Development Of An Enhanced End-To-End Communication Model With Elliptic Curve Cryptography

[1]Oguntunde T. & Fasan M.T.
Department of Computer Science
University of Ibadan, Ibadan, Nigeria
E-mail: [1]tantos557@yahoo.com

ABSTRACT

Secure communication has become a critical concern in modern digital systems due to increasing cyber threats. Elliptic Curve Cryptography (ECC) is one of the most powerful but least understood types of cryptography in wide use today. However, vulnerabilities in traditional ECC models, particularly in key exchange and integrity verification, limit its effectiveness. This study proposes an enhanced ECC-based end-to-end communication model incorporating a Key Derivation Function (KDF) and SHA-256 cryptographic hashing over a prime field F_p, implemented using Python socket programming over TCP. The data used for this study was gathered randomly from SMS, Facebook and a couple of text lines previously used in literature. The traditional ECC algorithm was enhanced using a computed shared secret (S) from the receiver's public key and introduction of a Key Derivation Function (KDF), which was used to generate a symmetric key from S. A cryptographic hash function, H(P_m), was used to ensure the message integrity. Performance evaluation of the enhanced model was carried out in comparison with the existing ECC model and Rivest-Sharmir-Adelman (RSA) scheme using encryption and decryption speed, key-generation time and memory usage as metrics. A message of 1, 10 and 100kb in sizes through the existing and improved ECC model had encryption time of 18.5 and 15.2 (ms), 168.4 and 152.7(ms), 1,650.7 and 1,520.3(ms), respectively, while their decryption time were 17.3 and 14.8(ms), 165.2 and 149.5 (m,s), 1,620.5 and 1,490.1(ms), respectively. Furthermore, key sizes of 256, 384 and 521 (bits) with the existing and enhanced ECC had key generation time of 13.1 and 12.4 (ms), 19.5 and 18.7 (ms), 26.8 and 25.3 (ms), respectively, indicating a 5.3, 4.1 and 5.6 % of the enhanced ECC's better performance than the existing, respectively. The memory utilisations of the existing and the enhanced ECC model were 2.3 and 2.1(MB), 3.8 and 3.5 (MB), 3.7 and 3.4(MB) for key generation, encryption (1kb data) and decryption (1kb data) operations, respectively. This research provides a quantitatively validated enhancement to ECC, balancing security and efficiency. The proposed model could be deployed for secure applications in IoT, blockchain, and messaging systems, offering a practical and forward-compatible cryptographic solution.

Keywords: Elliptic Curve Cryptography, Secure Communication, Key Derivation Function, Cryptographic Hashing, Performance Optimization

CISDI Journal Reference Format
Oguntunde T. & Fasan M.T. (2023): Development Of An Enhanced End-To-End Communication Model With Elliptic Curve Cryptography. Computing, Information Systems, Development Informatics & Allied Research Journal. Vol 14 No 4, Pp 63-78
Available online at www.isteams.net/cisdijournal. dx.doi.org/10.22624/AIMS/CISDI/V14N4P7

1. INTRODUCTION

Information security has its foundation in early civilization when non-complicated encryption procedures were deployed to protect personal transmission. Since the earliest time, cryptography has been progressively evolving, with the growth of better sophisticated methods and techniques. Encryption techniques' became importance has become pronounced in diplomatic transmissions, with governments and diplomatic representatives depending on coded conversations to hide confidential communications and maintain confidentiality. Cryptography at this age was mostly manual and took considerable capacity to encrypt and decrypt communications.

The 19th century was the time that showcased considerable developments with the innovation of mechanical cipher tools such as the Enigma machine. These appliances employed mechanical procedure to encrypt and decrypt communications, resulting to a more secure contents. Charles Wheatstone and Samuel Morse invented early cryptographic gadgets and techniques to protect telegraph transmission, revealing the need for more secure burgeoning field of electronic transmissions. The requirements of World War II and the Cold War dispensation led to all-encompassing cryptographic investigation. The decryption of the Enigma ciphertext by the Allies at Bletchley Park became a significant success, displaying the significance of cryptography in military art of war. The era of computers in the mid-20th century brought both fresh problems and chances, leading to the discovery of the TCP/IP communications suite and coding of the RSA encryption procedure which did a lot in protecting digital communications and confidential information from unauthorized usage and interference (Stallings, 2011). Elliptic Curve Cryptography (ECC) evolved as a promising cryptography substitute to earlier cryptographic ciphers like RSA and Diffie-Hellman. Its source dated back to the mid-1980s when Neal Koblitz and Victor Miller individually proposed elliptic curve-based cryptographic scheme (Koblitz, 1987).

2. RELATED WORKS

Sullivan (2013) discussed the most popular RSA and the Diffie-Hellman key exchange procedures. The new Algorithm represented the first far-reaching cryptographic ciphers whose strength depended on the number theory and was the first to activate protected transmission between two persons with no shared secret. These systems are known as public key cryptographic scheme. RSA, as a simple algorithm, multiplies two prime figures. Its difficult sister algorithm (Diffie-Hellman) factors the product of the multiplication into its two constituent primes. RSA and Diffie-Hellman procedures are characteristic of simple in one way, difficult on the other side and are otherwise called Trap door Functions, which is germane to producing a safe public key cryptographic scheme, the bigger the defference between going the easy way and the difficult one in the RSA and Diffie-Hellman attributes, the safer its cryptographic scheme. RSA's performance is based on the principle, factoring lags and multiplication is better. A public key encoding scheme has two constituents, a public key and a private key.

