Performance Analysis of Standard Elliptic Curves for the Implementation of Elliptic Curve Cryptography on Resource-Constrained IoT Applications

Elvis Yeboah-Duako
School of Computer Science and Engineering
University of Electronic Science and
Technology of China
yeboahduako770@gmail.com

Abstract* — The security of IoT infrastructure has generally relied on the Rivest, Shamir and Adleman (RSA) public key cryptosystem. However, the increasing security requirements in mitigating high computing adversary attacks often translate to an exponential increase in the key size of RSA thereby making it impractical to implement in resource constrained IoT applications where computational speed, storage and bandwidth are limited. Attention has therefore been drawn to the Elliptic Curve cryptography which offers similar security levels at a much smaller key size. In the ECC however, the choice of curve is the fundamental security framework of the cryptosystem and hence, much thought must be given to the selection of key factors such as the prime field size and the shape of the curve. In this study, sets of elliptic curves suggested by various international standards are chosen, and the selected curves are examined with an emphasis on their performance and security characteristics. Each curve's performance is evaluated in terms of computational cost. The analysis is carried out by taking into account each curve for the Elliptic Curve Diffie-Hellman (ECDH) method and Elliptic Curve Digital Signature Algorithm (ECDSA) implementations. The comparison table of the chosen curves is presented in accordance with how long it takes each curve to compute different operations when applied to the ECC algorithms.

Keywords— Elliptic Curve Cryptography (ECC), IoT, Data Confidentiality, RSA, ECDH, ECDSA.

I. Introduction

The Internet of Things (IoT), which enables global connection of people, devices, and environments, is radically changing our way of life. It is used in a variety of industries, including intelligent transportation, healthcare, and smart cities & homes. For instance, wearables like Fitbit or implantable medical devices like pacemakers can monitor our physiological status around-the-clock, encouraging a better lifestyle and enabling prompt medical intervention when required. IoT applications aim to significantly improve the comfort, convenience, and smartness of our lives [1]. Nevertheless, as IoT gradually invades our privacy spaces, its security, primarily in terms of data confidentiality and authenticity, is of significant importance given that IoT operates most in the wireless space.

Cryptograph, the science and art of secret writing [2] has been the backbone technology to meeting data confidentiality requirements in most applications. Several Cryptographic schemes exist with RSA being the most widely used. In recent times, however, Elliptic Curve Cryptography (ECC) has gained tremendous attention due to its ability to match the security of the famous RSA cryptosystem with relatively shorter key length and lower computation cost.

Hence, it is gradually becoming the de-facto cryptosystem for implementing PKC protocols such as digital signatures and key agreement protocols. Likewise, in resource constrained IoT applications such as wearables, smart cities & homes, intelligent transportation systems, healthcare, etc., the potency of ECC can be harnessed to provide significant security features even with its limited computing power.

Curve selection is essentially the first step to maximizing security in the implementation of ECC. As such, several global curve standards are available to aid in the selection of the appropriate ECC-curve. Factors that influence the choice of curve include the security requirement and the required efficiency in terms of computation speed that the curve provides in relation to the application environment it is to be implemented in. These standards are defined over either prime or binary extension fields for different security levels even though most elliptic curves in use today are defined on prime fields [3]. Hence, this study analysis is subject to selected curves defined over prime fields.

The rest of the paper are sectioned as follows: Section II gives a brief summary on Elliptic Curve Cryptography. In Section III, curves to be used for this analysis are introduced and Analysis of different elliptic curves are provided in sections IV and V. Security Analysis are also provided in sections VI and VII and finally, the paper is concluded in section VIII.

II. Elliptic Curve Cryptography

A. Overview

An Elliptic curve with respect to cryptography can be defined by the equation:

$$y^2 = x^3 + ax + b \qquad (1)$$

Where a and b are integers such that $\mathbf{4a^3 + 27b^2 \neq 0}$. Given such a curve, three properties define its operation:

- The curve is always symmetric about the x-axis.
- A line drawn through any two given points (P_1 & P_2)on the curve gives a third point (P_3) on the curve except when the points are vertically opposite in which case the third point is defined as a point at infinity (∞).
- A tangent to any point P_1 on the curve gives a second point P_2 on the curve except when the point lies on the x-axis in which case the second point is defined to be at infinity (∞).

