ARCHITECTURAL EVALUATION OF ALGORITHMS RSA, ECC AND MQQ IN ARM PROCESSORS

Gustavo da Silva Quirino[1] and Edward David Moreno[2]
[1]Department of Computer, IFBA, Bahia, Brasil
gucefet@gmail.com
[2]Department of Computer, UFS, Sergipe, Brasil
edwdavid@gmail.com

ABSTRACT

Embedded devices are present in many of our daily activities. In some of these activities is necessary to protect the data from possible attacks. Due to resource constraints, it is necessary to study security techniques that consume the least possible resources of embedded platforms. This paper presents the performance evaluation of asymmetric cryptographic algorithms in embedded systems with Advanced RISC Machine (ARM) processor. The algorithms studied were Rivest-Shamir-Adleman (RSA), Elliptic Curve Cryptography (ECC) and Multivariate Quadratic Quasigroup (MQQ). We have used three criteria in our comparison: the processing time, memory and processor usage. We used the SimpleScalar tool for our simulations analysis. The results showed that MQQ is a good algorithm for embedded systems since it is better than ECC and RSA.

KEYWORDS

Embedded systems, Public-key cryptosystems, Performance Analysis and Design Aids.

1. INTRODUCTION

We are surrounded by embedded systems that perform many different functions. Studies show that only 2% of the six (6) million processors produced annually are used in conventional computers. The remaining 98% are embedded in some kind of machine or device, including cell phones, appliances, vehicles, robots, and other application domains [1]. Furthermore, the embedded devices are playing increasingly sophisticated functions that require high processing power [2]. The battery storage technology not been following the demand of embedded systems, making imperative the development of applications that consume as little energy as possible.

Because the use of embedded systems in routine activities, requiring in many cases, confidentiality of information, it is essential that security techniques are incorporated in these systems to protect data from potential attacks. Requirements of confidentiality, integrity, availability and authentication must be guaranteed through effective security methods. Encryption is the security method most used in embedded systems [3]. In recent years have admitted using asymmetric cryptography systems of low computational power, because some asymmetric cryptographic algorithms like ECC, achieve high levels of security, ordering acceptable amount of resources [4].

The cryptographic algorithm RSA [5] is currently the most used among the asymmetric algorithms, working from the difficulty of factoring large prime numbers. Standardized by

NIST[1] , this algorithm is widely used in transactions on the Internet. The ECC [6], [7] algorithm was created in 80s, and are based on the difficulty of solving the discrete logarithm problem on elliptic curves. Despite its complexity, the algorithm based on elliptic curves have been extensively studied in academia. Recently, the public key algorithm called MQQ[8] was proposed in academia. Experiments performed in the FPGA and PC platforms showed that MQQ is faster than algorithms such as RSA and ECC [8], [9]. Algorithms involved in this study are asymmetric, but each one works with a specific encryption mode.
Studies on performance evaluation of cryptographic algorithms for ARM processors are often quite different in terms of methodology, platform, metrics and focus of analysis, what difficult a direct comparison among the obtained results. Thus, this paper describes a study of cryptographic such as RSA, ECC and MQQ as well as the performance analysis of these algorithms in ARM embedded platforms. This paper is organized as follows. The section 2 describes the characteristics and applications of ARM embedded system. The section 3 gives some background about asymmetric algorithms RSA, ECC and MQQ. The section 4 and 5 discusses on the simulation environment, implementations and observation in the performance evaluation. Finally, some concluding remarks and planning for future works are outlined in Section 6.

2. ARM EMBEDDED SYSTEM

The embedded systems are defined in [10] as any device that includes a programmable computer but is not designed to be a general purpose computer. So a personal computer is not an embedded system, but a fax machine or a clock constructed from a microprocessor is a embedded system.
According to [11] the embedded systems market is of great importance today, as it has doubled in six years, from U$45 billion in 2003 to U$88 billion in 2009. The embedded systems market produces about 10 billion processors per year, something around 98% of all processors produced, indicating that embedded computing is dominant in computer technology [12]. To get an idea of the size of this market, by 2015 the number of tablets with ARM processors will be approximately 258.3 million units [13].

The dominant architecture in the embedded electronics market is based in ARM processors, which are suitable for low power applications due to its relative simplicity [14]. Applications of ARM processors include electronic products like mobile phones, tablets, video games and computer peripherals such as hard drives and routers. The Intel dominates the desktop market, but the market for embedded systems is mastered by the ARM company that produces 3 billion processors per year, approximately 90 per second [12].
According to [15] while growing the complexity of software, hardware and network connectivity, it also increases the number of malicious attacks on embedded systems. The increasing of human dependence on trade for embedded systems, critical infrastructure and health, make these devices attractive for certain attackers. Embedded industrial control systems manage nuclear reactors, oil refineries and other critical structures. According to [15] in 2020 approximately 60% of embedded systems will require some kind of security technique. However, since the computational power of the embedded devices is smaller, [16] states that a prerequisite to ensure confidentiality in embedded systems is the performance evaluation of cryptographic algorithms, including analysis of time and energy cost. The next section describes the history and characteristics of the asymmetric cryptographic algorithm RSA, ECC and MQQ.

[1] U.S. Agency for technology that has a partnership with industry to develop and apply technology, measurements and standards. Further information: www.nist.gov

3. ASYMMETRIC ALGORITHMS

The IEEE 802.15.4 standard specifies physical and media access control layers that could be optimized to ensure low-power consumption and to guarantee real-time constraints[17]. The encryption mechanism specified in IEEE 802.15.4 standard is based on encryption symmetric key. Some authors like [18] , claim that symmetric key cryptographic techniques are preferred over public key techniques for communication in resource constrained platforms. But according to [19] recent studies have shown that it is possible to implement public key encryption using the right selection of algorithms and associated parameters, and optimization techniques for low power. In some cases the public-key cryptography efficiently obtained similar or even greater than symmetric key encryption using keys smaller. According to [20] is already proven that public-key algorithms developed are suitable for constrained devices.