RSA and Diffie-Hellman were so secure since they have rigorous security proofs, Sullivan showed that breakage of the ciphertext is synonymous to offering solution to an impossible mathematical problem. The factoring procedures are faster and less computational than the simple method of pairs of primes guess. As the capacity to decode numbers improves, key-size needed to grow even more fast but this situation is not the best for mobile and low-powered gadgets having constrained computational power. RSA is not the best scheme for the future of cryptography.

In RSA:

1. Determine a maximum figure, labeled max, a result of multiplying two random prime numbers
2. Pick another number to be the public key (pub)
3. Compute the private-key (priv) from the public key provided both prime numbers are available.
4. both the public (***pub***) and private (***priv***) keys have to be larger than zero and less than the maximum (max) value
3. To encode a number, do a multiplication of it by itself ***pub*** times, making sure to wrap around to a number in the valid range when the results of the calculation is a number larger than the maximum chosen.
4. While decoding a message, multiply the message by itself ***priv*** times to get back the original number

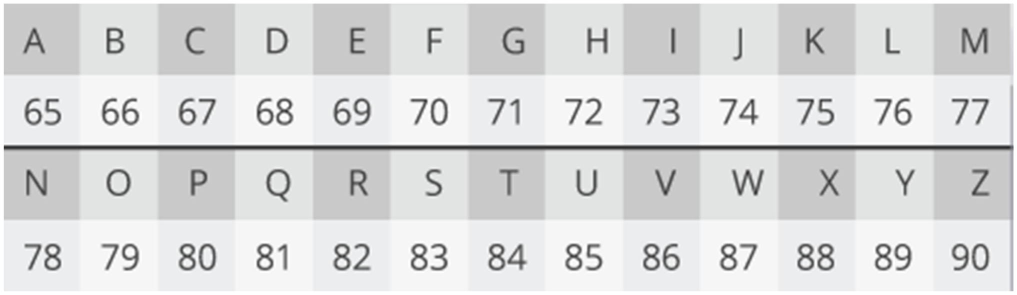
For instance:

1. Use prime numbers 13 and 7, whose multiplication gives max. result of 91
2. take the public-key to be 5
3. Having known that 7 and 13 are the factors of 91 and using a procedure known as the Extended Euclidean Algorithm, the private-key is calculated to be 29.

Note: The parameters: max = 91; pub = 5; priv = 29, make a fully workable RSA cipher

4. To encrypt a number, multiply it by itself 5 times
5. To realize the initial number again, then take that figure and multiply it by itself 29 times
   Employ these parameters, encode the plaintext "CLOUD":

1. To encode a plaintext mathematically, the letters should be turned into numbers, UTF-8. A character to a number is one-to-one



A	B	C	D	E	F	G	H	I	J	K	L	M
65	66	67	68	69	70	71	72	73	74	75	76	77
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
78	79	80	81	82	83	84	85	86	87	88	89	90

Fig 2.1: Encoding a Plaintext Mathematically

2. In this representation, CLOUD is 67, 76, 79, 85, 68.
   Note: Each of these numbers is smaller than the max, 91, encrypting each individually. Starting with the first letter:
3. Multiply 67 by itself five times to obtain the encoded result.
   67×67 = 4489 = 30 *
   * 4489 is larger than the max, so, it has to be wrapped around by dividing with 91 and taking the remainder.

4489 = 91×49 + 30
30×67 = 2010 = 8
8×67 = 536 = 81
81×67 = 5427 = 58
Meaning when 67 is encoded, it gives 58

. repeat the procedure for each letter in CLOUD, the encoded plaintext will be:
58, 20, 53, 50, 87

5. To decode this plaintext, let each figure be multiplied by itself 29 times:
58×58 = 3364 = 88, wrapping around if the result is more (>) than the max.
88×58 = 5104 = 8

6. 9×58 = 522 = 67
The calculation give 67 again. Repeat this for the rest of the digits, to produce the initial plaintext.

Sullivan (2013) recalled that in 1985, cryptographic procedures went complicated past factoring, a Trapdoor Functions built on a complex mathematics aspect, elliptic curves. An elliptic curve is a number of points agreeable with a particular mathematical expression having 2 changeable location with degree two in first variables and three in the second. The expression for an elliptic curve such as this: $y^2 = x^3 + ax + b$.

Elliptic Curve Cryptography (ECC) has top-bottom mirror imaging nature. A location on the curve can be mirrored along the x axis of that same curve. Furthermore, any non-vertical line may meet the curve on at least, three other locations. A Table game employs two locations "dot." meaning two given locations on a curve can be dotted together to get a new location, A dot B = C or a string "dotting" a location with itself over and over produce: A dot A = B, A dot B = C and A dot C = D. Moreover, if it happens that, in a two points, the initial point "dotted" with itself n times to provide a final location, calculating n is tough, since just the final and the first locations are given. For Elliptic Curve cryptography, RSA restricts only to whole numbers in a static range rather than allow any value for the points on the curve. Manipulating the formula for the elliptic curve ($y^2 = x^3 + ax + b$) uses same twist of wrapping numbers that is greater the maximum. If the maximum is a prime number, the elliptic curve is recognized as a prime curve and has perfect cryptographic nature. In elliptic curve representation, message could be represented as locations on the curve as x coordinate to obtain a location, y on the curve.

