Advances in Nonlinear Variational Inequalities
ISSN: 1092-910X
Vol 28 No. 7s (2025)

CLS-SE: Continuous Location Sharing using Geffe Generator based Stream Cipher and Elliptic Curve Cryptography

Nikhil Khandare[1*], Valmik Nikam[2]

[1*]nbk0918@gmail.com, Assistant Professor, Veermata Jijabai Technological Institute, Mumbai, India

[2]vbnikam@it.vjti.ac.in, Associate Professor,Veermata Jijabai Technological Institute, Mumbai, India

Article History:

Received: 5-01-2025

Revised: 19-02-2025

Accepted: 27-02-2025

Abstract:

With high end location enabled mobile devices and high speed internet connectivity, it is very common in today's world to share location continuously with people. However this information is very sensitive as small fraction of this information may reveal unwanted things, thus encrypting this information becomes crucial. Encryption protects this information from unwanted people and thus protects privacy of individual. For encrypting continuous location data, continuous keys are needed, which are provided by keystream generator. In this research Geffe generator is used for pseudorandom keystream generation, mutual authentication protocol is proposed for parties taking part into continuous location sharing, also elliptic curve DiffieHellam based protocol is proposed for initialization vector and polynomial of linear feedback shift registers. Stream cipher and elliptic curve cryptography is used to encrypt continuous location data and share with other parties. Proposed work is implemented in GIS software and continuous location was encrypted, security analysis was done it was found that proposed work is unbreakable and secured by mathematical hard problems like elliptic curve discrete logarithm problem.

Keywords: Linear feedback shift register (LFSR),Geffe generator, Authentication, Security, Diffie-Hellman, GIS, Elliptic curve cryptography (ECC).

1. Introduction

There is challenge of encrypting continuous location as the flow of data is continuous (the data never stops), it is continuously being sent to receiver. Thus continuous keys are needed to encrypt the location information,key should be random and should be able to convert location information into encrypted information. Linear feedback shift register can generate continuous stream of bits which can be used as keys

1.1 Linear Feedback Shift Register (LFSR)

Linear feedback shift register [1] is basically flips flops connected to each other as shown in figure 1, here s_1 to s_n are flip flops and these are connected via AND gates and x-or operation. This logical circuit is capable of generating continuous stream of bits at the output, however bits c_1 to c_n needs to be either zero or one and this defines the 'corresponding polynomial' of

LFSR. Also all the flip flops are given some initial value which is called as ‘initialization vector’ of LFSR.

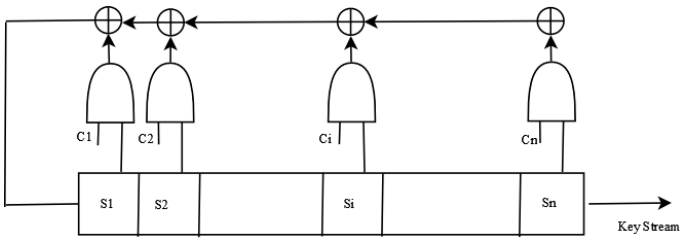

Figure 1: General structure of N-bit linear feedback shift register

For example, consider specific case of above LFSR where there are only four flip flops and $c_1=1$, $c_2=0$, $c_3=1$, $c_4=0$, thus corresponding polynomial of LFSR is $1+c_1x+c_2x^2+c_3x^3+c_4x^4$, which becomes $1+x+x^3$. If $s_1=1, s_2=0, s_3=1, s_4=1$, Initialization vector is 1011. This LFSR is useful in cryptography for encryption however there can be algebraic attack on this LFSR due to its linear nature. This problem is solved in non-linear feedback shift register.

1.2 Non Linear Feedback Shift Register (NLFSR) and Geffe generator

To address the issue of algebraic attack on LFSR, non-linear feedback shift registers (NLFSR) was proposed in [2], specific case of NLFSR is considered here called as Geffe generator as shown in figure 2. Here LFSR are combined and output of keystream generator (NLFSR) is some function of $LFSR_1$, $LFSR_2$, and LFSR3.Let keystream generated by $LFSR_1$ is x_1, keystream generated by $LFSR_2$ is x_2, and keystream generated by $LFSR_3$ is x_3. The output of Geffe generator (NLFSR) is given as $f(x_1, x_2, x_3)=(x_1 \wedge x_2) \oplus (x_2 \wedge x_3) \oplus x_3$.

