Earthline Journal of Mathematical Sciences
E-ISSN: 2581-8147
Volume 13, Number 2, 2023, Pages 451-472
https://doi.org/10.34198/ejms.13223.451472

A New Text Encryption Scheme Suitable for Combating Sniffing Attacks in IoT Applications via Non-supersingular Elliptic Curves over Binary Extension Fields

Zakaria Abukari[1,*], Edward Yellakuor Baagyere[2] and Mohammed Muniru Iddrisu[3]

[1] Department of Computer Science, Tamale Technical University, Ghana
e-mail: zack@tatu.edu.gh

[2] Department of Computer Science, C. K. Tedam University of Technology and Applied Sciences, Navrongo, Ghana
e-mail: ebaagyere@cktutas.edu.gh

[3] Department of Mathematics, C. K. Tedam University of Technology and Applied Sciences, Navrongo, Ghana
e-mail: middrisu@cktutas.edu.gh

Abstract

Several research works propose the use of Elliptic Curve Cryptography (ECC) to provide security for the Internet of Things (IoT) and cloud computing due to its shorter key requirement of approximately 160-571 bits vs. 1,024-15,360 bits of the others whilst achieving the same level of security. As a result, several ECC based text encryption schemes have been proposed in recent times. However, due to the mathematical foundations behind some of these schemes, there is the need for improvement to make them efficiently suitable for applications targeting IoT platforms. In addition, many of the existing schemes are either limited to some languages and/or use lookup tables which increase their computational overheads in terms of storage and processing. Against this background that this paper proposes a new ECC based text encryption scheme using efficient elliptic curve arithmetic to reduce the computational overheads. The scheme resists the major forms of sniffing attack in software implementation of ECC-based schemes. A test implementation proves that a very high key sensitivity is also achieved.

Received: June 27, 2023; Accepted: July 31, 2023; Published: August 19, 2023

2020 Mathematics Subject Classification: 68P25.

Keywords and phrases: elliptic curves cryptography, ECC text encryption, IoT.

[*]Corresponding author

1. Introduction

The Internet of Things (IoT) is a promising game changer in Information and Communication Technology that aims at making all manners of physical objects and devices converge at the Internet and thus making the current Internet even more pervasive by enabling easy access and interaction among a wide variety of devices such as home appliances, surveillance cameras, monitoring sensors, actuators, displays, vehicles, industrial equipment and so on [1]. Large amounts of data are expected to be generated to provide a myriad of new services to citizens, companies, and governments among others. However, secured communication among these heterogeneous objects is currently a major area of concern amongst stakeholders. For example, Hewlett Packard holds that about 70% of IoT devices are vulnerable to sniffing attacks and reliable solution is yet to be found [2]. Due to the complexity of the IoT ecosystem, utilization of the existing cryptographic solutions such as RSA and AES to combat sniffing attacks in IoT is questionably since some of the IoT devices present hardware and energy constraints for computationally expensive encryption schemes. The existing literature suggests lightweight schemes. However, the current ones recommended by NIST are symmetric schemes which are generally not suitable for transit data and therefore, the need for public key lightweight schemes suitable for secured communication in IoT cannot be overemphasized. Against this background that this paper proposes a new ECC based text encryption scheme using efficient ECC arithmetic.

2. Literature Review

Since the discovery that Elliptic Curve Cryptography (ECC) is gradually becoming a better alternative to the RSAs and Discrete Logarithm schemes amongst others in the field of public key cryptography [3], a number of ECC based text encryption schemes have emerged in the literature. The schemes are either implemented using elliptic curves defined over prime fields $GF(p)$ or binary extension fields ($GF(2^m)$. Table 1 gives a summary of some of the existing ones, taking into consideration the underlying fields of their binding elliptic curves, point representation used in point arithmetic method, encryption technique and character set of plaintext.

Table 1: Summary of existing ECC based Text Encryption schemes.

Referenced work	**Underlying field**	**Points representation**	**Encryption technique**	**Character set of plaintext**
Encryption of data using Elliptic Curves over Finite Fields [4]	$GF(p)$	Affine	Each character of plaintext M is mapped to points on the binding elliptic curve using a common lookup table between Alice and Bob. Then each mapped point is encrypted to a pair of cipher points using a shared key.	ASCII
Implementation of Text Encryption using Elliptic Curve Cryptography [5]	$GF(p)$	Affine	Each character of plaintext is converted to a 16 bit decimal, then the plaintext being decimals now, is partitioned into large integers and encrypted with a shared key.	UTF-16
Security Enhancement of Text Message Based on Matrix Approach Using Elliptical Curve Cryptosystem [6]	$GF(p)$	Affine	Each character of plaintext is transformed into its ASCII code and mapped to an affine point Pm(x, y) on the binding elliptic curve. The affine points are then mapped again using Matrix Mapping scheme then finally encoded using ElGamal encryption scheme.	ASCII
Implementation of Elliptic Curve Cryptography in Binary Field [7]	$GF(2^m)$	Affine	Plaintext is converted to binary strings using the ASCII values of each character then grouped. Each block is then encrypted with a shared secret	ASCII
Elliptic Curve Cryptography for Secured Text Encryption [8]	$GF(p)$	Affine	Plaintext M is converted into ASCII values then to HEXADECIMAL followed by grouping of the HEXADECIMAL into finite set of input size. The order of the HEXADECIMAL value is reversed and the ciphertext is obtained using an elliptic curve scalar multiplication algorithm.	ASCII
Text Message Encoding Based on Elliptic Curve Cryptography and a Mapping Methodology [9]	$GF(p)$	Affine	Each character of plaintext M is mapped to points on the binding elliptic curve using a common lookup table between Alice and Bob.	ASCII