3.1 RSA algorithm

In the introductory paper about RSA, [5] proposed a method to implement a public key cryptosystem whose security is based on the difficulty to be factoring large prime numbers. Through this technique it is possible to encrypt data and to create digital signatures. It was so successful that today is the RSA public key algorithm used most in the world. The encryption scheme uses RSA and signature of the fact that:

$$m^{ed} \equiv m(\text{mod } n) \qquad (1)$$

for m integer. The encryption and decryption schemes are presented in algorithms 1 and 2. The decryption works because $c^d \equiv (m^e)^d \equiv m(\text{mod } n)$. The safety lies in the difficulty of computing a clear text m from a ciphertext $c = m^e \text{ mod } n$ and the public parameters n (e).

Algorithm 1: RSA Encryption

```
Input: RSA public key (n,e), Plain text m  [0, n-1]
Output: Cipher text c
begin
    1. Compute $c = m^e$ mod n
    2. Return c.
End
```

Algorithm 2: Decryption RSA

```
Input: Public key (n,e),Private key d, Cipher text c
Output: Plain text m
begin
    1. Compute $m = c^d$ mod n
    2. Return m.
End
```

3.2 Elliptic curve cryptography (ECC)

The main idea of the algorithms based on curves is to build a set of points of an elliptic curve for which the discrete logarithm problem is intractable. According to [21] cryptosystems based on elliptic curves is an interesting technology because they reach the same level of security systems such as RSA, using minor keys, and thus consuming less memory and processor resources. Recently, elliptic curve cryptographic techniques show that public key cryptosystem is also feasible for resource constrained platforms [22]. This characteristic makes them ideal for use in smart cards and other environments where features such as storage, time and energy are limited.

The scenario of using public key cryptographic algorithms are changing, because according to [23] in terms of public key encryption algorithm RSA continues to lead the number of implementations, but the number of applications that are using algorithms elliptic curves is increasing considerably thanks to the standardization performed by NIST. The algorithms based on curves are standardized according to the ANSI X9.62, FIPS 186-2, IEEE 1363-2000 and ISO / IEC 15946-2. According to [24] public key encryption includes algorithms for key agreement, encryption and digital signatures. Among the algorithms that operate in key agreement, it can mention the Elliptic Curve Diffie-Hellman (ECDH), data encryption on the Elliptic Curve Integrated Encryption Standard (ECIES) and generating the digital signature Elliptic Curve Digital Signature Algorithm (ECDSA).

In the mid-80 [6] and [7] proposed a method of cryptography based on elliptic curves ECC. According to creators of the ECC, an elliptic curve is a plane curve defined by the following equation:

$$y^2 = x^3 + ax + b \quad (2)$$

The efficiency of this algorithm is based on finding a discrete logarithm of a random element that is part of an elliptic curve. To get an idea of the applicability of the algorithms based on elliptic curves on devices with computational constraints [25] argue that the efficiency of ECC cryptographic algorithm with key sizes of approximately 160 bits is the same obtained using the RSA algorithm with 1024 bit key. Algorithms several features are based on elliptic curves, including key management, encryption and digital signature. Key management algorithms are used to share secret keys, encryption algorithms enable a confidential communication and digital signature algorithms authenticate a participant communication as well as validate the integrity of the message.

The procedures of decryption and encryption through elliptic curve analogous to ElGamal encryption scheme are described in the algorithms 3 and 4. The pure text m is first represented as a point M, and then encrypted by the addition to kQ, where k is an integer chosen randomly, and Q is the public key.

Algorithm 3: ElGamal elliptic curve encryption

```
Input: Parameters field of elliptic curve ( p, E, P, n), Public key Q, Plain text m
Output: Cipher text (C1 , C2 )
Begin
    1. Represent the message m as a point M in E ($F_p$ )
    2. Select $k \in R^{[1,n-1]}$.
    3. Compute C1 = kP
    4. Compute C2 = M + kQ.
    5. Return (C1 , C2 )
End
```

```
Algorithm 4: ElGamal elliptic curve decryption
Input: Parameters field of elliptic curve ( p, E, P, n), Private key d, Cipher text (C1 , C2 )
Output: Plain text m
Begin
    1. Compute M = C2 − dC1 , and m from M.
    2. Return (m).
End
```

The transmitter transmits the points C1 = kP eC2 = M + kQ to receiver who uses his private key d to compute:

$$dC1 = d(kP) = k(dP) = kQ \quad (3)$$

and then calculating M = C2 − kQ. An attacker who wants to read of M need to calculate kQ. This model algorithm have been extensively studied since according to [24] in recent years the ECC has attracted attention as a security solution for wireless networks, because the use of small keys and low computational overhead.

3.3. Multivariate Quadratic Quasigroup (MQQ)

The cryptographic algorithms presented above have their security based on computationally intractable mathematical problems: computational efficiency of calculating the discrete logarithm and integer factorization [8]. In 2008, it was proposed a new scheme called multivariate quadratic public key near group (MQQ) [26]. This algorithm is based on multivariate polynomial transformations of nearly quadratic and groups having the following properties [26], [8].

- Highly parallelizable unlike other algorithms that are essentially sequential.
- The encryption speed is comparable to other cryptosystems public key based on multivariate quadratic.
- The decryption speed is typical of a symmetric block cipher.
- Post-Quantum Algorithm

According to [27], [26] MQQ gives a new direction for the cryptography field and can be used to develop new cryptosystems the public key as well as improve existing cryptographic schemes. Furthermore according to [26], [9] experiments showed that the hardware MQQ can be as fast as a typical symmetric block cipher, being several orders of magnitude faster than algorithms such as RSA, DH and ECC.

A generic description for the scheme is a typical system MQQ multivariate quadratic $T \circ P' \circ S : \{0, 1\}^n \rightarrow \{0, 1\}^n$ where T and S are two nonsingular linear transformations and P' is a multivariate mapping bijective quadratic over $\{0, 1\}^n$. The mapping $P' : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is defined in the algorithm 5.