Advantage of gaffe generator is that it has increased period of keystream, algebraic attack is not possible due to its non-linear nature and randomness of keystream is increased as we want truly random bit for perfect secrecy. Generating a truly random bit is not possible in practice, however cryptographers try to generate pseudo random bit for ensuring perfect secrecy.

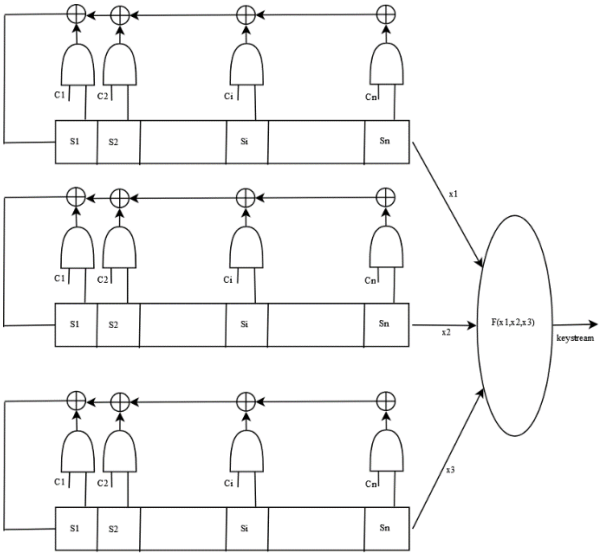

Figure 2: Geffe generator: Non-linear feedback shift register

Figure 2 mainly shows Geffe generator (specific case of NLFSR) and keystream generated at output is function of 3 LFSR's.

1.3 Geffe generator based stream cipher

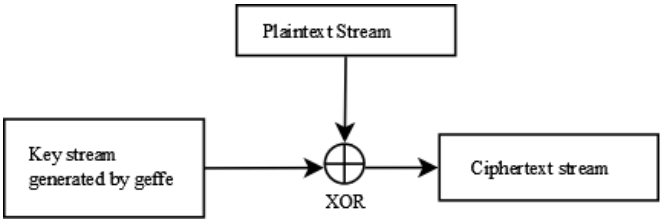

Figure 3: Geffe generator based stream cipher

Figure 3 shows encryption mechanism used by stream ciphers to encrypt continuous data stream. Here plaintext or data stream is continuously XORed with continuous keystream and resultant cipher text stream is generated and this mechanism is used in this research to encrypt the location data which is continuously being sent from sender to receiver. Let plaintext stream is 11001…..and keystream generated by Geffe generator is 01010….., then cipher text stream is 10011 (which is XOR of plaintext and cipher text stream).

1.4Diffie-Hellman key exchange

Secure way of sharing key among parties over insecure channel without actually sending key over channel was given in [3]. Here both parties agree over prime field and primitive element of the group 'g', let 'a' be secret key of party A, 'b' be secret key of party B. Shared secret key thus becomes a*b*g. There was issue of key sharing in private key cryptography and this was solved and Public key cryptography was started after [3]. Difficulty of this technique depends of difficulty of solving following problems,

Discrete logarithm problem (DLP)*: Given <G,*>, where 'G' is multiplicative cyclic group and 'α' is generator of that group, cardinality of G i.e. |G| is 'n',if'a' is chosen such that 0<=a<=n-1 and calculate β= α^a, then discrete logarithm problem is given (G, β, α) find 'a'.*

Computational Diffie Hellman problem (CDHP):*Given <G,*>, where 'G' is multiplicative cyclic group and 'α' is generator of that group, cardinality of G i.e. |G| is 'n', if'a' is chosen such that 0<=a<=n-1 and if'b' is chosen such that 0<=b<=n-1, then computational Diffie Hellman problem is given (G,α^a,α^b,α) find α^{ab}.*

Decision Diffie Hellman Problem (DDHP):*Given <G,*>, where 'G' is multiplicative cyclic group and 'α' is generator of that group, cardinality of G i.e. |G| is 'n', if'a' is chosen such that 0<=a<=n-1 and if'b' is chosen such that 0<=b<=n-1, then Decision Diffie Hellman Problem is given (G,α^a,α^b,α^k), decide whether (α^k==α^{ab}). Which means tip is given that 'α^{ab}' is the answer(shared secret key), but still 'Whether answer is correct or not?' cannot be verified.*

Above problems becomes more difficult when combined elliptic curve cryptography as it becomes elliptic curve discrete logarithm problem (ECDLP), elliptic curve computational Diffie Hellman problem (ECCDHP) and elliptic curve decision Diffie Hellman problem (ECDDHP).