B. *Useful Formulars for ECC Computations*

Elliptic curves use geometric operations for its calculations. Mainly point doubling, point addition and scalar multiplication are the key operations performed

1. *Point Doubling*

Addition is defined over a single point, $P_1(x_1, y_1)$ on an elliptic curve. Say $\boldsymbol{P_1 + P_1 = P_2}$. The coordinates of $P_2(x_2, y_2)$can be calculated using the formulars below:

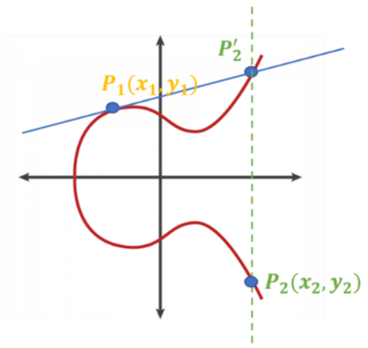


Fig 1: Point doubling operation for Elliptic curves

$$Slope\ (S) = \frac{3x_1^2 + a}{2y_1} \tag{2}$$

$$x_2 = S^2 - 2x_1 \tag{3}$$

$$y_2 = S(x_1 - x_2) - y_1 \tag{4}$$

2. *Point Addition*

Addition is defined over 2 points, $P_1 and\ P_2$, on an elliptic curve. Say $\boldsymbol{P_1 + P_2 = P_3}$. The coordinates of $P_2(x_2, y_2)$can be calculated using the formulars below:

$$Slope\ (S) = \frac{y_2 - y_1}{x_2 - x_1} \tag{5}$$

$$x_3 = S^2 - (x_1 + x_2) \tag{6}$$

$$y_3 = S(x_1 - x_3) - y_1 \tag{7}$$

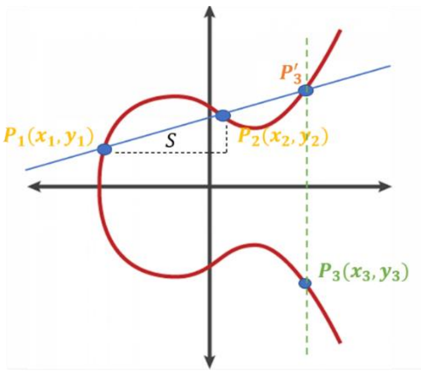


Fig 2: Point addition for Elliptic curves

3. *Scalar Multiplication*

Scalar multiplication is defined of Elliptic curves as "Repeated Point Addition". if $\boldsymbol{P}$ and $\boldsymbol{Q}$ be points on an Elliptic curve and $\boldsymbol{k}$ is an integer $(k\ \in R)$:

$$Q\ = k \cdot P \tag{8}$$

$$\Rightarrow\ Q\ = \underbrace{P + P + P + \cdots + P}_{k\ times}$$

III. Selection of Elliptic Curve

The selection of reliable elliptic curves for ECC implementation is the focus of all the current international standards. Every standard curve aspires to maintain the Elliptic Curve Discrete Logarithm Problem's level of difficulty (ECDLP). Each standard specifies a certain elliptic curve shape as well as recommended values for the prime fields and equation constants in order to maximize efficiency. The choice of curve has a significant impact on ECC security. The basic goal is to choose a curve that is secure from known ECDLP attacks. Any application's cryptographic security is weakened if a poor curve is chosen. Super-singular curves, a particular type of elliptic curves, for example, are unstable because the discrete logarithm problem (DLP) over them can be transformed over a small prime field [4]. The list of standards that can be used to choose an elliptic curve for ECC as demonstrated in [5] include: *ANSI X9.62/.63, IEEE P1363, SEC 2, NIST FIPS 186-2, Brainpool, NSA Suit B, and ANSSI FRP 256V1.*

Each of the aforementioned standards has a set of curves that are suggested for use with ECC. The group contains curves defined over both small and large prime fields. The order of the group must be large for the cryptography technique to be more secure [6]. There are numerous elliptic curve variations available today. Every curve has unique qualities and benefits. The most challenging aspect of setting parameters for the ECC to ensure robust application security and improved performance is determining the proper curve. ECC has been included into many standards, including *ANSI X9.62, ANSI X9.63, IEEE P1363, etc.*

The selected elliptic curves are defined over a prime field *GF(p)* where p is a prime number. The following are the general forms of elliptic curves which are used in ECC [7]: the *Weierstrass curve, the Montgomery curve and the Edward curve.*

