Proceedings of Science and Mathematics
Faculty of Science, Universiti Teknologi Malaysia
Vol. 9, 2022, page 22-32

The Elliptic Curve Cryptography Algorithm in Cryptocurrency Simulation for Bitcoin Transaction and Ownership

Amirruliman Abdul Halim*, Nur Syarafina Mohamed
Department of Mathematical Sciences, Faculty of Science
Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia
*Corresponding author: amirimanhalim@gmail.com

Abstract
Cryptocurrencies have grown in popularity in recent years, and their use is becoming wider in a variety of fields. The use of digital currencies for legal purposes is advancing in tandem with technological advancement, but criminal activities are also emerging to capitalise on this boom. Elliptic Curve Cryptography is a method of cryptography that employs elliptic curves over finite fields. When compared to other cryptographic schemes such as Rivest, Shamir, and Adleman (RSA) algorithm, this approach allows for smaller key sizes while maintaining the same level of security. The Elliptic Curve Digital Signature Algorithm (ECDSA) is the most widely used elliptic curve-based signature scheme, with applications in a wide range of fields. The Bitcoin protocol, which has seen a surge in popularity as an open source, digital currency, is one modern application of the ECDSA. The aim of this paper has been, first, to simulate the bitcoin transactions using cryptography algorithms by using ECDSA as a signature and SHA256 as an encryptor, all of which are contain in Algorithm Elliptic Curve Cryptography algorithm that secures the bitcoin transaction process and second, to prove the ownership of bitcoin used bitcoin's ECDSA uses the secp256k1 elliptic curve. Based on the result occur by using ECDSA coding on python, we manage to get signature matches and signature verification, which means only the one who have the signature key are allow to do the transaction and by doing this the user can prevent fake transaction by the outsider (hackers) and also can prove the ownership of the bitcoins.
Keywords: Cryptocurrencies; Elliptic Curve Cryptography; Bitcoin; cryptography algorithms; ECDSA; Secp256k1

1. Introduction

Cryptocurrency works in a similar way to computer data like music and movies in that it may be hidden and destroyed. Even if a cryptographic protocol is believed or even proven to be secure, implementing it efficiently and securely in practice is usually difficult. Cryptographic operations can have a significant impact on an application's performance. The impact varies depending on the application's requirements, the number of operations that must be performed, the availability of special-purpose computer instructions or cryptographic accelerators, and the operating environment's constraints. Many cryptographic primitives, for example, are not well suited for implementation on constrained Internet of Things devices like radio-frequency identification tags.

People, businesses, and government agencies have been dealing with the problem of data tampering for decades. Some businesses and individuals have had a difficult time as a result of data theft and manipulation To address this concern, researchers are developing lightweight cryptography—cryptographic primitives and protocols with a small computational footprint. . The blockchain technology has provided the world with a one-of-a-kind solution that protects data integrity. Data signatures can now be created, so ensuring data integrity and authenticity.

Each bitcoin transaction requires a high level of security to secure both user and transaction data. At this point, the suitable algorithm is used to secure the transaction while reducing the processing time. The Elliptic Curve Cryptography (ECC) algorithm is a high-security cryptography algorithm. It is frequently compared to the Rivest, Shamir, and Adleman (RSA) algorithm because it has a security level that is nearly identical, but there are some differences that make ECC superior to the RSA

algorithm, allowing the ECC algorithm to optimise cryptocurrency security during the transaction process. Bitcoin is the first and most popular cryptocurrency, laying the way for a long-standing and unaltered payment system technology. Because of similar way to computer data, it easy to be manipulated and steal the data. The ECC method is used as the ECDH and ECDSA key exchange algorithms for signing and verifying in this study. How to implement ECC algorithm in simulation to prove Fake Bitcoin Transactions and Proved the ownership of Bitcoins. The goal of this research is to use cryptography methods to imitate bitcoin transaction.

2. Literature Review

2.1 Simulation

Simulation is a way of putting the model to the test. The model's behaviour matches some features of the system's behaviour that stand out, allowing you to experiment with the model to infer the user's behaviour. This broad framework has been shown to be a valuable complement to learning, problem-solving, and design (White, Preston and Ingalls, Ricki, 2015).

