ISSN No:-2456-2165

Survey on Asymmetric Cryptographic Algorithms in Embedded Systems

Nelson Josias G. Saho
École Doctorale des Sciences de l'Ingénieur
Université d'Abomey-Calavi
Benin
nelson.saho@uac.bj

Eugène C. Ezin
Institut de Formation et de Recherche en Informatique
Université d'Abomey-Calavi
Benin
eugene.ezin@uac.bj

***Abstract*—Embedded systems spread out daily activities, both wireless sensors and mobile equipments that collect and process data for multiple purposes. Although data security on computers and servers appears to be well controlled, securing them for embedded systems with limited resources (e.g. memory, computational power, etc.) is a dare. In this paper, we investigated on the asymmetric cryptographic algorithms in embedded systems. We evaluated the performance of asymmetric cryptography algorithms through Elliptic Curve Cryptography (ECC) versus the RSA algorithm. Specifically, we investigated on some cryptosystems by using ECC protocols such as Elliptic Curve Integrated Encryption Scheme (ECIES), Elliptic Curve Digital Signature Algorithm (ECDSA), Elliptic Curve Nyberg-Rueppel (ECNR) and RSA. We then performed many tests that shown ECC algorithms are more advantageous in terms of computing speed and memory consumption over the RSA algorithm. We ended that ECC-based cryptosystems offer an attractive alternative to the classical asymmetric cryptography like RSA, as it uses a short key size to achieve an equivalent level of security. This makes it an attractive and efficient alternative for deployment in embedded systems.**

***Keywords*—Asymmetric cryptography algorithms, Elliptic curve cryptography, RSA algorithm, Embedded Systems, encryption scheme, digital signature**

I. Introduction

The increasing use of the Internet and the world wide web (www) for daily activities and the proliferation of data generation sources generate new threats to privacy. Web services, whether they are passive or active, in cyberspace can give others a great deal of information [1].

Both personal and business users are likely victims of these threats. How to prevent these potential threats is a major concern in this era of acceleration of digital transformation and, at the same time, the development of Big Data. For Tseng et al., the increasing popularity of digital media has ushered in concern over related issues [2]. Typically, the document confidentiality is achieved by encryption.

Since the tools used by malicious users increasingly fussy, data protection becomes more and more important. Whether preventing unauthorized access to personal data, or ensuring the integrity of corporate secrets, all applications require increased security to protect data from talented intruders [3]. Therefore securing data is becoming a major issue.

Moreover, nowadays, our life is pervaded by computer systems embedded inside many products. These embedded systems are found in almost everything today, from cars to robotic materials, embedded biomedical devices, mobile equipments or wireless sensors handling. In embedded biomedical devices and robotic materials handling, any subversion or denial of service could result in loss of human life ... [4]. These systems are becoming increasingly sophisticated and interconnected, both to each other and to the Internet [5]. Securing these data becomes a major problem given the multiplicity of their use.

For any user (personal or business), it is judicious to find the best way to keep their data safe, i.e. ensure data integrity, authenticity and confidentiality. Cryptography helps to provide these triple function. What is the judicious manner with current hardware and software available on markets to ensure data security especially with embedded systems?

Many asymmetric cryptography algorithms such as RSA algorithm [6], has been used for both encrypting data and digitally signing them. But one of the drawbacks for classical asymmetric cryptosystems is the use of large key lengths is required to increase such a cryptosystem security. On the other hand, protocol based on Elliptic Curve Cryptography (ECC) decrease the key length while providing securities at the same level as that of other cryptosystems provides [7]. In this paper, we propose to explore different asymmetric cryptography algorithms, to compare their performance to know the best way to more secure data in embedded systems.

This paper is organized as follows. In section two we will present the fundamental concepts including the materials and methods used. In Section three we will present the different results we have achieved. In Section four, a discussion and an analysis of them will be presented. We will end this paper with concluding remarks presented in Section five.

II. Fundamental concepts

To introduce this paper in Section one, we talked about asymmetric cryptography algorithms and also discussed elliptic curve cryptography. So, in this section we will present the concept of asymmetric cryptography, the two processes based on it, namely: encryption scheme and digital signature. We will finish by presenting ECC.

A. Overview of asymmetric cryptography

Asymmetric cryptography, also called public key cryptography, uses a pair of related keys to cipher and decipher data: one key is public and the second is private. The public key can be shared with everyone while the private key is kept secret. The purpose of the asymmetric cryptography is to protect a plaintext (message) from unauthorized access or use. Some of the examples for asymmetric key cryptosystem are RSA, El Gamal, and ECC [8].

The public key can be used to encrypt a message; the reversed key is used for decryption. The opposite operation is possible and is called digital signature. The latter ensures data authenticity and uses encryption mechanism to proof that data are not altered. It is therefore, a reliable engagement mechanism [9].

The benefits of public key cryptography include:

- the key distribution problem in case of symmetric cryptography is ruled out since there is no need to exchange our keys anymore;
- security of the cryptosystem is improved because the private keys are never sent to someone; and
- the possibly use of digital signature is guaranteed.

It also has drawback including:

- it is slower compared to symmetric cryptography; and
- if the private key is lost, a received message cannot be deciphered.

B. Overview of encryption scheme

Firstly, we have to generate the related keys: the private and the public keys. Assume that Bob wants to send to Alice encrypted message. Alice will generate the keys and will inform Bob about her public key. Let K_{pr} and K_{pb} respectively be the private and the public keys generated by Alice.

The plaintext Bob wants to send to Alice is encrypted into ciphertext by using Alice's public key, which will be decrypted back into plaintext by Alice with her private key. Then:

$$C = E_{K_{pb}}(P). \tag{1}$$

$$P = D_{K_{pr}}(C). \tag{2}$$

where P represents the plaintext, E the encryption method, D the decryption method and C the ciphertext.

(1) is computed by Bob and (2) is computed by Alice to recover the plaintext.

