Application of Lenstra–Lenstra–Lovasz on Elliptic Curve Cryptosystem Using IOT Sensor Nodes

Md Sameeruddin Khan[1], Thomas M. Chen[2], Mithileysh Sathiyanarayanan[3], Mohammed Mujeerulla[4,*] and S. Pravinth Raja[5]

[1]*Post-Doctoral fellow, City, University of London, United Kingdom and Pro-Vice chancellor & Dean at School of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, India*
[2]*Professor, Cyber Security, City, University of London, United Kingdom*
[3]*University of Brighton, U.K*
[4]*Associate Professor, School of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, India*
[5]*Post-Doctoral fellow, City, University of London, United Kingdom and Professor & Head of the Department, School of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, India*
E-mail: mohammedmujeerulla@presidencyuniversity.in; mujerroshan@gmail.com
[*]*Corresponding Author*

Received 17 January 2024; Accepted 06 September 2024

Abstract

The Internet of Things (IoT) model is presented in this paper with multi-layer security based on the Lenstra-Lenstra-Lovasz (LLL) algorithm. End nodes for the Internet of Things include inexpensive gadgets like the Raspberry Pi and Arduino boards. It is not practical to run rigorous algorithms on them, as opposed to computer systems. Therefore, a cryptography procedure is required that could function on this IOT equipment. Bitcoins and Ethereum are examples of cryptocurrency and Ripple employs techniques such as elliptic curve digital signature, Elliptic-Curve Diffie-Hellman (ECDH), and algorithm to sign any cryptocurrency on SECP256k1 elliptic curves transactions. By using Lenstra-Lenstra-Lovasz on a real-world Bitcoin blockchain and applying it to multiple dimensions, such as nonce leakage and weak nonces across several elliptic curves with different bit sizes on a Raspberry Pi, we can demonstrate the security of elliptic curve cryptosystems. Public key encryption techniques are seriously threatened by the development of quantum computing. Therefore, employing lattice encryption with Nth Degree Truncated Polynomial Ring Units (NTRU-NTH) on the Bitcoin blockchain will increase the resistance of Bitcoin blocks to quantum computing assaults. The execution time taken on SECP256k1 is 131.7 Milli seconds comparatively faster than NIST-224P and NIST-384P.

Keywords: RAG – Random number generator, EdDSA – edwards curve digital signature algorithm nonce – number only used once, The NIST – National Institute of Standards and Technology, SEC – U.S. securities and exchange commission, ECC – elliptic curve cryptography, IoT – Internet of Things.

1 Introduction

Security has taken on a greater significance because of networks' quick development. Digital signatures are now an essential component of security authentication since they may confirm the truthfulness and dependability of a message. Due to a few unique circumstances, the original signer must provide the authorized proxy signer the authority to forge an authentic signature on their behalf. The Shor algorithm [3] has demonstrated that current cryptography based on conventional number theory, such as the ElGamal algorithm and Rivest, Shamir, and Adleman (RSA) algorithm, will no longer be reliable under quantum attacks. Research on quantum computers has become a hot topic in recent years. The American NIST is likewise openly seeking postquantum cryptography techniques.

Existing cryptography typically involves RSA (Rivest, Adi Shamir and Leonard Adleman) which is used extensively in creating digital signatures, elliptic curve cryptography [1, 19] which is used in encrypting signatures and discrete log method which is used in ECDH key exchanges. Unfortunately, these cryptographic algorithms are at great risk from quantum computers. In RSA we have message M, two Prime numbers P and Q, and C and D are the cipher and decypher values. We then compute N = P*Q with ϕ = (P-1)(Q-1). The generated cipher and decypher for message M are $C = M^e MODN$ and

$D = C^d(MODN)$ respectively, where e is the encryption key value so that it is not share of values from P and Q. In elliptic curve cryptography we have equation $y^2 = x^3 + ax + b$ Mod P whose working is based on discrete logarithm values, where x and y values are from 0 to P-1, a and b are constants. We then use a private key d as a scalar value and multiply with generator point G. We then use ESDA and EdDSA to sign and verify the messages. In regards to the discrete log, Alice creates a =$g^a MODP$, Bob creates b =$g^b MODP$, and finally, both Alice and Bob recover the values of K = $a^b(MODP)$ and K = $b^a(MODP)$ respectively. An alternative to these cryptographic techniques discussed above is prone to quantum attacks and a technique that still creates a hard problem in the era of quantum computers is Nth degree truncated polynomial ring (NTRU) – Key Encapsulation Mechanism (KEM) [4]. NIST has defined many standards when it comes to encryption such as AES, SHA-3, lightweight cryptography, and post-quantum cryptography methods. In post-quantum cryptography methods, we have a key exchange mechanism to replace discrete logs and digital signatures. In digital signatures, we have NTRU, Dillitium, and Falcon. Both NTRU and Falcon employ lattice cryptography to be robust against quantum computing attacks.

Let's look at how cryptography works, a lattice is a collection basis that can vary from single to N dimensions [2, 18]. A vector from the point of the origin a vector value is represented as a polynomial value. A polynomial value looks something like this $(5x^2 + 1) * (3x + 2)$ we end up in $15x^3 + 10x^2 + 3x + 2(MOD11)$ on prime field P = 11. Substituting further we get $4x^3 + 10x^2 + 3x + 2(MOD11)$ We then pick up a value of N to constrain the values and in this case, we might have N = $4(x^4 - 1)$ this will enable us to constrain the values that we have into what is called the ring. These are the two main operations that we need to do when we have our message M of pattern M = [1,0,1,1] which can be easily represented as polynomial as $1 + x + x^3$.

Let us take an example to demonstrate lattice encryption and decryption with N = 11, p = 3, q = 32 we create factors $[-1, 0, 1]$ of the polynomials as selected in our case $(5x^2 + 1) * (3x + 2)$ and take the value of f as our first polynomial $f = -1 + x + x^2 - x^4 + x^6 + x^9 - x^{10}$ we can also represent f as $[-1, 1, 1, 0, -1, 0, 1, 0, 0, , 1, -1]$ and select second polynomial as $g = -1 + x^2 + x^3 + x^5 - x^8 - x^{10}$. We then find the inverse of f with MOD Q and MOD P as $f.f_q(MODq)$ we obtain 1 and $f.f_p(MODp) = 1$, we get $f_p = 9x^{10} + 5x^9 + 16x^8 + 3x^7 + 15x^6 + 15x^5 + 22x^4 + 19x^3 + 18x^2 + 20x + 5$. So Bobs private key and public key is (f, f_q) and $h = p.f_q.f(MODq)$ respectively. Now we take a random polynomial $r = 3x^2 + 5x - 1$ for message

M. To encrypt a message M we multiply a random polynomial r with Bob's public key and add the message M.i.e encryption = $(r.h + M)$. To decrypt this we multiply the encrypted value $(r.h+M)$ with one of Bob's private key elements like f. So we end up with decryption as $(r.h + M).f.(Modq)$.

When we expand the equation we get $r.p.f_q f + Mf(MODq)$ and we know that $f.f_q(MODq) = 1$ and our result is $r.p + M.f$. Then we multiply $r.p + M.f$ with $f_p(MODP)$ to get the message M.

