Comparative Study on the Performance of Elliptic Curve Cryptography Algorithms with Cryptography through RSA Algorithm

Nelson Josias G. Saho* — Eugène C. Ezin **

* École Doctorale des Sciences de l'Ingénieur
Université d'Abomey-Calavi
Calavi
BENIN
nelson.saho@uac.bj

** Institut de Formation et de Recherche en Informatique
Université d'Abomey-Calavi
Calavi
BENIN
eugene.ezin@uac.bj

ABSTRACT. Encryption method is an effective way to guarantee the confidentiality and in other ways (digital signature) the authenticity of data. The use of the digital signature derives from the methods of asymmetric cryptography. It appears as credible alternatives to guarantee the authenticity, non forgery, non reuse, inalterability and irrevocability of data. Data encryption and electronic signature by elliptic curve cryptography are now widespread and it is important to highlight the comparative advantages. In this paper, we evaluated the performance of asymmetric cryptography through elliptic curve cryptography versus that with the RSA algorithm. Then, we achieved some cryptosystems by using elliptic curve cryptography protocols : ECNR, ECDSA, ECIES, and RSA. Therefore, we perform tests that showed most of the elliptic curve cryptography algorithms are more advantageous in terms of memory consumption and computing speed over the RSA cryptosystem.

RÉSUMÉ. Si le chiffrement est un moyen efficace de garantir la confidentialité, la signature numérique assure quant à elle l'authenticité des données. Son utilisation découle des procédés de la cryptographie asymétrique. Elle est une alternative crédible pour garantir l'authenticité, la non falsification, la non réutilisation, l'inaltérabilité et l'irrévocabilité des données. Le chiffrement des données et la signature électronique par la cryptographie sur les courbes elliptiques sont maintenant très répandus et il est important d'en saisir les différents avantages. Dans cet article, nous avons évalué les performances de la cryptographie asymétrique sur les courbes elliptiques par rapport à celle de l'algorithme RSA. Les cryptosystèmes élaborés à cet effet (en utilisant les protocoles ci-après : ECNR, ECDSA, ECIES et RSA) nous ont permis d'effectuer des tests qui ont montré que la plupart des algorithmes de cryptographie à courbe elliptique sont plus avantageux en termes de consommation de mémoire et de vitesse de calcul que le cryptosystème RSA.

KEYWORDS : Asymmetric cryptography, Elliptic curve cryptography, RSA algorithm, Encryption sheme, Digital signature

MOTS-CLÉS : Cryptographie asymétrique, La cryptographie sur les courbes elliptiques, Algorithme RSA, Méthode de chiffrement, Signature numérique

1. Introduction

It's no longer a secret in our current ecosystem: data is an essential part of the business. Integrated into the heart of the strategic process of any enterprise, they are the object of paramount importance in any decidability. The acceleration of the digital transformation and, at the same time, the development of the Big Data, are pushing companies to find the best management system for managing these volumes of data. In this digital environment, securing data is becoming a major issue. For any business owner or individual (depending on the nature of data), it's a good idea to find the best way to keep their data safe. To guarantee this safety i.e data integrity, authenticity and confidentiality, one way is to encrypt and/or sign them before their delivering process. Digital signature is based upon asymmetric encryption algorithm methods [1].

Since the 1978's, the RSA algorithm, proposed by Rivest, Shamir, and Adleman [2], has been the most widely used algorithm for both encrypting data and digitally signing them. But it presents nowadays some drawbacks, such as large key lengths are required to increase such a cryptosystem security. Accordingly the computational time is high. Elliptic curve cryptography (ECC) corrects this disadvantage by using short key size to ensure similar security. Elliptic curve cryptography algorithms seem to fit well with public key cryptography by replacing the calculation on numbers (in the case of RSA) by the calculation on groups associated with the elliptic curve. So it most difficult to break the digital signature through elliptic curves because solving the problem of the discrete logarithm on the elliptic curve group appears to be more difficult than factoring big number into prime numbers.

