COMPARATIVE ANALYSIS OF DIGITAL SIGNATURES BASED ON RSA CRYPTOGRAPHY AND ELLIPTIC CURVE CRYPTOGRAPHY

MILOŠ S. DRAŽIĆ

Belgrade Metropolitan University, Faculty of Information Technologies, milos.drazic@metropolitan.ac.rs

MILOŠ JOVANOVIĆ

Belgrade Metropolitan University, Faculty of Information Technologies, milos.jovanovic@metropolitan.ac.rs

IGOR FRANC

Belgrade Metropolitan University, Faculty of Information Technologies, igor.franc@metropolitan.ac.rs

DRAGAN ĐOKIĆ

Belgrade Metropolitan University, Faculty of Information Technologies, dragan.djokic@metropolitan.ac.rs

BOJANA TOMAŠEVIĆ DRAŽIĆ

Belgrade Metropolitan University, Faculty of Information Technologies, bojana.tomasevic@metropolitan.ac.rs

Abstract: *Elliptic curve cryptography (ECC) as one of the most advanced cryptosystems is being used by many applications for encryption, public key exchange and digital signatures. The strength of this type of cryptography is in the significantly increased difficulty of solving the problem of discrete logarithm due to the replacement of numbers multiplication by the operation of points "addition" on an elliptic curve, without affecting the key size. The application of RSA to digital signatures is a method complementary to the ECC, in the sense that the trust in the procedure is built on the factorization of a large number of N and the difficulty of taking roots modulo N. Nevertheless, for the same level of security, the key lengths in RSA and ECC can differ drastically, resulting in a large difference in the load of the hardware components, which will be addressed in this paper.*

Keywords: *Digital signatures, elliptic curve cryptography, RSA*

1. INTRODUCTION

The emergence of public key cryptography, which came with the pioneering work of Diffie and Hellman [1], as well as the development of RSA (Rivest, Shamir, Adleman) public key cryptosystems [2], are the cornerstone of modern cryptographic procedures. Encryption solutions are focused on secure communication in an insecure environment, i.e. insecure network. While symmetric schemes, such as DES [3] and AES [4], have been built on ad hoc settings, for which there is no mathematical claim, which undoubtedly, formally rigorously, provides a clear assessment of the degree of security of the encryption method used, RSA has its status as a secure tool based on the paradigm of asymmetric encryption. The creators of this system added mathematical formalism to the entire cryptographic machinery, by which each step of the offered procedure, as well as its consequences, could be clearly mathematically grounded through definitions, theorems, propositions and lemmas. The rigor of this approach brought two benefits: on the one hand, mathematics was used to create the most advanced cryptosystem at the time, and on the other hand, the same mathematics gave very clear statements regarding security assessments and the strength of this solution. It was this second aspect, which had been missing until then, that brought additional confidence in the RSA. RSA quickly became the standard in the world of crypto protection and security, and it is important to mention here that its authors patented their result [5]. Protecting the originality of their results, they prevented insufficiently trained and educated individuals from dealing with "upgrades" and "improvements", which would certainly appear quickly in the case of open source.

This prevents confidence in this technology from being shaken. In addition to this fact, it is important to keep in mind that the computer resources of that time could not be a threat to the RSA. Therefore, the attempt to recognize the threat and weakness of the RSA was reduced to thought experiments, rather than in reality at all. Shor's algorithm is without a doubt the most important representative of such attempts [6]. However, it was Shor's algorithm that showed that with the application of power resources (quantum computers), RSA ceases to be the tool of choice for secure cryptography, because factorization could be achieved in polynomial time. A few years before the advent of the Shor’s algorithm, Victor Miller [7] and Neal Koblitz [8], independently introduced elliptical curves into cryptography. The disadvantage of their work lies in the fact that these are purely mathematical results, which did not clearly indicate that the problem of discrete logarithm would be significantly aggravated by the use of elliptic curves. Also, the authors did nothing to patent and protect their result. To the lost time that passed before the potential of elliptic curves was realized, should be added the time caused by the damage caused by mathematically unqualified people, whose "improvements", with the aim of obtaining the patent rights, in fact only questioned the security of such a cryptosystem. It should not be overlooked that the mathematical complexity of applying elliptic curves and the level of abstraction, such as the operation of points "addition” on a curve, made it even more difficult to build confidence in elliptic curves quickly. Due to all this, the real application of elliptic curves in technology had to wait more than 15 years after pioneering works. However, today it can be said that the lost time has been successfully compensated, so elliptical curves have become the tool of choice for many new technologies, such as cryptocurrencies [9], and that in some important aspects they greatly exceed RSA.