In the proposed work, the NTRU-Nth degree truncated polynomial ring-key Encapsulation Mechanism (KEM) is used. This algorithm is based on a secure lattice encryption method that is faster than other public key protocols like RSA and faster when decrypting due to the absence of polynomial factorization in the case of RSA and the discrete logarithm problem in the case of elliptic curves. Adding a layer of protection against quantum computing attacks on Bitcoin transactions for the Blockchain the work contributes a high level of security. This paper is organized as follows Section II presents related work. Section III provides a theoretical principle of elliptic curve digital signature (ECDSA) and the LLL Algorithm. Section IV describes the methodology in three parts, A A-Potential pitfall I: Crack ECDSA due to leak of nonce, B-Potential pitfall II: Crack ECDSA due to weak nonces using LLL, C- Lattice Encryption using NTRU – Nth degree Truncated polynomial ring – Key Encapsulation Mechanism (KEM), and Section V summarizes our conclusions and future work.

2 Related Work

The most popular LR method is known as the Lenstra-Lenstra-Lovász (LLL) algorithm [5]. It was given a name based on the names of the creators. The LLL method, however, poses numerous difficulties because of its increased processing complexity and unpredictable execution time. According to this [6], the lattice reduction algorithm's complexity is only dependent on the size of the matrix and the lattice basis norm. The LR algorithm's computational cost can be impacted by the matrix structures (i.e., the correlation between the columns) of a given lattice matrix, which is often determined by its condition number or determinant. In this study, obtained a tighter upper bound on the complexity of the LLL algorithm in terms of the condition number and determinant of a specific lattice matrix to examine how the matrix structures can affect the program's complexity. Here [7], an ABPS system was developed that can withstand quantum attacks. The new technique permits a signer whose attributes meet the access structure to assign their signing privileges to other individuals. This system offers a few characteristics, including absolute privacy and fine-grained access. The processing, storage, and security comparisons were assessed using performance analysis.

A Proxy Re-Signature (PRS) enhancement known as Attribute-based Proxy Re-Signatures (ABPRS) [8] enables a semi-trusted proxy to change a signature from one entity into another without disclosing the signing key or the identity of the original signer. The suggested primitive, which has various applications, combines PRS and Attribute-based Signatures (ABS) capabilities. Additionally, by utilizing a general transformation technique, a multi-hop unidirectional ABPRS scheme that fulfills a better security notion of co-selective unforgeability was produced [12]. [12] focuses on exploiting implicit key leakage caused by improper cryptographic primitive usage and searches the Bitcoin Blockchain for ECDSA nonce reuse. Additionally, systematically describes how an attacker might exploit duplicate r values to leak secret keys and nonces, which goes beyond the straightforward situation in which the same key and nonce have been used several times. The findings demonstrate that attackers have already taken advantage of ECDSA nonce reuse, which has been a persistent issue in the Bitcoin ecosystem [17]. The proposed work provides a resistance of Bitcoin blocks to quantum computing assaults. Also provides an additional security layer and increases the performance in terms of execution time.

3 Theoretical Principle

3.1 Elliptic Curve Digital Signature (ECDSA)

ECDSA is a digital signature scheme on digital currencies like Bitcoin and Ethereum. The security of the ECDSA relies on the selection of nonce values to be truly random otherwise it can significantly reduce the security of the signature. Looking at the table performance of EdDSA seems to be in and around ECDSA, Further it supports the clustering of signatures to merge them in the process of signing a message. This is due to the underlying technique used in EdDSA is based on the Schnorr signature method.

1. In Schnorr signature, signature (R, S) is created for message M.
2. Initially we generate the private key (x) and later we derive the public key using a point on the elliptic curve to obtain P = x. Generator point G
3. A random vale k is selected to obtain the signature values of R = random k* Generator G
4. Then s is derived from S = random k – Hash (Message M, R)*x.

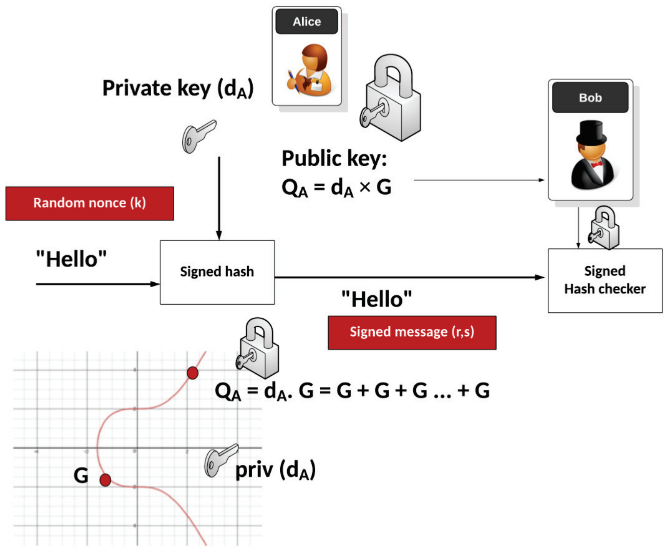

Figure 1 ECDSA [13].

5. We therefore have public key P and signature (S, R) for message M
6. To validate the signature we compute public key P*Hash (Message M, R)+S * Generator G
7. Therefore we obtain x*Generator G*Hash (M, R)+(k-Hash(M, R)*x)*G this gives x*G*Hash(M, R)+k*G–Hash(M, R)*x*G=k*G
8. The value of k*G is equal to R, and so if the result is the same as R, the signature matches.

Alice applies ECDSA to sign the message as follows:

1. Compute message hash $e = Hash(message)$
2. Let h be the L_n be the leftmost bits of e, L_n has a bit length of the group order N.
3. Select a random number nonce k such that 1 $<$ nonce k $<$ N – 1
4. Compute a point on the curve as (x_1, y_1)= k * Generator point G
5. Compute coordinate $r = x_1(MODN)$. If coordinate r=0, jump to step 3.
6. Compute coordinate $s = k^{-1}(h + rdA)(MODN)$ If coordinate s=0, jump to step 3.

7. The pair of signature obtained is (r, s)

Bob verifies the message as follows:

1. Compute message hash $e = HASH(msg)$
2. Let h be the least significant bits L_n of e
3. Compute c = coordinate $s^{-1}(MODN)$
4. Compute u_1 = h * c MOD N and u_2 = r.c $(MODN)$
5. The signature is invalid if the curve point coordinates $(x_1, y_1) = u_1*$Generator G $+u_2 * QA$ and if $(x_1, y_1) = 0$

6. If r $\cong$ x1 (MOD N) then the signature is valid and is shown in figure 1.

4 Methodology

Some of the Fortune 500 companies have identified quantum computing as a strategic project and significant differentiator in marketplace [15]. It's clear quantum computing represents a fundamentally new approach to data security but people realize the danger the quantum computers pose to confidential data. Lattice-based encryption relies on very difficult mathematical problems involving lattices. A lattice is a mathematical structure that can be used to represent an infinite grid of points. Every lattice is formed from a basis vector that, when multiplied together, can form any point in the lattice. A 2D lattice with a basis of [0, 2] and [1,0] would look like Figure 2,

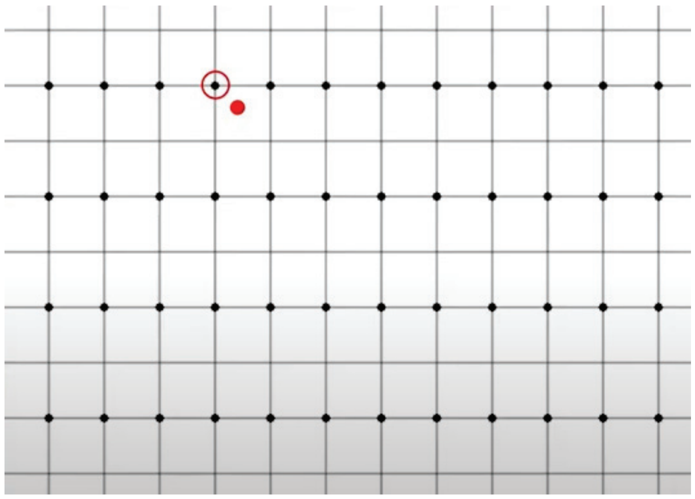

Figure 2 Lattice with Noise $\vec{e}$.