Algorithm 5: Non-linear mapping P’

```
Input:  A vector x = (f1 , ..., fn ) of n linear Boolean functions of n variable. We implicitly
suppose that a multivariate quadratic quasigroup * is previously defined, and that n = 32k, k =
5,6,7,8 is also previously determined.
Output: : 8 linear expressions P 0 (x1 , ..., xn ), i = 1,...,8 and n - 8 multivariate quadratic
polynomials i (x1 , ..., xn ), i = 9, ..., n.
begin
    1. Represent a vector x = (f1 , ..., fn ) of n linear Boolean functions of n variables x1 , ...,
    xn as a string x = (X1 , ..., Xn/s ) where Xi are vector of dimension 8;
    2. Compute Y = Y1 , ..., Yn/8 , where Y1 = X1 , Yj+1 = Xj □ Xj+1 ,for even
    j=2,4,...and
    Yj+1 = Xj+1 □ Xj , for odd j=3,5,...
    3. Output (y)
End
```

The algorithm for encryption with the public key is the direct application of the set of n multivariate polynomials P = {Pi (x_1 , ..., x_n) | i = 1, ..., n} on the vector x = (x_1 , ..., x_n), or is y = P (x), can be represented as y = P (x) ≡ y ≡ A.X .

3.4 Complexity of algorithms

According to [28] problems of integer factorization (RSA) generally allow algorithms that run in sub-exponential time.

According to [29], a subgroup generated by a point whose order **"r"** have **"k"** bits gives **k/2** bits of security, this value is related to the fact that the best algorithm for solving the discrete logarithm problem on an elliptic curve has complexity $O(2^{k/2})$. According [30], unlike the case of the integer factorization problem, there are no algorithms with time sub-exponential to the discrete logarithm problem over elliptic curves. The best known algorithm to date has exponential time. Perform the operations necessary to sum points on elliptic curves is a slower process than making exponentiation modulo a prime, which is the operation used in traditional systems. However, as for discrete logarithm problem over elliptic curves is complex, the same level of security can be achieved with a smaller key, so a smaller key implies faster operations, which effectively offsets the increased complexity of the operations.

According to [31] the computational complexity of the MQQ algorithm is polynomial with degree of regularity D_{reg} more precisely the complexity is: $N^{\omega D_{reg}}$ which basically consists in reducing complexity in an array of ≈ D_{reg}.

4. SIMULATION ENVIRONMENT

The SimpleScalar simulator [32] is a computational architecture that models a virtual computer with CPU, caches (instruction and data), Random Access Memory (RAM) and the entire hierarchy of memories, and can simulate real programs running on such platforms.

In this study, the RSA and ECC cryptographic algorithms were written based on Integer and Rational Arithmetic multi-precision C/C++ Library (MIRACL) [33]. According to [33] the library’s reference amongst MIRACL cryptographic tools provide facilities for implementation of algorithms for security applications in the real world. Furthermore, the MIRACL available codes are compact, fast and efficient, presenting high performance in any processor or platform.

Studies by [34], [35] and [36] showed that codes based on the MIRACL offers better performance than codes based on other cryptographic libraries such as Crypto++, LibTomCrypt, OpenSSL and Lydia+GMP . Regarding MQQ algorithm, was used the version written by [26] in his dissertation. The algorithms were written in the same programming language (C language) and subjected to charges of the same size. The encrypted message in all tests had a size of 44 bytes.
The next section shows the results obtained with the simulation of cryptographic algorithms on the ARM platform.

5. ARCHITECTURAL EVALUATION

According [31] and in accordance with Table 1, the platform sa1core used in ARM simulator is compatible with the platform StrongArm SA-110 which is a 32-bit RISC processor using 206MHz and 16KB in the L1 data cache and 16KB in the L1 cache instruction.

TABLE 1 . FEATURES OF ARM ARCHITECTURE SIMULATED

Fetch queue (instructions)	8
Branch prediction	Nottaken
Fetch & decode width	1
Issue width	2
ITLB	32-entry, fully associative
DTLB	32-entry, fully associative
Functional units	1 int ALU / 1 int MUL/DIV
Instruction L1 cache	16KB, 32 way
Data L1 cache	16KB, 32 way
L1 cache hit latency	1 cycle
L1 cache block size	16B
L2 cache	none
Memory latency (cycles)	64,1
Memory bus width (bytes)	4

The gcc compiler and library MIRACL were used in coding. The code was turned into ARM via compiler arm-linux-gcc. The encrypted files by algorithms were identical, respecting the equivalence between key sizes, which can be viewed in Table 2.

TABLE 2. EQUIVALENCE OF KEY SIZE TO RSA, ECC E MQQ [37][26]

RSA	1024	2048	3072	7680	15360
ECC (Prime field)	192	224	256	384	521
MQQ 160	160	—	—	—	—

The key generation was not part of this study because it was considered expensive for embedded platforms. The codes were simulated by SimpleScalar, which returned information about simulation time in cycles, number of reads and writes, memory pages, use of cache memory, among others. The performance evaluation was separated into sections like processing time, processor usage and memory consumption.

5.1 Processing Time

Considering the frequency of StrongArm processor in 206MHz and having the number of cycles of each algorithm, it was possible to calculate the time in milliseconds (ms). The Figure 1 shows that in the ARM platform, the processing time of RSA was slower than ECC and MQQ for all variations of key. The MQQ 160 had the best performance among the algorithms analyzed, since it had a time of 18.7 ms, against 305.8 ms of the ec-elg_p192 and 4302.8 ms RSA 1024. These data show that MQQ in the ARM platform is 16 times faster than ec-elg_p192 and 230 times faster than RSA 1024.