			Then each mapped point is converted to ciphertext using EC point operations.	
Text encryption using elliptic curve cryptography [10]	$GF(p)$	Affine	Each character of plaintext is converted to a 16 bit decimal, then the plaintext being decimals now, is partitioned into large integers and encrypted with a shared key.	UTF-16

3. Proposed Encryption and Decryption Scheme

3.1. Underlying mathematical foundation of the proposed scheme

The proposed scheme uses non-supersingular elliptic curves defined over binary extension fields, $GF(2^m)$. Enabled by the Elliptic Curve Discrete Logarithm Problem [11], the Elliptic Curve Diffie-Hellman Key Exchange (ECDHKE) protocol is implemented using ECC arithmetic proposed in [12] for its reported efficiency.

3.2. Character set of plaintext

The proposed algorithm is designed to work with the 16 bits Unicode Transformation Format (UTF-16). Unicode is a universal character set which contains all the alphabets, technical and literary symbols, punctuations and anything used in writing text of all modern and ancient languages in the world, including the European alphabetic scripts, Middle East right-to-left scripts, and scripts of Asia [13]. The Standard defines three encoding formats, being UTF-8 bits, UTF-16 bits and UTF-32 bits. UTF-8was designed for backward compatibility with existing systems based on the ASCII encoding system whilst the UTF-16 is quite convenient and compact whereby most character codes fit into 16 bits (two bytes), allowing fast processing. It is very popular and the most recommended standard for systems that target all languages [13]. UTF-32 has higher memory quotas and currently has limited practical usage. There is no known advantage for its usage [13].

3.3. Encryption and decryption

The process starts with the key generation stage (System Initialization).

i. A trusted party generates the ECC domain parameters and publishes them as the tuple (m, a, b, P, n) where m defines the binary extension field $GF(2^m)$, a non-supersingular elliptic curve E: $y^2 + xy = x^3 + ax^2 + b$ where $a, b \in GF(2^m)$, a base point P and its order n.

ii. The two communicating parties then select their private keys from the interval $[1, n-1]$ and use the ECDHKE protocol to exchange a secret key $k = (k_x, k_y)$ for the encryption and decryption.

3.3.1. Encryption

The encryption and decryption schemes involve three literary characters: Alice and Bob are the two legitimate communicating parties whilst Eve depicts their adversary, the sniffer (eavesdropper).

The encryption process is as follows:

i. The initiator, Alice, splits the plaintext P into n blocks

$$P = P_1, P_2, \cdots, P_n$$

ii. Each block P_i is transformed into a binary string B_i such that each character c in position j in block i, c_{ij}, is at most 16 bits and s is the length of P_i as:

$$B_i = \sum_{j=1}^{s} \left(U_{ij} + \beta_{ij}\right)$$

where U_{ij} is the UTF-16 bits code of c_{ij} and β_{ij} is a 16 bits Differential Factor for c_{ij}. It is computed as

$$\beta_{ij} = \left(\frac{i(i+1)}{2} + \frac{q(q+1)}{2}\right) \bmod f(z)$$

where q is the relative position of c_{ij} in the original plaintext P and $f(z)$ is a binary field reduction polynomial for $GF(2^{16})$.

iii. Each B_i is then encrypted using k to obtain the ciphertext C_i as follows:

$$C_i = B_i + (k_x \cdot k_y)$$

iv. Output complete ciphertext in hexadecimal as

$$C = \sum_{i=1}^{n} Hex(C_i)$$

v. $t = n + (k_x \cdot k_y)$

vi. C and t are transmitted to Bob through the insecure IoT network.

3.3.2. Decryption

i. Bob obtains ciphertext C and blocks count t.

ii. Decrypts t to obtain blocks count n as:

$$n = t - k_x \cdot k_y$$

iii. Splits C into $C_1, C_2, \cdots, C_n$ then computes

$$C_i' = \sum_{i=1}^{n} Bin(C_i)$$

where *Bin* is a function that converts its input from hex to binary.

iv. Each C_i' is decrypted using the shared secret k to obtain the plaintext B_i as:

$$B_i = C_i' - (k_x \cdot k_y)$$

v. Starting from the most significant bit (MSB) of B_i, each sixteen (16) bits, W_j, is transformed to a UTF-16 character code as follows:

$$U_{ij} = W_j - \beta_{ij}$$

where

$$\beta_{ij} = \left(\frac{i(i+1)}{2} + \frac{q(q+1)}{2}\right) mod\ f(z)$$

q is the relative position of W_j in C and $f(z)$ is the reduction polynomial of $GF(2^{16})$.

vi. Recover each character of the plaintext as $c_{ij} = Char(U_{ij})$ where $Char(x)$ is a function which returns the unique UTF-16 character for the code x.

vii. Accumulate the recovered characters c_{ij} to form plaintext P_i as follows:

$$P_i = \sum c_{ij}$$

viii. The complete plaintext P is obtained as

$$P = \sum_{i=1}^{n} P_i.$$

Algorithms 1 and 2 present pseudo codes of the encryption and decryption respectively.