C. Overview on digital signature scheme

Digital signature is a reliable engagement mechanism [9]. It proves to a third party that a particular document has been approved by an entity [10]. A digital signature has the following characteristics: authenticity, non forgery, inalterability, non reusable and irrevocability.

Let us assume Alice wants to send a message m to Bob. Bob must be able to verify the authenticity of the message of Alice. The classical method of digital signature can be described in four steps [11]:

- setting up signature architecture;
- using of hash function;
- preparing signed message; and
- receiving signed message.

1) Digital signature establishment: To set up digital signature, they agreed on the following choices:

- an asymmetric encryption function C and a decryption function D;
- a hash function denoted by H;
- since Alice wants to send a message m to Bob, she generates the related keys: a private key K_{pr} and a public key K_{pb}; and
- she transmits the public key K_{pb} to Bob by a channel which cannot be necessarily secured, and keeps K_{pr} secret.

2) Hash function: A hash function is a one-way function with the message as input and a digest message as output [12]. The output is called *hash message* or *condensate*. A valid hash function will have to guarantee the impossibility to recover the original message from the hash.

Adding a one-way function into the process allows its improvement since its absence would duplicate not only the size of the signature but also increase the number of calculations to be performed [13].

The most widely used hash algorithms are :

- *Message Digest (MD5)*; It produced 128 bits as output from a file of arbitrary size by processing it into blocks of 512 bits.
- *Secure Hash Algorithm (SHA)*. It created a 160 bits stream from a message of up to 2^{64} bits in length by also processing it into blocks of 512 bits.

Figure 1 illustrates the process of signing a message applying a hash function.

3) Preparing signed message: Alice prepares the signed message as follows [13]:

- she produces a condensate of m as follows: $H(m)$;
- she encrypts $H(m)$ with the function C by using K_{pr}. The output is the signature of the message denoted S_m which is defined by:

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

- she sends the signed message by placing the clear message m and the signature S_m in any container.

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

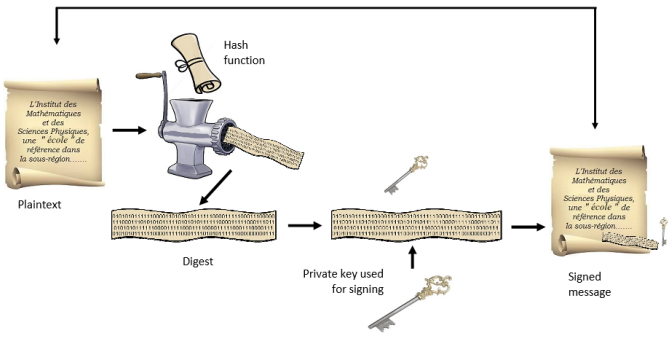

Fig. 1. Process of signing a message with hash function [13].

4) Receiving signed message: Once Bob received m_{signed}, to verify the message authenticity he proceeds as follows [13]:

- he computes the digest of the plaintext by using the agreed hash function H: $H(m)$;
- he decrypts the signature using the function D with Alice's public key K_{pb}. The decrypted message is $D_{Sm} = D(K_{pb}, S_m)$;
- he compares D_{Sm} with $H(m)$. The signature will be authentic if D_{Sm} and $H(m)$ equal.

Proof:

$$\begin{aligned} D_{Sm} &= D(K_{pb}, S_m) \\ &= D(K_{pb}, C(K_{pr}, H(m))) \\ &= H(m). \end{aligned} \tag{3}$$

■

D. Overview on elliptic curve cryptography

Elliptic Curve Cryptography (ECC) is independently proposed by Neal Koblitz and Victor Saul Miller in 1985 [14], [15]. It includes a set of techniques allowing to secure data by using less resources.

To explain the functioning of ECC, we will present firstly the mathematical concepts of them, as well as the scalar multiplication operation which is the most important operation on curves. After this, we will indicate the set of cryptographic protocols based on it.

1) Brief overview on elliptic curves: An elliptic curve E is an algebraic curve that can be represented by the Weierstrass' equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{4}$$

We assume that the parameters $a_1,\ a_2,\ a_3,\ a_4,\ a_6$ belong to a field K on which a curve is defined.

Two elliptic curves with different shapes are illustrated in Figure 2. The shape of the curve varies following to the chosen parameters.

Equation (4) can be simplified. Shou Yanbo in his thesis [16] showed that (4) becomes:

$$E : y^2 = x^3 + ax + b, \tag{5}$$

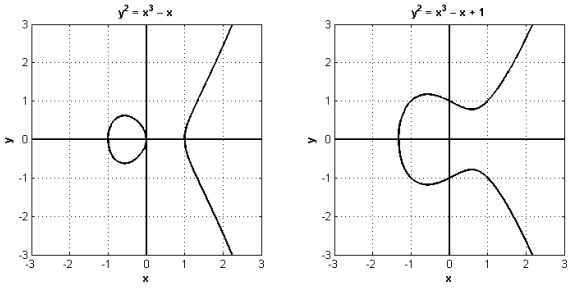

Fig. 2. Elliptic curve examples.

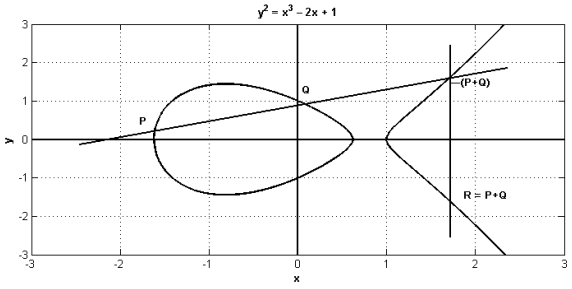

Fig. 3. Addition of points on elliptic curves.

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

It is necessary to remember how the addition of two points and the multiplication of a point by a scalar are carried out since the set of elliptic curves is an additive group.

a) Addition of points: Let us consider an elliptic curve E of (5) 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 follows:

i. If $x_1 \neq x_2$ then

$$\begin{cases} x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \\ y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1. \end{cases}$$

ii. If $x_1 = x_2$ but $y_1 \neq y_2$ then $P_3 = \infty$.

iii. If $P_1 = P_2$ *and* $y_1 \neq 0$ then

$$\begin{cases} x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \\ y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1. \end{cases}$$

iv. If $P_1 = P_2$ *and* $y_1 = 0$ then $P_3 = \infty$.