2. THE CONCEPT OF DIGITAL SIGNATURE

Unlike the idea of secure communication in an insecure environment, digital signature rests on a different concept. The idea itself is simple and is part of everyday experience. If there is a document and a person who wants to verify the items, provisions, or messages listed in that document, he/she will do so by affixing his signature, trademark, or any mark by which he/she wishes to certify the authenticity of that document. Here, an adequate analogy can make a clear distinction between public key cryptography and digital signature. Let the letters, which anyone can send, be inserted into the mailbox of a certain person. Although these letters can come from anyone, only the owner of the mailbox will have access to their contents, after opening them. If we imagine that among the received letters there is someone who is stamped, then we can assume that the owner of the box, as well as anyone who could take a look at that letter, will be convinced of the authenticity of that document, in the sense that it will not doubt that letter had just been compiled by the person whose stamp was on the document. On the other hand, only that person, using his own stamp, is capable of creating such an impression and trust. Now we can translate all this into the language used in cryptography. We can consider a mailbox to be a public encryption key, and a private decryption key is actually a mailbox key owned only by the owner. In this way, the concept of secure communication is created. On the other hand, for the physical stamp used, we can consider that in digital signing it represents a private signing key, while the imprint of that stamp on the document is a public signature, or public signing key. Also, the procedure by which the sender puts a stamp on a document is a type of signing algorithm, while the procedure by which the stamp on a given document is compared to a real, physical stamp can be considered a verification algorithm. The basic elements in the steps in digital signing are listed: A private and a public signing key, as well as a signing and a verifying algorithm.

Careful analysis allows us to identify what necessary conditions a successful digital signing should meet. If someone who has insight into the imprint of the stamp would want to create his own physical stamp based on this imprint to try to falsify the real stamp, he/she should not be able to do so. Likewise, the imprint should be so unique that the attacker, unable to create his stamp, must not be able to find among the existing physical stamps available to him, one that gives an identical imprint as the stamp he/she wants to forge. This analysis refers to the case of one printed document. Now imagine that the same stamp is imprinted on a number of documents. Each document and stamp print could potentially reveal more and more about the real, physical stamp to the attacker. The requirement that is imposed is precisely that no matter how many letters with the imprint of the stamp the attacker has an insight into, he/she cannot conclude anything more about the real stamp than if he/she had insight into only one letter with the imprint of the stamp.

This simple analysis clearly indicates that digital signatures are of great importance, because they confirm that the information comes from a reliable party.

3. RSA DIGITAL SIGNATURES

RSA, applied to digital signing, includes the following fundamental steps:

- The sender creates the key
- The sender signs the document
- The recipient is doing verification

In the first step, the following operations are performed:

- Selection of secret prime numbers p and q,
- Selection of the verification exponent V_e, such that V_e and $(p$-1) $(q$-1) are mutually prime,

- Calculation of private signing key S_{priv}, which is a modular inverse of the verification exponent per module (p-1) (q-1),
- Publishing the values of N and V_e, where N = pq.

The second fundamental step involves the following operations:

- Select a digital document D of length equal to or greater than 1 and less than N,
- Digital signature calculation, i.e. public signing key S_{pub}, which is congruent $D^{S_{priv}}$ modulo N.
- Publishing D and S_{pub}.