A. *Weierstrass Curve*

This curve is widely used curve for most practical ECC applications. It is the general form of cryptographic use-case elliptic curves and is represented by equation (9) below. NIST and Brainpool standards recommend this curve in their protocols.

$$y^2 = x^3 + ax + b \tag{9}$$

B. *Montgomery Curve*

Peter L. Montgomery proposed the Montgomery curve in 1987. Some standards specify the types of Montgomery curves that can be used in ECC. This type of curve is ideal for x-coordinate-only addition, which is used in scalar multiplication. The general form of the Montgomery curve is shown in equation (10):

$$By^2 = x^3 + Ax^2 + x \tag{10}$$

C. *Edward Curve*

Edwards curves are relatively new, having been introduced in 2007. When compared to the other two curve forms, this type of curve has the fastest arithmetic operations on the curve. The interesting thing about this curve is that it always has the same formula for point addition. Because of this feature, there is the possibility of side-channel attacks, in which one can guess the scalar coefficient using the calculation of time used for scalar multiplication. The general form of the Edwards curve is shown in equation (11):

$$ax^2 + y^2 = 1 + dx^2y^2 \tag{11}$$

IV. Analysis of Different Elliptic Curves

There are numerous elliptic curves available for use in ECC as suggested by various standards. Curve types are classified primarily based on prime field size and curve form. The following curves, which are very common in ECC applications, are considered for this analysis: *M221 curve, NIST P-244 curve, Curve 25519, BN (2,254) curve, Brainpool P256tl Curve, NIST P-256 curve, SECP256kl, SECP256rl, NIST P-384 curve* and *M-511 Curve.*

Each curve has different field sizes, which are nothing more than ECC key sizes. The selected curves are then analyzed using the two ECC algorithms, ECDH and ECDSA. The analysis is performed with SageMath [8], an open-source mathematical software, on a system with an Intel Core i5 3rd Gen processor and 12 GB of RAM. For each curve, the same algorithm is used to perform the analysis. Only the curve parameters vary depending on the type of curve. The computation time required for various operations during ECDH and ECDSA is calculated for each type of curve. The specifics of each algorithm are provided below.

A. *The Elliptic Curve Diffie Hellman Key Exchange Protocol (ECDH)*

ECDH is not an encryption algorithm, but rather a key-agreement protocol. The established keys are then used for encryption and decryption. The ECDH protocol specifies how the keys are generated and exchanged between two parties. Each curve is applied over the prime field suggested by its standard. *P* is chosen at random from the curve. The following steps are taken to exchange keys between Alice and Bob using the ECDH algorithm: follows:

1. Alice and Bob agree on the domain parameters which include the curve parameters (a, b), the field defined for use (p) and the base generator point $(G(x, y))$
2. Alice then selects an integer $\boldsymbol{a}$ as her private key such that $1 \le a \le n - 1$ where n is the order of the sub-group of $\boldsymbol{G}$ $(i.e. n = ord(G))$. She then computes her public key $\boldsymbol{A}$ as:
$$A = a \cdot G(x, y) \tag{12}$$
Bob also selects an integer $\boldsymbol{b}$ as his private key satisfying the same conditions as Alice's and computes his public key $\boldsymbol{B}$ as:
$$B = b \cdot G(x, y) \tag{13}$$
3. Alice and Bob then exchange their public keys $(A \text{ and } B)$ over the unsecure channel.
4. Finally, Alice Computes the shared secret $\boldsymbol{K_A}$ as a product of her private key and Bob's public key:
$$K_A = a \cdot B(x_b, y_b) \tag{14}$$
Bob also calculates the secret key $\boldsymbol{K_B}$:
$$K_B = b \cdot A(x_a, y_a) \tag{15}$$

These two keys $K_A \text{ and } K_B$ are proven to be the same.

Proof of Correctness

From (19), Alice computes K_A as:

$$K_A = a \cdot B(x_b, y_b)$$

But $B(x_b, y_b) = b \cdot G(x, y)$

$$\Rightarrow K_A = a \cdot \left(b \cdot G(x, y)\right)$$
$$\therefore K_A = (a \cdot b) \cdot G(x, y) \tag{16}$$

Again, from (20), Bob computes K_B as:

$$K_B = b \cdot A(x_a, y_a)$$

But $A(x_a, y_a) = a \cdot G(x, y)$

$$\Rightarrow K_B = b \cdot (a \cdot G(x, y))$$
$$\therefore K_B = (b \cdot a) \cdot G(x, y) \tag{17}$$

Since (16) = (17), it is confirmed that, indeed, Alice and Bob have the same secret key to be used for the encryption and decryption process.