The steps for creating simulation models, designing simulation experiments, and analysing simulations are shown below (Ichsani, Yuditha and Audia, Resisca and Bahaweres, Rizal, 2019):

1. Determine the objectives and scope
2. Determine the conceptual model
3. Obtain and analyse data
4. Building a simulation model program
5. Verification of simulation models and validation
6. Experiment design
7. Running simulation

2.2 Cryptography

Cryptography is a branch of computer science and mathematics concerned with methods for secure communication between two parties while a third party is present. It is based on methods such as encryption, decryption, signing, creation of random numbers.

2.3 Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography is a type of public-key cryptography that is based on the area of an elliptic curve (Kapoor, Abraham & Singh, 2008), Elliptic Curve Cryptography is a cryptographic procedure in which each side has a private and public key pair. Only a few people have access to the private key, whereas everyone has access to the public key. Elliptic Curve Cryptography (ECC) is a cryptographic technique based on the algebraic structure of an elliptical curve on a finite surface.

2.4 Elliptic Curve Diffie-Hellman (ECDH)

ECDH is a method of determining the private and public keys through key exchange. Both the sender and the receiver perform the same action with a separate private key and the same public key, yet the outcomes are identical. The most well-known technique for addressing discrete issues is ECDH, which is based on a difficult mathematical formula (Sio-long Ao, Alan Hoi-shou Chan, Hideki Katagiri, Li Xu, 2011).

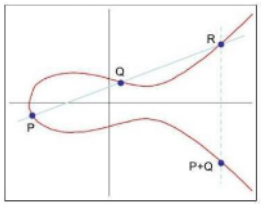
Operation Group in ECDH is divided into two, namely:

1. Addition

$$E = \{(x, y \mid y_2 = x_3 + ax + B) \cup \{0\}\ O\} = \text{point of infinity} \quad (1)$$

Figure 2.1 Point Addition

$$s = \left(\frac{(Y_P - Y_Q)}{(x_P - x_Q)}\right) \quad (2)$$



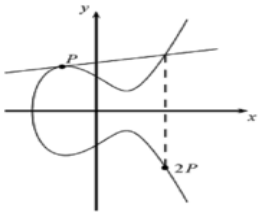
$$x_R = s^2 - (x_P + x_Q) \quad (3)$$

$$Y_R = s(x_P - x_R) - Y_P \quad (4)$$

2. Doubling Point

Figure 2.2 Point Doubling

$$P + P = R = 2P \quad (5)$$



$$s = \left(\frac{3x_P^2 + a}{(2Y_P)}\right) \quad (6)$$

$$x_R = s^2 - 2x_P \quad (7)$$

$$Y_R = s(x_P - x_R) - Y_P \quad (8)$$

2.5 Elliptic Curve Digital Signature Algorithm (ECDSA)

Scott Vanstone proposed the Elliptic Curve Digital Signature Algorithm in 1992, and it is the elliptic curve analogue of the Digital Signature Algorithm (DSA). The fundamental benefit of ECDSA is that it provides the same level of security as DSA while using smaller keys. Smaller keys allow for faster calculations and smaller public keys to be passed around. In 1998, 1999, 2000, and 2000, the ECDSA was accepted as ISO, ANSI, IEEE, and FIPS standards, accordingly. One-way hash function or one-way hash function is used in the Digital Signature Algorithm (DSA). The signing and verification processes make up the digital signature process. Digital signatures are used to verify that information is signed by the person who claims to be who it claims to be and to see whether the information has been tampered with by bad actors. The process of creating signature lists and verifying signatures. The signing of a communication involves producing a signature with the private key and verifying it with the public key. Instead of signing the message directly, cryptographic hash functions are used to create a message digest in the original message for efficiency reasons (Wang, 2014).

2.6 Bitcoin's Koblitz Curve

1. Bitcoin used the elliptic curve Secp256k1, which is defined by the sextuple $T=(p,a,b,G,n,h)$.
2. The prime, $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$.
3. The Curve $E : y^2 = x^3 + ax + b \; over \; F(p)$ id defined by: *a = 0, b = 7*
4. The base point G in compressed form is: *G = 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798*
5. The order *n = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141*
6. The cofactor h = 01

Every Bitcoin address is a cryptographic hash of a public key, and the person who owns this specific Bitcoin is the person who has the associated private key, according to ECDSA. To send a Bitcoin to someone else, you must first provide them the private key that corresponds to the coin. Breaking the ECDSA is the same as solving the discrete logarithm issue for elliptic curves (ECDLP). This means that if someone solves the ECDLP or finds a backdoor in the Secp256k1 curve, they will have access to 4.5 billion dollar worth of Bitcoins. It's worth mentioning that Secp256k1 isn't currently on NIST's list of recommended curves.