A lattice can be comprised of an infinite number of dimensions and an infinite number of basis vectors. To use lattice for encryption we came up with a really hard problem that involves them and the best problem we have so far is the closest vector problem. The closest vector problem simply is, when you are given a lattice find the closest point in that lattice to any given nonlattice point. In other words, we pick a random point that is not exactly a lattice point but is close to one of the points in the lattice for example the red dot in the below lattice we need to find out the closest black dot to the red dot in the given lattice. In this example, it is pretty clear that the black dot encircled with a red boundary is the closest to the red dot. This problem can get complicated with an increase in several lattices and dimensions. Then we try to encode the data in a secure lattice point which is difficult to find unless you derive a solution to the closest vector problem. In lattice-based cryptography, lattice structures are created by scaling several dimensions every time, and a random point on the lattice is selected as our vector. Then we add a small error to this lattice vector and a new problem is created to compute the original vector in the lattice. The existing pub key ciphers like RSA and ECC [9] have a serious threat due to advancements in quantum computers. The underlying problem on which ECC [10, 11, 14] and RSA are based like discrete logarithm problems and modulus arithmetic would no longer be a challenge to quantum computing. As an infant step, we begin the application of lattice-based cryptography on 2 – a dimensional lattice having an x-axis and a y-axis. Suppose we select two points such as $p1 = (2, 0)$ and $p2 = (0, 2)$ and we represent them as vector $\vec{v_1} = [2, 0]$ and vector $\vec{v_2} = [0, 2]$ respectively, where (0,0) is the origin point of the vector. Where we have vector $\vec{v_3} = \vec{v_1} + \vec{v_2} = [2, 2]$ and vector $\vec{v_4} = 2\vec{v_1} + \vec{v_2} = [4, 2]$, this is shown in Figure 3.

We introduce difficulty in lattice cryptography by having small noise $\vec{e}$ upon the original vector, this leads to difficulty in finding the actual vector for our defined lattice in Figure 2. With this, we have the closest vector problem and short vector problem. In regards to the closest vector problem, we need to find the grid point in the lattice closest to our original point. In regards to the shortest vector problem, we need to find a short vector that is close to the origin and not a long vector that is far from the origin.

4.1 Potential Pitfall I: Crack ECDSA Due to Leak of Nonce

Figure 4 pictorially shows the Leak of Nonce, Tables 2 and 3 show the crack of ECDSA, due to the leak of nonce using NIST-224P and SECP112R2 recommended parameters.

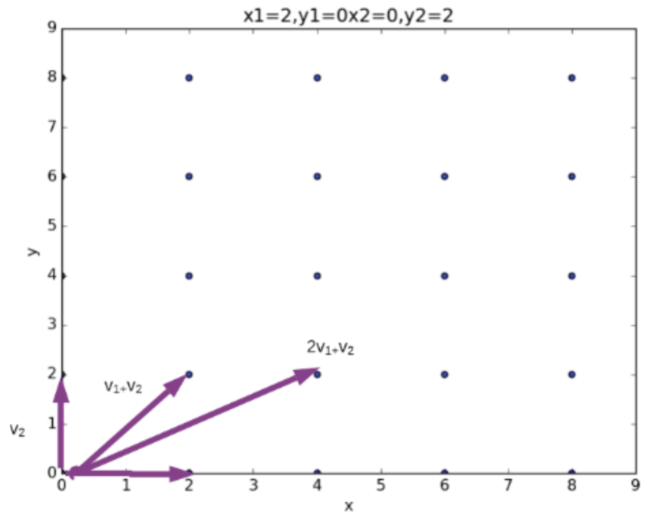

Figure 3 Example 1- Lattice with three different vectors.

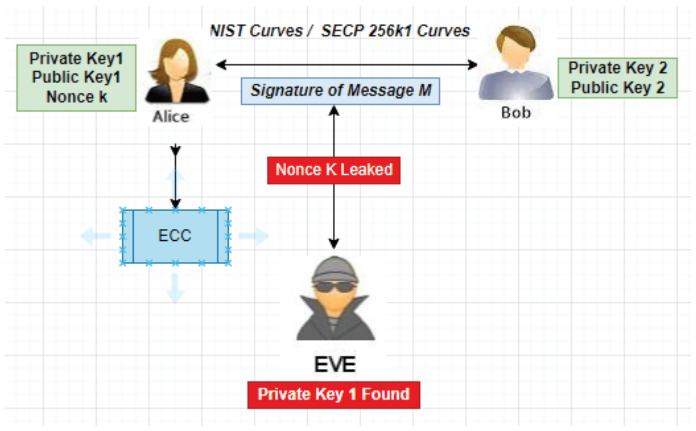

Figure 4 Leak of nonce.

4.2 Potential Pit Fall II: Crack ECDSA Due to Weak Nonces Using LLL

Lattice attacks to find nearest vectors: Suppose we have two vectors $\vec{v_1}$ and $\vec{v_2}$ which connect from the origin to two points and are orthogonal (if there are 90 degrees between vectors) one can easily find the nearest point. By applying Baini's algorithm we find the solution to $w = a\vec{v_1} + \vec{v_2}$ if not we get an incorrect answer. To test the orthogonal match between vectors $\vec{v_1}$ and

Table 1 Lattice nearest point

('Find nearest to: ', 40, 10)
('Cos(theta): ', -0.1351132047333135)
('Grid point: ', 8, 2)
('Grid point: ', -4, 10)
('Solution (not rounded): ', 5.0, 0.0)
('Solution: ', 5.0, 0.0)
('Nearest: ', 40.0, 10.0)
== Try for other vectors on lattice ==
('Grid point: ', -2, 8)
('Grid point: ', 3, 9)
('Cos(theta): ', 0.8436614877321074)
('Solution (not rounded): ', -5.0, 6.0)
('Solution: ', -5.0, 6.0)
('Nearest: ', 28.0, 14.0)
== Try for other vectors on lattice ==
('Grid point: ', -4, 10)
('Grid point: ', 4, 12)
('Cos(theta): ', 0.7633862853691145)
('Solution (not rounded): ', -5.0, 5.0)
('Solution: ', -5.0, 5.0)
('Nearest: ', 40.0, 10.0)
== Try to solve for the point generated in the first part ==
('Grid point: ', 56, 58)
('Grid point: ', 156, 160)
('Solution (not rounded): ', -55.0, 20.0)
('Solution: ', -55.0, 20.0)
('Nearest: ', 40.0, 10.0)

$\vec{v_2}$ we compute

$$\cos\theta = \left(\frac{\vec{v_1}\vec{v_2}}{|\vec{v_1}|.|\vec{v_2}|}\right)$$

We want $\cos\theta$ to be zero. Figure 5 pictorially represents weak nonce and table IV shows a crack of ECDSA with weak nonce using NIST-384P recommended parameters. Suppose $\vec{v_1} = (5,1)$ and $\vec{v_2} = (-2,8)$ and we want to find nearest point (27,8), we initially determine the angle between the points (5,1) and $(-2$,8) is around 90 degrees, then we solve for the nearest neighbor problem using Babai's algorithm then we have the lattice shown in Figure 6. Figures 7 and 8. show a lattice with three different vectors and angles between the basis in the lattice respectively.