Algorithm 1: Proposed encryption scheme

INPUT: Plaintext P and shared secret key k_x, k_y

OUTPUT: Ciphertext C, blocks count t

$n \leftarrow BlocksCount$

$P_x \leftarrow P/n$

$q \leftarrow 1$

$for\ i \leftarrow 1\ to\ n\ do$

$begin$

$P_i \leftarrow P_x[i]$

$for\ j \leftarrow 1\ to\ length(P_i)\ do$

    $begin$

    $ch \leftarrow P_i[j]$

    $u \leftarrow UTF16Code\ for\ the\ character\ ch$

    $\beta \leftarrow Differential\ Factor\ of\ i, q$

    $Add\ u,\ \beta\ to\ block\ B_i$

    $Increase\ q\ by\ 1$

    $end\ \ //\ end\ for\ j \ldots\ldots\ldots$

$C_i \leftarrow B_i + (k_x \cdot k_y)$ // $encryption\ of\ block\ B_i\ to\ obtain\ ciphertext\ C_i$

$Convert\ C_i\ to\ hex\ and\ Accumulate\ in\ C$

$end\ \ //\ end\ for\ i \ldots\ldots\ldots$

$t = n + (k_x \cdot k_y)$

$\boldsymbol{Return}\ (C, t)$

Algorithm 2: Proposed decryption scheme

INPUT: Ciphertext C, blocks count t, and shared secret key k_x, k_y

OUTPUT: Plaintext P

$n \leftarrow t - (k_x \cdot k_y)$

$C_x \leftarrow C/n$

$q \leftarrow 1$

$for\ i \leftarrow 1\ to\ n\ do$

$begin$

$C_i \leftarrow Convert\ C_x[i]\ into\ binary$

$B_i \leftarrow C_i - (k_x \cdot k_y)$

$while\ length(B_i) > 0\ do$

  $begin$

  $w \leftarrow copy(B_i, 16)$

  $\beta \leftarrow\ Differential\ Factor\ of\ i, q$

  $u \leftarrow w - \beta$

  $c \leftarrow\ Char\ (u)$

  $Add\ c\ to\ P_i$

  $B \leftarrow\ remaining\ bits\ of B_i$

  $B_i \leftarrow\ B$

  $Increase\ q\ by\ 1$

  end

$Accumulate\ P_i\ in\ P$

end

$\boldsymbol{Return}\ (P)$

3.4. Test implementation

The proposed scheme was coded in Borland Delphi programming language for testing and analyses. Best practice testing approaches which include unit testing, system testing, volume testing and integration testing were adopted to test key generation, encryption and decryption.

3.4.1. Key generation

The ECDHKE unit is tested first. The domain parameters for the experimentation are as follows:

1) A non-supersingular elliptic curve E: $y^2 + xy = x^3 + ax^2 + b$, where $x, y, a, b \in GF(2^{163})$ represented in polynomial basis
2) Curve coefficients
   a = 1
   b = 20A601907B8C953CA1481EB10512F78744A3205FD
3) Reduction polynomial
$$f(z): z^{163} + z^7 + z^6 + z^3 + 1$$
4) Base point G (Gx, Gy)
   Gx = 3F0EBA16286A2D57EA0991168D4994637E8343E36
   Gy = 0D51FBC6C71A0094FA2CDD545B11C5C0C797324F1
5) Order of G
   n = 40000000000000000000292FE77E70C12A4234C33

These setup values correspond to Curve B-163, the smallest of the non-supersingular curves recommended by the NIST for Elliptic Curve Cryptography [14].

Alice and Bob randomly chose their private keys *nA* and *nB* respectively.

nA = 65538D7CF99B387FB6A29925BFDE5F709AED5F58F

nB = 7996EB9643237AB85FEFE54E5287B20F8A69AFF44

Using the private keys together with the public parameters, private computations are done to obtain:

Qa = (2E81A8F4BC4C4664D1215ADB05454A7D3A0CC1EB0 , B5F503FF5E24C51640ED4BBBD4CC6A9120A7D30F)

Qb = (26E9E86D4657170BA21A2F50F6495F5FE3ACA7B2E , 1BD0AC394BB864605E4539C7A46EA7F07B090B3C9)

Sa = (348DA663457DA026D475CE907F0A2DE9A11875449 , 778D84BD7A08A7051362953A11BE093FA5A396269)

Sb = (348DA663457DA026D475CE907F0A2DE9A11875449 , 778D84BD7A08A7051362953A11BE093FA5A396269)

The values **Sa** and **Sb** are equal and as such the ECDHKE is successful. Alice could now use Sa to encrypt and Bob would use Sb to decrypt.

3.4.2. Encryption and decryption

Following the origin and grouping of spoken and written languages in the World, a special plaintext is designed in order to ascertain whether the scheme can really work on UTF-16 which covers all languages. The text is multilingual, containing phrases from five European languages (English, German, French, Russian and Spanish); three Asian including right-to-left scripts (Arabic, Chinese and Japanese) and an African language (Ewe from Ghana) in addition to technical symbols and punctuations. Figure 1shows the screenshot before encryption, whilst Figure 2displays the ciphertext and the recovered plaintext.

The sample plaintext was

English: "No comment" is a big comment!

German: "KeinKommentar" isteingroßerKommentar!

French: "Aucuncommentaire" est un groscommentaire !

Spanish: "Sin comentario" esun gran comentario!

Russian: ″Безкомментариев″ – этобольшойкомментарий!