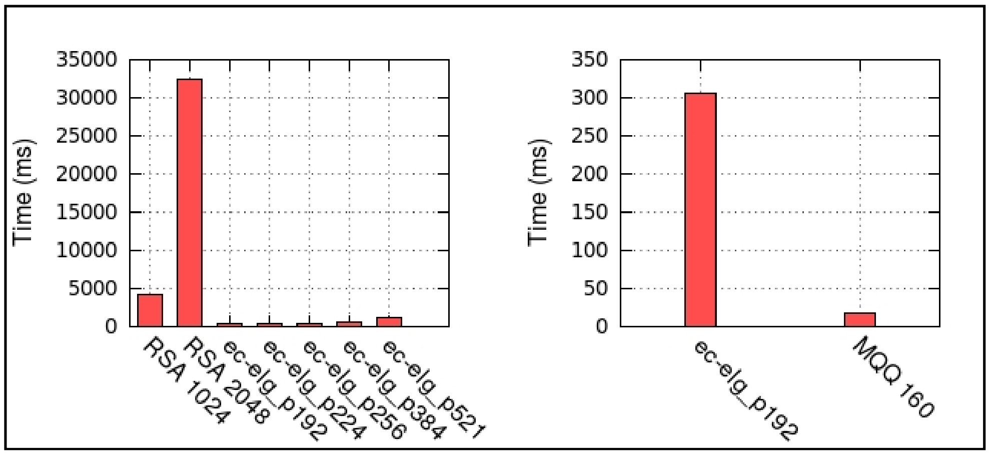

Figure 1. Processing Time

The Figure 2 shows that in the RSA and MQQ algorithms, the enciphering consumes more time than deciphering, but the ECC algorithm showed equivalent time of encryption and decryption. The decryption of algorithm RSA1024 consumes 28% of the total execution time of the algorithm. The decryption algorithm MQQ 160 consumes only 23% of the total execution time of the algorithm. In contrast, the decryption algorithm ec-elg_p192 consumes 48% of the total time of the algorithm.

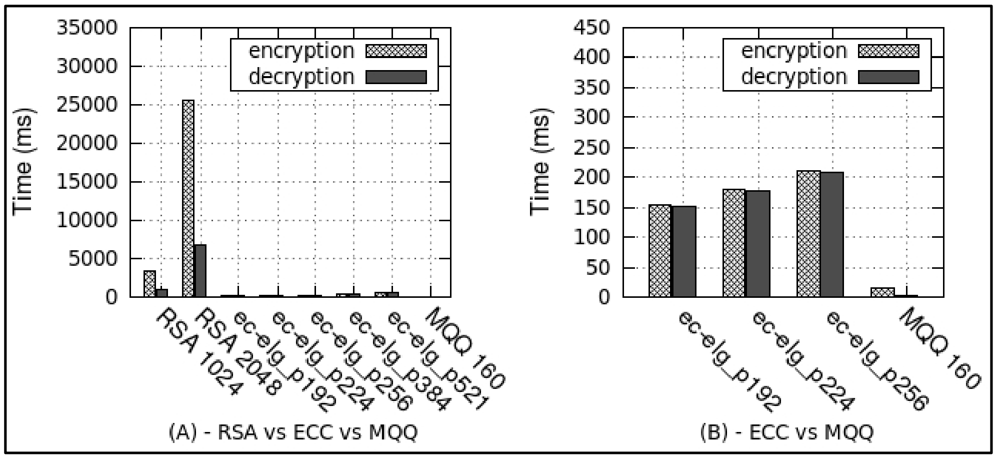

Figure 2. Processing Time divided by encryption and decryption

5.2 Evaluation of processor

The performance evaluation took into account criteria such as number of instructions, number of reads and writes, as well as the amount of branches, which are a code sequences that are conditionally executed according to a flow control. When the number of branches is high, the program presents a lower performance because tests require conditional processing time, and increase the program code.

TABLE 3. NUMBER OF INSTRUCTIONS

Algorithm	Instructions (bilion)
RSA 1024	1,110
RSA 2048	8,426
ec-elg p192	0,055
ec-elg p224	0,067
ec-elg p256	0,080
ec-elg p384	0,140
ec-elg p521	0,248
MQQ 160	0,004

The Table 3 shows that the number of instructions executed by the MQQ 160 was 277 times smaller than the number of instructions executed by the algorithm RSA 1024. Regarding ec-elg_p192, the number of instructions executed by MQQ 160 was 13 times smaller.

Regarding the number of cycles per instruction (CPI), Figure 3 shows that the algorithm based on curves showed the highest mean related to CPI values, followed by the RSA algorithm and finally, the MQQ algorithm. The MQQ algorithm presents a good value of CPI, emphasizing that the algorithm is fast and has less processing time.