We now solve a (5) + b(1) = 27 and a(-2)+ b(8) = 8. By solving the linear equation for a and b we get a=5.52 and b=0.309. By rounding a=6 and b=0 we

Table 2 ECDSA: CRACK ECDSA, if nonce known (NIST-224P recommended parameters)

N=26959946667150639794667015087 0196259404578077144243917216827223680 61
a= -3
b=18958286285566608000408668544493 9 26415504680968679321075787234672564
h=1
Gx=19277929113566293071110308034 69 94880268319342194524401566497843520 33
Gy=19926808758034470970197974370 88 87491842059919906039495376373431987 72
Message 1: Journal of The Institution of Engineers (India): Series B
Sig 1(R,S):20295994277414429292950706516 58 17360889742256827078853020199717379663117379 5 32558753336121063760245326848558369490648244 88859151416487
Private Key:54129281547145903062437588754417504 9 60726350003600491543255563946 51
The private key is found:54129281547145903062437588754417504 9 60726350003600491543255563946 51
Execution time:162.4 Milli seconds

Table 3 ECDSA: CRACK ECDSA, if nonce known (SECP112r2 recommended parameters)

N=36DF 0AAFD8B8 D7597CA1 0520D04B
a=6127 C24C05F3 8A0AAAF6 5C0EF02C
b=51DE F1815DB5 ED74FCC3 4C85D70
h=04
G=04 4BA30AB5 E892B4E1 649DD092 8643ADCD 46F5882E 3747DEF3 6E956E9
Message 1: Journal of The Institution of Engineers (India): Series B
Sig 1(R,S):527349671015913583819624162679876 317691106918931019007002247480372
Private Key:527349671015913583819624162679876 317691106918931019007002247480372
The private key is found:527349671015913583 81 9624162679876 317691106918931019007002247480372
Execution time: 131.7 Milli seconds

get vector point (30, 6) which is near to (27,8). Let us take another example with lattice vectors (102,113) and (37,41) then the angle (Cosθ) between them is closer to zero and therefore unable to solve and we get a solution of (-53,10) which gives us the wrong point (-4,-26) and is not closer to (27,8). This is shown in Figure 8.

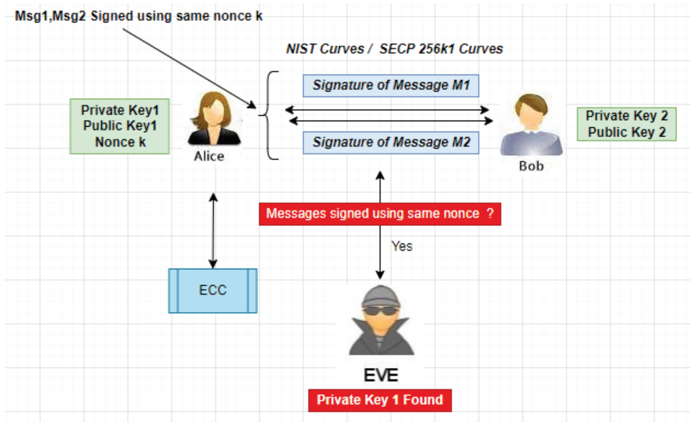

Figure 5 Weak nonce.

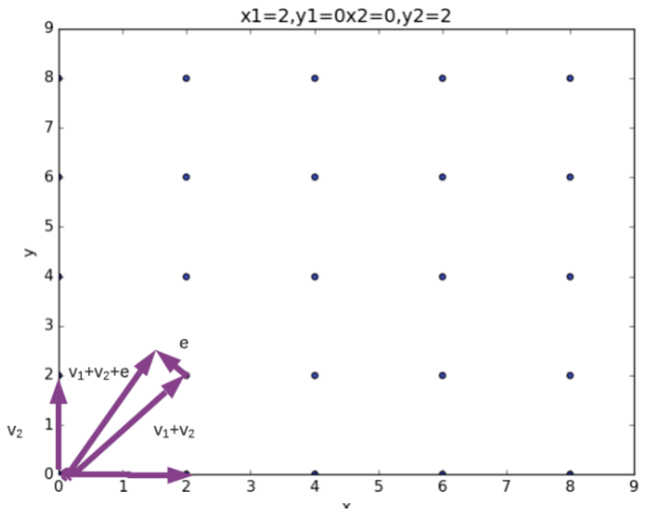

Figure 6 Example 3 – Lattice with three different vectors and noise vector $\vec{e}$.

Let us consider an example to find a reasonably orthogonal basis for the lattice spanned by the vectors using LLL

Step 1: $\vec{v1} = (15, 23, 11)$ $\vec{v2} = (46, 15, 3)$ $\vec{v3} = (32, 1, 1)$

$\vec{v1}^*$ = (15,23,11) $\vec{v2}^* \approx (27.69, -13.07, -10.43)$

Table 4 ECDSA: CRACK ECDSA, with weak nonce (NIST-384P recommended parameters)

N=394020061963944792122790401001 436138050797392704654466679 4
a=-3
b=2758019355995970587784901184 0389048093056905856361568521 42
h=01
Gx=2624703509579968926862315674 4 566981891852923491109213387 81
Gy=8325710961489029985546751289 5 201081792878530488613155947 0
Message 1: Journal of The Institution of Engineers (India): Series A
Message 2: Journal of The Institution of Engineers (India): Series B
Sig 1(R,S):342326145854884889885254094893134480 7 0856003877366432728946930309982753983144581479655 98 5593209398584422401171173255162588572669706618117 01 5417949490740035277436062654182624362565311800538 73 1934361326017179689401068877422834637391 40
Sig 2(R,S): 311641734663763730497757529074880912 01 0170439394662760498602995905438685059640925244144 84 0948102483554129248522463884855535812568880049366 21 3115156900728464606031313206717483899402977126940 45 347455566476060871970222078555038561 539
K1:1642709644932997069077598375492591741 62
K2:1325487128568524215751164837283687907 9
Private Key:179641037423842868867582096685 4052295961607500727369417229093392295123351591938 0 888328428588819991925102919506687486
The private key is found:179641037423842868867582096685 4052295961607500727369417229093392295123351591938 0 888328428588819991925102919506687486
Execution time: 1761.9 Milli seconds

we take $\vec{v1}^{*} = (46, 15, 3) - \frac{(46,15,3).(15,23,11)}{(15,23,11).(15,23,11)}(15, 23, 11)$

$\approx (27.69, -13.07, -10.43)$

in practice if $\vec{vk}$ is the only working vector, we only need the Gram-Schmidt basis = $\vec{v1}^{*}$, $\vec{v2}^{*}$,.......$\vec{vk}^{*}$ and we won't need the rest.