Arabic: "لا تعليق" هو تعليق كبير!

Chinese: “沒有評論”是一個大評論！

Japanese: 「ノーコメント」は大コメント！

Ewe: "No comment" nyenyagãaɖe!

English: "No comment" is a big comment!

You need $O(\sqrt{n})$ steps to break the key

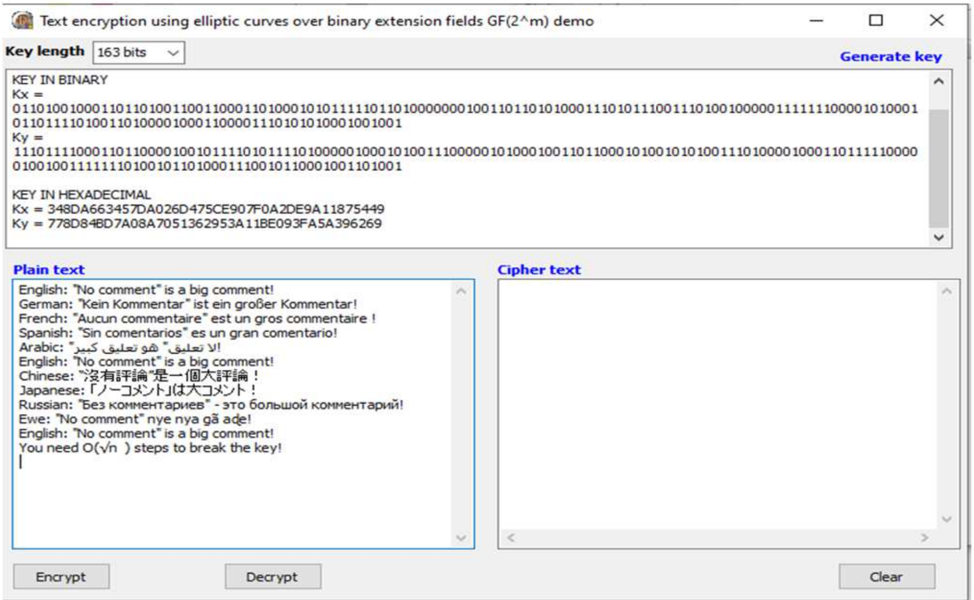

Figure 1: Encryption key and sample plaintext.

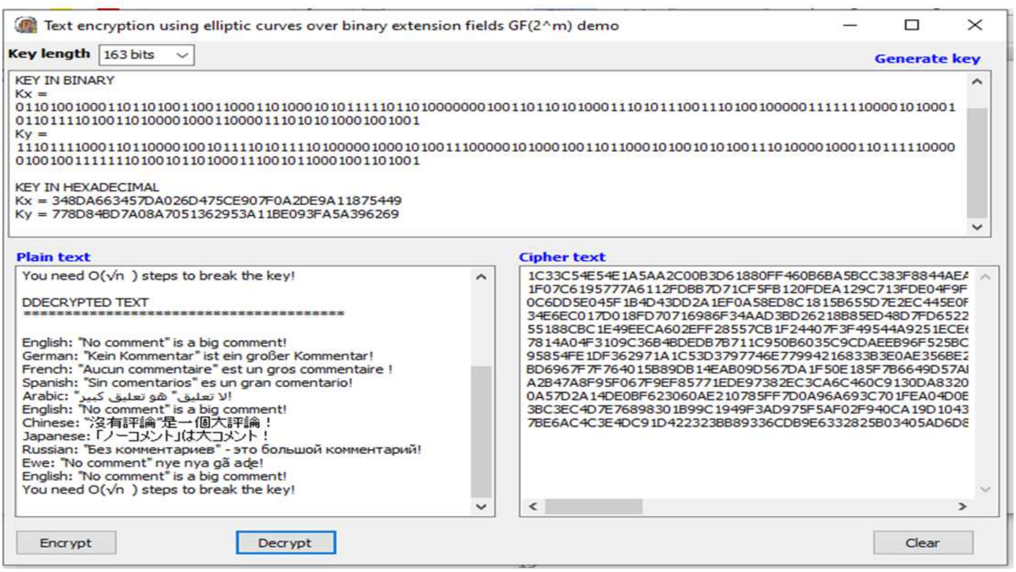

Figure 2: Ciphertext and recovered plaintext.

3.5. Security and performance analyses of the proposed scheme

3.5.1. Known forms of attack on ECC-based schemes

Three different forms of attack on software implementation of ECC-based schemes are found in the literature. These include brute-force attack, plaintext-only attack and ciphertext-only attack.

3.5.2. Thwarting brute force attack

All elliptic curve-based cryptographic schemes inherit their primary security prowess from the apparent hardness of the Elliptic Curve Discrete Logarithm Problem [11]. The best known ECDLP brute-force attack on non-supersingular curves is the combined method of the Pohlig-Hellman algorithm and the Pollard's Rho method. This combinatory effort has an exponential algorithmic complexity of $O(\sqrt{n})$ steps, where n is the order of the elliptic curve base point P. The number of steps increases proportionally as n grows (Figure 3).

To thwart this form of attacks, the curve E and the base point P should be selected such that n is sufficiently large to make the $\sqrt{n}$ steps an infeasible computational amount, for example $n > 2^{160}$ [14, 15].