According to the advantages of elliptic curve cryptography, we propose to compare the performance of elliptic curve cryptography algorithms with the RSA algorithm to know the best way for everyone to secure more their data. The paper is organized as follows. In section 2 we will present the fundamental concepts including encryption scheme, digital signature, cryptography based on elliptic curves, the materials and methods used. In Section 3 we will describe the different results we have achieved. A discussion and an analysis of these findings will be presented in Section 4. We ended this paper with concluding remarks presented in Section 5.

2. Fundamental concepts

In this section we will present the concepts of encryption scheme and digital signatures and those of cryptography based on elliptic curves so-called elliptic curve cryptography (ECC).

2.1. Overview of asymmetric cryptography

Asymmetric cryptography, also known as public key cryptography, uses public and private keys to encrypt and decrypt data. The keys are simply large numbers that have been paired together but are not identical (asymmetric). One key in the pair can be shared with everyone; it is called the public key. The other key in the pair is kept secret; it is called the private key. Either of the keys can be used to encrypt a message; the opposite key from the one used to encrypt the message is used for decryption. Some of the examples for asymmetric key cryptosystem are RSA, ELGAMAL, and ECC etc [3].

A digital signature is based on asymmetric cryptography and guarantees the authenticity of an electronic document or message in digital communication and uses encryption techniques to provide proof of original and unmodified documentation. It is therefore, a reliable engagement mechanism [4]. A digital signature has the following characteristics: authenticity, non forgery, non reusable, inalterability, irrevocability.

Assume Alice wants to send a message to Bob. The latter must be able to verify the authenticity of such a message. The message m is assumed to be the binary file with contents such as text, image, executable file, etc. We propose to describe the classical method of digital signature into four steps: setting up signature architecture, preparing the signed message, receiving the signed message and the use of hash function.

2.1.1. Setting architecture of digital signature establishment

Alice and Bob have agreed on the following choices:

– an asymmetric encryption consisting of an encryption function C and a decryption function D;

– a hash function we denote H;

– an encryption method in which Alice generates a private key K_{pr} and a public key K_{pb}. She transmits the public key K_{pb} to Bob by a not necessary secured channel and keeps the private key K_{pr} secret.

2.1.2. Preparing signed message

Alice prepares the signed message in the following manner:

– she produces a condensate of the message m in using the chosen hash function [5] $H(m)$;

– she encrypts this condensate with the encryption function C using her private key K_{pr}. The result obtained is the signature of the message we denote S_m which is defined by:

$$S_m = C(K_{pr}, H(m));$$

– she prepares the signed message by placing the clear message m and the signature S_m in any container. The signed message is defined by:

$$m_{signed} = (S_m, m).$$

Alice then sends m_{signed} to Bob whatever the communication channel used (secured or not).

2.1.3. Receiving signed message

Once Bob received the signed message, to verify its authenticity he proceeds as follows:

– he produces a condensate of the plaintext by using the agreed hash function $H(m)$;

– he decrypts the signature using the decryption function D with Alice public key K_{pb}. The decrypted message is $D_{Sm} = D(K_{pb}, S_m)$;

– he compares D_{Sm} with $H(m)$.

If the signature is authentic, D_{Sm} and $H(m)$ will be equals due to the property of the asymmetric encryption:

$$\begin{aligned} D_{Sm} &= D(K_{pb}, S_m) \\ &= D(K_{pb}, C(K_{pr}, H(m))) \\ &= H(m) \end{aligned} \tag{1}$$

If (1) is satisfied then the message will be authenticated.

2.2. Overview on elliptic curve cryptography

ECC is independently proposed by Neal Koblitz and Victor Saul Miller in 1985 [6, 7].

To explain the functioning of ECC, we first present the mathematical concepts of the elliptic curves, as well as the most important operation on curves i.e. the scalar multiplication. We then present the set of cryptographic protocols that are based on elliptic curves.

2.2.1. Short overview on elliptic curves

Yanbo Shou in his thesis [8] showed that Weierstrass' equation of an elliptic curve can be simplified over a prime finite field $\mathbb{F}_p$ and becomes:

$$E : y^2 = x^3 + ax + b, \tag{2}$$

where a et $b \in \mathbb{F}_p$.