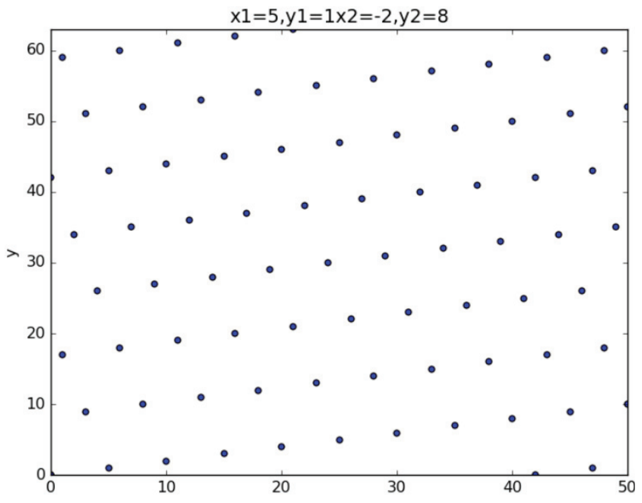

Figure 7 Example 3 – Lattice with three different vectors.

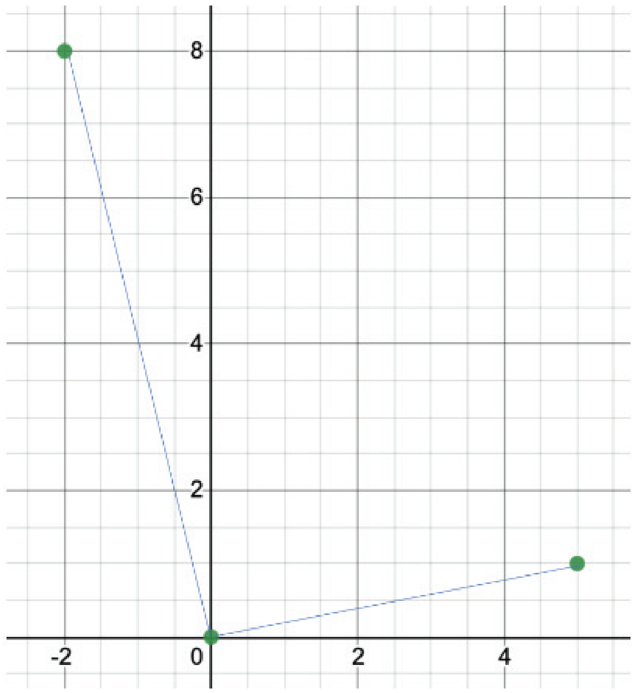

Figure 8 Angle between basis in lattice.

Step 2: $\vec{v1} = (15, 23, 11)$ $\vec{v2} = (46, 15, 3)$ $\vec{v3} = (32, 1, 1)$
$\vec{v1}^*$ = (15,23,11) $\vec{v2}^* \approx (27.69, -13.07, -10.43)$

Now use $\vec{v1}$ to reduce $\vec{v2}$ $\vec{v2} = (46, 15, 3)$ $--\lfloor\mu\rceil$ (15,23,11)

This is our new $\vec{v2} = (31, -8, -8)$

Checking the Lovaz condition:

$||\vec{v1}^*||^2 = 875||\vec{v1}^*||^2 \approx 1046.43$

$$\mu_{2,1} = \frac{\vec{v2}.\vec{v1}^*}{\vec{v1}^*.\vec{v1}^*}\vec{v1}^{*2} = 875\vec{v1}^{*2}\mu_{2,1} =\approx 0.221$$

$$\frac{3}{4}] - \mu_{2,1}^2 \approx 0.701$$

4.3 Lattice Encryption Using NTRU – Nth Degree Truncated Polynomial Ring – Key Encapsulation Mechanism (KEM)

A lattice-based public key method NTRU- Nth degree Truncated polynomial ring uses shortest vector problem and is three twice faster than other public key protocols like RSA and three times faster while decrypting due to non-usage of polynomial factorization in regards with RSA or discrete logarithm problem in regards with elliptic curves. To send a secret message between Alice and Bob requires the generation of private and public keys. The private key is known only by Bob while the public key is known by Alice and Bob. To generate private and public key pair we select two polynomials f and g with degree N-1on coefficients $-1, 0, 1$. The polynomial $f \in Lf$ selected by Bob must satisfy a requirement that inverse modulo p and modulo q must satisfy while computed using the Euclidean algorithm. In simple words $f * fq = 1$ and $f * fp = 1MODp$. If the selected f is not invertible Bob has to try with another f. Bob's private keys are polynomials $f$and f_p. The public key h is computed using h= polynomial$pfq * g(MODq)$. Alice and Bob mutually agree on the largest polynomial N, p, and q and then two short polynomials f and g are generated by Bob to generate his key pairs. The coefficients for these polynomials are $-1, 0$ and 1.

1. Let us consider an example with p=2, q=32 and N=11

Table 5 NTRU for key encapsulation mechanism Ntruhps 2048509

Public key size: 699
Secret key size: 935
Cipher text size: 699
Public key (only showing 1/8 of key):= bc2ea5d3461cc42f37bd3c6f8b8d598606b174800b21445f83d89 60101e4afb3a71e99dc54635a044f1e9ee3790098bc33a094d06e 0a7b1ce8525cc64650603fd6826eb9084200f827fe94abfa9494b a67a5acc0a11453
Secret key (only showing 1/8 of key):= af4a26827c639017d9a5224ecfbb18915d466b44745ccfe245c 9c40a5fdeb7ae5608bc11dd3c29dbbf011b8954490ad95e8cbe b07e6f41de025b887d9b4c3f606e2816c22590cae81d7c47544b bfde5ce4afd6395999c65270e564e7659f10606486ac02d0174a 070d8cd3279ccad75d4a6b0eec
Cipher text: : 61afb3a85ad1931380de5ce5f1d 8b5612cffa1f66d2437cc3f21ebafe22e227e6688b179ab79db 0c5e100efae821b2862ec0baf538d07047a920eb2c435992795 3d2191c25d3ab2b97a947a8217e872c5f92899e8a6f51
Key (A): 4b8cfd5a5c49d8ede5c343c7ad2058568 153ad0a1ecdeee7e330318fc662f800
Key (B):4b8cfd5a5c49d8ede5c343c7ad2058568153ad0 a1ecdeee7e330318fc662f800
Keys are the same

2. If Bob has values for polynomials $f = [-1, 1, 1, 0, -1, 0, 1, 0, 0, 1, -1]$ then polynomial representation is $f = -1 + a + a^2 - a^4 + a^6 + a^9 - a^{10}$ then has picks $g = -1 + a^2 + a^3 + a^5 - a^8 - a^{10}$
3. The we should be able to compute f^{-1} for MOD p and MOD q and therefore polynomial $f * polynomial f_q(MODq) = 1$ and polynomial $f * polynomial f_p(MODp) = 1$ Due to inverse function we get $f_p = 9a^{10} + 5a^9 + 16a^8 + 3a^7 + 15a^6 + 15a^5 + 22a^4 + 19a^3 + 18a^2 + 29a + 5$
4. The public key $h = p * polynomial fq * polynomial f(MODq)$, then obtain $fq * g = [-5, -9, -1, -2, 11, 12, 13, 3, 22, 15, 16, 29, 60, 19, -2, -7, -36, -40, -50, -18, -30]$. In order to create a ring we then divide by aN-1 and get Bob's Public Key H : $[24, 19, 18, 28, 4, 8, 5, 17, 4, 17, 16]$ with private key$f$ and f_q
5. Encryption is done by taking Bob's public key (h), random polynomial, and message M to compute $Encrypt = r * h + MessageM$