For a practical demonstration, the NIST recommendations for the selection of domain parameters were adopted in the test implementation. Using the smallest recommended curve, B-163 ($GF(2^{163})$), and the other corresponding domain parameters being the generator point G, order of G, and the curve coefficients a and b in polynomial representation, it was found out that an adversary using the Pohlig-Hellman-Pollard's Rho algorithm, requires approximately $2.41785163922925834941235\ 3 \times 10^{24}$ steps to discover the key.

Assuming the adversary spends as small as one nanosecond (1 billionth of a second (0.000000001 or 10–9) to execute a step, then, the time complexity will be approximately 76,617,095 years to find the shared secret. A proof to this assertion is as follows:

For

$$\mathrm{n} = 000000040000000000000000000292FE77E70C12A4234C33$$

whose decimal equivalence is

$$\mathrm{n} = 5846006549323611672814742442876390689256843201587,$$

the number of steps to resolve the ECDLP is:

$$total\ steps = \sqrt{5846006549323611672814742442876390689256843201587}$$

$$= 2{,}417{,}851{,}639{,}229{,}258{,}349{,}412{,}353$$

$$= 2.417851639229258349412353 \times 10^{24}$$

$$time\ complexity = 2.417851639229258349412353 \times 10^{24} \times 10^{-9} \text{ seconds}$$

$$= 2.417851639229258349412353 \times 10^{15} \text{ seconds}$$

A day has 86,400 (8.64×10^4) seconds. Converting the time complexity to days, we have

$$time\ complexity = \frac{2.417851639229258349412353 \times 10^{15}}{8.64 \times 10^4} \text{ days}$$

$$= 27{,}984{,}393{,}972.56 \text{ days}$$

$$= 76{,}617{,}095.07 \text{ years}$$

$$= 7.661709507 \times 10^7 \text{ years}$$

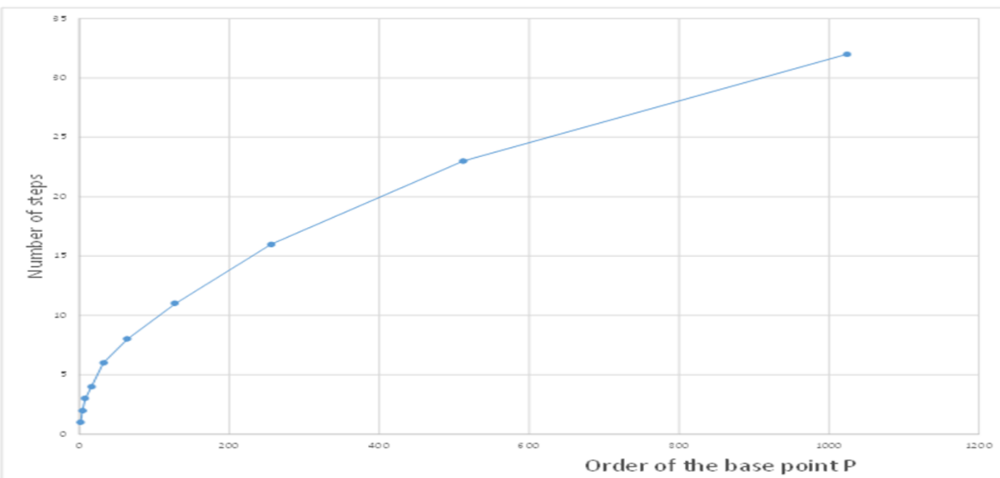

Figure 3: Algorithmic complexity of Pohlig-Hellman-Pollard's Rho attack.

Similar analyses with the higher degree curves yield higher time complexities. In this regard, ECDLP appears to be intractable using conventional computers, and the situation is believed to continue as such until the arrival of *quantum computers.* Shor (1997), as cited in [15], theoretically found an efficient algorithm for fast computation of discrete logarithms and integer factorization on an empirical quantum architecture. Though a lot of research is ongoing in quantum computing, there is no evidence in the literature that gives timelines or pointers to when the first quantum computer will appear.

3.5.3. Thwarting plaintext-only attack

Another form of possible attack is through cryptanalysis. This is the science of analyzing and deciphering codes and ciphers without any knowledge of the key. A cryptanalyst will normally know the encryption algorithm, the ciphertext and may form some plaintext-ciphertext pairs. To resist this form of attack, the proposed scheme introduces a novelty called Differential Factor (DF) in step (ii) of the encryption process which ensures that different ciphertext is always generated for each letter occurrence in the plaintext. To this end, the plaintext-ciphertext pairings, if it is even possible, will be fruitless.

3.5.4. Thwarting ciphertext-only attacks

In a ciphertext-only attacks, the attacker has access to only the ciphertext. No idea of the plaintext nor the encryption algorithm. The approach is possible owing to the fact that the alphabets of every language have known frequencies in the formation of words in the language. For example, Table 2 shows the frequency of letters occurrence in a typical English plaintext according to [11].

Table 2: Frequency of letters in English text.

Letter	Freq.		Letter	Freq.
E	13.11 %		M	2.54 %
T	10.47 %		U	2.46 %
A	8.15 %		G	1.99 %
O	8.00 %		Y	1.98 %
N	7.10 %		P	1.98 %
R	6.83 %		W	1.54 %
I	6.35 %		B	1.44 %
S	6.10 %		V	0.92 %
H	5.26 %		K	0.42 %
D	3.79 %		X	0.17 %
L	3.39 %		J	0.13 %
F	2.92 %		Q	0.12 %
C	2.76 %		Z	0.08 %

Taking undue ride on this vulnerability, the cryptanalyst will normally count the letters in the ciphertext and construct frequency tables or use other statistical methods to identify character patterns and eventually decrypt the message.