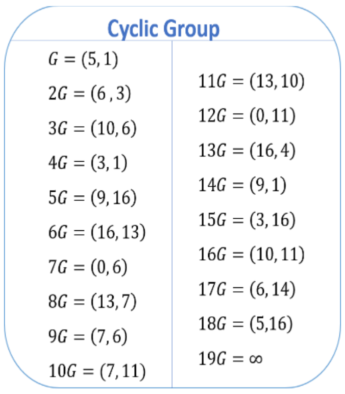
Illustration

Let Domain parameters be:

$p = 17, \quad a = 2, \quad b = 2, \quad G(x, y) = (5,1)$

$$\Rightarrow \boldsymbol{E: y^2 \equiv x^3 + 2x + 2 \ \ (mod\ 17)}$$

The cyclic group of the generator point $G(5,1)$ are calculated using $(2) - (8)$ and results are shown in fig 3 below:



Cyclic Group

$G = (5,1)$	$11G = (13,10)$
$2G = (6,3)$	$12G = (0,11)$
$3G = (10,6)$	$13G = (16,4)$
$4G = (3,1)$	$14G = (9,1)$
$5G = (9,16)$	$15G = (3,16)$
$6G = (16,13)$	$16G = (10,11)$
$7G = (0,6)$	$17G = (6,14)$
$8G = (13,7)$	$18G = (5,16)$
$9G = (7,6)$	$19G = \infty$
$10G = (7,11)$	

Fig 3: Cyclic group points for Generator point

The shared secret key is calculated following the steps outlined this section and represented on Table 1.

Table 1: Illustration of ECDH key Exchange protocol

Alice	Eve	Bob
Chooses private key $a = 3$	p = 17 G (5,1)	Chooses private key $b = 9$
Computes public key $A(x_a, y_a) = 3G$ $A = (10,6)$		Computes public key $B(x_b, y_b) = 9G$ $B = (7,6)$
Receives $\boldsymbol{B(x_b, y_b)}$ from Bob	**A** **B**	Receives $\boldsymbol{A(x_a, y_a)}$ from Alice
Computes Secret key $K = 3B = (3 \cdot 9)G$ $K = 27G \bmod 17$ $\boldsymbol{K = 8G = (13,7)}$	**??**	Computes Secret key $K = 9A = (9 \cdot 3)G$ $K = 27G \bmod 17$ $\boldsymbol{K = 8G = (13,7)}$

TABLE 2:
ECDH Algorithm Computation Times of Selected Curves

	Type of Curve	Time taken for point addition (ns)	Time taken to calculate Alice's public key $A(K_{pb}) = aP$ (ms)	Time taken to calculate Alice's public key $B(K_{pb}) = bP$ (ms)	Time to calculate secret on Alice side $aB(K \times pb)$ (ms)	Time to calculate secret on Bob side $bA(K \times pb)$ (ms)	Total time (ms)
1	Curve 25519	121.7	13.1	12.8	13.1	14.1	53.1
2	M221	102.3	11.9	12.3	12.6	13.2	50.0
3	SECP 256r1	62.6	11.6	11.1	11.4	12.7	47.9
4	BN (2,254)	61.9	12.7	11.3	11.9	13.4	48.3
5	NIST P-224	58.4	12.2	11.2	12.3	12.4	48.1
6	M-511	58.1	16.3	13.9	13.8	15.1	59.1
7	NIST P-384	53.2	13.8	12.2	14.3	14.6	54.9
8	Brainpool P256t1	51.9	14.0	13.4	14.8	14.1	56.3
9	NIST P-256	47.7	12.4	17.9	15.3	16.4	62.0
10	SECP 256k1	45.2	11.7	12.5	11.7	12.2	48.1

As seen, Alice and Bob have been able to agree on a common secret key over the unsecure channel. And though Eve gets access to the public keys and the domain parameters, she is unable to compute the secret key without the knowledge of Alice and/or Bob's private keys.