The set of elliptic curves being an additive group, it is important to remember the addition of two points and the multiplication of a point by a scalar.

2.2.1.1. Addition of points

Let us consider an elliptic curve E of (2) and two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ belonging to this curve. The addition of P_1 and P_2 is the point $P_3(x_3, y_3)$ of E ($P_3 = P_1 + P_2$) defined as Y. Shou illustrates it in [8].

Fig. 1 illustrates the addition of two points P and Q ($R = P + Q$) on the elliptic curve defined by:

$$y^2 = x^3 - 2x + 1.$$

In that figure (Fig. 1), the line that goes through both P and Q intersects the elliptic curve in a third point which is the negation of the sum of P and Q.

2.2.1.2. Scalar multiplication

Based upon point addition, we can perform the multiplication, denoted by $Q = kP$ on an elliptic curve E where $k \in \mathbb{Z}^+$ and $(P, Q) \in E^2$. Scalar multiplication is in fact a sequence of addition of points [9]:

$$Q = k\,P = \underbrace{P + P + \cdots + P}_{\text{k times}} \tag{3}$$

2.2.2. Elliptic curve cryptography protocols

Below are the main cryptographic protocols which are based on the theory of elliptic curves.

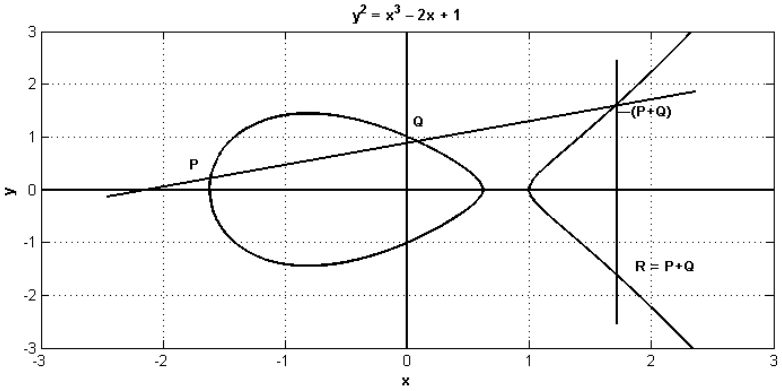

Figure 1. *Addition of points on elliptic curves.*

Table 1. *The key lengths of different algorithm.*

Algorithms	Key lengths (bits)				
Key lengths label	1	2	3	4	5
Symmetric	80	112	128	192	256
ECC	163	233	283	409	571
RSA	1 024	2 240	3072	7 680	15 360

– Diffie-Hellman key exchange protocol [10];

– Elliptic Curve Integrated Encryption Scheme (ECIES) [11]

– El Gamal method [12];

– Elliptic Curve Nyberg-Rueppel (ECNR) [13]

– Elliptic Curve Digital Signature Algorithm (ECDSA) [14];

– Elliptic Curve Menezes-Qu-Vanstone (ECMQV) [15].

The resistance of ECC is related to the problem of the discrete logarithm on the corresponding group for the elliptic curve. ECC is an alternative to conventional public key cryptography.

In section 3 we will make a comparative study between an algorithm based on elliptic curves and RSA algorithm. First, let us present the used materials and methods.

2.3. Materials and methods

Arjen K. Lenstra and Eric R. Verheul achieved a work identifying equivalent key lengths, for three algorithms (symmetric, RSA and ECC algorithms) [16], which can provide the same level of robustness. For better understanding, key lengths in the same column are expected to provide the same level of robustness (see Table 1). Considering key lengths label 2 in Table 1, a key of a symmetric algorithm of 112 bits would produce the same robustness as that of an asymmetric algorithm ECC of 233 bits and that of RSA of 2240 bits.

The computer used by A. Lenstra et al. is characterized by a 2.0 GHz Intel processor and a RAM of 512 MB [16]. Results are available at the URL[1].

1. *http://www.keylength.com/fr/1/*. Last accessed June, 24th 2019.