Again, the introduction of the Differential Factor (DF) prevents this form of attack on the grounds that even if a letter is doubled in a word, it will have different cipher representations. Automatically, letters frequency tables are invalidated and character patterns from the ciphertext cannot be formed. This guarantees a 100% security from ciphertext-only attacks.

3.5.5 Key sensitivity of the proposed scheme

Key sensitivity of the proposed scheme is very high. Decrypting with a wrong key is always fruitless. A test was done with a wrong key that was very close to the correct key. In other words, a wrong key *wk* that was very close to the correct key *ck* was used. In this particular simulation, only the last bit of K_x was changed from 1 to 0 and, as is illustrated in Figure 5, this little change generated a decrypted message that had no semblance at all with the correct message.

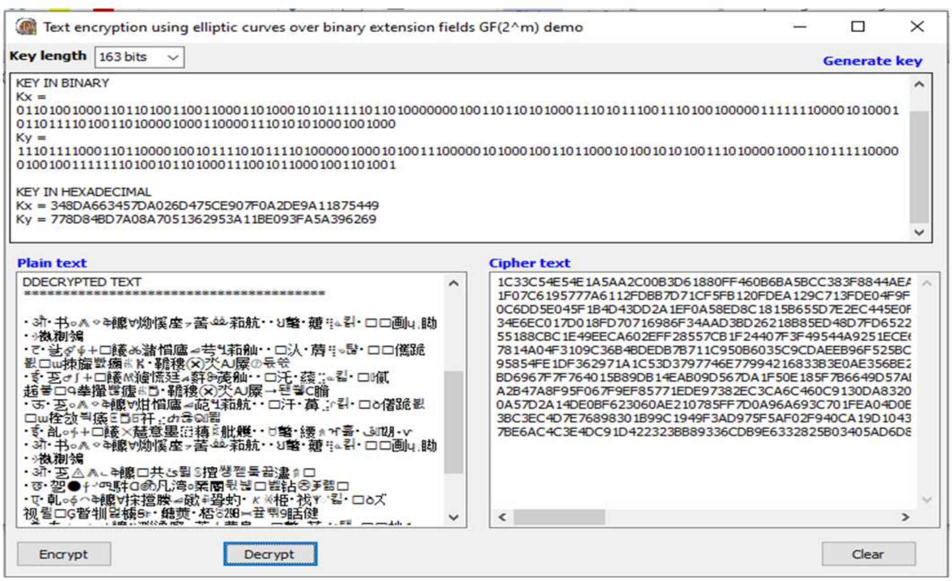

Figure 4: Key sensitivity.

3.5.7. Encryption and decryption speed of the proposed scheme

A test run was done on an Intel Core i7 with 12GB RAM whose CPU was averagely 10% busy. After several cycles of encryption and decryption of a15, 923 words (23 pages), an average of 5.5 seconds was recorded for encryption whilst decryption mean time was 4 seconds.

3.6. Novelty of the proposed scheme

i. The proposed scheme introduces a novelty called Differential Factor (DF) in step (ii) of the encryption process which ensures that different ciphertext is always generated for each letter occurrence in the plaintext. Many of the existing schemes do not present this novelty and as such plaintext-only and ciphertext-only attacks are possible in such schemes.

ii. Unlike most of the existing systems, the proposed scheme does not require extra memory to store common lookup tables. There has been a predominant use of mapping the characters of the plaintext to some calculated points on the binding elliptic curve. Though this technique was initially revered as a breakthrough, it results in the need for common lookup tables between Alice and Bob. The ripple effect is that there is always additional huge cost on storage and time waste in lookups. Maybe for conventional computers of today, it might not be a deficiency but for some IoT devices, it's expensive.

iii. In addition, the size of the ciphertext is always equal to that of the plaintext. This is achieved through the use of the field reduction polynomial for $GF(2^{16})$ in step (ii) and step (iii) of the encryption and decryption processes respectively.

iv. Further to the above, the scheme proposes the use of efficient elliptic curve arithmetic in [12] which uses the polynomial basis to represent the elements of the underlying binary extension field, allowing very large integers to be stored as simple bit strings, and thus the field operations are reduced to string routines without the need for large CPU registers. The elliptic curve scalar multiplication is achieved through an improved López-Dahab projective coordinate method in [12] which is devoid of the computationally expensive field inversions reducing the overall computational overhead of encryption and decryption processing.

3.7. Suitability of the proposed scheme for IoT

The ecosystem of Internet of Things is very complex, ranging from conventional computers to all manners of ‘Things’. Some of these devices have limited processing and storage capabilities. The existing literature suggests lightweight schemes. However, the existing ones recommended by NIST are symmetric schemes which are generally not suitable for transit data and as such the need for public key lightweight schemes cannot be overemphasized. Lightweight cryptographic schemes are those that operate on considerably shorter operands, smaller key sizes, fewer computational overheads and bandwidth [16].

Comparing ECC based schemes with the other public key schemes suchas the discrete logarithm and integer factorization schemes, ECC is touted with its shorter key requirement of approximately 160-571 bits vs. 1,024-15,360 bits of the others whilst achieving the same level of security [17].