B. *The Elliptic Curve Digital Signature Algorithm (ECDSA)*

ECDSA is used to sign the message's hash, which is shortened to make the hash's bit length equal to the bit length of n, which is the order of the subgroup [9]. The truncated hash is an integer and is denoted by the letter z. To implement the ECDSA between Alice and Bob, a prime number q, an elliptic curve $E \bmod q$, a base point from the curve G, Alice's private key d, and Bob's public key H_A are used. ECDSA is used by Alice to sign the message as follows:

1. Alice takes a random integer k from $\{1, \ldots, n-1\}$, where n is the subgroup order.
2. She calculates the point $P = kG$
3. She then calculates $r = x_P \bmod n$, such that $r \neq 0$, where x_P is the $x - coordinate$ of P
4. She then calculates $s = \frac{z+rd}{k} \bmod n$, such that $s \neq 0$
5. The pair (r, s) constitutes the ECDSA signature

To verify the signature, Bob needs Alice's public key H_A, the hash z and the signature (r, s) to perform the following verification computations:

1. Bob calculates $u_1 = s^{-1} z \bmod n$
2. And $u_2 = s^{-1} r \bmod n$
3. He then calculates the point $P = u_1 G + u_2 H_A$

The signature is valid only if $r = x_P \bmod n$.

V. Performance Analysis

Each chosen curve is subjected to the ECDH method for curve analysis. We notice the calculation times needed to compute the shared secret during decryption of the ECDH key exchange algorithm and the computation times needed to compute point multiplication on the curve during encryption. The computation time required for each curve to carry out numerous operations during the ECDH key exchange procedure is displayed in Table 2. When compared to other curves, the SECP256r1 curve takes less time to perform the point multiplication operation during encryption and decryption. Therefore, SECP256r1 is a viable option when a quick computation is needed.

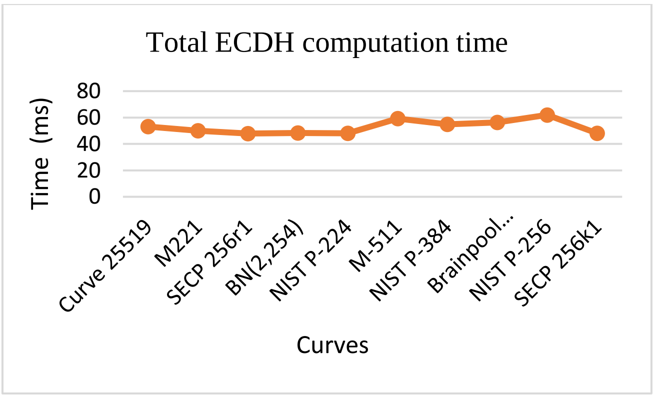


Fig. 4: Total ECDH computation time per selected curve

The examination of the chosen curves using the ECDSA is carried out, much like with the ECDH method. Once more, the findings are taken into account in terms of how long it takes for each curve to generate and verify a signature. Table 3 presents the numerical outcomes.

Figures 5 and 6 show, respectively, the graphical representations of the calculation time for each curve to generate the signature and to verify that signature when executing ECDSA. Figures and Table 3 both show that, when the ECDSA is used, the curve M221 takes less calculation time for the signature generation and verification operations than other chosen curves. IoT Applications with limited resources can use the M221 curve. The benefit of adopting M221 is that it has a smaller prime field size than other curves, at 221 bits.

TABLE 3:
ECDSA Computation times of Curves

	Type of Curve	Signature Generation Time (r, s)		Signature Verification time (ms)	Total time (ms)
		Time to compute r (μs)	Time to compute s (μs)		
1	Curve 25519	1.06	2.47	21.7	21.704
2	M221	0.98	2.32	15.1	15.103
3	SECP 256r1	2.01	2.31	20.3	20.304
4	BN (2,254)	0.68	2.45	18.8	18.803
5	NIST P-224	0.73	2.52	15.5	15.503
6	M-511	0.91	4.26	61.7	61.705
7	NIST P-384	0.68	3.09	34.7	34.704
8	Brainpool P256t1	0.78	2.41	21.0	21.003
9	NIST P-256	2.37	5.64	19.9	19.98

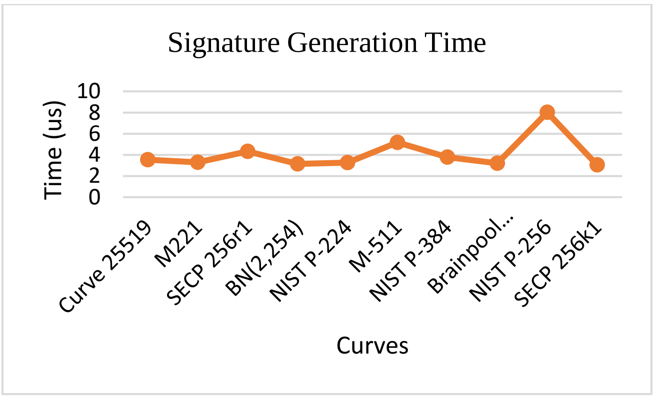


Fig. 5: Computation time taken by each curve during the signature generation process