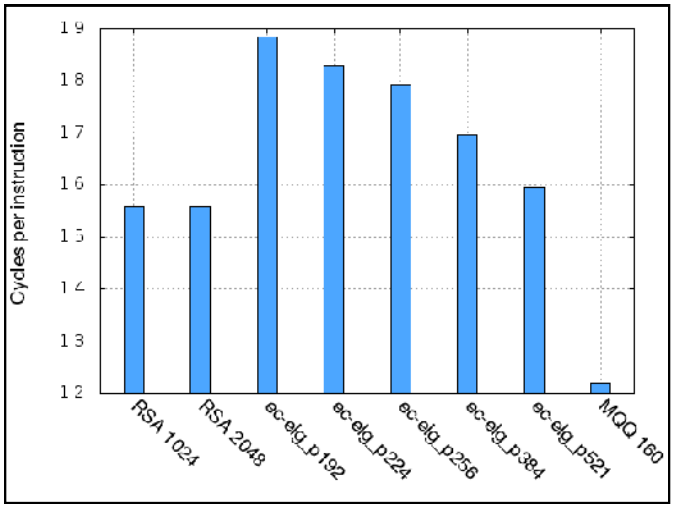

Figure 3. Cycles per instruction

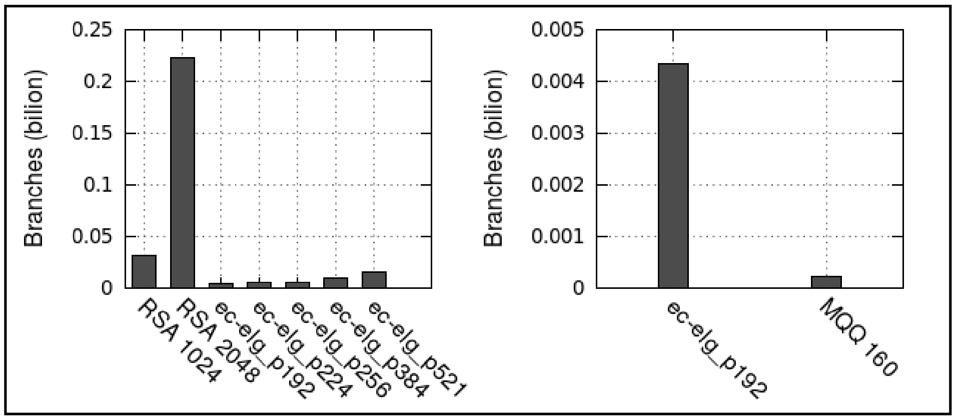

Figure 4. Number of branches

The Figure 4 shows that the algorithm RSA 2048 showed 0.22 billion of branches, against 0.03 billion from RSA 1024. The algorithms based on curves and the MQQ showed smaller number of branches. The ec-elg_p192 presented 0,004 billions of branches compared to 0,0002 billions presented by MQQ 160. The figure shows that the MQQ 160 presented a number of branches 21 times lower than presented by ec-elg_p192 and 150 times smaller than presented by RSA 1024. This is another reason that explains why the MQQ presents better results when compared to RSA and ECC.

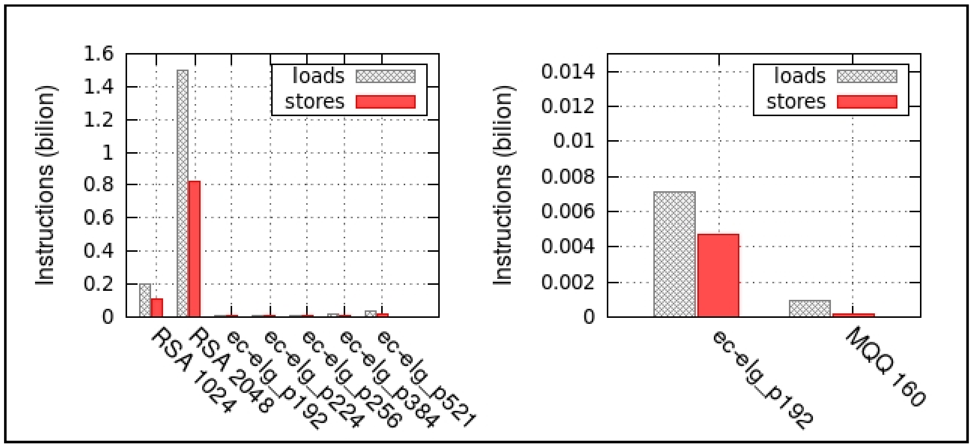

Figure 5. Number of loads and stores

The verification of the amount of loads and stores performed by each algorithm on ARM platform has not changed the scenario of the benchmark, since according to Figure 5, the RSA algorithm performed more operations of this kind regardless of variations in key from our algorithms studied.

5.3. Evaluation of memory

The evaluation of memory consumption aimed to study not only the amount of memory occupied by each algorithm, as well as specific issues of cache performance that can determine which algorithm is more suitable for ARM embedded platforms. Among these questions we have the replacements and writebacks. Replacements occur when a data block is removed from the main memory and goes to the cache memory. Writebacks is writing information that was in the cache, but by conflicts in the use of the cache had to be transferred to the main memory. Both replacements as writebacks are detrimental to system performance. So, the RSA has a high quantity of these operations and the performance is lower when compared to ECC and MQQ algorithms.

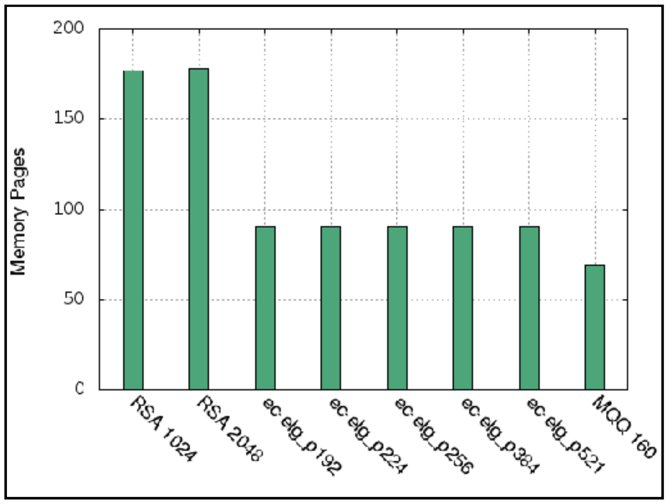

Figure 6. Memory pages

In relation to the number of memory pages used by each algorithm, it's possible note from Figure 6 that RSA 1024 occupied 177 pages and RSA-2048 occupied 178 pages. All variations of ECC occupied 90 pages and MQQ-160 occupied 69 pages. These numbers shows that MQQ-160 is 61% more economical in memory consumption when compared to RSA 1024 and 23% more economical when compared to ec-elg_p192.

The Figure 7 and Figure 8 refers to the percentage of processor query which were considered misses, i.e. loss of information in the cache memory. This failure occurs when data or instructions are not available in the cache, causing the query has to be performed directly in RAM.