2.7 ECDSA Performance

The ECDSA's performance is mostly determined by how well we can implement point multiplication on the specified elliptic curve. These operations are more complicated than the modular inversion stages, and they cost variable amounts of field inversions, multiplications, and squaring depending on the elliptic curve chosen. In other words, the elliptic curve chosen will have an impact on the ECDSA implementation's performance.

Exploiting group endomorphisms is one way to speed up point multiplication. A map $\emptyset: E \mapsto E$ such that $\emptyset(\infty) = \infty$ and $\emptyset(P) = (g(P), h(P))$ for every $P \in E$, where *g* and *h* are rational functions with coefficients in *K*, is an endomorphism of *E* over *K*. The endomorphism ring of *E* over *K* is formed by all endomorphisms of *E* over *K*. The fact that certain endomorphisms allow for very fast computation of operations that are costly on the regular curve is the key to improving point multiplication speed. Significant reductions in computing time can be realised if the mapping to the endomorphism is sufficiently inexpensive.

For ECDSA, Bitcoin uses a Koblitz curve, which is a good choice in terms of efficiency. The Frobenius map is an endomorphism that transfers the Koblitz curve to the condition $\tau(\infty) = \infty, \tau(x, y) = (x^2, y^2)$.. Furthermore, $(\tau^2 + 2)P = \mu\tau(P)$ for all $P \in (F2m)$. Any integer *k* can be represented by an extension into powers of if we consider the Frobenius map to be a complex number.

$$[m]P = [m_0]P + [m_1]\tau(P) + [m_2]\tau^2(P) + \cdots + [m_r]\tau^R(P) \text{ with}$$

$$m_i \in \{0, 1, -1\} \qquad (9)$$

Because squaring in the normal basis provides an essentially free left shift of the bits, combining this attribute with normal basis representation for elements of $E(F2m)$ is extremely helpful.

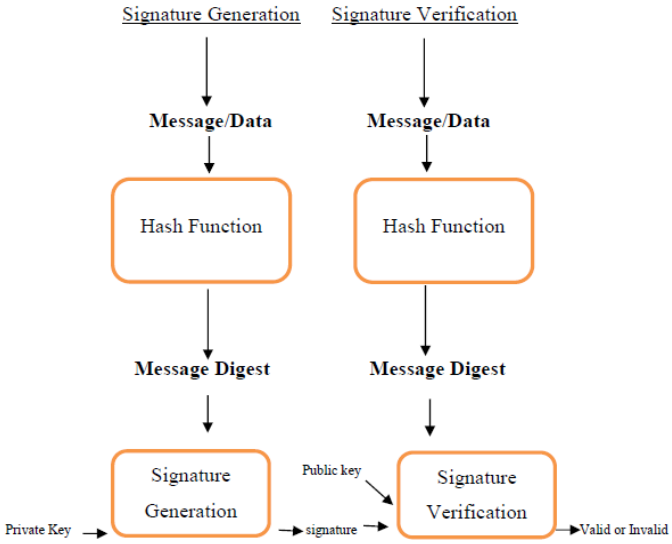
For endomorphisms that are relatively easy to compute, point multiplication can be sped up by approximately 33%, with the Frobenius map over Koblitz curves outperforming even this estimate. This makes Koblitz curves such as the one used in Bitcoin especially attractive for ECDSA.

3. Research Methodology

3.1 Flow Generator Signature – Signature Verification

Figure 3.1 Flow Generator Signature – Signature Verification

The two steps of the digital signature procedure are signing and verifying. Digital signatures are used to verify that the information is signed by the person who claimed to be signing it and to check for malicious modifications. The procedure for creating signature lists and checking signatures. When signing a message, the private key is used to create the signature, which is then confirmed using the public key. Figure 3.1 shows the overall process. In fact, instead of signing the message directly, cryptographic hash functions are applied in the original message to produce a message digest for performance reasons.



One of the cryptographic primitives that has seen a surge in use recently is hash functions. The fact that such functions do not encrypt or decrypt communications should be noted. Apart from other fascinating applications, they are a necessary instrument for ensuring data integrity. The protocol was more efficient because the calculations were simpler and less bandwidth was required when sending the signed data because the hash was a much shorter element.