The third fundamental step consists of:

- Taking public values published by the sender, N and V_e,
- Calculating $S_{pub}^{V_e}$ mod N and confirming that the obtained value is the same as the value of D.

3. ECC AND DIGITAL SIGNATURES

The elliptic curve formalism, applied to digital signing, is a direct application of the rules established by the Digital Signature Algorithm (DSA) [10], which represents a successor and an improved version of the Elgamal solution [11]. Therefore, the steps that are specific to DSA will be listed here, and the difference between standard DSA and Elliptic curve DSA (ECDSA) will be that in the case of ECDSA, the binary operation involves points addition on the curve.

Fundamental steps include:

- The sender creates public parameters and a key
- The sender signs the document
- The recipient performs the verification

In order for the whole procedure to be successfully implemented, it is necessary to start from the construction of a finite field F_p, where p is a prime number. When a construction is made by removing 0 from the field F_p, a group structure F_p^* of order $p-1$ is obtained, which is closed to a binary operation. For such a group, there is certainly one element G such that all other elements of the group, F_p^* = {1, G, G^2, G^3, ..., G^{p-1}}, can be generated from it by successive application of the binary operation. The speed of signing depends on the order of the group F_p^*, so the question arises whether and to what extent security is sacrificed if the order of the group is reduced, i.e. if a subgroup is taken. In general, the index calculus will not simplify the problem of solving the discrete logarithm if a subgroup of the group F_p^* is selected. Working with subgroups simplifies the procedure and shortens it. It remains to define the subgroup of order $q < p$, where q is a prime number. This subgroup is easiest to determine by assuming that it can be generated from the element g which is the q-th root of $(p-1)$-th power of the generator of the group, i.e. $g = G^{(p-1)/q}$. This laid the groundwork for the beginning of the procedure.

As part of the first step, the following operations are performed:

- Selection of prime numbers p and q, such that p is congruent to the number 1 modulo q, as well as the element g of the subgroup of the group F_p^*, which is of order q,
- The private signing key S_{priv} and verification key V_k are created, as follows: first, S_{priv} is chosen such that it is greater than or equal to 1 and less than or equal to q-1, and then a verification key is generated, satisfying $V_k \equiv g^{S_{priv}}$(mod p).
- The verification key is then published (V_k is equivalent to the verification exponent in the RSA procedure).

The second step involves:

- Selection of a document D and calculation D mod q
- Selection of a random number r, which satisfies $1 < r < q$
- Calculation of signature pair:
  S_{pub1} = (g^r mod p) mod q,
  $S_{pub2} \equiv$ (D + $S_{priv}S_{pub1}$) r^{-1} (mod q)
- Publication of document D and pair (S_{pub1}, S_{pub2})

The third step consists of the following procedures:

- Calculating the value pair (V_1, V_2) as follows: $V_1 \equiv$D S_{pub2}^{-1} (mod q) and $V_2 \equiv S_{pub1}S_{pub2}^{-1}$ (mod q)
- Confirm that (g^{V_1} ${V_k}^{V_2}$ mod p) mod q=S_{pub1}.

4. ECDSA VS. RSA DIGITAL SIGNATURE-METHODS

To perform a comparative analysis of the hardware load as well as the time required for digital signing and verification, a computer with the following performances listed was used:

- Processor: Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz (2 processors)
- No. of virtual processors: 8
- System type: 64-bit operating system, x64-based processor
- Edition: Windows 10 Pro
- Installed RAM: 6GB
- Hard disk capacity: 100GB

The comparison was made on the basis of data from official documents related to ECC [9] and RSA [12]. **Table 1** lists the curves used, their size (the length in bits of the field order) and the strength (number of bits of security), as well as the values of $N = pq$ corresponding to the RSA algorithm of the same security strength.