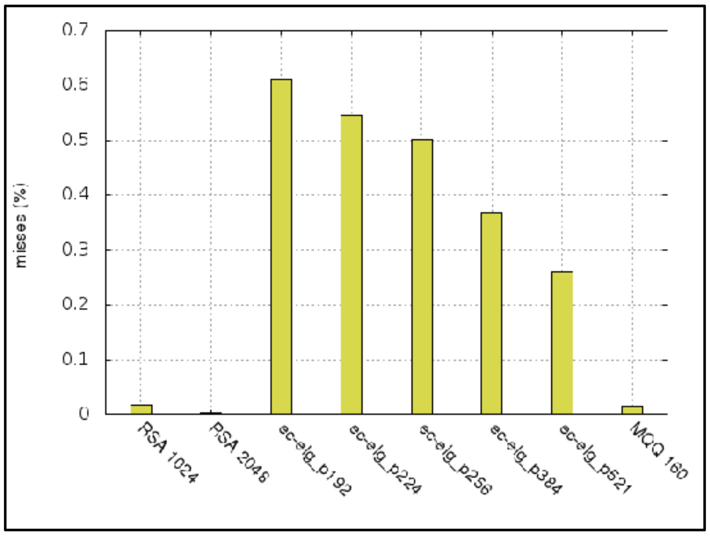

Figure 7. Percentage of misses in the instruction cache

It can be seen in Figure 7 that most of misses in instruction cache occur in algorithms based on elliptic curves. But in general all algorithms presented low percentage of misses per access this cache. The algorithm RSA 1024 showed a percentage misses of 0.018%, compared to 0.61% of ec-elg_p192 and 0.015% of MQQ 160.

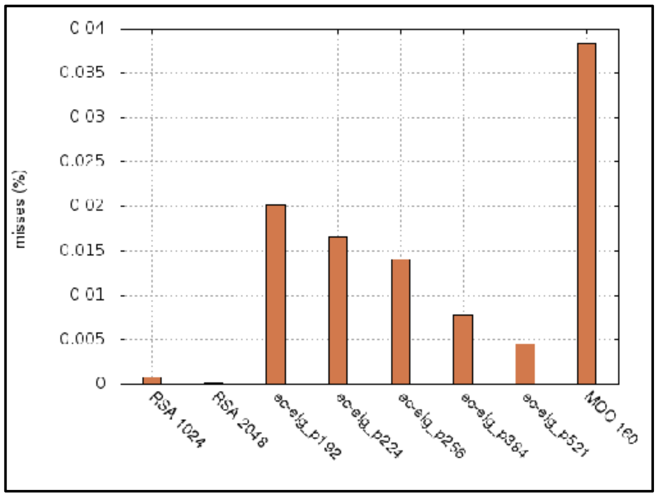

Figure 8. Percentage of misses in the data cache

Unlike the percentage of misses in the instruction cache, the MQQ algorithm has the highest percentage of misses in the data cache. However, as can be seen in Figure 8, the levels achieved by all algorithms are considered irrelevant because they represent less than 1% relative to the amount of hits [38].

The number of memory accesses that can be seen in Figure 9 was significantly higher in the RSA algorithm. This number is the result of the sum of satisfactory accesses (hits) with the considered (misses). As the number of misses was low in this evaluation, we chose not to show the number of hits, since this number is very close to the number of access. Through Figure 9, it's possible see that the number of accesses in instruction cache was larger than in data cache. Further in accordance with Figure 9 and taking as the focus the algorithms MQQ and ECC, it is apparent that the MQQ had the lowest number of accesses among all algorithms evaluated. In comparison with the algorithm based on elliptic curves, the algorithm MQQ 160 showed only 8% of the amount of accesses of algorithm ec-elg_p192 in the instruction cache and 9% of the number of access in the data cache. The data presented on the use of the instruction and data cache, show that the programs evaluated don’t need of caches larger than those used on the ARM platform, and also that there is a good spatial localization in these algorithms. The high number of accesses of the RSA algorithm shows that it is not suited for embedded applications that utilize ARM platform, and is even more critical in WSNs, where the platforms are less computational power. This indicates that for WSN, the algorithms more suitable would be ECC and MQQ. According to [39], implementation of RSA in embedded devices is not flexible enough to support scalability.

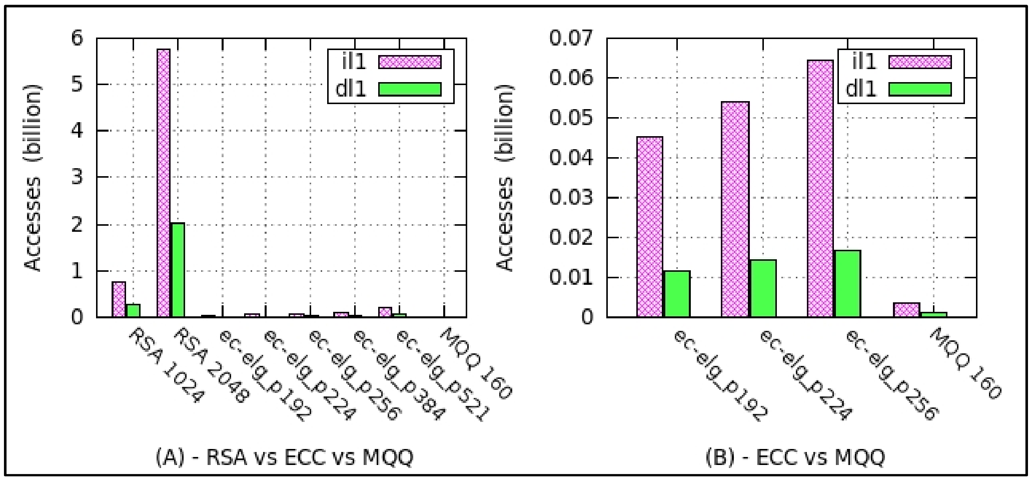

Figure 9. Number of Acesses

TABLE 4. NUMBER OF REPLACEMENTS

Algorithm	**il1 x (10^{-5})**	**dl1 x (10^{-3})**
RSA 1024	1,37	1,67
RSA 2048	2,62	1,98
ec-elg p192	2,74	1,85
ec-elg p224	2,95	1,84
ec-elg p256	3,22	1,86
ec-elg p384	4,09	1,86
ec-elg p521	5,09	1,90
MQQ 160	0,001	0

The replacements, both in instruction and data level, were minimal. The Table 4 shows that only RSA and ECC presented computable data in this evaluation criteria. In instruction cache, ECC algorithm has an increased according the key size increases, moreover, the ECC always present levels higher than RSA. In the data cache, all algorithms analyzed showed an approximate number of substitutions, with the exception of MQQ, that in two caches analyzed obtained irrelevant values (near to zero replacements). Again this behavior explains why MQQ has a good performance and usage of the processor.