3.2 Secp256k1's Parameter

Secp256k1 refers to the parameters of the elliptic curve used in Bitcoin's public-key cryptography. The name represents the specific parameters of curve:

- **sec:** stands for **S**tandards for **E**fficient **C**ryptography.
- **256:** length in bits of the field size.
- **k: K**olbitz curve, as opposed to random. The non-random construction allows for efficient construction.
- **1:** sequence number

The general equation of the secp256k1 curve is $y2=x3+7$. Let's represent this curve using the classes we have already defined.

Below are the public specs for Bitcoins's curve – Secp256k1:

- Pcurve = 2256−232−29−28−27−26−24−1
- 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
- Acurve = 0, Bcurve =7, This is define the curve: $y2=x3+Acurve*x+Bcurve$

- Gx= 55066263022277343669578718895168534326250603453777594175500187360389116729240
- Gy= 326705100207588169780830851305070431844712733806592432759389043357573374824 24
- $GPoint$=(Gx,Gy)

3.3 Operation in ECDSA

I. Point Addition

To add two points *P* and *Q* on an elliptic curve, find the third point *R* where line joining *P* and *Q* is equal to – *(P + Q).* Reflecting the point along the X-axis will give us *P + Q*. Example in fig. 1 Point Addition ECDSA.

II. Point Doubling

Point doubling or "adding a point to itself' follows a very simple equation and of course it requires the slope and the domain parameter (refers to fig. 2 Point doubling).

$$For\ X\ coordinate: R_x = S^2 - 2 * P_x \quad (10)$$

$$For\ Y\ coordinate: R_y = -\left(P_y + S(R_x - P_x)\right) \quad (11)$$

$$R_x = S^2 - 2 * P_x R_Y = -1 * (P_y + S * (R_x - P_x)) \quad (12)$$

And against the same catch, R becomes the negatives reflection of R, so it looks something like:

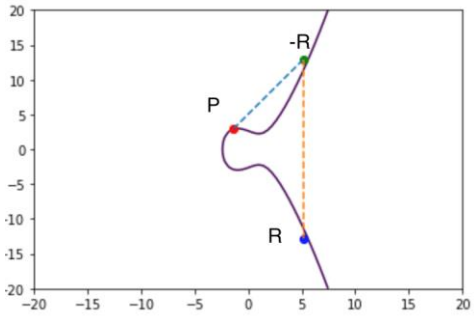
Figure 3.2 Point Doubling

III. Point Multiplication

The straightforward way of computing a point multiplication is through repeated addition. However, there are more efficient approaches to computing the multiplication.

i. Double and add

The simplest method is the double-and-add method, similar to square-and-multiply in modular exponentiation. The algorithm works as follows:



To compute sP, start with the binary representation for s:

$$s = s_0 + 2s_1 + 2^2 s_2 + \cdots + 2^m s_m, \text{where } s_0 .. s_m \in \{0,1\},$$

$$m = \log_2 S \quad (10)$$

ii. Window Method

In the window method version of this algorithm, one selects a window size w and computes all $2w$ values of dP for $d = 0,1,2,\cdots,2^w - 1$. The algorithm now uses the representation $d = s_0 + 2^w d_1 + 2^{(2w)} d_2 + \cdots + 2^{mw} d_m$.

3.4 Proved the Ownership of Bitcoins

I. Setup

1. The elliptic curve group *E(a, b, p)* with parameters a, b, p order either prime *n* or divisible by prime n.
2. The primitive element *P* ∈ *E*, Which is of order n
3. The private key which random integer *d* ∈ [2, 2*n*]
4. The public key which is a point on the curve *Q* = [d]*P*

II. Signing

1. Generate a random integer r ∈[2,2n]
2. Compute [r]P = ($x1$,$y1$)
3. Compute the integer $s1$=$x1(mod\ n)$
4. If $s1$=0, stop and go to Step 1
5. Compute $r-1(mod\ n)$
6. Compute $s2$=$r-1(H(m)+d*s1)(mod\ n)$
7. If $s2$=0, stop and go to Step 1
8. The signature on the message *m* is the pair of integers ($s1$,$s2$)