We will set up cryptosystems with elliptic curve algorithm and RSA algorithm by using equivalent key lengths obtained by Lenstra et al. to evaluate the performance of these types of algorithms for both encryption and digital signature.

3. Simulation results

We have firstly presented the encryption scheme with the RSA algorithm and one with the elliptic curve algorithm i.e. the ECIES. Secondly, we have given the results for the digital signature. Then, we have compared these results to determine the best way to secure documents.

3.1. Encrytion scheme

Two steps are in concern when dealing with encryption scheme which are encryption and decryption processes.

3.1.1. ECIES encrytion

The ECIES protocol is indeed a standardized version of Elgamal. Suppose that Alice wants to send a message m to Bob in a secure way, they must first have all the following information:

– KFC: key derivation function that allows to generate several keys from a reference secret value;

– MAC message authentication code transmitted with data to ensure its integrity;

– SYM: Symmetric encryption algorithm;

– $E(\mathbb{F}_p)$: elliptic curve used with the generator G whose $ord_p(G) = n$;

– K_B: public key of Bob $K_B = k_B\,G$ where $k_B \in [1, n-1]$ is his private key.

3.1.1.1. Encryption process

To encrypt the message m, Alice proceeds as follows:

1) Choose a random number k belonging to the interval $[1, n-1]$ and compute $R = k\,G$.

2) Calculate $Z = k\,K_B$.

3) Generate the keys k_1 and k_2 such that $(k_1, k_2) = KDF(abscisse(Z), R)$.

4) Encrypt the message m by doing $C = SYM(k_1, m)$.

5) Generate the MAC code $t = MAC(k_2, C)$.

6) Send (R, C, t) to Bob.

3.1.1.2. Decryption process

To decrypt the message (R, C, t), Bob must perform the following calculations:

1) Verify if $R \in E(\mathbb{F}_p)$. Otherwise reject the message.

2) Compute $Z = k_B\,R = k_B\,k\,G = k\,K_B$.

3) Generate the keys k_1 and k_2 such as $(k_1, k_2) = KDF(abscisse(Z), R)$.

4) Generate the MAC code $t' = MAC(k_2, C)$.

5) Verify if $t = t'$. Otherwise reject the message.

6) Decrypt the message by computing $M = SYM^{-1}(k_1, C)$.

3.1.2. RSA encrytion

We have firstly to generate the RSA keys. The key generation process in RSA algorithm is as follows:

1) Choose two distinct prime numbers p and q such as the number of bits of the two integers is approximately equal.

2) Compute their product $n = p \times q$ called the encryption module.

3) Compute $\varphi(n) = (p-1)(q-1)$ the value of the Euler indicator in n.

4) Choose an integer e which is the encryption exponent such as $e \in]1, \varphi(n)[$ and $gcd(e, \varphi(n)) = 1$.

5) Calculate the deciphering exponent d such as $e\,d \equiv 1(modulus\ n)$.

The pair (n, e) is the public key of the encryption whereas the number d is the corresponding private key.

3.1.2.1. Encryption process

After Bob obtains Alice's public key, he can encrypt the message m and then sends it to Alice. He computes the ciphertext C, using Alice's public key e, corresponding to:

$$C \equiv m^e\ (modulus\ n). \tag{4}$$

Bob then transmits C to Alice.

3.1.2.2. Decryption process

Alice can recover m from C by using her private key (exponent d) by computing D such that:

$$\begin{aligned} D &\equiv C^d\ (modulus\ n) \\ &\equiv [(m)^e]^d\ (modulus\ n) \\ &\equiv m. \end{aligned} \tag{5}$$

This can be done reasonably quickly, even for very large numbers, using modular exponentiation.

3.2. Digital signature

3.2.1. Digital signature with RSA algorithm

Three steps are in concern when dealing with digital signature embedded in RSA algorithm which are key generation, signature generation and signature verification.

