/ICCTD.2009.192

JAIN, R. 1991. *The Art of Computer Systems Performance Analysis*. New York, John Wiley & Sons, 685 p.

JENA, D.; PANIGRAHY, S.K.; JENA, S.K. 2009. A novel and efficient cryptosystem for long message encryption, *In:* INTERNATIONAL CONFERENCE ON INDUSTRIAL AND INFORMATION SYSTEMS, 4, Sri Lanka. *Proceedings...* Sri Lanka, ICIIS, p. 7-9.

http://dx.doi.org/10.1109/ICIINFS.2009.5429899

KUROSE, J.F.; ROSS, K.W. 2006. *Redes de Computadores e a Internet*. 3rd ed. São Paulo, Pearson Addison Wesley. 634 p.

LENSTRA, A.K.; VERHEUL, E.R. 2001. Selecting Cryptographic Key Sizes. *Journal of Cryptology*, **14**(4):255–293.

http://dx.doi.org/10.1007/s00145-001-0009-4

MORENO, E.D.; PEREIRA, F.D.; CHIARAMONTE, R.B. 2005. *Criptografia em Software e Hardware*. São Paulo, Novatec, 288 p.

NETRINO. 2011. Embedded Systems Glossary. Available at: http://www.netrino.com/Embedded-Systems/Glossary-E. Accessed on: 02/22/2011.

PENG, Z.; FANG, J.J. 2010. Comparing and Implementation of Public Key Cryptography Algorithms on Smart Card. *In:* INTERNATIONAL CONFERENCE ON COMPUTER APPLICATION AND SYSTEM MODELING, Taiyuan, 2010. *Proceedings...* Taiyuan, ICCASM, **12**:508-510.

http://dx.doi.org/10.1109/ICCASM.2010.5622377

RAMACHANDRAN, A.; ZHOU, Z.; HUANG, D. 2007. Computing Cryptographic Algorithms in Portable and Embedded Devices. *In:* INTERNATIONAL CONFERENCE ON PORTABLE INFORMATION DEVICES, 7, Orlando, 2007. *Proceedings...* Orlando, p. 1-7.

http://dx.doi.org/10.1109/PORTABLE.2007.47

RELIC. 2011. Relic Toolkit – Efficient Library for Cryptography. Available at: http://code.google.com/p/relic-toolkit/. Accessed on: 02/10/2011.

STALLINGS, W. 2008. *Criptografia e Segurança de Redes: Princípios e Práticas*. 4 ed. São Paulo, Pearson Prentice Hall, 512 p.

STAPKO, T. 2008. *Practical Embedded Security.* Burlington, Elsevier, 480 p.

TANENBAUM, A.S. 2003. *Redes de Computadores.* 4. ed. São Paulo, Campus, 945 p.

VAHID, F.; GIVARGIS, T. 2002. *Embedded System Design: A Unified Hardware/Software Introduction.* Riverside, John Wiley & Sons. 352 p.

WOLF, W. 2008. *Computers as Components: Principles of Embedded Computing System Design.* Burlington, Morgan Kaufmann Publishers, 507 p.

Submitted on October 10, 2011.
Accepted on December 14, 2011.