Table 1: Elliptic curves used, their size, strength and modulus N corresponding to RSA algorithm of the same strength

Elliptic curve	Size (bits)	Strength (bits)	N (bits)
Secp192r1	192	96	1536
Secp224r1	224	112	2048
Secp256r1	256	128	3072
Secp384r1	384	192	7680
Secp521r1	521	256	15360

Tools used:

• Development environment: NetBeans IDE 8.2

• Performance Monitor within which the following counters were monitored:

1. Processor > % Processor Time,
2. Memory > Pages/sec,
3. Physical Disk > % Disk Time,
4. Physical Disk > Avg. Disk Queue Length,
5. System > Processor Queue Length,
6. Network Interface > Bytes Total/sec.

Testing was done in the Java programming language with the BouncyCastleProvider [13] as a security provider in ECDSA implementation.

4. RESULTS AND ANALYSIS

Based on the measurements performed using the Performance Monitor, as well as working in the NetBeans environment, which gave us the values of the time it took to generate the keys, signing the digital document (message) as well as verification, we obtained the results listed in the table below.

Table 2: Comparison of time in case of using ECDSA and RSA, which are required for keys generation (Keys gen. column), signing (Sign. column) and verification (Ver. column) for keys of different sizes (Size column) and for the appropriate security level (Strength column)

	Strength (bits)	Size (key length in bits)	Keys gen. (s)	Sign. (s)	Ver. (s)
EC DSA	96	192	0.497	0.039	0.004
	112	224	0.498	0.039	0.005
	128	256	0.5	0.04	0.005
	192	384	0.522	0.043	0.007
	256	521	0.543	0.044	0.011
RSA	96	1536	0.398	0.03	0.002
	112	2048	0.89	0.035	0.002
	128	3072	2.805	0.055	0.004
	192	7680	126.77	0.353	0.005
	256	15360	1245.65	2.663	0.014

What is immediately noticeable is that the key generation speed, which is the first fundamental step in digital signing, is for the lowest level of security on the RSA side compared to ECDSA and that RSA is about 25% faster in execution than ECDSA. This is somewhat understandable, if we take into account that to perform the first step in the case of RSA, it is enough to apply Euclidean, i.e. extended Euclidean algorithm, with the index N being a relatively small number. In addition, we must take into account that in our example, it was not taken into account that any primality test, such as the Rabin-Miller test [14], [15], was performed when selecting prime numbers. On the other hand, in the first step in ECDSA, a reduction had to be done to a subgroup whose order has a well-defined property given by the modular equation, to conduct a random number generation, and finally to solve another modular equation. Already at this point we can notice that the very first step in the case of RSA, with an increase in the value of the N index, with the aim of increasing the level of security, will increase nonlinearly over time. On the other hand, the increase in the value of the finite group order in the case of ECDSA, in order to achieve a higher level of security, will not be the same as the growth rate of index N. This will result in only a slight increase in key generation time. One of the reasons is the efficiency of the double and add algorithm which performs points addition on elliptic curves (the speed of this operation would be even higher if we worked on Koblitz, not Weierstrass curves, due to the high efficiency of Frobenius mapping [8]). In the case of raising the security level, it is noticed that the key generation time in the case of ECDSA has only slightly increased, while in the case of RSA this increase is dramatic and for a 256-bit security level, it is over 20 minutes! We also note that the time required for signing and verification in the case of ECDSA is between 0.043s and 0.055s and is, as expected, the shortest for the lowest and the longest for the highest security level considered. For ECDSA, we can conclude that the signing time is always longer than the verification time, but that the increase of that time with increasing levels of security is more pronounced in the case of verification time. We can understand this behaviour if we take into account that three modular equations must be solved for both signing and verification, with the difference that in case of signing a document of fixed length (actually a Latin sentence “Festina lente” in our case) is always taken, while in case of verification the parameters are variable, i.e. they grow numerically, so as the order of the group increases, so does the complexity of the modular equations. We can assume that with a further increase in the order of the group (security level), the verification time would become longer than the signing time. Based on the obtained results for ECDSA, we can conclude that of the total time required for key generation, signing and verification, about 90% of the time was spent on key generation and that computer resources are the most loaded during this activity. In the case of RSA, the signing time is comparable to ECDSA for the lowest security levels, but the time-nonlinear behaviour with further increase of the N index is clearly visible and that for 192-bit security level key generation in RSA is very inefficient in relation to ECDSA. Computer resources are almost entirely spent on this activity. In contrast, the verification time is the same as in the case of ECDSA, while the signing time has an increase that is more pronounced with the increase in security levels. This is to be expected if one considers the modular equation that must be solved in the second fundamental step. Based on these results, it can be said that the price for increasing the level of security in RSA was paid by the increase in the value of the N index, which made it significantly more difficult to generate key pairs. In the case of ECDSA, the increase in the level of security was achieved through the introduction of two public keys (signatures), so in order to increase security, it was not necessary to increase the order of the group significantly. An additional reason for this is that the problem of discrete logarithm on elliptic curves is a far more complex than it is in standard modular arithmetic. The fact that the order of the group does not have to increase drastically, as well as the mentioned efficiency of binary operation on elliptic curves (double and add), results in an almost constant time required for key generation, signing and verification, regardless of the required security level.