1.5 Elliptic curve cryptography

Elliptic curve cryptosystem (ECC) was proposed in [4] and today most cryptosystems are using ECC since it is fast and secure. Elliptic curve over real is given by equation

$$E= \{(x,y) \mid y^2=x^3+ax+b, \text{ where } a,b \text{ € } Rs.t.\ (4a^3+27b^2)!=0 \text{ U } \{o\} \}\ldots\ldots..A$$

here, $(4a^3+27b^2)!=0$ is non singularity condition and {o} is point at infinity

after drawing the curve, addition of point ($P=(x_1,y_1),Q=(x_2,y_2)$) over elliptic curve is defined as,

Point addition: Straight line is drawn through two points ($P=(x_1,y_1),Q=(x_2,y_2)$) on curve and this line will intersect curve at some point R, reflection of this point R over x-axis is denoted by $R'=(P=(x_1,y_1) + Q=(x_2,y_2))$.There are various cased in this point addition.It can be easily verified that elliptic curve E over operation '+' will form a group, Abelian group and cyclic group. Curve can be drawn by using points and there are cryptosystems defined over these points are curve.Elgamal cryptosystem and Massey Omuracryptosystem is defined over elliptic curve points and is more secure because for breaking the cipher one needs to solve more difficult problems,

Elliptic curve discrete logarithm problem (ECDLP): let E be elliptic curve over finite field F_q, consider point P and Q on curve, where Q=k*P, thus finding this number 'k' is elliptic curve discrete logarithm problem.

Elliptic Curve Computation Diffie Hellman Problem (ECCDHP):*let E be elliptic curve over finite field F_q, consider point P, $P_a=(a*P)$, $P_b=(b*P)$ on curve, ECCDHP is given (P,P_a,P_b) find $P_{ab}=a*b*P$.*

Elliptic Curve Decision Diffie Hellman Problem (ECDDHP):*let E be elliptic curve over finite field F_q, consider point P, $P_a=(a*P)$, $P_b=(b*P)$ on curve. Intruder gets a tip that P_k is required point, ECDDHP is given (P,P_a,P_b,P_k) to verify that $P_k=P_{ab}$ or not.*

This paper is organizes as follows, section 1 introduces to topics used in this research and sets the ground, section 2 discussed work done in past related to this research, proposed work and its modules are discussed in section 3, section 4 discusses the implementation of this work in GIS software, security analysis of proposed work is done in section 5, section 6 concludes the paper.

2. Related Work

Lot of work is done on protecting the privacy of individual when it comes to sharing geographic data or location information, this section discusses work related to this research.

In location based services (LBS), location service provider (LSP) and cloud service provider (CSP) play very important role, users location can be misused by LSP, thus to address this issue, 'MobiShare+' was proposed to protect the privacy of individual from LSP in online social network which are mobile[5]. Fog computing is extended version of cloud computing, there are many security issues in cloud computing which are also present in Fog computing,

all major security issues in fog computing and solution to these issues was discussed in [6]. Information communicated by IoT devices also needs to be shared securely, thus secure method for sharing information of IoT devices was given in [7], and this method used blockchain technology, simulation of above method indicated that system is safe and feasible.In mobile hospital service, patient should share his location with the hospital, to handle privacy issue in this case, blockchain based protocol was proposed to preserve privacy and secure sharing of location [8].