The proposed scheme is an ECC-based scheme. The memory and computational overhead efficiency as detailed in section 3.5.6 and 7, positions the proposed scheme to be very suitable for applications that target to run on IoT devices.

3.8. Comparison with existing works

In order to compare the existing schemes and the proposed scheme in this work, it is important to look at the following fundamental metrics of efficiency in ECC-based schemes for software implementation:

i. **Underlying finite field:** The two major fields commonly used in cryptography are prime fields $GF(p)$, and binary extension fields, $GF(2^m)$ but the latter introduce higher efficiencies as compared to the formerin software implementation [18].

ii. **Representation of elements:** The choice of the basis by which field elements are represented in schemes that are implemented over binary extension fields has a major effect on the efficiency of the finite field operations [19]. The polynomial basis, according to [20], yields better results in software implementation whereas the Normal Basis is more efficient when it comes to hardware ECC implementation.

iii. **Point Arithmetic and scalar arithmetic:** The elliptic curve scalar multiplication (ECSM), kP, is the most dominant and time consuming operation, taking about 80% of the total execution time of any ECC scheme [21]. The operation involves point additions and point doublings. Projective coordinate systems are devoid of the expensive field inversion operations which are responsible for the high computational overhead in ECSM [22, 23, 24, 25].

iv. **Points mapping and common lookup tables of plaintext:** The technique involves mapping the characters of the plaintext to some calculated points on the binding elliptic curve. This approach initially sounded like a breakthrough but the need for common lookup tables makes it unattractive due to additional storage and processing overheads.

v. **Unicode compliance for all languages:** Modern text encryption schemes should serve all languages of today and the past. Systems that are designed to handle only the ASCII character set cannot work with many languages from Asia and the Middle-East right-to-left scripts.

Leveraging on the above fundamentals, Table 3 gives a comparative summary of some existing elliptic curve-based text encryption schemes with our proposed scheme. A score of plus (+) is awarded for metrics that favour software implementation whilst a minus (-) for those that do not. A plus (+) is therefore a positive merit whilst a negative (-) is a disincentive. The Total Score column holds the sum of the pluses and the minuses.

Table 3: Comparative summary of some existing elliptic curve based text encryption schemes.

Referenced work	**Underlying field**	**Point arithmetic and scalar arithmetic require field inversions**	**Mapping of plaintext to points**	**Common lookup tables required**	**Unicode compliance for all languages**	**Total score**
Encryption of data using Elliptic Curves over Finite Fields [4]	$GF(p)$ (-)	Yes (-)	Yes (-)	Yes (-)	No (-)	-5
Implementation of Text Encryption using Elliptic Curve Cryptography [5]	$GF(p)$ (-)	Yes (-)	No (+)	No (+)	Yes (+)	+1
Security Enhancement of Text Message Based on Matrix Approach Using Elliptical Curve Cryptosystem [6]	$GF(p)$ (-)	Yes (-)	Yes (-)	Yes (-)	No (-)	-5
Implementation of Elliptic Curve Cryptography in Binary Field [7]	$GF(2^m)$ (+)	Yes (-)	No (+)	No (+)	No (-)	+2
Elliptic Curve Cryptography for Secured Text Encryption [8]	$GF(p)$ (-)	Yes (-)	No (+)	No (+)	No (-)	-1

Text Message Encoding Based on Elliptic Curve Cryptography and a Mapping Methodology [9]	$GF(p)$ (-)	Yes (-)	Yes (-)	Yes (-)	No (-)	-5
Secure and Efficient Text Encryption Using Elliptic Curve Cryptography [26]	$GF(p)$ (-)	Yes (-)	No (+)	No (+)	No (-)	-1
Proposed	$GF(2^m)$ (+)	No (+)	No (+)	No (+)	Yes (+)	+5

From the data in Table 3, one can appreciate that most of the existing systems are implemented over the prime fields that cannot currently avoid the expensive field inversions. In [7] a scheme over $GF(2^m)$ was proposed but the researchers failed to use the alternative point representation, the projective coordinate system, which has the advantage of point arithmetic devoid of field inversions. Even though the projective coordinate systems require more mathematics, their use is highly recommended due to the elimination of the unpleasant field inversions and divisions.

Another major weakness of majority of the existing systems is the Unicode compliance for all languages. The ability of a today's text encryption scheme to serve all languages is admitted. Except [5] and [10], the rest are only ASCII schemes.

On the issue of mapping plaintext characters to points on the elliptic curves, and the use of common lookup tables, a number of researchers have realized the need to eliminate such computational disadvantaged strategies.

The *Total score* for software implementation suggests that the proposed scheme in this work has the highest score.

4. Conclusion

In this work, an ECC based text encryption scheme for all languages was proposed. The Elliptic Curve Diffie-Helman Key Exchange (ECDHKE) was used by the partners to exchange a secret key for encryption and decryption. The scheme was implemented over binary extension fields hence the field arithmetics were carried out in $GF(2^m)$. The elliptic curve Point arithmetics were achieved with an improved version of the López-Dahab Projective coordinate method. The predominantly traditional technique of mapping plaintext characters to the curve points was removed and so no need for storage to keep common lookup tables nor computational time waste for lookups.

The NIST recommendations for domain parameters were followed. The test implementation proved the scheme is resistant to the best known attacks.

References

[1] J. Holdowsky, M. Mahto, M. E. Raynor, and M. Cotteleer, Inside the Internet of Things (IoT), 2015.