An elliptic curve cryptosystem can be explained by determining a prime number as the maximum, a curve equation and a public location on the curve. A private key, ***priv***, and a public key, a location dotted with itself ***priv*** times. Determining the private-key from the public-key is referred to as the Elliptic Curve Discrete Logarithm (ECDL) function, the sought Trapdoor Function. The ECDL is spectacularly tougher to solve than factoring, need greater computational power, meaning it is a stronger, harder to break cryptographic scheme than RSA and Diffie-Hellman. With ECC, you can use smaller keys to get the same levels of security especially for mobile devices. RSA security operates with increasing key length, at the cost of slower cryptographic performance but ECC operates having high security with short and fast keys. Elliptic curve cryptography is applied in wide variety of applications: U.S. government's internal communications, the Tor project's assurance of anonymity, bitcoins' ownership, Apple's signatures in iMessage, encryption of DNS information with DNSCurve and secure web browsing authentication over SSL/TLS. A number of sites' pages' SSL certificate

built with elliptic curves, associated to RSA or DSA key pair state ECDHE_RSA or ECDHE_ECDSA, respectively.
ECDHE stands for Elliptic Curve Diffie-Hellman Ephemeral, an elliptic curves key exchange mechanism to give excellent forward secrecy in SSL. The RSA component means that RSA proves the identity of the server while in others, ECDSA (the Elliptic Curve Digital Signature Algorithm) proves the servers' identity. Use of ECC saves time, power, computational resources and more secure for both the server and the browser. Dual Elliptic Curve Deterministic Random Bit Generator (Dual_EC_DRBG) generates haphazard numbers with the Elliptic Curve mathematics. The algorithm itself involves taking points on a curve and repeatedly performing an elliptic curve "dot" operation.

According to Sanders (2022), ECC is a public key cryptographic procedures employed for critical security functions: encryption, authentication, and digital signatures. It produces keys from elliptic curve equation while the traditional technique factors very large prime numbers. ECC keys, much shorter than RSA's, making them a lot easier to run, persist and requiring less processing power to encode and decode data. Elliptic Curve Cryptography's benefits include: ECC is considered more secure than RSA as it would take timeless age to reverse engineer ECC generated key. Moreover, shorter keys mean less processing power to encrypt and decrypt data. Smaller keys requirement than other methods to achieve the same level of security with lower computing power and battery resource usage, adaptable widely in cryptocurrency platforms, including Bitcoin and Ethereum, and IoT devices. ECC are suitable for digital signatures and key exchange. Elliptic curves' promising mathematical attributes include: First, "Point addition" and "Point doubling." A point can be kept doubling until the "the infinity point", O.

ECC functions built upon the Elliptic Curve Discrete Logarithm Problem (ECDLP) asserted that it is tough finding x if y = g^x **mod** p is known and where g is some known integer and p is a prime number. Therefore, if parameters g and p are chosen carefully, it should be hard for an individual who does not know the secret exponent, x to compute x given y (or vice versa). ECC is very resourceful in Pretty Good Privacy (PGP), a well known email encoding tool using ECC against unauthorized access. PGP functions by generating a public/private key pair for individual sender and receiver. To encode an email, the recipient's public key (that can be divulged to any other user) is needed; conversely, and the private key (must be kept secret) will be needed to decode the email received.

Kumar (2025) recognized numerous germane kinds of procedures employed in public-key cryptography, e.g. RSA (uses factorization), ECC (employs elliptic curves determined over numerable fields), DSA, DH (leverages discrete logarithms), and ElGamal, each hinging on various mathematical phenomena to enforce safety. ECC is a Public-key Cryptography with two keys: one is a public key and the other, a secret and private key. Kumar reiterated that Neal Koblitz and Victor S. Miller invented the use of elliptic curves to enable safer security: small key sizes especially, implemented on Mobile, IoT, modern secure protocols. He asserted that, for safe communications over an untrusted network, an undisclosed key needed to be provided for efficient and safe data transformation or for access controls. The shared point of the two parties on the elliptic curve is their undisclosed key in a way that, if someone is spying and taps into the curve, the initial location 'G', party_1's public key 'A', and party_2's public key 'B', it's practically tough for them to unfold party_1's secret number, 'a' or party_2's secret number, 'b'. The reason for the difficulty being the fact that, discovering the distance to reach 'A' or 'B' from 'G' is a tough mathematical problem referred to as the Elliptic Curve Discrete Logarithm Problem (ECDLP).

The strength of ECC are practically shorter key lengths at comparable safety level, less storage usage, more capacity for transmission plus faster computation; mathematics of elliptic curves affords stronger and advanced safety features such as Perfect Forward Secrecy (PFS) when deployed in communication rules such as ECDHE, promoting the strength of safe communication. The drawbacks of ECC are that it's relatively new, its security relies on the determination of the elliptic curve locations to use and that some earliest communication rules might not support ECC like the more popular RSA. Areas of applications of Elliptic Curve Cryptography are Digital Signatures and Code Signing, Securing the Web (HTTPS) and Cryptocurrencies & Blockchain. It is used to Secure wallets, Verify transactions, Internet of Things (IoT) Devices support, Secure Shell (SSH) and Secure Key Exchange.

Callan (2023) defined Elliptic Curve Cryptography as a public and private key encryption built upon elliptic curve concept employing small but greatly effective encryption keys for safety. RSA procedure encrypts textual messages or data with factored prime number, ECC establishes public keys on the winding symmetrical lines along the x-axis on a chart and non-vertical lines overlap the curve in at most, three locations. ECC has an abridged equation of the form: $y^2 = x^3 + ax + b$. ECC cryptography's growing popularity stems from the fact that its keys are extensively small with the same encryption effectiveness resulting in, less payload transmission and faster websites. ECC withstands levels of quantum computing. The small key facilitates quicker key generation and signing, with reduced latency, a benefit for IoT devices having constrained storage space and computing capacity for the elliptic curve algorithms solution. ECC facilitates speedy SSL/TLS handshakes to interchange and authenticate ordinal credentials for web pages. Furthermore, ECC certificates occupy lesser memory facilitating network performance. Callan noted that in ECC is threatened by Side-Channel Attack (SCA), which facilitates brute force attacks but SCA attacks can be countered by randomizing some factors to hide the side channels information. Moreover, Callan submitted that, Twist-security is another countermeasure, which provides a mathematically invalid public key to deny an attacker the access to reverse engineer the target's private key from the connection's shared key.