Most location privacy preservation algorithms are applying some cryptographic algorithm at location based server end or social network server end or at both, however this requires huge computing power and time, to overcome this distributed processing, centralized algorithm 'CenLocShare' was proposed for location privacy combining the two servers into one[9]. To address the issue of node compromise in wireless sensor network, key based on location were introduced by incorporating ID and private key of individual with location also a fault tolerant algorithm was proposed based on location which protects the system from bogus data injection attacks[10]. City sense application used traces from all metropolitan area to show most popular places in the city, the question of how to apply the research in location privacy on this application was addressed, various areas of location privacy were categorized and existing solution inappropriateness was highlighted [11]. Data of 21 months was collected from commercial location based online social network and it was analyzed, it was found that privacy concerns of individual are related with their age, location, gender to name few[12]. User sometimes cloak the region using for privacy protection mechanism without taking into consideration the nearby cells, malicious user who is aware about the geography of region may guess the location correctly, to address this issue, new framework for location privacy was proposed called cloaking of location information semantically[13].

While sharing location with friends, user cannot trust all of his friends also user location privacy and network privacy should be protected, thus to address these issues user defined location sharing privacy was introduces in [14] and validation of protocol was done through simulations. Data of 30 participants for 3 weeks was taken and intruder tried to get information about location attributes of user like age, sex, education to name few, attack was successful and was done on large scale with thousands of users, to solve this issue, 'SmartMask' was proposed for system level security and privacy protection[15]. Order retrievable encryption was proposed for PPLSS (location sharing service privacy preservation) which gave advantage of directly sharing location with friends, preserving privacy of individual against service provider, low computation cost and runtime updates of location [16]. Novel algorithm for encryption of latitude and longitude was proposed in [17], where user can decrypt the encrypted location information when his location coordinates matches with the key, inaccuracy of current GPS devices was also considered and some tolerance was allowed, this algorithm was named as location dependent algorithm for data encryption.Privacy preservation by encrypting historical geographical data was done in [18], location, time and identity of individual was encrypted using elliptic curve cryptography.

In literature, huge efforts have been made to protect privacy of individual by encrypting location coordinates, however, little attention has been paid towards encrypting location information continuously for long time. In this paper continuous stream of location

information is encryption using stream cipher and elliptic curve cryptography. The detailed discussion about this research is done in section 3 which discussed various modules of this work which include authentication, Initialization vector sharing, LFSR polynomial sharing, encryption and decryption. Effort has been made to implement this work in GIS software and location was encrypted successfully by gaffe generator based stream cipher.Sending of data is done over insecure channel by elliptic curve cryptography which is secured by mathematical hard problem discussed in section 1.

3. Proposed Work

This section discusses proposed work. Here, overall architecture of proposed work is given, how user interacts with proposed system is given through use case diagram, authentication protocol, Initialization vector sharing is described and finally how encryption and decryption takes place is described.

3.1 Overall architecture of proposed work

Overall architecture of proposed work is given in figure 4, here two party A is sending continuous location data to party B over insecure channel using stream cipher and elliptic curve cryptography.