Furthermore $\forall\ P \in E,\ P + \infty = P$.

Figure 3 illustrates the addition of two points P and Q ($R = P + Q$) on the elliptic curve defined by $y^2 = x^3 - 2x + 1$.

b) Scalar multiplication: Based on addition of points, 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 [17]:

$$Q = k\,P = \underbrace{P + P + \cdots + P}_{\text{k times}}. \tag{6}$$

TABLE I
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

It is important to specify that ECC resistance is linked to the problem of the discrete logarithm on the group of the curve. ECC seems to be a credible alternative to conventional public key cryptography.

2) Elliptic curve cryptography protocols: Here are the main cryptographic protocols based on elliptic curves theory.

- Diffie-Hellman key exchange protocol [18]
- El Gamal method [19]
- Elliptic Curve Integrated Encryption Scheme (ECIES) [20]
- Elliptic Curve Menezes-Qu-Vanstone (ECMQV) [21]
- Elliptic Curve Digital Signature Algorithm (ECDSA) [22]
- Elliptic Curve Nyberg-Rueppel (ECNR) [23]

E. Materials and methods

Our goal is to explore different asymmetric cryptography algorithms, compare their performance to know the best way to further secure data in embedded systems. We will show why asymmetric cryptography and not symmetric cryptography should be more suitable for embedded systems since the latter is faster, and not greedy. We will survey the different asymmetric cryptography algorithms and perform a comparison study between them to identify which one seems better for embedded systems.

For the comparison study, we will use as inputs, the result of the work archived by Arjen K. Lenstra and Eric R. Verheul [24]. They have identified equivalent key lengths, for symmetric, RSA and ECC algorithms, which can provide the same level of robustness. For better understanding, key lengths in the same column (refer Table I) are expected to provide the same level of robustness. 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 for both encryption and digital signature.

The computer used by Lenstra et al. is equipped as follows [24]:

- 2.0 GHz Intel processor; and
- a RAM of 512 MB.

Their results are available at this URL[1].

III. SIMULATION RESULTS

We will first present the reasons for the preference of asymmetric cryptography over symmetric cryptography. Then, it will be presented asymmetric cryptography algorithms non based on elliptic curves before presenting the encryption scheme with the RSA algorithm and one with the elliptic curve algorithm i.e. the ECIES. Secondly, we will present the results of digital signature. Then, we will compare these results to determine the best way to secure data for embedded systems.

A. The preference of asymmetric cryptography

Let us recall that symmetric encryption uses a unique key that must be shared between the people who need to take part in communication. Unlike symmetric encryption, asymmetric encryption, as mentioned in subsection II-A, uses a pair of keys. Symmetric encryption is an old technique (the earliest known evidence of the use of cryptography was found in an inscription carved around 1900 BC in Egypt [25]).

Whether a beginner or a non-techie in the cryptography science, choosing an encryption algorithm to secure data can be a difficult task. This choice becomes more complex if he has to opt between symmetric or asymmetric encryption. We will discuss the major differences between these two encryption methods, which one is more secured or more suitable as needed.

We retain here four major differences between symmetric and asymmetric encryptions. Most of them are related to the key and the computational time.

Symmetric and asymmetric encryption are both very efficient depending on the task at hand, and this in different ways. Asymmetric encryption is the more secured one, while symmetric encryption is faster [26]. Either or both can be deployed alone or together.

The Table II below compares symmetric and asymmetric encryptions.

We cannot conclude that one is more adequate than the other because it is related to the need. Let us mention, however, that in a configuration where on-board equipment will have to give the guarantee that a datum collected or generated comes from itself, asymmetric cryptography is indicated because in this case the principles of the digital signature will be used. It is for this reason that in this paper to identify a cryptosystem that will ensure both the encryption and the digital signature, we focus our attention on asymmetric cryptosystems.

B. Survey on asymmetric cryptography

M. A. Al-Shabi discussed the most important algorithms used for the encryption and decryption process; he makes a comparative study for most important algorithms in terms of speed (implementation) and security (special keys) to determine whether an encryption algorithm is good [28]. Moreover, the computational resources, such a size of RAM memory, are an integral consideration since they affect the algorithm efficiency. So a consequent allocation of resources is necessary. He produced the result, part of which is recorded in the Table III.

By analysing this result, we have understood that the major asymmetric algorithms he considered, like other authors are:

- RSA published and developed by Ron Rivest et.al [6];

[1]http://www.keylength.com/fr/1/ visited on August, 24th 2020.

[2]Since 2015, National Institute of Standards and Technology (NIST) recommends a minimum of 2048-bit keys for RSA [27] an update to the widely-accepted recommendation of a 1024-bit minimum since at least 2002.

TABLE II
MAJOR DIFFERENCES BETWEEN SYMMETRIC AND ASYMMETRIC ENCRYPTION.

Difference element	Symmetric encryption	Asymmetric encryption
1. Used key	A single key is used for both encryption and decryption of message.	Two different keys, a public key and a private key, are used. The public key is used for encryption while the second is used for decryption.
2. Complexity and execution speed	It's a simple process. So the encryption process is faster.	It is more difficult than symmetric encryption, and its process is less quick.
3. Length of keys	A shorter key length than an equivalent asymmetric cryptosystem is used. The length of the used keys depends on security requirements.	The length of the keys is much larger than an equivalent symmetric cryptosystem. The recommended RSA key size is 2048 bits or higher[2].
4. Security	Since the secret key is shared, then its sharing mechanism is the flaw of symmetric cryptosystems.	The process is more secure than to symmetric encryption. There is nevertheless the use of the key which is the weakness of any cryptosystem, both symmetric and asymmetric.