3.2.1.1. RSA key generation

The key generation using RSA algorithm is the same as described in subsection 3.1.2

3.2.1.2. RSA signature generation

To sign a message m with RSA, it is sufficient to encrypt the digest of the message with the private key d. The signed message becomes:

$$s \equiv [hash(m)]^d\ (modulus\ n), \tag{6}$$

where $hash$ is a hash function. For the purpose of this paper we use $SHA-256$ as recommended by FIPS[2] 180-4 [20]. Once the message m is signed, the sender will transmit it with its signature to the recipient.

3.2.1.3. RSA signature verification

To verify the received signature, the recipient must decrypt the signature with the sender public key according to the following relation:

$$\begin{aligned} h &\equiv s^e \ (modulus\ n) \\ &\equiv [hash(m)^d]^e \ (modulus\ n) \\ &\equiv hash(m). \end{aligned} \tag{7}$$

To authenticate the author of the message, the recipient applies the same $hash$ function to the message m and checks whether the result is equal to the previously calculated value which is h.

3.2.2. Digital signature with elliptic curves

As we previously presented, there are several algorithms that implement the digital signature with elliptic curves. We will use these algorithms for the performance study. Let us present the EDCSA algorithm.

3.2.2.1. ECDSA key generation

We consider the elliptic curve retained in (2). To generate the keys using the elliptic curves, one proceeds as follows:

1) Select a number x, strictly smaller than the order n of the curve.

2) Compute P such as $P = x\,G$, G being the generator of the curve.

Thus, our key pair is (P, x) where P is the public key and x the private key. Let us recall that P as well as the parameters n, a, b and G must be published.

3.2.2.2. Generating an ECDSA signature

To create the signature S of a message m, one proceeds as follows:

1) Choose a random number k belonging to the interval $[1, n-1]$.

2) Calculate $R(x_1, y_1)$ such as $R = k\,G$.

3) Compute $r = x_1 \ (modulus\ n)$. If $r = 0$ then return to the step i.

4) Calculate $k^{-1} \ (modulus\ n)$.

5) Calculate $SHA-1(m)$, and convert it into an integer e (instead of the SHA-1 one may use SHA-256 or SHA-512).

6) Calculate $s = k^{-1}(e + xr) \ (modulus\ n)$ where x is the private key. If $s = 0$ then return to the step i.

The signature S of the message m is the pair (r, s).

3.2.2.3. ECDSA signature verification

To verify the signature $S = (r, s)$ of the message m using the public parameters, we

2. Federal Information Processing Standards

Table 2. *Running time for key generation.*

Key lengths (bits)		Computational time (sec)	
ECC	RSA	ECC	RSA
163	1 024	0.219	0.958
233	2 240	0.257	1.615
283	3 072	0.197	6.122
409	7 680	0.224	162.357
571	15 360	0.156	2029.909

proceed as follows:

1) Check if r and s are integers within the interval $[1, n-1]$.

2) Calculate $SHA-1(m)$, and convert it into an integer e.

3) Calculate $w = s^{-1}\ (modulus\ n)$.

4) Calculate $u_1 = e\,w\ (modulus\ n)$ and $u_2 = r\,w\ (modulus\ n)$.

5) Calculate $X(x_1, y_1)$ such as $X = u_1 G + u_2 P$.

6) If $X = 0$ then S is not valid. Otherwise, calculate $v = x_1\ (modulus\ n)$.

The signature S of the message m is valid if $v = r$.

3.3. Performance comparison

In this subsection we make the comparison between RSA and ECIES algorithms for the encryption scheme, ECDSA and ECNR algorithms over RSA algorithms for digital signature. The comparison is done in terms of execution time of each algorithm. The machine on which the various tests are performed has a dual core processor of 2.4 GHz and equipped with 2 GB RAM. As other processes running on the computer, the computation time retained in this paper is the average time after a certain number of execution of the different cryptosystems.

3.3.1. Encryption scheme

For this process, we have two steps: encryption and decryption. But we have firstly to generate a pair of key. The comparison in terms of execution time is also done according to the same steps. We then wrote two programs in Java[19] that implement these algorithms. We have encrypted one text file of 4 Ko and after we have decrypted it. Then, we can calculate the computational time of each step.