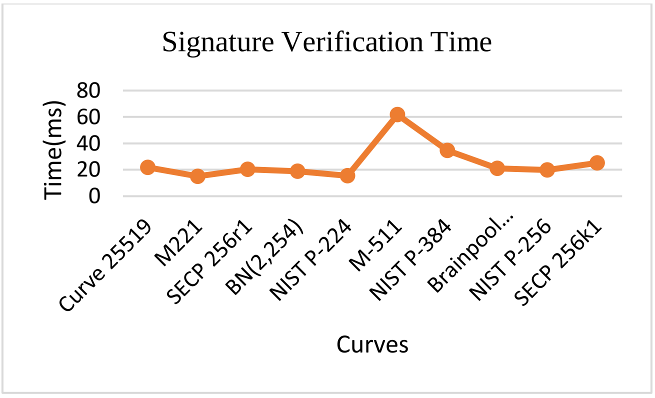


Fig. 6: Computation time taken by each curve during the signature verification process of ECDSA

VI. General Security Analysis

Elliptic curves, like most cryptosystems, define their own discrete logarithm problem, which serves as the foundation for their security. Given $A(x_a, y_a) = k \cdot G(x, y)$, the Elliptic curve Discrete Logarithm Problem (ECDLP) is defined as finding the value of k, given the initial point $G(x, y)$ and the final point $A(x_a, y_a)$. Until proven otherwise, this is thought to be a very expensive operation. And, because it is computationally impossible to successfully compute k in discrete polynomial time, all known attacks can be resisted with carefully chosen elliptic curve parameters for the cryptographic scheme.

The use of exhaustive search mechanisms is the most naïve approach to solving the Elliptic curve discrete log problem. To withstand such an attack, the order of the generator base, n, must be sufficiently large. $(n \geq 2^{80})$ [10]**.**

Another well-known ECDLP attack is a hybrid of the Pohlig-Hellman and Pollard's rho algorithms, with a running time of $O(\sqrt{P})$ where p is the largest prime divisor of n**.** To defeat this type of attack, the elliptic curve parameters should be chosen so that n is divisible by a significantly large prime number p. The size of p should be so large that $\sqrt{p}$ steps is an infeasible amount of computation $(p \geq 2^{160})$ [10].

Some of the recommended curves in the international standards can become weaker due to a rare procedural attack [11]. The ECDLP can also be solved via Shank's Baby Step Giant Step attack, although it has a significant memory cost. Side channel attacks are particularly useful for disabling the encryption employed by embedded devices [4]. Additionally, the Elliptic curve discrete log issues do not benefit from precomputation attacks, which are relatively common in exponential discrete log problems. Increasing the Elliptic curves' overall security performance in comparison to other cryptosystems.

However, it should be highlighted that there is no mathematical evidence to support the claim that there isn't an effective algorithm for solving the ECDLP. As a result, it would be implied that $P \neq NP$ if it were demonstrated that such an effective polynomial-time algorithm does not exist. Such a proof would be revolutionary because this question is now regarded as one of the most important and outstanding open ones in computer science (and not very likely to appear). ECDLP is not known to be NP-hard, hence there is either no evidence that it is intractable. This is also unlikely to be demonstrated [10].

Elliptic curves' short history—they were invented in 1985—means that less study has been done on their discrete log issues than has been done on conventional discrete log problems, which are employed in most cryptosystems and have sub-exponential solutions. This raises certain unanswered queries and lingering reservations about the ECDLP's overall security, as does the absence of evidence supporting its hardness.

VII. Security of selected ellpitic curves

The authors of [6] evaluated the ECDLP and ECC security of each curve. In order to ensure ECDLP security, four factors were taken into account: the curve's safety against Pollard's rho algorithm, its security against multiplicative transfer, the curve's stiffness, and finally the complex multiplication field discriminator. The authors deemed a curve unfit for use in the ECC application if it failed to withstand any of the aforementioned parameters. The safety or otherwise of each curve is assessed based on the ECDLP security parameters. The SECP256k1 curve is used in the Bitcoin protocol even though [5] claims that it is unsafe since it does not meet the ECDLP security parameters' rigidity requirement. According to the security criteria of the ECDLP, certain curves are categorized as safe or not as shown in Fig. 7.