TABLE III
COMPARATIVE ANALYSIS OF ASYMMETRIC CRYPTOGRAPHY [28].

Algorithms	Battery Consumption	Time Consumption	Block Size	Round	Structure	Attack
RSA	Low	Slowest	Variable	1	Public Key Algorithm	Cycle attack
Diffie Hellman	High	Medium	Variable	1	Festial & Substitution	Man in the middle
ECC	Medium	Fast	Variable	1	Public Key Algorithm	Side channel

- Diffie Hellman published and developed by Whitfield Diffie and Martin Hellman [18]; and
- ECC published and developed by by Neal Koblitz and Victor Miller [14], [15].

Diffie Hellman algorithm battery consumption is high. This is not indicated for the embedded systems. This bad parameter accumulated with the medium computational time will heat up the processor and thus affect the performance of the equipment. We therefore do not recommend for embedded systems the algorithms of Diffie Hellman. Then, our attention will be focused on RSA algorithm and that is based on elliptic curves. We will analyze more closely the performance of these two algorithms to retain which of them is useful for embedded systems.

This comparative study on the performance between elliptic curve cryptography algorithms over cryptography through RSA algorithm will be done for both encryption scheme and digital signature. After describing one algorithm in each case on which the comparison study focused, we will indicate the performance of each of them.

C. Description of encryption and digital signature algorithms

We will explain for the encryption scheme the algorithm of RSA and an algorithm of elliptic curves and for the digital signature we will do the same. So, for encrytion scheme, we will describe RSA encryption and Elliptic Curve Integrated Encryption Scheme (ECIES) and for the digital signature, we will describe RSA digital signature and Elliptic Curve Digital Signature Algorithm (ECDSA).

1) RSA encrytion: We have to generate the RSA keys first of all. It is processed as follows [13]:

i. choose two distinct prime numbers p and q such as the number of bits of the two integers is approximately equal;
ii. compute the encryption module $n = p \times q$;
iii. determine the value of the Euler indicator in n by computing $\varphi(n) = (p-1)(q-1)$;
iv. choose the encryption exponent, an integer e such as $e \in]1, \varphi(n)[$ and $gcd(e, \varphi(n)) = 1$; and
v. compute the deciphering exponent d such as $e\,d \equiv 1(modulus\ n)$.

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

a) Encryption process: Once Bob obtains Alice's public key, he can encrypt the plaintext m and then sends the cryptogram to Alice. He computes that cryptogram C using e, the public key of Alice, by doing:

$$C \equiv m^e \ (modulus\ n). \tag{7}$$

Then, Bob transmits the cyphertext C to Alice.

b) Decryption process: Alice can recover m from C by using the exponent d. She computes D such that:

$$D \equiv C^d \ (modulus\ n). \tag{8}$$

Proof: Let us recall that $C \equiv m^e \ (modulus\ n)$. Then,

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

■

2) ECIES encrytion: The ECIES protocol is indeed a standardized version of Elgamal [29]. Suppose Alice wants to send a message m to Bob in a secure way, they must agree on 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 $ord_p(G) = n$; and
- K_B: Bob's public key $K_B = k_B\,G$ where $k_B \in [1, n-1]$ is his private key.

a) Encryption process: To encrypt the message m, Alice proceeds as follows:

i. choose a random number k belonging to the interval $[1, n-1]$ and compute $R = k\,G$;
ii. compute $Z = k\,K_B$;
iii. generate the keys k_1 and k_2 such that $(k_1, k_2) = KDF(abscissa(Z), R)$;
iv. encrypt the message m by doing $C = SYM(k_1, m)$;
v. generate the MAC code $t = MAC(k_2, C)$; and
vi. send (R, C, t) to Bob.

b) Decryption process: To decrypt the message (R, C, t), Bob proceeds as follows:

i. verify if $R \in E(\mathbb{F}_p)$. Otherwise reject the message;
ii. compute $Z = k_B\,R$;

$$\begin{aligned} Z &= k_B\,R \\ &= k_B\,k\,G \\ &= k\,K_B. \end{aligned}$$

iii. generate the keys k_1 and k_2 such as $(k_1, k_2) = KDF(abscissa(Z), R)$;
iv. generate the MAC code $t' = MAC(k_2, C)$;
v. verify if $t = t'$. Otherwise reject the message; and then
vi. decrypt the message by computing $M = SYM^{-1}(k_1, C)$.

3) RSA Digital signature algorithm: Generally, there are three stages involved in digital signature processing which are key generation, signature generation and signature verification.

a) RSA key generation: The key generation using RSA algorithm is the same as described in subsection III-C

b) RSA signature generation: To sign any message m with RSA, it is sufficient to encrypt the hash with d, the private key. Let s be this signature.

$$s \equiv [hash(m)]^d \ (modulus\ n), \tag{9}$$

where $hash$ is a hash function.

Once the message m is signed, the sender will transmit it with its signature to the recipient. For the purpose of this paper we use $SHA-256$ as recommended by Federal Information Processing Standards (FIPS) 180-4 [30].

c) RSA signature verification: To verify if the received message is authentic, the recipient must decrypt the received signature with the public key of the sender according to the following relation:

$$h \equiv s^e \ (modulus\ n). \tag{10}$$

Proof: Let us recall that $s \equiv [hash(m)]^d \ (modulus\ n)$, and $e \times d = 1$.

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

■

To verify the authenticity the sender of the message, the recipient has to apply the retained hash function ($hash$) to the plaintext m and checks whether the previously calculated value (h) is equal to the current result.

4) ECDSA algorithm:

a) ECDSA key generation: Let us consider the retained elliptic curve in (5). For the keys generation, one proceeds as follows:

i. select a number x, strictly smaller than the order n of the curve; and
ii. 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.

b) Generating an ECDSA signature: Generating the ECDSA signature S of a message m is doing as follows:

i select a random number k belonging to the interval $[1, n-1]$;
ii compute $R(x_1, y_1)$ such as $R = k\,G$;
iii compute $r = x_1 \ (modulus\ n)$. If $r = 0$ then return to the step i;
iv compute $k^{-1} \ (modulus\ n)$;
v compute $SHA-1(m)$, and convert it into an integer e (instead of the SHA-1 one may use SHA-256 or SHA-512); and
vi compute $s = k^{-1}(e + x\,r) \ (modulus\ n)$. If $s = 0$ then go to the step i.