Banerjee et al. (2019) defined the Internet of Things (IoT) as an dynamic connection of wireless devices ever relating to the cloud manipulating data. Datagram Transport Layer Security (DTLS) facilitates the establishment of mutually authorized safe media in-between wireless sensor nodes and the untrusted cloud, compromised network facility in the security of an end-to-end transmission, and protocols. Banerjee et al presented DTLS 1.3, based on protocol version 18 with a re-configurable elliptic curve cryptography (ECC) accelerator, saving up to two folds of energy, reducing programming workload and memory usage while employing an high-end processor to exhibit secured systems. The DTLS protocol's handshake begins with the station, then the server agreeing upon protocol factors, a Diffie-Hellman shared secret key transmission over an untrusted channel, while consecutive handshake messages are totally encoded employing keys computed from the shared secret, a jointly authorized safe medium was put in place between a station and the server possibly useful in the systems data stage.

The TLS ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 cipher scheme was implemented in this study, where elliptic curve cryptography was used for nodes validation and key transmission, AES-128-GCM (Advanced Encryption Standard in Galois/Counter Mode), employed for authorized encoding, and SHA2-256 (Secure Hash Algorithm 2) was deployed for communication hashing, key

and pseudo-random figure generations.
It was discovered that the total computation energy for a software execution of DTLS is of the order of 0.1- 0.5 J, which is controlled by either handshake calculations or systems data encoding based on the systems attributes hence, it is pertinent to plan an low-power hardware to promote both handshake and application data manipulations for low-power IoT devices secured by DTLS. Moreover, Bernstein (2006) discussed the invention of a maximum safe Elliptic-Curve-Diffie-Hellman (ECDH) cipher achieving very great speeds at the same imagined safety level. The Curve25519 function is Fp-restricted x-coordinate scalar multiplication on E(Fp^2), where p is the prime number 2255 − 19 and E is the elliptic curve $y^2 = x^3 + 486662x^2 + x$. Each Curve25519 user has both 32-byte secret key and public key. Each set of two Curve25519 users has a 32-byte common secret used to authorize and encode messages between two actors. A hash of the common hidden Curve25519(a, Curve25519(b, 9)) is deployed representing the key for a hidden-key authorization system, or as the key for a hidden-key authorized-encoding system. Curve25519 application gives tremendously high speed, consistent, short hidden keys, short public keys, free key authentication, and short code. At a lower level, the development of an ECDH function connotes making decisions that impact speed.

The assumed tasks of the authorized users are to produce autonomous uniform arbitrary hidden keys, make the hidden keys hidden since Diffie-Hellman hidden keys should be used many times with numerous public keys to give common-hidden hashes. The safety of any spectacular key-generation system H with, e.g., 512-bit output is not assured but with a little mix, a safe hidden-key algorithm, breaks the common threats. The standard choices of secret-key Cryptosystem, C, are perceived to be secure. Addressing the security attributes a software required, It is better to make a sophisticated but valid assumption on security, rather than a demystified but simple assumption. Curve25519 is at a very high security level where determining the first key is, by assumption, difficult enough, such the ease of discovering consecutive keys is not an issue.

Fadia et al. (2024) categorically stated that, mobile wireless devices communicating through the Internet connect numberless devices to the internet but its widespread welcomes serious safety issues, particularly for lightweight devices with scarce resources. At the same time, fostering safe transmission and data correctness in such limited-resources locations is important. Usual algorithms like RSA are not conducive for IoT locations, which requires alternative ECC for communication contents' confidentiality, integrity, and authenticity. It was discovered in 1985, which built around the elliptic curves over predetermined fields plus the toughness of the Elliptic Curve Discrete Logarithm Problem (ECDLP) bringing about secure cryptographic operations with negligible resource consumption.

The existing ECC-based encryption systems fail to ensure total safety against common encoding attacks. ECC and RSA are two popular public-key cryptosystems. ECC gives the same safety level with RSA but with smaller key sizes, it becomes more efficient with mobile IoT systems. RSA requires much more RAM than the more memory-efficient ECC. RSA's key generation time is directly proportional to the safety level but ECC's generation time demonstrates that its scalable and efficient. RSA's consumes energy significantly more at higher security levels, while ECC is more energy saving. RSA has remarkably fast encryption times throughout all security levels but decryption time. RSA is more efficient for signatures at lower security levels, ECC scales better as security requirements increase.

Moreover, Mahto et al. (2017) asserted that, digital world facilitates the transmission of messages between end-points through the cloud. The Internet has some weaknesses, which aids passive or active intruders execute cyber-attacks on transmitted message. It was discovered that ECC-160 point multiplication is much more proficient than RSA-1024 private key operation. Furthermore, as per key length of RSA and ECC, use of 1024-bit RSA result in some risks while 160-bit ECC could safely be useful for a much more longer time. RSA is more speedy than ECC, but on security, ECC performs better than RSA. RSA is a better choice for message verification software than computation of digital signatures. ECC outperforms in functional proficiency and security over RSA. RSA's decoding capability is not effective as its encoding. ECC is more appropriate for limited-memory gadgets and requires fewer factors for encoding and decoding than RSA for the same security level.