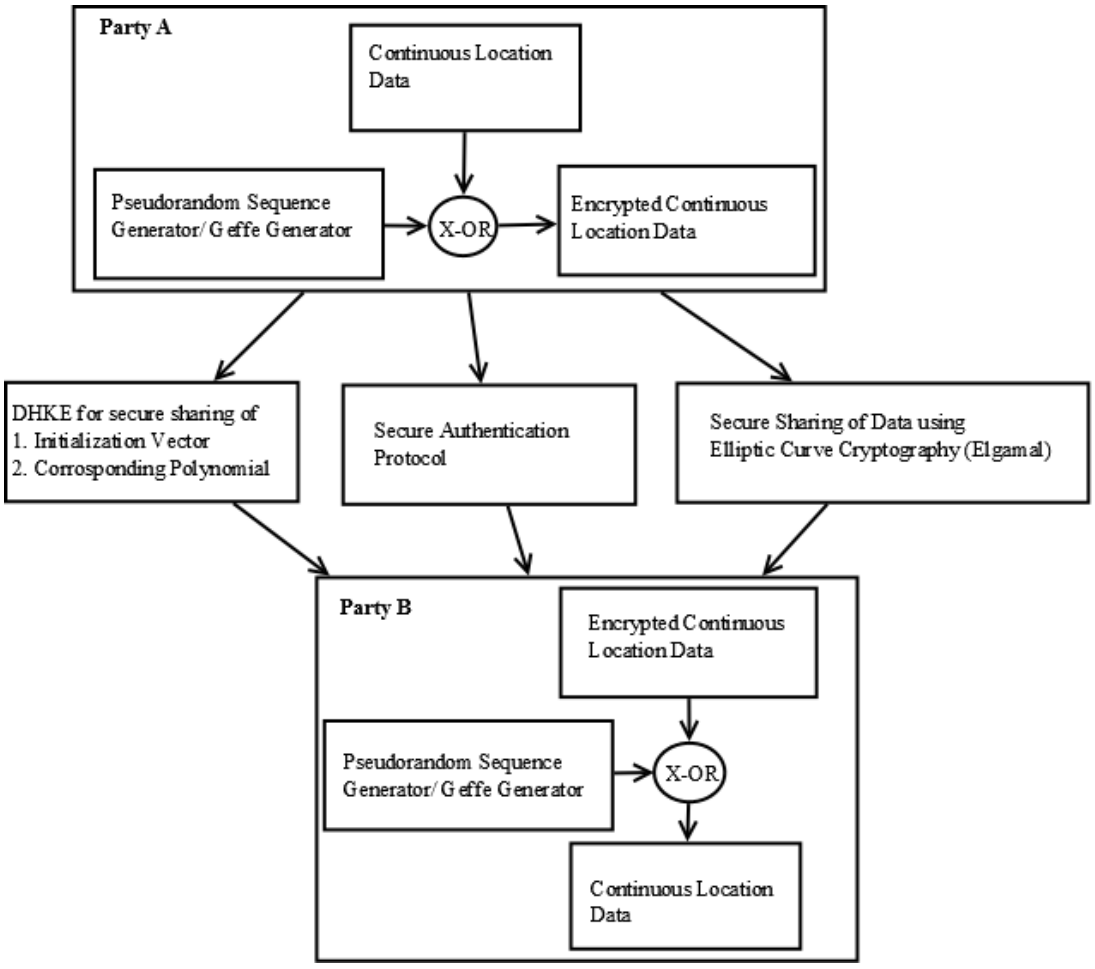

Figure 4: Overall Architecture of proposed work

Detailed explanation about overall architecture shown in figure 4 is as follow,

1. User B request the continuous location of Party A, now based on anyone's request party A cannot share his continuous location. To check authenticity of request made by B is authentic authentication protocol is used. Party A should be authenticated to party B and Party B should be authenticated to party A i.e. mutual authentication should take place.Mutual authentication protocol used will be discussed in section 3.3.

2. Party A wants to send continuous location to party B, on each clock pulse location (x=latitude=longitude) is taken and converted to binary stream of bits (eg. 101001….). This continuous stream of bits is encrypted using stream cipher, but stream cipher needs continuous random keys which are generated by Geffe generator, however Geffegenerator has LFSR's which needs to be initialized using bits called as 'Initialization vector' and these LFSR's need polynomial according to which keystream is generated. Now question of how to agree on initialization vector and how to give polynomial to LFSR is discussed in section 3.4

3. Now continuous location information (stream of bits) is X-OR'ed with continuous keystream (generated by Geffe generator) and output is called ciphertextstream.

4. This output stream is then shared to party B using Elgamal elliptic curve cryptosystems. On the receiver side decryption is done using elliptic curve cryptography and ciphertext stream is obtained which is further decrypted (X-OR operation) using stream cipher. Now keystream is required for decryption is provided by Geffe generator, which is initialized using initialization vector and polynomial shared using protocol discussed in section 3.4. Detailed process of encryption and decryption is discussed in section 3.5.

3.2 Use case diagram of proposed work

Figure 5 shows use case diagram of proposed work, it shows how user interacts with proposed system, major use cases include,