TABLE IV
RUNNING TIME DURING KEY GENERATION STAGE.

Key lengths (bit)		Running 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

$$S = (r, s).$$

c) ECDSA signature verification: To verify S, one proceeds as follows:

i. check if r and s are integers within the interval $[1, n-1]$;
ii. compute $SHA-1(m)$, and convert it into an integer e;
iii. compute $w = s^{-1}\ (modulus\ n)$;
iv. compute $u_1 = e\,w\ (modulus\ n)$ and $u_2 = r\,w\ (modulus\ n)$;
v. compute $X(x_1, y_1)$ such as $X = u_1 G + u_2 P$; and
vi. if $X = 0$ then S is not valid. Otherwise, calculate $v = x_1\ (modulus\ n)$.

S is valid if $v = r$.

D. Performance studies

We make the comparison between RSA and ECIES algorithms for the encryption scheme, ECDSA and ECNR algorithms over RSA algorithm for digital signature. The comparison is done by considering the execution time of each algorithm. The computer which is used for the various tests is equipped with:

- a dual core processor of 2.4 GHz; and
- 2 GB of RAM.

Since other processes are running on the computer, the computational time retained is the average time after a certain number of execution of the different cryptosystems.

1) Encryption scheme: For this process, we have two steps: encryption and decryption. But we have firstly to generate a pair of key. The different comparisons are done following also the same steps, namely:

- key generation step;
- encryption step; and
- decryption.

We wrote programs in Java [31] implementing the algorithms. We encrypted one text file of 4 KB and after we have decrypted it. Then, we can calculate the computational time of each step.

a) Key generation: The results of the comparisons carried out in terms of computational time during the key generation stage are mentioned in Table IV.

From these results, we got the curve in Figure 4.

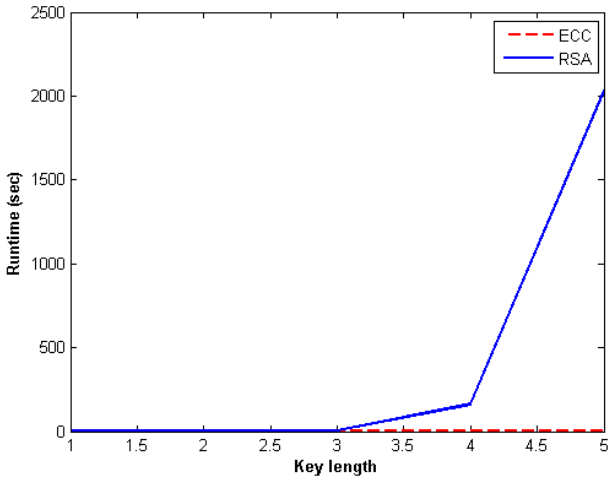

Fig. 4. ECC and RSA curves respect to running time during key generation stage.

TABLE V
RUNNING TIME DURING ENCRYPTION STAGE.

Key lengths (bit)		Running 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

Remark 1: By minutely analyzing the Figure 4, we have noted that the key generation by the elliptic curves algorithm ECIES is much faster than the key generation in the RSA algorithm. While this time increases exponentially in the case of RSA, it grows linearly with an almost zero slope at the level of the ECC algorithm. Therefore, from the point of view of key generation stage, it is evident that the algorithm based on elliptic curves is more optimal.

b) Encryption process: The results of the comparisons carried out in terms of computational time during the encryption stage, with RSA and ECIES algorithms, are mentioned in Table V.

From these results, we got the curve in Figure 5.

Remark 2: A careful analysis of the Figure 5 shows us that the data encryption by the RSA algorithm is faster than the encryption with an algorithm based on elliptic curves. But notice that the gap between the two computational times is relatively one second, then not so significant. Hence, from the point of view of encryption stage, we can conclude that both algorithms have the same performance but the RSA-based algorithm is a

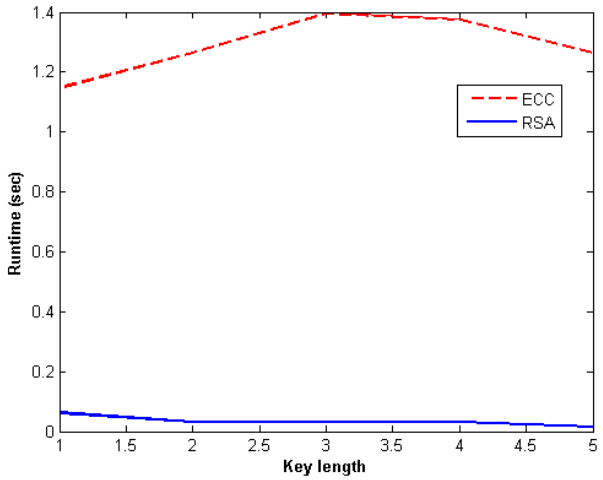

Fig. 5. ECC and RSA curves respect to running time during encryption stage.

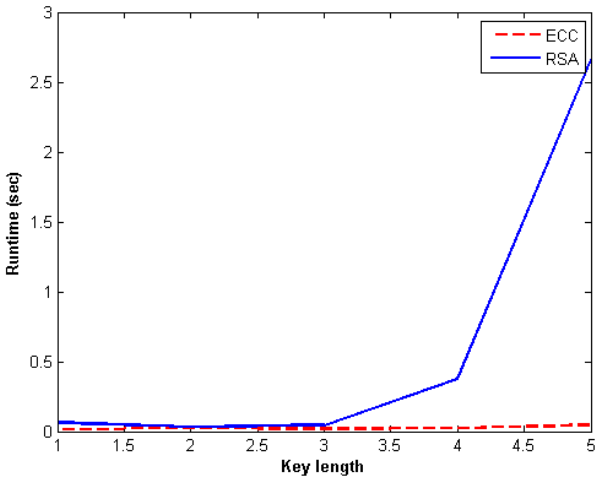

Fig. 6. ECC and RSA curves respect to running time during decryption stage.