Table 6 NTRU for key encapsulation mechanism Ntruhps 2048677

Public key size:930
Secret key size:1234
Cipher text size:930
Public key (only showing 1/8 of key):=
3d30ad496e845fbe86b1bd731d550ec77fbcebf47578c2
4510c18f42329dce118b2c852b8acc977aa63609153780
68d49d7cd8d631c3a5701907f882fd8b71f76c13046fe3
e0c7beb5c88866cb15455c9d8deb3f82dc2c493a9fed5f
63dde2574bab7f0f20db0a91d20f9e0e3d4f56a0112af600
Secret key (only showing 1/8 of key):=
8479e9e09c3d27bfb9a786650f433b2fb128d4ee0d73a2
c869bcd18b909f46d25f2d112174e51f8b576467ea43b
dd26bd60ba0003d0a13469fe7ccdec8d38136bdb8308a
de6398b2c21fb7cf2d87989212d0848952130298ad249
f71e743e2e317daeddb6cab9a4072a645818b776e49ae
1652801322a13c3f1bd513b7d21f0ad2738179b833ae0
11d504838addd040888c1abdb815a1e19d9f1
Cipher text:
8479e9e09c3d27bfb9a786650f433b2fb128d4ee0d73a
2c869bcd18b909f46d25f2d112174e51f8b576467ea43
bdd26bd60ba0003d0a13469fe7ccdec8d38136bdb8308
ade6398b2c21fb7cf2d87989212d0848952130298ad24
9f71e743e2e317daeddb6cab9a4072a645818b776e49a
e1652801322a13c3f1bd513b7d21f0ad2738179b833ae
011d504838addd040888c1abdb815a1e19d9f1
Key (A):
47f897119dcb315df84ad5a9bb59a710af36bcc6d425cb415c069924f094661d
Key (B):
47f897119dcb315df84ad5a9bb59a710af36bcc6d425cb415c069924f094661d
Keys are the same

6. To decrypt we initially multiply Bob's private key fq and Mod q we follow the below steps,
    - Decrypt = $(r * publickeyh + MessageM) * polynomialf(Modq)$
    - Decrypt = $(p * r * polynomialfq * g + M) * polynomialf(Modq)$
    - Decrypt = $(p * r * g + MessageM * polynomialf)$

Table 7 Elliptic curve point generation times in milli seconds

Algorithm	Ciurves	Execution Time
ECDSA Signatures	NIST-P192	178.78
	NIST-P224	185.74
	NIST-P256	191.84
	SECP-160k1	151.4
	SECP-192k1	157.36
	SECP-224k1	163.18
	SECP-256r1	176.74
	Brainpool-p160r1	157.94
	Brainpool-p192r1	166.04
	Brainpool-p224r1	187.66
	Brainpool-p256r1	190.1
	ECDSA RFC6979	281.4
EdDSA Signatures	Ed25519	174.5
	Ed448	285.54

- Decrypt = $(p * r * g + MessageM * polynomialf) * polynomialfp(Modp)$
- Decrypt = $p * r * polynomialf * polynomialfp + MessageM * polynomialf * polynomialfp(Modp)$
- Decrypt = $0 + MessageM * polynomialf * polynomialfp(Modp)$
- Since polynomial $f * polynomialfp(Modp)$ will be 1, we obtain Decrypt = Message M.

Tables 5 and 6 demonstrate NTRU for key encapsulation mechanism on NTRUHPS 2048509 and NTRUHPS 2048677. Table VIII demonstrates lattice encryption using NTRU – Nth degree Truncated polynomial rings respectively. Figure 9 pictorially shows the performance of ECDSA vs. EdDSA across various curves, TABLE VII shows point generation times in milliseconds. Table IX provides the execution time of ECDSA on standard curve databases like NIST-224P, SECP112R2, NIST384P, and SECP256K1 by keeping the N size constant. The table concludes that the proposed ECDSA on SECP112R2 consumes less execution time compared to other curves as mentioned.

Table 8 Lattice encryption using NTRU – Nth degree truncated polynomial ring

Algorithm	**n**	**p**	**q**	**Result**
Moderate Security	250	5	256	==== Bob generates public key ===
				f(x)= [1, 1, -1, 0, -1, 1]
				g(x)= [-1, 0, 1, 1, 0, 0, -1]
				d = 2
				Bob's Public Key:
				[235L, 117L, 220L, 164L, 205L, 97L, 210L, 87L, 164L, 71L, 206L, 254L, 29L, 132L, 32L, 148L, 171L, 80L, 247L, 205L, 65L, 49L, 183L, 80L, 219L, 101L, 252L, 2L, 133L, 7L, 21L, 248L, 160L, 218L, 212L, 233L, 147L, 144L, 253L, 168L, 255L, 166L, 198L, 139L, 146L, 160L, 18L, 83L, 198L, 155L, 157L, 63L, 209L, 67L, 144L, 85L, 205L, 250L, 32L, 159L, 249L, 211L, 76L, 6L, 160L, 64L, 217L, 33L, 82L, 111L, 124L, 59L, 114L, 230L, 153L, 12L, 196L, 188L, 187L, 116L, 255L, 109L, 145L, 149L, 135L, 124L, 47L, 81L, 208L, 118L, 13L, 139L, 1L, 48L, 104L, 70L, 152L, 221L, 243L, 200L, 125L, 144L, 3L, 98L, 86L, 31L, 170L, 212L, 202L, 211L, 130L, 123L, 253L, 135L, 37L, 91L, 76L, 153L, 81L, 126L, 196L, 7L, 117L, 191L, 252L, 6L, 100L, 236L, 181L, 65L, 210L, 247L, 164L, 223L, 86L, 174L, 85L, 148L, 56L, 180L, 43L, 200L, 7L, 61L, 65L, 153L, 231L, 232L, 3L, 61L, 20L, 42L, 5L, 95L, 125L, 248L, 96L, 242L, 140L, 225L, 19L, 96L, 77L, 104L, 23L, 158L, 102L, 83L, 194L, 24L, 114L, 147L, 78L, 155L, 13L, 175L, 25L, 227L, 168L, 221L, 53L, 114L, 136L, 31L, 193L, 155L, 60L, 246L, 232L, 232L, 161L, 1L, 146L, 111L, 220L, 243L, 122L, 86L, 145L, 220L, 60L, 124L, 251L, 204L, 143L, 125L, 145L, 189L, 151L, 20L, 151L, 169L, 200L, 94L, 237L, 131L, 137L, 144L, 136L, 158L, 240L, 181L, 51L, 152L, 237L, 112L, 251L, 218L, 118L, 231L, 26L, 172L, 10L, 19L, 42L, 123L, 13L, 119L, 173L, 27L, 36L, 97L, 249L, 214L, 44L, 231L, 221L, 231L, 76L, 86L]

(*Continued*)