As for the counters that monitored the system load, we can say that in the case of ECDSA in the first place are time-localized jumps in value, and the reason for this is the fact that execution times are very short. Therefore, only the highest counter values are listed in the ECDSA case (**Table 3**). The System\Queue Length counter was zero for all cases, so it is not shown in the table.

Table 3: Counters in the case of ECDSA for different key lengths.

Key	Mem. (pages/s)	Net. (bytes/s)	\%disk time	\%avg. queue length	\%proc. time
192	16.9	17949	2.13	0.021	35.48
224	24.2	20607	2.02	0.02	36.32
256	17.2	28862	2.53	0.025	35.86
384	16.9	18891	2.17	0.022	33.17
521	24.9	20034	2.38	0.024	35.35

Based on the obtained values, we can see that these are events that are quite stable and uniform, and that they show that the ECDSA loads the system almost equally, regardless of the level of security. This is in favour of the constancy in execution times that was observed earlier.

In the case of RSA, for lower security levels, significant counter changes are, as with ECDSA, localized jumps, but as the security level increases, the system load becomes continuous. In this sense, two modes can be recognized: one corresponds to time-localized jumps in the system load, while the other is characteristic of a continuous load. Based on the analysis so far, it is not difficult to conclude that the first mode includes RSA signatures with key values of 1536, 2048 and 3072 bits, while the second mode includes lengths of 7680 and 15360 bits. The **Table 4** shows the characteristic values: maximum for the first mode, and mean values for the second mode. System > Processor Queue Length counters in the case of 7680 and 15360 show non-zero behaviour, which is understandable given the length of execution, and amounts to 0.18 and 0.77, respectively.

Table 4: Counters in the case of RSA for different key lengths and two different modes. In the case of the first mode (key lengths 1536, 2048 and 3072) the maximum values are displayed, and in the case of the second mode (7680 and 15380) the mean values are shown.

Key	Mem. (p/s)	Net. (by/s)	\%disk time	\%avg. queue length	\%proc. time
1536	16.94	38299	2.27	0.008	30.22
2048	16.89	46791	2.62	0.026	25.83
3072	16.77	39620	2.6	0.006	25.11
7680	1.126	22536	0.276	0.003	15.351
15360	5.1498	44501	0.332	0.003	13.282

We notice that in cases of lower security level, the system load is similar to ECDSA, but it is also important to note that the mean values of Memory> Pages / sec and Physical Disk>% Disk Time counters in continuous load mode are about an order of magnitude smaller in relation to the pulse mode, while in the case of Processor>% Processor Time, the counters are only about 2 times smaller. Considering the execution length in continuous load mode, it is clear that the hardware components are significantly loaded in the case of RSA than in the case of ECDSA.