TABLE VI
RUNNING TIME FOR DECRYPTION STAGE.

Key lengths (bit)		Running 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

TABLE VII
RUNNING TIME FOR KEY GENERATION STAGE.

Key lengths (bit)		Running 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

bit faster than that-based on the elliptic curves.

c) Decryption process: The results of the comparisons carried out in terms of computational time during the decryption stage are mentioned in Table VI.

From these results, we got the curve in Figure 6.

Remark 3: We obtain opposite results in comparison with those obtained during the encryption stage (refer Remark 2). The decryption by the ECC algorithm is more faster than the one with RSA. With a RSA key size greater than 409 bits, the decryption time becomes more and more considerable. Therefore, from a decryption point of view, we conclude that ECC-based algorithms perform better.

2) Digital signature: As previously mentioned in the description of algorithms in subsection III-C, we have three main stages in their implementation, namely:

- the key generation stage;
- the signature generation stage; and
- the signature verification stage.

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 running time is also done according to the three stages. The written programs in Java using the same library JCE [31] implement these algorithms.

a) Key generation: The results of the comparisons carried out in terms of computational time during the key generation stage are mentioned in Table VII.

From these results, we got the curve in Figure 7.

Remark 4: By analyzing the Figure 7, we have noted that the key generation by the elliptic curves algorithms is much faster than the key generation stage on RSA-based algorithm. The performance between both elliptic curve algorithms i.e. ECDSA and ECNR, is not very different. As we noted in Remark 1 during key generation stage of encryption process, while this time increases exponentially in the case of RSA, it grows linearly with an almost zero slope at the level of the ECC algorithm. Therefore, according to the generation key, the ECC-based algorithms are much more optimal.

b) Signature generation: The results of the comparisons carried out in terms of computational time during the signature generation stage are mentioned in Table VIII.

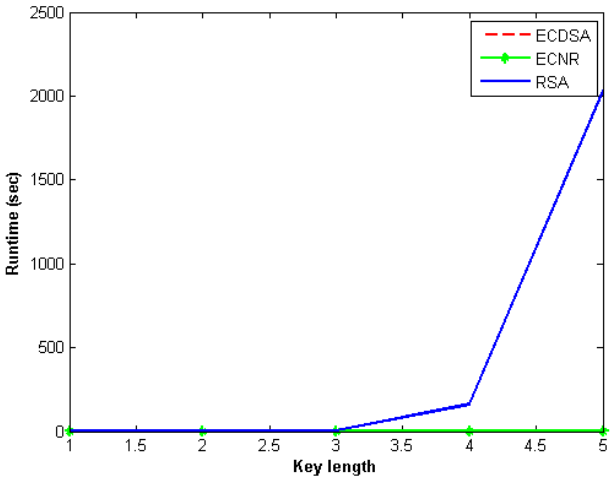

Fig. 7. ECC (ECDSA and ECNR) and RSA curves respect to running time during key generation curve.

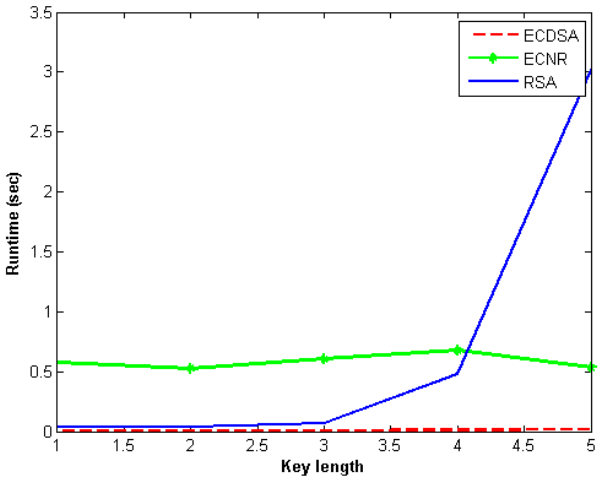

Fig. 8. ECC (ECDSA and ECNR) and RSA curves respect to running time during signature generation stage.

TABLE VIII
RUNNING TIME OF SIGNATURE GENERATION STAGE.

Key lengths (bit)		Running 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

TABLE IX
RUNNING TIME OF SIGNATURE VERIFICATION STAGE.

Key lengths (bit)		Running 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

From these results, we got the curve in Figure 8.

Remark 5: The table VIII shows that ECDSA, which is ECC-based algorithm, produces better results than both algorithms ECNR and RSA. It can remark however that when the length of the key of the ECNR algorithm is greater or equal than 409 bits, the signature generation time of this algorithm is better than RSA-based algorithm. Elsewhere, the RSA algorithm is a bit faster than the ECNR algorithm. In any case, the performance of both algorithms is not too different up to a given key length (409 bits) and beyond, ECNR becomes more efficient. In fact, we basically understand that ECC performs better than RSA in terms of signature generation.

c) Signature verification: The results of the comparisons carried out in terms of computational time during the signature verification are presented in Table IX.

From these results, we got the curve in Figure 9.

Remark 6: They all have almost the same performance especially the ECDSA and RSA algorithms. But the RSA results are better and the ECNR algorithm is the most greedy in terms of running time. When it comes to signature verification, the RSA algorithm is much more efficient than ECC-based algorithms. In fact, it just does a modular exponentiation.

IV. DISCUSSION AND ANALYSIS

We first ruled out symmetric cryptosystems in the context of this paper because they do not allow us to guarantee authenticity and non-repudiation. Gaur et al. achieved that public-key cryptography is viable on small devices without hardware acceleration [32]. Among the most used asymmetric cryptosystems like RSA, Diffie-Hellman, ECC, recommended by Al-Shabi [28], we discarded the Diffie-Hellman algorithm because it was not adequate to embedded systems because of its energy consumption.