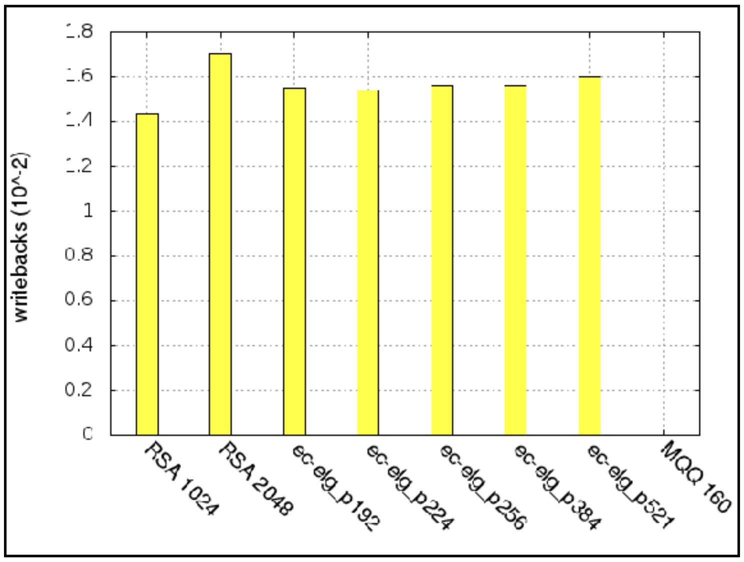

Figure 10. Number of writebacks

These characteristics make the MQQ is on the same level of ECC and present better performance. Regarding to writebacks, it is possible to observe in Figure 10 that RSA 1024 and ECC algorithms maintained a range between 0.014 and 0.016. As expected, the RSA 2048 algorithm obtained approximately 0.017 while MQQ algorithm doesn't show writebacks.

6. CONCLUSION

Embedded systems are increasingly found in various applications, with emphasis on mobile devices, telecommunications, Wireless Sensor Networks(WSN) and general purpose applications. Critical issues in design and development of such systems are the runtime, physical size and power consumption, since the computing platforms have limited resources. This work showed that in ARM platform, the algorithm MQQ-160 is faster, consumes less amount of memory and uses less the architectural resources of the processor when compared to RSA 1024 and ec-elg_p192. Thus, the results confirm that among the algorithms studied, the MQQ-160 is more suitable for embedded platforms with limited resources since it uses lower quantity of resources and gives a good performance. As future work we plan to (i) study the key generation phase, since this analysis considered only the encryption and decryption processes, (ii) to study programming techniques that reduce the power consumption of the components presented in this work as a critical platform ARM, and (iii) more detailed studies of the safety level of MQQ algorithm.

REFERENCES

[1] T. Moreira, "Geração automática de código vhdl a partir de modelos uml para sistemas embarcados de tempo-real," Master's thesis, UFRGS, 2012.

[2] J. Urriza, C. Buckle, F. Pa´ez, D. Barry, L. Diaz, E. Schorb, L. Schorb, S. Lucas, E. Constabel, J. Orozco et al., "Aplicabilidad de sistemas de tiempo real con requerimientos eterogeneos," in V Congreso de Tecnología en Educacion y Educacion en Tecnología, 2012.

[3] T. Yan, K. Xu, M. Wei, and J. He, "Studies on security communications of embedded devices," Advanced Materials Research, vol. 216, pp. 609–612, 2011.

[4] Z. Guo, W. Jiang, N. Sang, and Y. Ma, "Energy measurement and analysis of security algorithms for embedded systems," in Proceedings of the 2011 IEEE/ACM International Conference on Green Computing and Communications. IEEE Computer Society, 2011, pp. 194–199.

[5] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120–126, 1978.

[6] N. Koblitz, "Elliptic curve cryptosystems," Mathematics of computation, vol. 48, no. 177, pp. 203–209, 1987.

[7] V. Miller, "Use of elliptic curves in cryptography," in Advances in Cryptology - CRYPTO Proceedings. Springer, 1986, pp. 417–426.

[8] D. Gligoroski, S. Markovski, and S. Knapskog, "A public key block cipher based on multivariate quadratic quasigroups," Arxiv preprint arXiv:0808.0247, 2008.

[9] M. El-Hadedy, D. Gligoroski, and S. Knapskog, "High performance implementation of a public key block cipher-mqq, for fpga platforms," in Reconfigurable Computing and FPGAs, 2008. ReConFig'08. Inter- national Conference on. IEEE, 2008, pp. 427–432.

[10] M. Wolf, Computers as components: principles of embedded computing system design, 3rd ed. Morgan Kaufmann, 2012.

[11] G. Ascia, V. Catania, A. Di Nuovo, M. Palesi, and D. Patti, "Performance evaluation of efficient multi-objective evolutionary algorithms for design space exploration of embedded computer systems," Applied Soft Computing, vol. 11, no. 1, pp. 382–398, 2011.

[12] M. Wang, V. Callaghan, M. Lear, and M. Colley, "Teaching next generation computing kills; the challenge of embedded computing," Intelligent Campus, 2011.

[13] R. Shim, "Tablets impact the notebook market: Enter the ultrabook," DISPLAY, vol. 2, no. 12, p. 12, 2012.

[14] J. Fernandez, S. Orjuela, J. Alvarez, a nd W. Philips, "Fabric defect detection using the wavelet transform in an arm processor," Proceedings of the SPIE on Electronic Imaging, Image Processing: Machine Vision Applications V, pp. 83 000N–1, 2012.

[15] D. Kleidermacher and M. Kleidermacher, Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development. Newnes, 2012.

[16] Y. Ma, W. Jiang, N. Sang, and Z. Zhong, "An adaptive risk control and security management for embedded real-time system," in Availability, Reliability and Security (ARES), 2012 Seventh International Conference on. IEEE, 2012, pp. 11–17.