3.3.1.1. Key generation

The results, in terms of computational time, at the level of key generation are reported in Table 2.

We found that the key generation by the ECIES algorithm is much faster than the key generation in RSA algorithm. While this generation time increases linearly with an almost zero slope, it exponentially increases in case of RSA. Therefore, from a key generation point of view, the ECC-based algorithm is much more optimal.

3.3.1.2. Encryption process

The results, in terms of computational time, at the level of encryption using RSA and ECIES algorithms are reported in Table 3.

Table 3. *Running time for encryption.*

Key lengths (bits)		Computational time (sec)	
ECC	RSA	ECC	RSA
163	1 024	1.147	0.063
233	2 240	1.265	0.031
283	3 072	1.395	0.031
409	7 680	1.375	0.031
571	15 360	1.265	0.015

Table 4. *Running time for decryption.*

Key lengths (bits)		Computational time (sec)	
ECC	RSA	ECC	RSA
163	1 024	0.015	0.062
233	2 240	0.029	0.032
283	3 072	0.019	0.047
409	7 680	0.026	0.375
571	15 360	0.047	2.66

We noted that the encryption by the RSA algorithm is faster than the one with ECC algorithm. But the gap between the two computational times is not so great. Therefore, from an encryption point of view, we can conclude that the two algorithms present the same performance but RSA-based algorithm performs better.

3.3.1.3. Decryption process

The results, in terms of computational time, at the level of decryption are reported in Table 4.

We obtain opposite results in comparison with those obtained during the encryption step. The decryption by the ECC algorithm is more faster than the one with RSA.Therefore, from an decryption point of view, we can conclude that ECC-based algorithm performs better.

3.3.2. Digital signature

As previously elucidated in the description of algorithms, we have three main steps in their implementation namely the key generation, the signature generation and the signature verification. We will use two common algorithms of elliptic curves – ECDSA and ECNR – and compare their performance with RSA-based algorithm.

The comparison in terms of execution time is also done according to the same steps. We then wrote three programs in Java using the same library JCE [19] to implement these algorithms.

3.3.2.1. Key generation

The results, in terms of computational time, at the level of key generation are reported in Table 5.

We found that the key generation by the ECC algorithm is much faster than the key generation in RSA algorithm. The performance between ECDSA and ECNR is not very

Table 5. *Running time for key generation.*

Key lengths (bits)		Computational time (sec)		
ECC	RSA	ECC		RSA
		ECDSA	ECNR	
163	1 024	0.466	0.528	0.958
233	2 240	0.337	0.65	1.615
283	3 072	0.318	0.579	6.122
409	7 680	0.374	0.538	162.357
571	15 360	0.317	0.559	2029.909

Table 6. *Signature generation execution time.*

Key lengths (bits)		Computational time (sec)		
ECC	RSA	ECC		RSA
		ECDSA	ECNR	
163	1 024	0.009	0.057	0.036
233	2 240	0.01	0.061	0.037
283	3 072	0.01	0.060	0.067
409	7 680	0.014	0.067	0.479
571	15 360	0.018	0.083	3.015

different. As we noted in 3.3.1 during key generation, while this generation time increases linearly with an almost zero slope, it exponentially increases in case of RSA. Therefore, from a key generation point of view, the ECC-based algorithm is much more optimal.

3.3.2.2. Signature generation

The results at the level of the signature generation are reported in Table 6.

The Table 6 shows that ECDSA performs better than ECNR and RSA algorithms. When the length of the ECNR key is greater than 409 bits, the signature generation time of this algorithm is not better than its equivalent in terms of security with RSA. Elsewhere, the RSA algorithm is a little faster than the ECCNR algorithm. In any case, the performance of the two algorithms is not too different up to a given key length (409 bits) and beyond, ECNR becomes more efficient. In fine, we have understood that ECC performs better than RSA in terms of signature generation.

3.3.2.3. Signature verification

The results in terms of computational time at the level of the signature verification are presented in the Table 7.