[2] S. A. Kumar, T. Vealey and H. Srivastava, Security in internet of things: challenges, solutions and future directions, 2016 49th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, USA, 2016. https://doi.org/10.1109/hicss.2016.714

[3] D. Hankerson, J. López Hernandez and A. Menezes, Software implementation of elliptic curve cryptography over binary fields, in: Koç, Ç.K., Paar, C. (eds.), *Cryptographic Hardware and Embedded Systems — CHES* 2000, Lecture Notes in Computer Science, vol. 1965, Springer, Berlin, Heidelberg, 2000, pp. 1-23. https://doi.org/10.1007/3-540-44499-8_1

[4] D. Sravana Kumar, Ch. Suneetha and A. Chandrasekhar, Encryption of data using elliptic curve over finite fields, *International Journal of Distributed and Parallel Systems* (*IJDPS*) 3 (2012), 301-308. https://doi.org/10.5121/ijdps.2012.3125

[5] L. D. Singh and K. M. Singh, Implementation of text encryption using elliptic curve cryptography, *Procedia Computer Science* 54 (2015), 73-82. https://doi.org/10.1016/j.procs.2015.06.009

[6] V. Kamalakannan and S. Tamilselvan, Security enhancement of text message based on matrix approach using elliptical curve cryptosystem, *Procedia Materials Science* 10 (2015), 489-496. https://doi.org/10.1016/j.mspro.2015.06.086

[7] D. R. Susantio and I. Muchtadi-Alamsyah, Implementation of elliptic curve cryptography in binary field, *Journal of Physics: Conference Series* 710 (2016), 012022. https://doi.org/10.1088/1742-6596/710/1/012022

[8] K. Keerthi and B. Surendiran, Elliptic curve cryptography for secured text encryption, in: 2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Kollam, 2017. https://doi.org/10.1109/iccpct.2017.8074210

[9] O. Reyad, Text message encoding based on elliptic curve cryptography and a mapping methodology, *Information Sciences Letters* 7(1) (2018), 7-11. https://doi.org/10.18576/isl/070102

[10] M. A. Naji et al., Cryptanalysis cipher text using new modeling: text encryption using elliptic curve cryptography, in: The 2nd International Conference on Applied Photonics and Electronics 2019 (InCAPE 2019), Putrajaya, Malaysia, 2020. https://doi.org/10.1063/1.5142095

[11] J. Hoffstein, J. Pipher and J. H. Silverman, *An Introduction to Mathematical Cryptography*, Springer, New York, 2008.

[12] Z. Abukari, E. Y. Baagyere and M. M. Iddrisu, Efficient elliptic curve arithmetic for lightweight cryptographic schemes for IoT applications, *Asian Journal of Research in Computer Science* 14(4) (2022), 228-237. https://doi.org/10.9734/ajrcos/2022/v14i4307

[13] Unicode Consortium, The Unicode Standard, Version 15.0, 2022. Available online: https://www.unicode.org/versions/Unicode15.0.0/ch01.pdf

[14] NIST, Digital Signature Standard (DSS), Federal Information Processing Standards Publication, FIPS PUB 186-4, 87-101, 2019.

[15] D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, New York, USA: Springer-Verlag, 2004.

[16] D. P. Bai et al., Elliptic curve cryptography based security framework for internet of things and cloud computing, *International Journal of Computer Science and Technology* [*IJCST*] 6 (2015), 223-229.

[17] R. M. Avanzi et al., *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, Boca Raton, FL, USA: Chapman & Hall/CRC, 2006.

[18] P. Hosseinzadeh Namin, Efficient implementation of finite field multipliers over binary extension fields, Electronic Theses and Dissertations, 5828, 2016. Retrieved from https://scholar.uwindsor.ca/etd/5828

[19] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications*, 2nd ed., NY, USA: Cambridge University Press, 1997.

[20] T. F. Al-Somani and A. Amin, Hardware implementations of $GF(2^m)$ arithmetic using normal basis, *Journal of Applied Sciences* 6 (2006), 1362-1372. https://doi.org/10.3923/jas.2006.1362.1372

[21] Y. R. Venturini, Performance analysis of parallel modular multiplication algorithms for ECC in mobile devices, *Revista de Sistemas de Informaçao da FSMA* 13 (2014), 57-67.

[22] D. Pamula, Arithmetic operators on $GF(2^m)$ for cryptographic applications: performance - power consumption - security tradeoffs, Computer Arithmetic, Université Rennes 1, HAL Open Science, 2012.

[23] M. Rasmi et al., A survey on single scalar point multiplication algorithms for elliptic curves over prime fields, *IOSR Journal of Computer Engineering* (*IOSR-JCE*) 18 (2006), 31-47.

[24] M. Rivain, Fast and Regular Algorithms for Scalar Multiplication over Elliptic Curves, 2011. Available online: https://eprint.iacr.org/2011/338.pdf

[25] E. Karthikeyan, Survey of elliptic curve scalar multiplication algorithms, *Int. J. Advanced Networking and Applications* 04 (2006), 1581-1590.

[26] N. T. Raja and K. M. Singh, Secure and efficient text encryption using elliptic curve cryptography, in: *Evolution in Computational Intelligence. Smart Innovation, Systems and Technologies*, vol. 267, Springer, Singapore, 2022. https://doi.org/10.1007/978-981-16-6616-2_51