[17] L. Chaari and L. Kamoun, "Performance analysis of ieee 802.15.4/zigbee standard under real time constraints," International Journal of Computer Networks & Communications (IJCNC), vol. 3, no. 5, 2011.

[18] P. Sarkar, A. Saha, and S. Bag, "Highly secure key predistribution using affine planes and reed muller codes in wireless sensor networks," Inter- national Journal of Computer Networks & Communications (IJCNC), vol. 3, no. 5, 2011.

[19] J. Sen, "A Survey on Wireless Sensor Network Security," International Journal of Communication Networks and Information Security (IJC- NIS), vol. 1, no. 2, pp. 55–78, 2009.

[20] R. Struik, "Cryptography for highly constrained networks," in NIST - CETA Workshop 2011, 2011.

[21] I. Blake, G. Seroussi, and N. smart, Elliptic Curves in Cryptography, C. U. Press, Ed. Cambridge, 1999.

[22] A. K. Das, "A key establishment scheme for mobile wireless sensor networks using post-deployment knowledge," International Journal of Computer Networks & Communications (IJCNC), vol. 3, no. 4, 2011.

[23] C. Koc, N. Boston, and M. Darnall, About Cryptographic Engineering - Elliptic and Hyperelliptic Curve Cryptography. Springer, 2009.

[24] F. Amin, A. H. Jahangir, and H. Rasifard, "Analysis of Public-Key Cryptography for Wireless Sensor Networks Security," World Academy of Science, Engineering and Technology, vol. 31, no. July, pp. 530–535,2008.

[25] K. Chatterjee, A. De, and D. Gupta, "Software Implementation of Curve based Cryptography for Constrained Devices," International Journal of Computer Applications, vol. 24, no. 5, pp. 18–23, 2011.

[26] R. Maia, "Análise da viabilidade da implementação de algoritmos pó s-quânticos baseados em quase-grupos multivariados quadráticos em plataformas de processamento limitadas." Master's thesis, USP, 2010.

[27] R. Ahlawat, K. Gupta, and S. Pal, "From mq to mqq cryptography: Weaknesses new solutions," in Western European Workshop on Re- search in Cryptology, 2009.

[28] I. S. Torres, "Elliptical curve cryptography - segurança e privacidade em sistemas de armazenamento e transporte de dados," junho 2007, mSDPA, Univ. do Minho.

[29] C. L. P. Gouvea, "Implementação em Software de Criptografia Baseada em Emparelhamentos para Redes de Sensores Usando o microcontrolador MSP430," mestrado, Unicamp, 2010.
[30] J. Stapleton, Information Security Management Handbook, 6th ed.USA: CRC Press, 2012. Ed. Elliptic Curve Cryptosystems.
[31] J. Faugere, R. Ødega˚rd, L. Perret, and D. Gligoroski, "Analysis of the mqq public key cryptosystem," Cryptology and Network Security, pp.169–183, 2010.
[32] T. Austin, E. Larson, and D. Ernst, "Simplescalar: An infrastructure for computer system modeling," Computer, vol. 35, no. 2, pp. 59–67, 2002.
[33] M. Scott, "Miracl–multiprecision integer and rational arithmetic c/c++ library," Shamus Software Ltd, Dublin, Ireland, URL¡ http://www.shamus.ie, 2003.
[34] N. Uto and D. Reis Jr, "Um survey sobre bibliotecas criptográficas com suporte à criptografia de curvas elípticas," Cad. CPqD Tecnologia, vol. 1, no. 1, pp. 133–141, 2005.
[35] D. Pigatto, N. Silva, and K. Branco, "Performance evaluation and comparison of algorithms for elliptic curve cryptography with elgamal based on miracl and relic libraries," Journal of Applied Computing Research, vol. 1, no. 2, pp. 95–103, 2011.
[36] D. F. Pigatto, "Segurança em sistemas embarcados críticos - utilização de criptografia para comunicação segura," Master's thesis, USP, 2012.
[37] I. Branovic, R. Giorgi, and E. Martinelli, "A workload characterization of elliptic curve cryptography methods in embedded environments," in ACM SIGARCH Computer Architecture News, vol. 32, no. 3. ACM,2003, pp. 27–34.
[38] L. Diaz, H. Posadas, and E. Villar, "Obtaining memory address traces from native co-simulation for data cache modeling in system," in XXV Conference on Design of Circuits and Integrated Systems (DCIS-2010),2010.
[39] Z. Wang, Z. Jia, L. Ju, and R. Chen, "Asip-based design and implementation of rsa for embedded systems," in High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on. IEEE, 2012, pp. 1375–1382.

Authors

Gustavo da Silva Quirino holds a degree in Computer Science from Federal University of Tocantins (UFT - 2003), an expert in Linux Network Administration from the Federal University of Lavras (UFLA - 2010). He is currently a professor at the Federal Institute of Bahia (IFBA) and a master student in Computer Science at the Federal University of Sergipe (UFS). Conducts research in embedded systems, sensor networks and security systems.

Edward David Moreno holds a degree in Electrical Engineering - Universidad Del Valle (1991), and Ph.D. degrees in Electrical Engineering from the University of São Paulo (1994 and 1998). The doctorate was Sandwich with Univ. Toronto, Canada (1996) and Chalmers University of Technology, Goteborg, Sweden (1997). He did postdoctoral studies at UFSCAR Univ. Federal de São Carlos (2000). He is currently a Professor in DCOMP (Department of Computing) - UFS (Univ. Federal de Sergipe), in the city of Aracaju, Brazil. It Editoral Board member of 4 international journals: IJCSNS (Intl. Journal of Computer Science and Network Security), TCS Transactions on Computational Science Springer Verlag, the JUCS (Journal on Universal Computer Science), JCP (Journal of Computers). It also assessor courses in computing and informatics and Institutional INEP / MEC. He has participated as program committee, approx., 100 international events and published 6 books in the area of systems design and digital reconfigurable hardware security, and the 3 of them by Springer.