III. Verification

1. The verifier receives the message and the signatures: [m,$s1$,$s2$]
2. The verifier knows the system parameters and the public key Q
3. The integers $s1$,$s2$ are in the range [1, *n*1]
4. Compute w=$s2$ −1 $(mod\ n)$
5. Compute $u1$=$H(m)*w(mod\ n)$
6. Compute $u2$=$s1*w$ $(mod\ n)$
7. Compute $u1P \oplus [u2]Q$=($x2$,$y2$)
8. Compute the integer v=$x2$ $(mod\ n)$

   The signature is valid if v=$s1$

4. Result and Disccussion

4.1 Secp256k1's Parameter and Data

1. Prove Ownership

- P = 115792089237316195423570985008687907853269984665640564039457584007908834671663
- N = 115792089237316195423570985008687907852837564279074904382605163141518161494337
- $Gx=$ 55066263022277343669578718895168534326250603453777594175500187360389116729240
- $Gy=$ 32670510020758816978083085130507043184471273380659243275938904335757337482424
- $G=(Gx,Gy)$

2. Fake Transaction

- Pcurve = $2256-232-29-28-27-26-24-1$
- N = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
- Acurve = 0, Bcurve = 7
- $Gx=$ 55066263022277343669578718895168534326250603453777594175500187360389116729240
- $Gy=$ 32670510020758816978083085130507043184471273380659243275938904335757337482424
- Point =(Gx,Gy)
- privKey = 75263518707598184987916378021939673586055614731957507592904438851787542395619
- randNum = 28695618543805844332113829720373285210420739438570883203839696518176414791234
- HashOfThingToSign = 86032112319101611046176971828093669637772856272773459297323797145286374828050

4.2 Result

1) Result of python to prove the ownership of bitcoin.

```
C:\Users\user\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/user/PycharmProjects/pythonProject/main.py
******* Public Key Generation *********
 G(x,y) * privatekey(scalar) == publicKey(x,y)
public key x =  40512939985856747849916395927822149728201583913717859810043523594654503692227
public key y =  88166831356626186178414913298033275054086243781277878360288998796587140930350
public key (x,y) is on curve True
*************************************
publicKey / privatekey == G  True

Process finished with exit code 0
```

Figure 4.1 Result

2) Result of python for Bitcoin's Fake transaction

- Private key Result

```
C:\Users\user\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/user/PycharmProjects/pythonProject/main.py
Private Key:
752635187075981849879163780219396735860556147319575075929044388517875423956l9

Public Key public key (uncompressed):
04 (616831830690341450731307316444055058805950887490692587988451894619830471716O0, 385853073628473005567626877408442428212710429005313579590097126733918626262l)

Public Key (compressed):
02885f71c4561e1733119c66fce72d2209771e096a8305ff8fd36a405afbcbbe10
```

Figure 4.2 Private Key Result

- Result

```
******* Signature Generation *****
r = 879172294281107893665614225873070729700886951502146039003512946368042982907l8
s = 106502088372271315150771838464197128617761506858172050145503629250806341274845

******* Signature Verification ********>>
False

Process finished with exit code 0
```

Figure 4.3 Result

4.3 Chapter Summary

Based on the result we got by using coding on Python, we know that only the holder of the private key can use or open the Bitcoin's wallet and the result of the coding showing the private key we enter is true and prove the ownership of the Bitcoin. For the second coding we use private key and random number to do the transaction and the result is false, which mean the outsider try to do a transaction without the owner knowing about the transaction because the signature generation we got are not the same as the owner hold**.**

Acknowledgement
The researcher would like to thank all people who have supported the research. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my thesis supervisor, Dr. Syarafina Mohamed, for encouragement, guidance, critics and friendship. I am also very thankful to my final year project's member for their guidance, advices and motivations.