Furthermore, Haddad et al. (2024) stated that, "Cloud Computing avails the capacity to decisively manage and save Electronic Health Records (EHRs), medical images, pharmaceutical information systems, and laboratory information systems. The rules that enable computers to persist files and transmit them from one-point to the other is IPFS. Safe share of medical data across the Internet could provide improved public information accessibility, privacy and integrity to healthcare data and reporting using an encryption system. The secret detail for encrypting and decrypting data must be hidden from thir-parties. Hybrid AES and ECC ciphers were used by Haddad et al to ascertain that health data records are kept safer and secure in the cloud. Haddad et al. came up with an Ethereum blockchain-based patient-centric system, which uses smart contract facility, distributed systems, unrestricted environment, and common features grant access to patients' electronic health records by authentication and ensuring data correctness. An ensemble E2EE encryption procedure was employed for secure data storage, transmission and reception from the network of networks, which ensures that data is not divulged to non-authorized users.

Practical Networking (2015) declared that the modern art of encoding data is hinging upon the phenomenon, Asymmetric Encryption whose backbone are 3 algorithms: RSA, Diffie-Hellman (DH), and Digital Signature Algorithm (DSA). Asymmetric Encryption is a set of mathematical operations that can be executed with one key and demystified with another key while Symmetric Encryption can be achieved and undone with the same key. RSA, DH, and DSA are Asymmetric Ciphers, employed for 3 uses: encryption, key exchanges, and signatures. The RSA Algorithm can do all three: Encryption, Key Exchange, and Signatures, The Diffie-Hellman (DH) algorithm is only used for key exchange, and the Digital Signature Algorithm (DSA), for Signatures. The RSA algorithm computes two "commutative" keys to "encode with one, and decode with the other." RSA keys have functionality weaknesses for encoding and decoding but for insignificant content. RSA could generate signatures and interchange secret keys. Symmetric encryption uses identical keys for encoding and decoding but for the key-exchange challenge whose common solution is the Diffie-Hellman key interchange.

Diffie-Hellman allows two endpoints to interchange two un-identical public values, which combines with their private values resulting to a private third value. The third value is the Diffie-Hellman shared secret, which is not fit directly as a symmetric key but as a seed value, from which to produce a number of desired symmetric keys needed for safe communication rules. Symmetric Key cannot directly be employed as DH shared secret because Diffie-Hellman can only ascertain that side_A has same secret key as side_B but Diffie-Hellman cannot prove the authenticity of who the other side is, which makes it susceptible to Man in the Middle attack.

The Digital Signature Algorithm (DSA) is just a signature procedure with no commutative keys, neither encoding and decoding nor likelihood of a key exchange. DSA signature production operation receives input data plus a private key to produce signature as output. The DSA signature authentication operation receives input, a Public Key (correlating to the Private Key which created the Signature), and the Signature. The output is either a 1 or a 0, where 1 means signature is valid, or 0, signature not valid. If the Signature is valid, this proves the Integrity and Authentication of the Data.

1KOSMOS (2026) defined Elliptic Curve Cryptography (ECC) as a recent public-key cryptography hinging on elliptic curves over a limited number of elements that produces a better substitute to cryptography ciphers like RSA and Diffie-Hellman. ECC has been popularly deployed for safe transmissions in different usage, including SSL/TLS, blockchain equipment, and safe communication applications. ECC hinges on the trouble of solving for a scalar k (given only P and Q) such that Q = k * P, where P and Q are locations on an elliptic curve, and * represents a computationally efficient scalar multiplication. Solving for scalar k is believed to be computationally impossible for appropriately selected elliptic curves. Elliptic curve, represented by y^2=x^3 + ax + b, where a and b are constants over a limited number of elements, which defines the probable values for x and y, pairs of coordinates (x, y) that fulfill the curve's equation.

Four important constituents of ECC are elliptic curves, points, point addition and scalar multiplication. The ECC depends on the mathematical dissymmetry between scalar multiplication and its converse problem, the elliptic curve discrete logarithm problem (ECDLP), the challenge which is non-algorithm solvable for appropriately selected elliptic curves and big key sizes, rendering ECC-based ciphers safe when conventionally attacked except with very cutting-edge hybrid computers. Implementation challenges, poor curve selection and quantum computing threat are some drawbacks for ECC while advantages are smaller key sizes, efficiency and stronger security per bit. ECC has been deployed in numerous cryptographic ciphers and communication rules such as digital seals, key transmission and encoding. Some commonly deployed elliptic curve cryptography standards and protocols are ECDH (Elliptic Curve Diffie-Hellman), ECDSA (Elliptic Curve Digital Signature Algorithm)and EdDSA (Edwards-curve Digital Signature Algorithm).

Moreover, Alshar'e et al. (2025) described Lightweight Cryptography (LWC) as a scientific improvement that deploys cryptographic ciphers, which can function efficiently in numerous resource-limited situations. Elliptic Curve Cryptography (ECC) is a dissymmetric cryptographic cipher tagged as LWC, operating on elliptic curves with two usage areas: key transmission and e-signature verification. ECC expends little time in key creation, encoding and decoding plaintext than RSA technique. The objective of information and communication security is to guarantee the privacy, reliability, and data accessibility while on virtual communications and transmissions. Elliptic curves are vital structure for elliptic curve cryptography (ECC) that permits smaller keys usage for encoding and decoding contrary to earlier asymmetric encryption techniques. It is challenging to compute the discrete logarithm of the random point on the elliptic curve. The elliptic curve two keys are linked with given set of domain factors D, which include q, FR, S, a, b, G, n, and h. A point PU is selected at random from the group generated by G and is utilized as the public key. There exists a corresponding private key, PR. This cipher starts with creating a public-private key pair using elliptic curve mathematics.