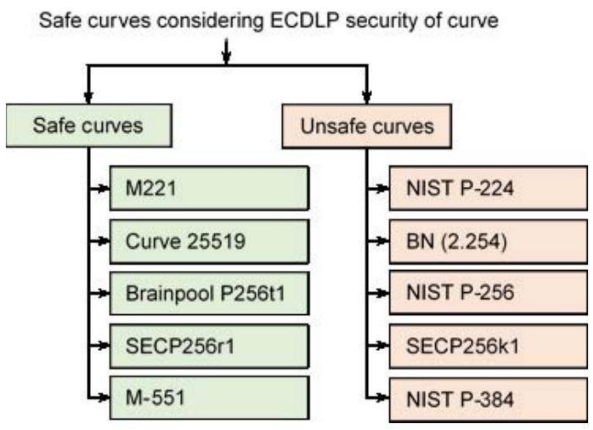


Fig.7: Classification of selected curves based on the ECDLP security level.

VIII. Conclusion

The numerous curve types recommended by various standards are chosen and evaluated in this research. Each curve is examined using the ECDH and ECDSA ECC algorithms. The comparative tables display how long each curve needs to compute when these two algorithms are applied to various operations. It is evident from the analysis that the amount of time needed for computing rises along with the curve's field size. The proper elliptic curve can be chosen based on the security and processing needs of a particular application. Studying Tables 2 and 3 reveals that the M221 curve performs well for the ECDSA since the generation and verification of signatures require less computational time. The SECP256r1 curve for the ECDH method is also a good choice for IoT applications with limited resources. Moreover, the SECP256r1 is a good choice for IoT applications given that it also performs better relatively with other curves. The M221 curve can be used for the ECDSA of elliptic curve cryptography where less storage is available and rapid processing is necessary.

Given that the elliptic curve parameters are carefully selected to thwart the known attacks against the ECDLP, it is believed that the ECDLP is infeasible given the state of computer technology in use today. A general-purpose, sub-exponential-time algorithm for solving the ECDLP has not yet been found. As a result, given the proper curve parameters, elliptic curve cryptography is thought to be exceedingly safe and with its low computation requirements, it is a great choice for resource constrained IoT applications.

References

[1] Zhang, J., Rajendran, S., Sun, Z., Woods, R., & Hanzo, L. (2019). Physical Layer Security for the Internet of Things: Authentication and Key Generation. IEEE Wireless Communications Magazine, 26(5), 92-98. https://doi.org/10.1109/MWC.2019.1800455

[2] Simson Garfinkel and Gene Spafford, Practical UNIX & Internet Security (ISBN 1-56592-148-8), Second Edition, April 1996.

[3] J. Bos, C. Costello, P. Longa and M. Naehrig, “Selecting Elliptic Curves for Cryptography: An Efficiency and Security Analysis”, 2014,eprint.iacr.org/2014/130.pdf

[4] C. Vuillaume, “Side Channel Attacks On Elliptic Curve Cryptosystems”, Technische Universitaet Darmstadt, Fachgebiet Informatik, Fachbereich Kryptographische Protokolle, 2004.

[5] D. J. Bernstein and T. Lange, “Safe Curves: choosing safe curves for elliptic-curve cryptography”, accessed 29 June 2022, safecurves.cr.yp.to/.

[6] Edward Yin, “Curve Selection in Elliptic Curve Cryptography”, San Jose State University, Project, Spring, 2005

[7] J. Bos, C. Costello, P. Longa and M. Naehrig, “Selecting Elliptic Curves for Cryptography”, A presentation for the Crypto Forum Research Group (CFRG), 2014www.ietf.org

[8] Website: www.sagemath.org

[9] Andrea Crbellini, “Elliptic Curve Cryptography: ECDH and ECDSA”, 30 May 2015.

[10] Rakel Haakegaard and Joanna Lang, The Elliptic Curve Diffie-Hellman (ECDH), (2015)

[11] Tetsuya Izu and Tsuyoshi Takagi, “Exceptional Procedure Attack on Elliptic Curve Cryptosystems”, Springer-Verlag Berlin, pp. 224–239, 2003.