There is nevertheless the Elgamal [19] algorithm. This cryptosystem has been used by the free software GNU Privacy Guard. But its current versions only implement elliptical curves. It has the same year of publish with the elliptical curves. Unlike RSA encryption, it has never been under patent protection. It

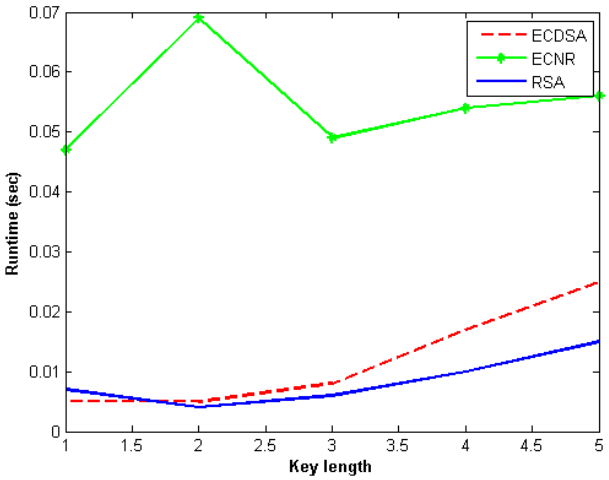

Fig. 9. ECC (ECDSA and ECNR) and RSA curves respect to running time during signature verification stage.

is nowadays considered like ECC algorithms. In this paper, however, our focus was on the RSA and ECC cryptosystems.

After an analysis of the key lengths suggested by Lenstra et al. in [24] (see table I), we denote that for equivalent cryptosystems, algorithms based on elliptic curves require less bits in terms of key length than those based on RSA. Nonetheless, for the same algorithms on elliptic curves, to offer the same robustness it is necessary about twice the key length of the symmetric algorithm key. From our achieved results, the generation of the keys and the calculation of the message signature are faster with the algorithms on elliptic curves than with the RSA algorithm.

In their paper to evaluate and compare the performance of algorithms based on RSA and elliptic curves, Nicholas Jansma and Brandon Arrendondo have proposed two implementations using two cryptosystems [33]. They came to the conclusion that algorithms based on elliptic curves can have the same level of security as those based on RSA with a much shorter key.

R. Sinha et al. achieved the paper "Performance Based Comparison Study of RSA and Elliptic Curve Cryptography" [34] in which they analyzed the results obtained by Nicholas Jansma and Brandon Arrendondo. In this paper, we examined the cryptosystem to identify the best way to secure more the data.

Considering Bandwidth saving, ECC offers considerable bandwidth savings over RSA and considering computational overheads, ECC offers Roughly ten times than that of RSA can be saved [35]. 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 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].

V. Concluding Remarks

We surveyed asymmetric cryptographic algorithms and performed a comparison study between them. We achieved RSA cryptosystems and some others by using cryptography protocols based on elliptic curves like: ECNR, ECDSA and ECIES.

From the obtained results, we deduced that the systems based on elliptic curves, thanks to the mathematical approaches advantages offered by said curves, are an efficient alternative compared to RSA-based cryptosystems.

Setting up a robust cryprosystem no longer necessarily involves the use of significant machine resources. Indeed, for an equivalent level of robustness between two cryptosystems, one based on the elliptic curves algorithms and the other on the RSA-based algorithm, the first (that based on the elliptic curves) uses a shorter key length.

Since embedded systems do not have sufficient memory and computational power to perform the calculations required by RSA-based cryptosystems with big numbers, elliptic curve cryptosystems are perfectly suitable for them. And even if this machine resources could allow it, they should be used for other purposes for better performance of these embedded equipment.

However, let us remember that in both cases, these cryptosystems are based on the use of keys, pledge of their security. The use of a key is in fact the weakness of all cryptographic systems. Blockchain technology appears to correct this weakness. We can therefore study in future work whether its use is possible with embedded systems to guarantee ever more the security not only of this equipment but also and above all of data that it collects since they are now widely used for this purpose.

References

[1] Attaran, M., VanLaar, I.: Privacy and security on the Internet: how to secure your personal information and company data. Information Management & Computer Security **7**(5) 241–247 (1999), https://doi.org/10.1108/09685229910292907

[2] Tseng, Y.-C., Chen, Y.-Y., Pan, H.-K.: A secure data hiding scheme for binary images. IEEE Transactions on Communications **50**(8), 1227–1231 (2002), https://doi.org/10.1109/TCOMM.2002.801488

[3] Stanton, P., Yurcik, W., Brumbaugh, L.: Protecting multimedia data in storage: a survey of techniques emphasizing encryption. In: Proceedings SPIE, Storage and Retrieval Methods and Applications for Multimedia, **5682**, pp. 18–29. SPIE, California, United States (2005), https://doi.org/10.1117/12.587207

[4] Skowyra, R., Bahargam, S., Bestavros, A.: Software-Defined IDS for securing embedded mobile devices. In: 2013 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, pp. 1–7, (2013) https://doi.org/10.1109/HPEC.2013.6670325

[5] Koscher, K. A.: Securing Embedded Systems: Analyses of Modern Automotive Systems and Enabling Near-Real Time Dynamic Analysis. University of Washington, (2014)

[6] Rivest, R., Shamir, A., Adleman, L. M.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM **21**(2), 120–126 (1978), https://doi.org/10.1145/359340.359342