They all have almost the same performance except that the RSA results are better than those of ECC (ECDSA and ECNR). For signature verification, the RSA algorithm becomes much more efficient than ECC since it performs a modular exponentiation.

Table 7. *Execution time of signature verification.*

Key lengths (bits)		Computational time (sec)		
ECC	RSA	ECC		RSA
		ECDSA	ECNR	
163	1 024	0.0053	0.047	0.007
233	2 240	0.005	0.069	0.004
283	3 072	0.008	0.049	0.006
409	7 680	0.017	0.054	0.01
571	15 360	0.025	0.056	0.015

4. Discussion and analysis

We observed that memory consumption is much lower with the ECC algorithm (see Table 1), since shorter keys are allowed. Secondly, the key generation and message signature computation are faster with the ECC algorithm than the RSA algorithm. To evaluate and to compare the performance of the RSA and ECC algorithms, Nicholas Jansma and Brandon Arrendondo proposed two (02) implementations using respectively two cryptosystems [21]. They concluded that ECC can have the same level of security as RSA with a much shorter key.

By analyzing the suggested key length in [16] (see Table 1), it emerges that:

– Firstly, symmetric algorithms use a small key size to ensure the same level of security. For instance, the key of 256 bits length is enough for the symmetric algorithm, to reach the same level of security for the RSA algorithm with the key length of $15\,360$ bits. It means that the key length of RSA is 60 times the key length of the symmetric algorithm. But the difficulty in these algorithms lies in the preservation of the secret key.

– Secondly, the ECC algorithm requires fewer bits in terms of key length than the RSA algorithm to provide the same level of security. However, it requires about twice the key length of the symmetric algorithm to offer the same robustness.

R. Sinha et al. achieved the paper "Performance Based Comparison Study of RSA and Elliptic Curve Cryptography" [17] in which they analyzed the results obtained by Nicholas Jansma and Brandon Arrendondo. In this work, we performed by ourself with the nowaday ressources the cryptosystem to identify the best way to secure more data.

Due to the major drawback of the symmetric algorithms, we therefore concluded the algorithms implementing the elliptic curves are less costly in terms of key length. This is the main reason why the ECC algorithm is increasingly becoming the preferred choice for embedded systems with very limited memory and computing power.

Considering Bandwidth saving, ECC offers considerable bandwidth savings over RSA and considering computational overheads, ECC offers Roughly 10 times than that of RSA can be saved [18]. Considering key sizes, System parameters and key pairs are shorter for the ECC than RSA and After the different results obtained at the level of the encryption and decryption and those obtained for the digital signature, we have deduced that the elliptic curve algorithms are more efficient than the one based on the RSA. Also, we found that the digital signature ECDSA is more efficient than the ECNR. This is why among the algorithms based on the elliptic curves, the ECDSA algorithm was adopted and published as an international standard in ANSI X9.62[3].

3. Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)

5. Concluding remarks

We evaluated the performance of asymmetric cryptography through elliptic curve cryptography versus that with the RSA algorithm. Then, we achieved RSA cryptosystems and some others by using elliptic curve cryptography protocols – Elliptic Curve Nyberg-Rueppel (ECNR), Elliptic Curve Digital Signature Algorithm (ECDSA) and Elliptic Curve Integrated Encryption Scheme (ECIES).

Systems based upon elliptic curves are an effective alternative to the RSA cryptosystems since they involved different mathematical approaches. Elliptic curve cryptosystems are reputed for robustness equivalent to RSA cryptosystems with shorter key length. Accordingly, elliptic curve cryptosystems are perfectly suitable for embedded systems, e.g., smart cards, documents in which memory and power of the processors are not sufficient to achieve computation as required by RSA cryptosystems.

However, ECC and RSA cryptosystems involved the use of keys for security. We will in our future work investigate this drawback by developing a hybrid blockchain.

6. References

[1] W. M. DALEY, R. G. KAMMER, "Digital signature standard (dss), Technical report ", *DTIC Document*, 2000.

[2] R. RIVEST, A. SHAMIR, L. ADLEMAN, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, vol. 21 num. 2, pp. 120–126, 1978.