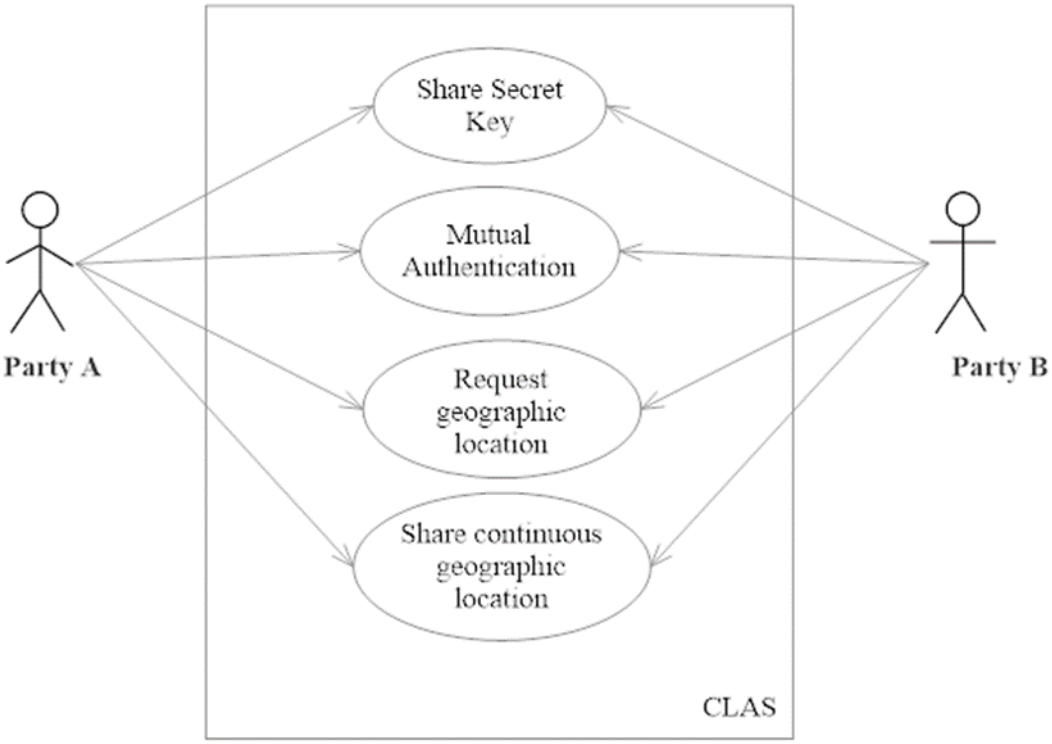

Figure 5: Use case diagram of proposed work

1. Mutual authentication: both parties should be authenticated to each other in order to send request or send data.

2. Share secret key: both parties should agree on key using Diffie Hellman based key exchange for initialization vector and corresponding polynomial of LFSR

3. Any party can send request to other party for getting continuous locationinformation.

4. Any partycan share continuous geographical information with other party using stream cipher and elliptic curve cryptography.

3.3 Authentication protocol for proposed work

Only authenticated parties can request location from other parties and send continuous location to other parties, thus strong authentication protocol is needed to achieve highest security, protocol is shown in figure 6. Conventions used in authentication protocol are listed below,