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

[7] Sharma, S., Krishna, C. R.: An Efficient Distributed Group Key Management Using Hierarchical Approach with Elliptic Curve Cryptography. In: IEEE International Conference on Computational Intelligence & Communication Technology, pp. 687–693, Ghaziabad, (2015), https://doi.org/10.1109/CICT.2015.116
[8] Sasi, S. B., Dixon, D., Wilson, J.: 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), **04**, (2014)
[9] Techopedia dictionary, https://www.techopedia.com/definition/5426/digital-signature. Last accessed 31 Aug 2020
[10] Universign website, https://www.universign.com/fr/2017/ signature-electronique-expliquee-patron/ Last accessed 31 Aug 2020
[11] Merkle R.C.: A Digital Signature Based on a Conventional Encryption Function. In: Pomerance C. (eds) Advances in Cryptology — CRYPTO '87. CRYPTO 1987. Lecture Notes in Computer Science, **293**, Springer, Berlin, Heidelberg. (1988) https://doi.org/10.1007/3-540-48184-2_32
[12] Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy B., Meier W. (eds) Fast Software Encryption. FSE 2004. Lecture Notes in Computer Science, **3017**. Springer, Berlin, Heidelberg, pp. 371–388 (2004), https://doi.org/10.1007/978-3-540-25937-4_24
[13] Saho, N. J. G., Ezin, E. C.: Securing Document by Digital Signature through RSA and Elliptic Curve Cryptosystems. In 2019 International Conference on Smart Applications, Communications and Networking (SmartNets), pp. 1-6, Sharm El Sheik, Egypt, (2019) https://doi.org/10.1109/SmartNets48225.2019.9069749
[14] Hankerson, D. R., Vanstone, S. A., Menezes, A. J.: Introduction and Overview. In: Guide to Elliptic Curve Cryptography. Springer Professional Computing. Springer, New York, pp. 1–23 (2004) https://doi.org/10.1007/0-387-21846-7_1
[15] Koblitz, N.: Elliptic curve cryptosystems. Mathematics of computation, **48**(177) 203–209 (1987) https://doi.org/10.1090/S0025-5718-1987-0866109-5
[16] Shou, Y.: Cryptographie sur les courbes elliptiques et tolérance aux pannes dans les réseaux de capteurs. UFC, pp. 27–29 (2014)
[17] Joye, M.: Introduction élémentaire à la théorie des courbes elliptiques. UCL Crypto Group Technical Report Series, (1995) http://sciences.ows.ch/mathematiques/CourbesElliptiques.pdf
[18] Diffie, W., Hellman, M. E.: New directions in cryptography. IEEE Transactions on Information Theory, **IT-22**(6), 644-–654 (1976).
[19] Elgamal, T: A public key cryptosystem and a signature scheme based on discrete logarithms. In IEEE Transactions on Information Theory, **31**(4), pp. 469-472, (1985), https://doi.org/10.1109/TIT.1985.1057074
[20] Martínez, V. G., Encinas, L. H., Ávila, C. S.: A Comparison of the Standardized Versions of ECIES. In : Proceedings of the Sixth International Conference on Information Assurance and Security, Atlanta, (2010)
[21] Crypto++ page, https://www.cryptopp.com/wiki/Elliptic _Curve_Menezes-Qu-Vanstone. Last accessed 30 Aug 2020
[22] Bassham, L., Johnson, D., Polk, T.: Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public Key Infrastructure Certificates. Internet Draft. Available at http://www.ietf.org, (1999)
[23] Ateniese, G., de Medeiros, B.: A Provably Secure Nyberg-Rueppel Signature Variant with Applications. In: Proceedings of Advances in Cryptology – ASIACRYPT 2004. Revised version: https://eprint.iacr.org/2004/093.pdf
[24] Lenstra, A., Verheul, E.: Selecting cryptographic key sizes. Journal of cryptology, **14**, 255-–293, (2001). Available at http://infoscience.epfl.ch/record/164526/files/NPDF-22.pdf
[25] Prerna, P., Agarwal, P.: Cryptography Based Security for Cloud Computing System. International Journal of Advanced Research in Computer Science, **8**(5), pp. 2193–2197, (2017) https://doi.org/10.26483/ijarcs.v8i5.3388
[26] Chandra, S., Paira, S., Alam, S. S., Sanyal, G.: A comparative survey of Symmetric and Asymmetric Key Cryptography. 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE), Hosur, India, pp. 83–93, (2014) https://doi.org/10.1109/ICECCE.2014.7086640
[27] Barker, E., Dang, Q.: NIST Special Publication 800-57 Part 3 Revision 1: Recommendation for Key Management: Application-Specific Key Management Guidance. National Institute of Standards and Technology, pp. 12, (2015) https://doi.org/10.6028/NIST.SP.800-57pt3r1
[28] Al-Shabi, M. A.: A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security. International Journal of Scientific and Research Publications (IJSRP) **9**(3), ISSN: 2250-3153, pp. 576–589, (2019) https://doi.org/10.29322/IJSRP.9.03.2019.p8779
[29] Saho, N. J. G., Ezin, E. C.: Comparative Study on the Performance of Elliptic Curve Cryptography Algorithms with Cryptography through RSA Algorithm. CARI 2020 - African Conference on Research in Computer Science and Applied Mathematics, Oct 2020, Thiès, Senegal (hal-02926106)
[30] FIPS 180-4, https://nvlpubs.nist.gov/nistpubs/FIPS/ NIST.FIPS.180-4.pdf. Last accessed 17 Sep 2019
[31] DOUDOUX, J.-M.: Développons en Java, JCE (Java Cryptography Extension), https://www.jmdoudoux.fr/java/dej/chap-jce.htm. Last accessed 31 Aug 2020
[32] Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.: Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In: Marc Joye and Jean-Jacques Quisquater, Cryptographic Hardware and Embedded Systems - CHES 2004, **3156** of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 119–132 (2004)
[33] Jansma, N., Arrendondo, B.: Performance comparison of elliptic curve and rsa digital signatures. (2004)
[34] Sinha, R., Srivastava, H. K., Gupta, S.: Performance Based Comparison Study of RSA and Elliptic Curve Cryptography.International Journal of Scientific & Engineering Research, **2**(5), 720-–725 (2013)
[35] Kute, V. B., Paradhi, P. R., Bamnote, G.: A software comparison of RSA and ECC. International Journal Of Computer Science And Applications, **2**(1), 61-–64 (2009)