Table 8 Continued

Algorithm	n	p	q	Result
				==== Alice generates public key =====
				Alice's Original Message : [1, 0, 1, 0, 1, 1, 1]
				Alice's Random Polynomial : [-1, -1, 1, 1]
				Encrypted Message :
				[187L, 74L, 177L, 96L, 97L, 155L, 55L, 25L, 24L, 54L, 126L, 155L, 226L, 215L, 83L, 161L, 249L, 157L, 216L, 19L, 29L, 154L, 190L, 23L, 177L, 227L, 242L, 74L, 66L, 58L, 23L, 123L, 148L, 223L, 146L, 177L, 250L, 2L, 171L, 118L, 126L, 0L, 39L, 164L, 139L, 155L, 23L, 1L, 253L, 20L, 101L, 153L, 200L, 232L, 49L, 235L, 117L, 150L, 40L, 40L, 138L, 191L, 93L, 98L, 93L, 58L, 193L, 126L, 62L, 29L, 168L, 50L, 54L, 219L, 230L, 127L, 107L, 185L, 189L, 149L, 20L, 207L, 73L, 94L, 106L, 175L, 53L, 143L, 178L, 34L, 22L, 102L, 211L, 3L, 196L, 143L, 162L, 29L, 70L, 162L, 183L, 102L, 122L, 72L, 71L, 176L, 171L, 211L, 215L, 101L, 109L, 32L, 81L, 93L, 252L, 20L, 25L, 7L, 177L, 110L, 72L, 20L, 222L, 243L, 197L, 250L, 149L, 122L, 237L, 194L, 198L, 225L, 88L, 94L, 254L, 123L, 250L, 135L, 19L, 241L, 161L, 221L, 80L, 107L, 149L, 18L, 246L, 55L, 233L, 203L, 2L, 10L, 170L, 66L, 159L, 171L, 148L, 175L, 66L, 121L, 178L, 226L, 99L, 182L, 230L, 0L, 103L, 236L, 171L, 91L, 183L, 41L, 77L, 140L, 29L, 225L, 96L, 192L, 49L, 83L, 93L, 86L, 120L, 0L, 130L, 119L, 45L, 210L, 221L, 234L, 169L, 230L, 206L, 37L, 104L, 250L, 86L, 251L, 158L, 239L, 11L, 137L, 37L, 181L, 140L, 167L, 129L, 182L, 162L, 47L, 77L, 23L, 34L, 130L, 190L, 142L, 59L, 179L, 196L, 191L, 178L, 133L, 62L, 66L, 239L, 38L, 130L, 168L, 135L, 88L, 139L, 243L, 119L, 77L, 93L, 88L, 137L, 165L, 244L, 172L, 121L, 79L, 121L, 142L, 184L, 172L, 54L, 139L, 213L, 170L]
				==== Bob decrypts =====
				Decrypted Message : [1L, 0L, 1L, 0L, 1L, 1L, 1L]

Table 9 ECDSA execution time on different standard curves

ECDSA on different standards	**Execution time in milli seconds**
NIST-P224 (Proposed)	162.4
SECP112R2 (Proposed)	131.7
NIST-384P (Proposed)	1761.9
SECP256k1 [20]	2763

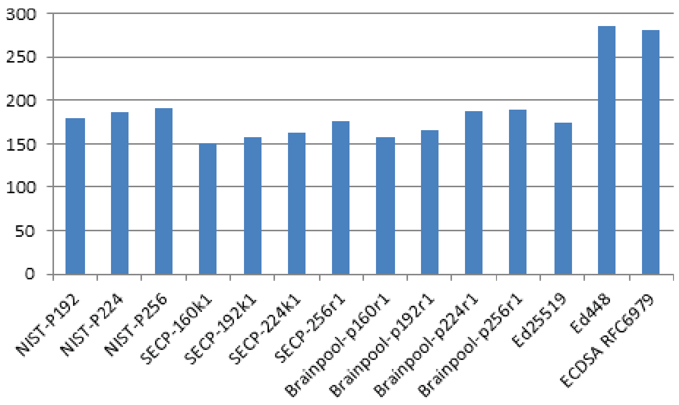

Figure 9 Performance of ECDSA vs EdDSA across various curves.

5 Conclusions

Looking at potential pitfall I – the crack of ECDSA due to the leak of nonce and potential pitfall II – the crack of ECDSA due to weak nonces using LLL in an earlier section of this paper it is very obvious that ECDSA is not secure over Bitcoin blockchain on curves like SECP256k1. Authors urge to keep the Bitcoin blockchain transaction secure by using lattice encryption using NTRU – Nth degree truncated polynomial ring – Key Encapsulation Mechanism (KEM) which is three twice faster than other public key protocols like RSA and three times faster while decrypting due to the non-usage of polynomial factorization regarding RSA or discrete logarithm problem in regards to elliptic curves. Thus providing an added level of security for the Bitcoin blockchain against quantum computing attacks on Bitcoin transactions.

Acknowledgement

The authors would like to acknowledge the support provided by Presidency University – Bengaluru,India.

References

[1] Chintan Patel, Nishant Doshi. Secure Light Weight Key Ex-Change Using ECC For User Gateway Paradigm "*IEEE Transactions on Computer*" DOI: 10.1109/TC.2020.3026027 (2021).

[2] Mohammed Mujeer Ulla; Preethi; Md. Sameeruddin Khan; Deepak S. Sakkari. Implementation of Elliptic Curve Cryptosystem with Bitcoin Curves on SECP256k1, NIST256p, NIST521p, and LLL "*Journal of ICT Standardization, River Publications*," ISSN: 2246-0853. DOI: 10.13052/jicts2245-800X.1141 (2024).

[3] Mohammed Mujeer Ulla; Preethi; Md. Sameeruddin Khan; Deepak S. Sakkari. Demerits of Elliptic Curve Cryptosystem with Bitcoin Curves Using Lenstra–Lenstra–Lovasz (LLL) Lattice Basis Reduction "*Arabian Journal for Science and Engineering*". https://doi.org/10.1007/s13369-023-08116-w (2023).

[4] Mohammed Mujeer Ulla and Deepak S. Sakkari. Application of Elliptic Curve Crypto System to Secure Multi-Signature Bitcoin Block Chain "*Journal of Computer Science, Science Publications*", ISSN: 1552-6607. https://doi.org/10.3844/jcssp.2023.112.125 (2023).

[5] Kamal, A., Ahmad, K., Hassan, R., Khalim, K. NTRU Algorithm: Nth Degree Truncated Polynomial Ring Units. In: Ahmad, "*EAI/Springer Innovations in Communication and Computing*". Springer, Cham, https://doi.org/10.1007/978-3-030-60890-3_6 (2021).

[6] Nizar Ouni and Ridha Bouallegue. Performance and Complexity Analysis of Reduced Iterations LLL Algorithm "*International Journal of Computer Networks Communications (IJCNC)*" May Vol.8. DOI: 10.5121/ijcnc.2016.8309 (2016).

[7] Yunju Park and Jaehyen, Analysis of the upper bound on the complexity of LLL Algorithm, "*Journal of the Korean Society for Industrial and Applied Mathematics*" Vol. 20, No. 2, 107–121. DOI: 10.12941/jksiam.2016.20.107 (2016).

[8] Wei, L.; Li, D.; Liu, Z. Provable Secure Attribute-Based Proxy Signature Over Lattice Small Integer Solution Problem in Random Oracle Model.

"*Electronics*", 12, 1619. https://doi.org/10.3390/electronics12071619. (2023).