References

[1] Ajeena, R. K. (2014). Point multiplication using integer sub- decomposition for elliptic curve cryptography. Applied Mathematics & Information Sciences, 517-525.
[2] Antony, S. N. F. M. A. and Kamarulhaili H. (2018). ISD method implementation over curves with j-invariant 0. AIP Conference Proceedings.
[3] Asep S., Teguh P. N. (2020). Implementation of Elliptic Curve Diffie-Hellman (ECDH) for Encoding Messeges Becomes a Point on the GF($\boldsymbol{pp}$). International Journal of Advanced Science and Technology, 3264-3273.
[4] Bahaweres, Y. I. (2019). The Cryptocurrency Simulation using Elliptic Curve Cryptography Algorithm in Mining Process from Normal, Failed, and Fake Bitcoin Transactions. 7th International Conference on Cyber and IT Service Management (CITSM), 1-8.
[5] Berutu. (2013). Peramalan Penjualan Dengan Metode Fuzzy Time Series Ruey Chyn Tsaur.
[6] Dawahdeh, Z. a. (2016). A new modification for Menezes-Vanstone elliptic curve cryptosystem. Journal of Theoretical and Applied Information Technology, 290-297.
[7] Dua M. Ghadi and Adil Al-Rammahi. (2020). Improvement of Menezes-Vanstone Elliptic Curve Cryptosystem Based on Quadratic Bézier Curve Technique. in Journal of Computer Science, 715-722.
[8] Farris Ian Gillies, Chun-Teck Lye, Lee-Ying Tay. (2020). Determinants Of Behavioral Intention to Use Bitcoin in Malaysia. Journal Of Information System and Technology Management (JISTM), 25-38.
[9] Gallant R. P. (2001). Faster point multiplication on elliptic curve with efficient endomorphism. CRYPTO2001, 190-200.
[10] Ichsani, Y. A. (2019). The Cryptocurrency Simulation using Elliptic Curve Cryptography Algorithm in Mining Process from Normal, Failed, and Fake Bitcoin Transactions. International Conference on Cyber and IT Service Management (CITSM), (pp. 1-8).
[11] Johnson, D. M. (2014). The Elliptic Curve Digital Signature Algorithm (ECDSA). International Journal of Information Security.
[12] K. G, P. P. (2020). IOT applications for high security WSN using Elliptic Curve Cryptography Algorithm. Journal of emerging technologies and innovative research.
[13] Kapoor, Abraham & Singh. (2008). Elliptic curve cryptography. ACM Ubiquity, 1-8.
[14] Kavyashree B., G. D. (2016). Outline Of Modified Menezes Vanstone Elliptic Curve. International Research Journal of Engineering and Technology (IRJET) , 1044-1048.
[15] Koblitz, N. (1987). Elliptic curve cryptosystem. Mathematics of computation, 203-209.
[16] Martínez, V. G.-Á. (2020). Analysis of the Cryptographic Tools for Blockchain and Bitcoin. Mathematics, 131.
[17] Radek F., Jiri M., Petr M., Leonard J. (2016). Cryptograph key distribution with elliptic curve Diffie-Hellman algorithm in low-power devices for power grids. Revue Roumaine des Sciences Techniques - Serie Électrotechnique et Énergétique, 84-88.
[18] Singh, S. R. (2007). A Simple Method of Forecasting Based on Fuzzy Time Series. Applied mathematics and computation, 330-339..
[19] Sio-long Ao, A. H.-s. (2011). Elliptic Curve Cryptography Sensor Networks with Optimum Controlling. IAENG Transactions on Engineering Technologies, 374-388.

[20] Song, W. W. (2017). privacy-preserved full-text retrieval algorithm over encrypted data for cloud storage applications. J. Parallel Distrib. Comput., 17-27.
[21] Stepen A. B., J. B. (2020). Elliptic Curve Diffie-Hellman (ECDH) Analogy for Secured Wireless Sensor Networks. International Journal of Computer Applications, 1-8.
[22] Subramanian, E.K., Tamilselvan, L. (2020). Elliptic curve Diffie–Hellman cryptosystem in big data cloud security. Cluster Comput 23, 3057-3067.
[23] V.S., M. (1986). Use of Elliptic Curves in Cryptography. Advances in Cryptology — CRYPTO ’85 Proceedings, 417-416.
[24] Wang, D. (2014). Secure Implementation of ECDSA Signatures in Bitcoin. Computer Science.
[25] White, Preston and Ingalls, Ricki. (2015). Introduction to simulation. 2015 Winter Simulation Conference (WSC), (pp. 1741-1755).
[26] Yang, C. H. (2018). File changes with security proof stored in cloud service systems. Pers. Ubiquit. Comput, 45-83.
[27] Yoshida, H. a. (2005). Analysis of a SHA-256 Variant. 12th International Workshop, (pp. 245-260). Canada.
[28] Zhou, Z., Hu, Z., Xu, M. and Song, W. (2010). Information Processing Letter, 1003-1006.