[3] S. B. SASI, D. DIXON, J. WILSON, "A General Comparison of Symmetric and Asymmetric Cryptosystems for WSNs and an Overview of Location Based Encryption Technique for Improving Security", *Journal of Engineering (IOSRJEN)*, vol. 04, 2014.

[4] TECHOPEDIA DICTIONARY, *https://www.techopedia.com/definition/5426/digital-signature*. Last accessed sep 17^th 2019

[5] P. ROGAWAY, T. SHRIMPTON, "Cryptographic hash-function basics: Definitions, implications and separations for preimage resistance, second-preimage resistance, and collision resistance", 2004.

[6] D. R. HANKERSON, S. A. VANSTONE, A. J. Menezes, "Guide to elliptic curve cryptography", *Springer-Verlag New York Inc*, 2004.

[7] N. KOBLITZ, "Elliptic curve cryptosystems", *Mathematics of computation*, pp. 203–209, 1987.

[8] Y. SHOU, "Cryptographie sur les courbes elliptiques et tolérance aux pannes dans les réseaux de capteurs", UFC, pp. 27–29, 2014.

[9] M. JOYE, "Introduction élémentaire à la théorie des courbes elliptiques", *UCL Crypto Group Technical Report Series*, 1995. Available at *http://sciences.ows.ch/mathematiques/CourbesElliptiques.pdf*

[10] W. DIFFIE, M. HELLMAN, "New directions in cryptography", *IEEE Transactions on Information Theory*, vol. 22, num. 6, pp. 644-–654, 1976.

[11] V. G. MARTÍNEZ, L. H. ENCINAS, C. S. ÁVILA, "A Comparison of the Standardized Versions of ECIES", *Proceedings of the Sixth International Conference on Information Assurance and Security – IAS 2010*, Atlanta, 2010.

[12] T. ELGAMAL, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *Crypto*, Springer, 1984.

[13] G. ATENIESE, B. DE MEDEIROS, "A Provably Secure Nyberg-Rueppel Signature Variant with Applications", *Proceedings of Advances in Cryptology – ASIACRYPT* 2004. Revised version: *https://eprint.iacr.org/2004/093.pdf*

[14] L. BASSHAM, D. JOHNSON, T. Polk, "Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public Key Infrastructure Certificates", *Internet Draft*, 1999. Available at *http://www.ietf.org*

[15] CRYPTO++ PAGE, *https://www.cryptopp.com/wiki/Elliptic_Curve_Menezes-Qu-Vanstone*. Last accessed 20 Apr. 2019

[16] A. LENSTRA, E. VERHEUL, "Selecting cryptographic key sizes", *Journal of cryptology*, vol. 14, pp. 255-–293, 2001. Available at *http://infoscience.epfl.ch/record/164526/files/NPDF-22.pdf*

[17] R. SINHA, H. K. SRIVASTAVA, S. GUPTA, "Performance Based Comparison Study of RSA and Elliptic Curve Cryptography", *International Journal of Scientific & Engineering Research*, vol. 2, num. 5, pp. 720-–725, 2013.

[18] V.B. KUTE, P R PARADHI, G. BAMNOTE, "A software comparison of RSA and ECC", *International Journal Of Computer Science And Applications*, vol. 2, num. 1, pp. 61-–64, 2009.

[19] J.-M. DOUDOUX, "Développons en Java, JCE (Java Cryptography Extension)". Available at *https://www.jmdoudoux.fr/java/dej/chap-jce.htm*. Last accessed Jul 31^{st} 2019

[20] FIPS 180-4, "The Secure Hash Standard", Available at *https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf*. Last accessed Sep. 17^{th} 2019

[21] N. JANSMA, B. ARRENDONDO, "Performance comparison of elliptic curve and rsa digital signatures", 2004.

[22] N. GURA, A. PATEL, A. WANDER, H. Eberle, S. SHANTZ, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs", *Marc Joye and Jean-Jacques Quisquater, Cryptographic Hardware and Embedded Systems - CHES 2004*, vol. 3156 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 119–132, 2004.