PUB_a: Public Key of A,

PUB_b: Public Key of B,

K_a: Private key of A,

K_b: Private key of B,

ID_A: Identity of A,

BU_A: Biometric of A,

loc_A: location of A,

Ts_A: Instantaneous location of A,

E(): Encryption function,

D():Decryption function,

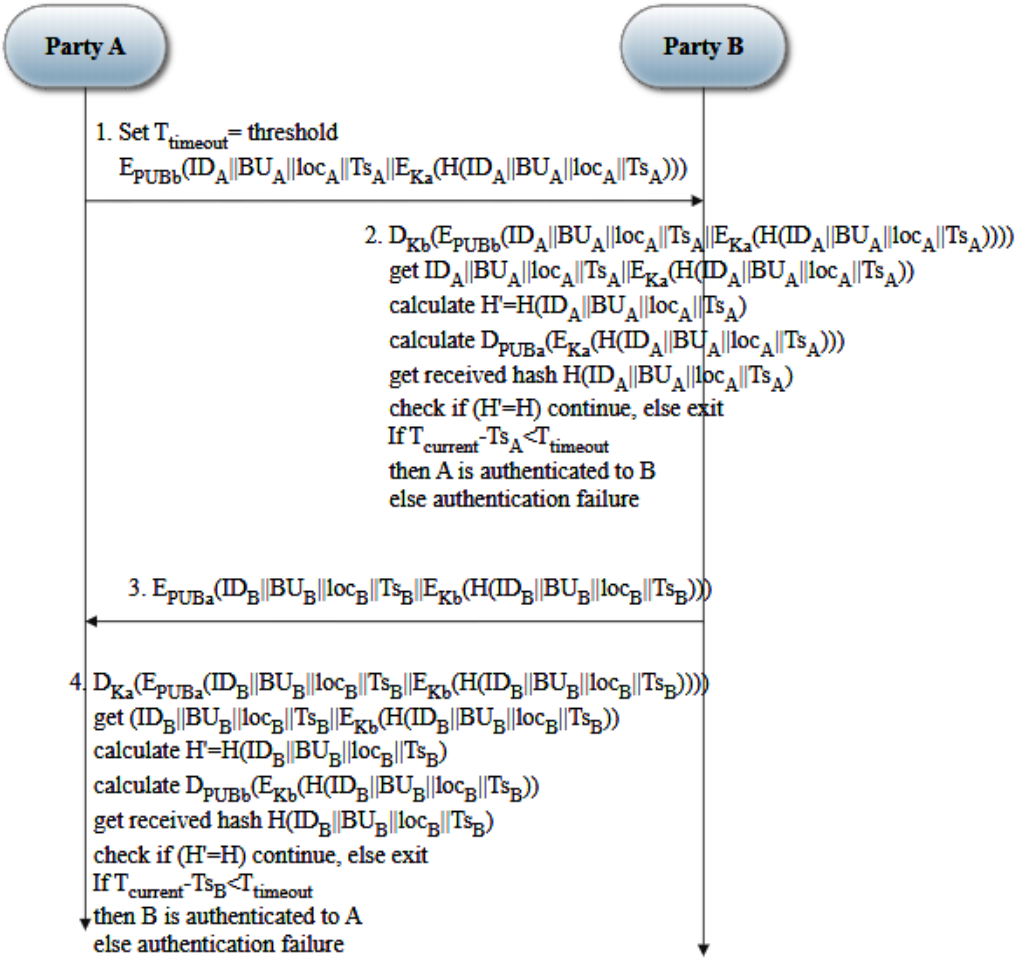

Figure 6: Mutual authentication protocol

Figure 6 mainly shows steps required to complete mutual authentication, these steps are described below,

1. Session should not be longer than threshold time thus threshold is set for example 120 seconds (i.e.Set$T_{timeout}$= threshold). Party A sends to party B$E_{PUBb}(ID_A||BU_A||loc_A||Ts_A||E_{Ka}(H(ID_A||BU_A||loc_A||Ts_A)))$, i.e. Message encrypted using public key of B, so that it can be decrypted only by private key of B. Message

has (identity, biometric, timestamp) and (identity, biometric, timestamp) encrypted using private key of A or digital signature of A (so that it can be decrypted only using public key of A).

2. Party B decrypts the message using private key of B i.e. it calculates $D_{Kb}(E_{PUBb}(ID_A||BU_A||loc_A||Ts_A||E_{Ka}(H(ID_A||BU_A||loc_A||Ts_A))))$ and gets $(ID_A||BU_A||loc_A||Ts_A||E_{Ka}(H(ID_A||BU_A||loc_A||Ts_A)))$. Then it calculates hash of $(ID_A||BU_A||loc_A||Ts_A)$ and check if it is equal to received hash, but hash is encrypted by private key of A, thus party B decrypts using public key of A. If (two hash are equal) then proceed else exit, also check,

   If (biometric details of A, if matching) then proceed else exit, also check,
   If ((Current time-Time of sending message by A) < Threshold time) then A is authenticated to B else authentication failure.

3. Party B should also be authenticated to Party A, thus Party B sends to party A, $E_{PUBa}(ID_B||BU_B||loc_B||Ts_B||E_{Kb}(H(ID_B||BU_B||loc_B||Ts_B)))$, i.e. Message encrypted using public key of A, so that it can be decrypted only by private key of A. Message has (identity, biometric, timestamp) and (identity, biometric, timestamp) encrypted using private key of B or digital signature of B (so that it can be decrypted only using public key of B).

4. Party A decrypts the message using private key of A i.e. it calculates $D_{Ka}(E_{PUBa}(ID_B||BU_B||loc_B||Ts_B||E_{Kb}(H(ID_B||BU_B||loc_B||Ts_B))))$and gets $(ID_B||BU_B||loc_B||Ts_B||E_{Kb}(H(ID_B||BU_B||loc_B||Ts_B)))$. Then it calculates hash of $(ID_B||BU_B||loc_B||Ts_B)$ and check if it is equal to received hash, but hash is encrypted by private key of B, thus party A decrypts using public key of B. If (two hash are equal) then proceed else exit, also check,

   If (biometric details of B, if matching) then proceed else exit, also check,
   If ((Current time-Time of sending message by B) < Threshold time) then B is authenticated to A and mutual authentication success, else authentication failure.