[9] Mohammed Mujeer ulla, Md Sameeruddin Khan, Preethi, and Deepak S.Sakkari. Security and Performance Analysis of Elliptic Curve Crypto System using Bitcoin Curves. "*IAENG International Journal of Computer Science*", ISSN: 1819-656X. (2023).

[10] Badis Hammi, Achraf Fayad, Rida Khatoun, Sherali Zeadally and Youcef Begriche. A Lightweight ECC-Based Authentication Scheme for Internet of Things (IoT), "*IEEE Systems Journal*" Pages: 3440–3450 DOI: 10.1109/JSYST.2020.2970167 Volume: 14. (2020).

[11] Xiaoqiang Zhang And Xuesong Wang. Digital Image Encryption Algorithm Based on Elliptic Curve Public Cryptosystem. "*IEEE Access Pages: 70025 – 70034*" ISSN: 2169-3536 Volume:6. DOI: 10.1109/ACCESS.2 018.2879844 (2018).

[12] Mohammad Ayoub Khan, Mohammed Tabrez Quasim, Norah Saleh Alghamdi, Mohammad Yahiya Khan, A Secure Framework for Authentication and Encryption Using Improved ECC for IoT- Based Medical Sensor Data. "*IEEE Access Pages: 52018 – 52027*" ISSN: 2169-3536 Volume: 8. DOI: 10.1109/ACCESS.2020.2980739 (2020).

[13] Michael Brengel and Christian Rossow. Identifying Key Leak- Age of Bitcoin Users. International Symposium on Research in Attacks, Intrusions, and Defenses. "*Open Access LNSC*", ISBN: 978-3-030-00470-5 volume 11050. DOI: 10.1007/978-3-030-00470-5_29. (2018)

[14] Joachim Breitner and Nadia Heninger Biased Nonce Sense: Lattice Attacks against Weak ECDSA Signatures in Cryptocurrencies, "*Lecture Notes in Computer Science Springer International Publishing -Financial Cryptography and Data Security*". DOI: 10.1007/978-3-030-32101-7_1. (2019).

[15] Javed R. Shaikh, Maria Nenova, Georgi Iliev and Zlatka Valkova-Jarvis. Analysis of Standard Elliptic Curves for the Implementation of Elliptic Curve Cryptography in Resource-Constrained E-commerce Applications. "*IEEE-COMCAS*" ISBN:978-1-5386-3169-0. DOI: 10.1109/COMCAS .2017.8244805. (2017).

[16] Shen Guicheng, Yu Zhen. Application of Elliptic Curve Cryptography in Node Authentication of Internet of Things. "*IEEE-IIHMSP*" ISBN:978-0-7695-5120-3 DOI: 10.1109/IIH-MSP.2013.118. (2013).

[17] Ravi Kishore Kodali and Ashwitha Naikoti. ECDH-based Security Model for IoT using ESP 8266. "*IEEE- ICCICCT*" DOI: 10.1109/IC CICCT.2016.7988026 (2016).

[18] Deepak S. Sakkari Mohammed Mujeer ulla. Review on Insight into Elliptic Curve Cryptography. "*Modern Approaches in Machine Learning Cognitive Science: A Walkthrough*" DOI: 10.1007/978-3-030-96634-88 (2022).

[19] Deepak S. Sakkari Mohammed Mujeer ulla Design and Implementation of Identifying Points on Elliptic Curve Efficiently Using Java. "*Modern Approaches in Machine Learning Cognitive Science: A Walkthrough*" DOI: 10.1007/978-3-030-96634-88. (2022).

[20] Deepak S. Sakkari Mohammed Mujeer ulla. Design and Implementation of Elliptic Curve Digital Signature Using BitCoin Curves Secp256K1 and Secp384R1 for Base10 and Base16 Using Java. "*Innovation in Electrical Power Engineering, Communication, and Computing Technology*" DOI: 10.1007/978-981-16-7076-328 (2022).

Biographies

Md Sameeruddin Khan (Senior Member, IEEE) is currently serving as Pro-Vice chancellor and Dean at school of computer science and engineering, Presidency university, Bangalore. In 2019 he pursued his Doctor of Philosophy in Computer Science and Engineering from Rayalaseema University, Kurnool, Andhra Pradesh and in 2001 he pursued his master of Technology in Computer Science and Engineering from Visveswaraih Technological University, Belgaum. He has many papers to his credit in reputed international journals and conferences. His areas of expertise include internet of things, wireless sensor network and cryptography.

Thomas M. Chen (Senior Member, IEEE) is currently a Professor in cyber security with the City, University of London. Previously, he was a Senior Technical Member of Staff with the GTE Laboratories (now Verizon), Waltham, MA, USA. He joined Southern Methodist University, Dallas, as an Associate Professor and then Swansea University, Wales, as a Professor in networks. He has written or edited seven books, 41 book chapters, and owns two U.S. patents. His research interests include cyber security, online extremism, and computer networks. He was a co-recipient of the Fred W. Ellersick Best Paper Award, in 1996. He has served as the former Editor-in-Chief for IEEE Communications Magazine, IEEE Network, and IEEE Communications Surveys and Tutorials.

Mithileysh Sathiyanarayanan (Member, IEEE) received the Ph.D. degree from the City, University of London. He completed his pre-doctoral fellowship from the University of Brighton, U.K. He is currently an Indian-Born Research Scientist and an Entrepreneur in London. He closely works with “MAKE IN INDIA” initiatives through his organization, MIT Square. Being the Founder and the CEO, his vision is to build Indian products and take it to the world. His research was in collaboration with the Nokia Research, Finland, which won him "Young Scientist Award." He has also won several research awards, entrepreneurship awards, and has been invited as a research consultant at various industries for his research excellence. At an age of 19, he started his career as a Social Entrepreneur and now mentoring several institutions and budding engineers in India. Recently, he was awarded International Achievers Award 2020 and the CEO of the Year 2021 for his innovation and product development. His diligence and stark sight have achieved his startup the recognition as the one of the fastest growing startups in India.,Dr. Sathiyanarayanan has been awarded with several prestigious awards, being a Scientist, a CEO, an Entrepreneur, a Mentor, an Author, and an Educationist among other hats, he wears. This gentleman has accumulated knowledge and experience across multiple industries at a young age and is now putting it all to together to brighten India's tomorrow.

Mohammed Mujeerulla has perceived his doctoral in 2023 from Presidency university, Bangalore and his masters and bachelors degree in computer science and engineering from R.V college of engineering, Bangalore, India in 2015 and 2008 respectively. Since 2017 he is working as associate professor in presidency university Bangalore. He has many papers to his credit in reputed international journals and conferences. His areas of expertise include internet of things, wireless sensor network and cryptography.

Dr. S. Pravinth Raja has over 18 years of experience in teaching and research. He is currently serving as a Professor and the Head of the Department of Computer Science and Engineering at Presidency University. Dr. Pravinth Raja completed his Ph.D. in 2017 at Anna University, Chennai, and later pursued a Postdoctoral Fellowship at City University of London, UK, which he successfully completed in 2023. He has received a research grant of 26 lakhs from the Government of India under the T10KT training program. With more than 50 research articles published in prestigious Q1, Q2, Q3, and Q4 journals and conferences, he has made significant contributions to his field. Additionally, he holds over 16 patents, of which 4 have been granted in the UK, India, and Australia. Dr. Pravinth Raja has supervised 3 Ph.D. scholars and his primary research interests include image processing and machine learning.