Private key is a safely created arbitrary figure. Public key, PU is a location along the elliptic curve, obtained through the private key. The essence of the Elliptic Curve Discrete Logarithm Problem (ECDLP) is the task of determining a private key, PR utilizing the equivalent public key, PU. Hence, it is germane to choose the domain factors, D in such a way where the ECDLP is very tough to resolve. RSA requires a key space of 1,024 bits, while ECC, 160 bit with the same security level of 80-bit advanced encryption standard (AES). ECC requires less memory utilization than RSA. RSA's public and private keys are depicted as numbers. In ECC, the private key is denoted as numbers, while the public key is depicted as a location on a curve. ECC produces a smaller ciphertext, enables faster key generation, and allows for speedier signature production. The decoding and encoding period is lightning-fast. The ECC records low delay in comparison with undergoing signature calculation in two stages.

Verkhovsky (2010) proposed a cryptosystem, which constitutes 3 properties: discrete logarithm problem intricacy, intricacy of number factorization of product of two big primes and the ensemble of symmetric and dissymmetric keys. In an attempt to prevent cryptosystem from cryptologic assaults, an idea of quantum entanglements (two or three interconnected particles sharing a lone non-separable quantum state) was introduced. The proposed cryptologic system has four layers: entanglement, encryption, decryption and disentanglement). "Entanglements" in cybersecurity is of two separate, tangential areas: Quantum Entanglement and Cyber Deterrence Entanglement. The introduction of entanglements is to hasten the encoding-decoding process. When combined with additional techniques, the degree of computations for encoding/decoding process is reduced. The developed communication cryptocol is multiple times faster than the RSA cryptosystem. The communication between the plaintext and the corresponding ciphertext is not one-to-one mapping, which makes the cipher less vulnerable to plaintext attacks. The existing public-key cryptographic (PKC) communication rules are seriously slow, not advisable for used in simultaneous transmission of confidential information but with Verkhovsky's new cryptosystem, PKC protocols is improved for speed in communications. The overhead of the entanglements is on the stage of information recovery.

Koblitz (1987) discussed analogs hinging on elliptic curves over a limited number of elements of public key cryptosystems that employ the multiplicative group of a limited number of elements, which should be much more safe, since the analog of the discrete logarithm problem on elliptic curves will be tougher to solve than the classical discrete logarithm problem. Koblitz described 2 elliptic curve public key cryptosystems for information transformation: Elliptic curve analog of the Massey-Omura system and ElGamal system but maintained that both Massey-Omura's and ElGamal's structures are modifications of Diffie and Hellman's initial key transmission scheme. Breaking either the elliptic curve of Massey-Omura or the ElGamal scheme involves the solving of the elliptic curve analog of the discrete logarithm problem.

3. METHODOLOGY

This chapter provides a description of the research design, data collection methods, tools, and techniques used to achieve the objectives of the study. The methodology is structured to ensure the development of an enhanced Elliptic Curve Cryptography (ECC)-based communication model, its implementation, and evaluation.

3.1 Data Collection

The data used for this study was gathered through experimental analysis and simulations.

3.2 Overview of the Current (ECC) Model

In the existing encryption scheme, as shown in Fig 3.1, a plaintext message, *m* is represented as a point $P_m(x, y)$ on an elliptic curve. Instead of directly encrypting the plaintext, the system encodes it as the elliptic curve point P_m. The encryption and decryption processes rely on elliptic curve point multiplication and key exchange mechanisms.

To establish secure communication, the sender and receiver generate key pairs:

- Sender's private key: *a*
- Sender's public key: $P_a = a \times G$
- Receiver's private key: *b*
- Receiver's public key: $P_b = b \times G$

Here, *G(x, y)* is a predefined base point on the elliptic curve, and multiplication is performed using elliptic curve scalar multiplication.

Encryption Process

To encrypt a message, P_m before transmission, the sender selects a random integer *k* (the session key) and computes the ciphertext as a pair of points: $C_m = \{P_m + kP_b, kG\}$
where:

- kP_b is the shared secret derived from the receiver's public key.
- *kG* ensures randomness and security in the encryption process.

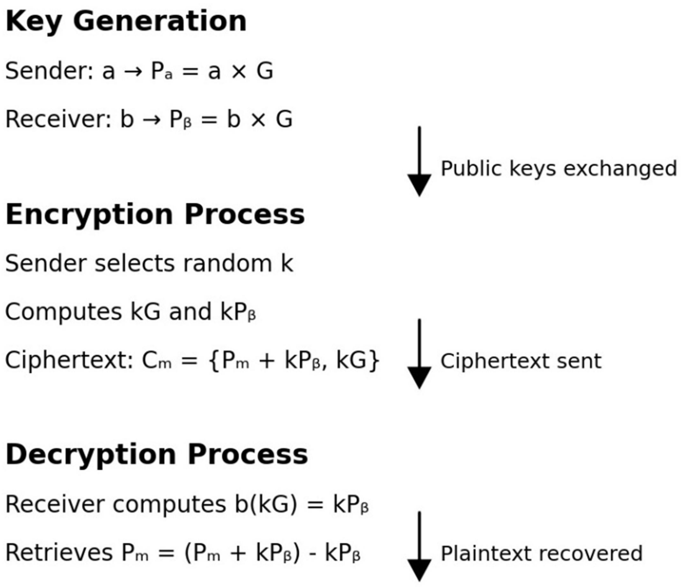

Key Generation
Sender: a → $P_a = a \times G$
Receiver: b → $P_\beta = b \times G$
Public keys exchanged
Encryption Process
Sender selects random k
Computes kG and kP_β
Ciphertext: $C_m = \{P_m + kP_\beta, kG\}$
Ciphertext sent
Decryption Process
Receiver computes $b(kG) = kP_\beta$
Retrieves $P_m = (P_m + kP_\beta) - kP_\beta$
Plaintext recovered

Figure 3.1: Key Generation, Encryption, and Decryption process of the Existing Model

Decryption Process

Upon receiving the ciphertext $C_m = \{P_m + kP_b, kG\}$, the receiver retrieves the plaintext as follows:

1. Multiply the last component by the receiver's private key, *b*:

$$b(kG) = kP_b$$

2. Subtract this result from the first component of the ciphertext: $P_m = (P_m + kP_b) - kP_b$

Since the term kP_b cancels out, the receiver successfully recovers the original plaintext point P_m.