5. CONCLUSION

The primary difference between RSA and ECC is in the strength of encryption. ECC provides an equivalent level of encryption power as an RSA algorithm with a shorter key length. The speed and security offered by ECC is higher than the RSA for Public Key Infrastructure (PKI) and Digital Signature. RSA requires much longer key lengths to implement encryption. ECC requires a much shorter key length compared to RSA. RSA increases key lengths by increasing security. Currently, the standard in the RSA is a 2048-bit key length. As various operations are performed on the computer/ server while simultaneously generating keys, signing, encrypting and decrypting data, this certainly puts an additional burden on the computer. The ECC does not face this type of challenge. The fact that its keys are much smaller requires less load on computers / servers. ECC with a 224-bit key achieves the same level of security as RSA with a standard 2048-bit key, which is almost 10 times smaller than the RSA. RSA is resource hungry as a cryptosystem. Encryption standards are becoming stricter. A higher degree of key pair protection is required. RSA can increase key length, but it does not improve security. Its security is not proportional to the growth of the key. A 3072-bit key does not provide double security compared to a 2048-bit key. More time to generate keys affects the security drop. The time lost in this way is given to the attacker, and on the other hand, computer resources are more burdened, which endangers the life of hardware components that can lead to the collapse of computer systems. With a much shorter key length for the same level of security through an efficient load on the hardware components, the ECC beats the RSA.

Based on all this, we can say that the ECC is a qualitative and quantitative step forward in relation to the RSA and that in case of a high level of security, the ECC should be the tool of choice.

REFERENCES

[1] W. Diffie and M. E. Hellman, “New Directions in Cryptography,” IEEE Trans. Inf. Theory, vol. IT-22, pp. 644-654, Nov. 1976.

[2] R. L. Rivest, A. Shamir and R. Adleman, “A Method for Obtaining Signatures and Public-Key Cryptosystems,” Commun. ACM, vol. 21, pp. 120-126, Nov. 1978.

[3] NIST–DES, “Data Encryption Standard (DES),” FIPS Publication 46-3, National Institute of Standards and Technology, Oct. 1999.

[4] NIST–AES, “Advanced Encryption Standard (AES),” FIPS Publication 197, National Institute of Standards and Technology, Nov. 2001.

[5] R. L. Rivest, A. Shamir and R. Adleman, “Cryptographic communications system and method,” U.S. Patent 4 405 829, Sep. 20, 1983.

[6] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124-134, Nov. 1994.

[7] V.S. Miller, “Use of elliptic curves in cryptography,” Advances in Cryptology-CRYPTO ’85, in Lecture Notes in Computer Science, vol. 18, Ed. Springer, Berlin, 1986, pp. 417-426.

[8] N. Koblitz, “Elliptic curve cryptosystems,” Math. Comput. vol 48, pp. 203–209, Jan. 1987.

[9] Standards for Efficient Cryptography, “SEC 2: Recommended Elliptic Curve Domain Parameters, “ https://www.secg.org/sec2-v2.pdf , Jan. 2010.

[10] NIST–DSS, “Digital Signature Standard (DSS),” FIPS Publication 186-2, National Institute of Standards and Technology, Jan. 2000.

[11] T. Elgamal, “A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” IEEE Trans. Inf. Theory, vol. 31, pp. 469-472, Jul. 1985.

[12] K. Moriarty, B. Kaliski, J. Jonsson and A. Rusch, “PKCS #1: RSA Cryptography Specifications Version 2.2, “RSA Laboratories’ Public-Key Cryptography Standards (PKCS) series (RFC8017), Nov. 2016.

[13] The Legion of the Bouncy Castle [Online]. Available: https://www.bouncycastle.org/latest_releases.html (Accessed: Nov. 2021).

[14] G. L. Miller, "Riemann's Hypothesis and Tests for Primality," J. Comput. Syst. Sci., vol. 13, pp. 300-317, May 1975.

[15] M. O. Rabin, "Probabilistic algorithm for testing primality", J. Number Theory, vol. 12, pp. 128-138, Feb. 1980.