3.2 Overview of the Improved ECC Model

The improved encryption model builds upon the existing elliptic curve cryptographic scheme while addressing potential vulnerabilities and optimizing performance. The primary enhancements include a more secure key exchange mechanism, improved resistance to known attacks, and efficiency improvements in encryption and decryption.

Similar to the previous method, a plaintext message, *m* is represented as an elliptic curve point $P_m(x, y)$. The encryption and decryption processes are enhanced with an additional layer of security through an optimized Key Derivation Function (KDF) and integrity verification.

Key Generation

Both the sender and receiver generate their key pairs as follows:

- **Sender's private key**: a
- **Sender's public key**: $P_a = a \times G$
- **Receiver's private key**: b
- **Receiver's public key**: $P_b = b \times G$

where $G(x, y)$ is the base point on the elliptic curve, and multiplication is elliptic curve scalar multiplication.

Encryption Process

To encrypt a message P_m, the sender:

1. Selects a random integer *k* (session key).
2. Computes a shared secret using the receiver's public key: $S = kP_b$
3. Uses a Key Derivation Function (KDF) to generate a symmetric key from *S*:

$$K = KDF(S)$$

4. Computes the ciphertext as: $C_m=\{P_m + S, kG, H(P_m)\}$

where $H(P_m)$ is a cryptographic hash function to ensure message integrity.

Decryption Process

Upon receiving the ciphertext $C_m = \{P_m + S, kG, H(P_m)\}$, the receiver:

1. Computes the shared secret using their private key: $S' = b(kG) = kP_b$
2. Derives the symmetric key: $K = KDF(S')$
3. Recovers the plaintext: $P_m = (P_m + S) - S$
4. Verifies integrity by computing $H(P_m)$ and comparing it to the received hash.

The use of a KDF prevents key leakage and makes the encryption more resistant to attacks, the cryptographic hash ensures the message has not been altered while the optimized key exchange mechanism reduces computational overhead while maintaining security.

3.3 Tools and Technologies

To ensure a robust development, implementation, and evaluation of the proposed enhanced ECC-based communication model, various tools and technologies were employed.

- Kali Linux (Mounted on VMware), a Debian-based Linux distribution, was used as the primary operating system for cryptographic analysis and testing.
- Microsoft Visual Studio (VS studio) was used as the primary Integrated Development Environment (IDE) for writing, debugging, and executing code
- Python Programming Language was selected as the core programming language due to its extensive cryptographic libraries and ease of prototyping security algorithms
- PyCryptodome, a Python-based cryptographic library, was used for implementing ECC key generation, encryption, and decryption operations.
- OpenSSL, an open-source cryptographic toolkit, was leveraged for implementing and testing TLS/SSL protocols
- Wireshark, a widely used network protocol analyzer, was employed for monitoring and analyzing network traffic during testing
- Valgrind was utilized for memory usage analysis and performance profiling of the ECC-based system.

4. RESULTS AND DISCUSSION

This chapter presents the results obtained from the implementation and evaluation of the enhanced Elliptic Curve Cryptography (ECC)-based communication model. The findings are analyzed and discussed in terms of security, efficiency, and performance improvements compared to the existing ECC model.

4.1 The Developed Model's Evaluation Metrics

The proposed enhanced ECC model was evaluated based on key performance metrics, including encryption/decryption speed, key generation time, memory consumption, and resistance to cryptographic attacks.

4.1.1 Encryption and Decryption Speed

The time taken to encrypt and decrypt messages of varying sizes was measured and compared between the existing and improved ECC models in Table 1.

Table 1: Encryption and Decryption Time Comparison

Message Size (KB)	Existing ECC Encryption (ms)	Improved ECC Encryption (ms)	Existing ECC Decryption (ms)	Improved ECC Decryption (ms)
1	18.5	15.2	17.3	14.8
10	168.4	152.7	165.2	149.5
100	1,650.7	1,520.3	1,620.5	1,490.1

In Table 1, for a message of 1, 10 and 100kb in size, the Existing and improved ECC model had encryption time of 18.5 and 15.2 (ms), 168.4 and 152.7(ms), 1,650.7 and 1,520.3(ms), respectively, while their decryption time were 17.3 and 14.8(ms), 165.2 and 149.5 (m,s), 1,620.5 and 1,490.1(ms), respectively.

These show a significantly better performance of the improved ECC model at both encryption and decryption computational overhead, due to the additional Key Derivation Function (KDF) and hash verification steps.

4.1.2 Enhanced ECC Key Generation Time

The key generation process was analyzed to determine the efficiency of the improved model.

Table 2: Key Generation Time Comparison

Key Size (bits)	Existing ECC Key Generation (ms)	Improved ECC Key Generation (ms)
256	13.1	12.4
384	19.5	18.7
521	26.8	25.3

Table 2 has the tabulated key generation time of both the existing and the enhanced ECC model. For key sizes of 256, 384 and 521 (bits), the existing and enhanced ECC had key generation time of 13.1 and 12.4 (ms), 19.5 and 18.7 (ms), 26.8 and 25.3 (ms), respectively, indicating a 5.3, 4.1 and 5.6 % of the enhanced ECC's better performance than the existing, respectively.

4.1.3 Memory Usage Analysis for both Existing and enhanced ECC Model

Memory utilisation of both the existing and the enhanced ECC model was evaluated using Valgrind.

Table 3: Memory Utilisation of both Existing and Enhanced ECC Model

Operation	Existing ECC Memory Usage (MB)	Improved ECC Memory Usage (MB)
Key Generation	2.3	2.1
Encryption (1KB Data)	3.8	3.5
Decryption (1KB Data)	3.7	3.4

In Table 3, the memory utilisations of the existing and the enhanced ECC model were 2.3 and 2.1(MB), 3.8 and 3.5 (MB), 3.7 and 3.4(MB) for key generation, encryption (1kb data) and decryption (1kb data) operations, respectively. This is evident that the enhanced ECC had better memory maximization better than the existing by 8.6, 7.9 and 8.1% across the operations, recpectively.

4.1.4 Comparative Analysis with Existing and RSA Models

The enhanced ECC model was compared with traditional ECC and RSA (Rivest-Shamir-Adelman) cryptographic models in terms of security and performance.

Table 4: Comparative Analysis of the Existing, Enhanced ECC and RSA Public Key Cryptographic Models

Parameter	Existing ECC	Improved ECC	RSA (2048-bit)
Encryption Speed (ms)	18.5	15.2	210.4
Decryption Speed (ms)	17.3	14.8	185.7
Key Size (bits)	256	256	2048
Resistance to MiMA	High	Moderate	High

Table 4 has the comparative analysis of the existing and enhanced ECC as well as RSA cryptographic models. For encryption speed, decryption speed and key size; existing ECC, enhanced ECC and RSA had 18.5, 15.2 and 210.4 (ms); 17.3, 14.8 and 185.7 (ms); 256, 256 and 2,408 (bits), respectively. It is evident that the enhanced ECC model performed better than both the existing ECC and RSA models in encryption and decryption speed. In key size especially, it outperformed RSA (2048-bit) with smaller key size (256-bit) with the same security level. Even though, the Existing ECC & RSA models were highly resistant to MiMA attack, the enhanced ECC model was still moderate in performance.

5. CONCLUSION

This study was able to develop an enhanced ECC, balancing security and efficiency. The developed model could be deployed for secure applications in IoT, blockchain, and messaging systems, offering a practical and forward-compatible cryptographic solution.

REFERENCES

1. Banerjee, U., Wright, A., Juvekar, C., Waller, M., Arvind, & Chandrakasan, A.P. (2019). An Energy-Efficient Reconfigurable DTLS Cryptographic Engine for Securing Internet-of-Things Applications.
2. Bernstein, D.J. (2006). Curve25519: New Diffie-Hellman Speed Records. In Public Key Cryptography, Pp. 207-228.
3. Taleb Fadia and Larid Toufik (2024). Elliptic curves cryptography for lightweight devices in IoT system. Brazilian Journal of Technology, Curitiba, vol.7, no.4, pp. 01-19.
4. Tim Callan (2023). Elliptic Curve Cryptography Explained. SeCTIGO. https://www.sectigo.com/blog/what-is-elliptic-curve-cryptography
5. Practical Networking (2015). RSA, Diffie-Hellman, DSA: the pillars of asymmetric cryptography. https://www.practicalnetworking.net/practical-tls/rsa-diffie-hellman-dsa-asymmetric- cryptography-explained/
6. 1KOSMOS (2026)What Is Elliptic Curve Cryptography (ECC)? Explained https://www.1kosmos.com/security-glossary/elliptic-curve-cryptography/
7. Haddad et al. (2024). “E2EE enhanced patient-centric blockchain-based system for EHR management”. PLoS ONE. vol. 19, No.4. https://doi.org/10.1371/journal.pone.0301371.
8. Koblitz, N. (1987). Elliptic Curve Cryptosystems. Mathematics of Computation, vol. 48, No. 177, pp. 203-209.

9. Kumar, R., & Tripathi, R. (2025). Elliptic Curve Cryptography: Applications, Challenges, and Recent Advances. Journal of Systems Architecture, 122, 102337.
10. Sanders (2022). “Elliptic Curve Cryptography: What is it? How does it work?”. The Source|Insights for the Future of Digital Trust. https://www.keyfactor.com/blog/elliptic-curve-cryptography-what-is-it-how-does-it-work/. 6 November, 2025.
11. Sullivan (2013). “A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography”. Cloudfare. https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/ . Fetched:2 December, 2025
12. Mahto et al. (2017). “RSA and ECC: A Comparative Analysis”. International Journal of Applied Engineering Research. Vol. 12, No. 19. pp. 9053-9061. http://www.ripublication.com
13. Marwan Alshar’e (2025). Elliptic curve cryptography based light weight technique for information security. Bulletin of Electrical Engineering and Informatics Vol. 14, No. 3, pp. 2300~2308 ISSN: 2302-9285, DOI: 10.11591/eei.v14i3.85872300 Journal homepage: http://beei.org
14. Boris S. Verkhovsky (2010). Hybrid Authentication Cybersystem Based on Discrete Logarithm, Factorization and Array Entanglements.International Journal of Communications, Network and System Sciences, vol.3, pp.579-584 doi:10.4236/ijcns.2010.37077 (http://www.SciRP.org/journal/ijcns/)
15. Williams Stallings (2011). Cryptography and network security principles and practice. fifth edition. Pearson Education, Inc., publishing as Prentice Hall.