3.4 Initialization vector of LFSR and corresponding polynomial of LFSR sharing

As discussed in section 3.1, there is issue of how to initialize LFSR's in Geffe generator and how to give polynomial to LFSR so that they can generate continuous keystream which is required for initial encryption of continuous location data by stream cipher. Method for calculating IV and polynomial of LFSR is shown in figure 7. Notations used are given here,

d_A: Private key of A,

d_B: Private key of B,

g: generator of cyclic group <G,>,*

g_{AB}: shared secret key,

IV: Initialization Vector

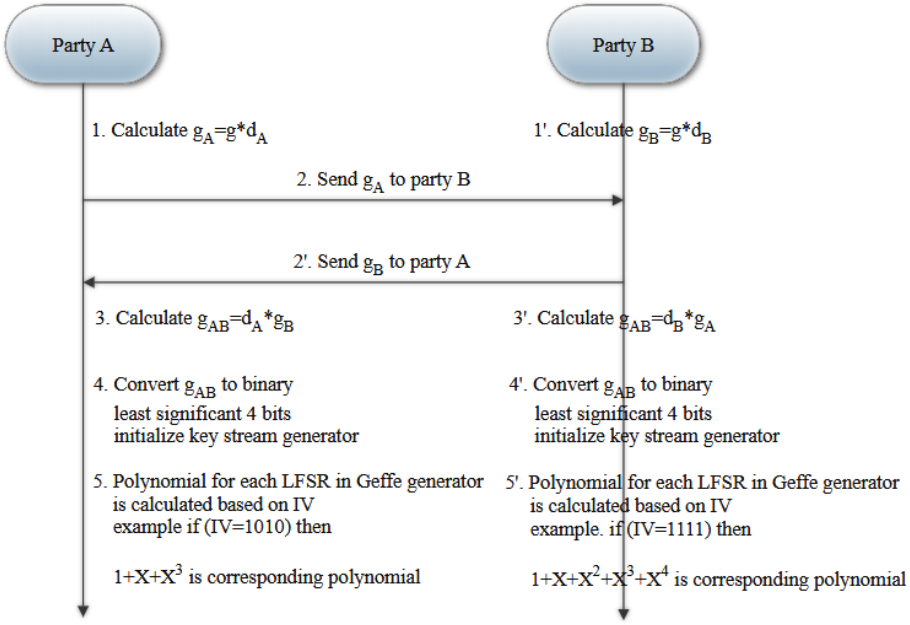

Figure 7: Shared secret key for parties, Initialization vector for LFSR and polynomial for LFSR agreement

Figure 7 mainly shows steps to be followed to get IV and polynomial of LFSR, details of steps are described below,

1. Let <G,*> be any cyclic group and let 'g' be generator of that group, party A calculates $g_A=g*d_A$. At the same time party B calculates $g_B=g*d_B$.

2. Party A send g_A to party B and party B send g_B to party A.

3. Party A calculates $g_{AB}=g_B*d_A$ and party B calculates $g_{AB}=g_A*d_B$. Thus both get same shared secret key.

4. Both parties convert shared secret key to binary and least significant 4 bits becomes initialization vector of each LFSR. For example if shared secret key is 29, then converting to binary it becomes (11101), thus initialization vector is (1101) we indicate it by ($c_1c_2c_3c_4$)

5. Polynomial for LFSR is calculated on the basis of initialization vector, polynomial is of the form $1+c_1x^1+c_2x^2+c_3x^3+c_4x^4$, where constants c_i's are taken from initialization vector. Thus for the same example polynomial of LFSR becomes $1+x+x^2+x^4$.

3.5 Encryption and Decryption of continuous location data

Encryption and decryption of continuous location data is discussed in this section,

3.5.1 Encryption of continuous location data using stream cipher and elliptic curve cryptography

Process of encrypting continuous location data is shown here in figure 8,

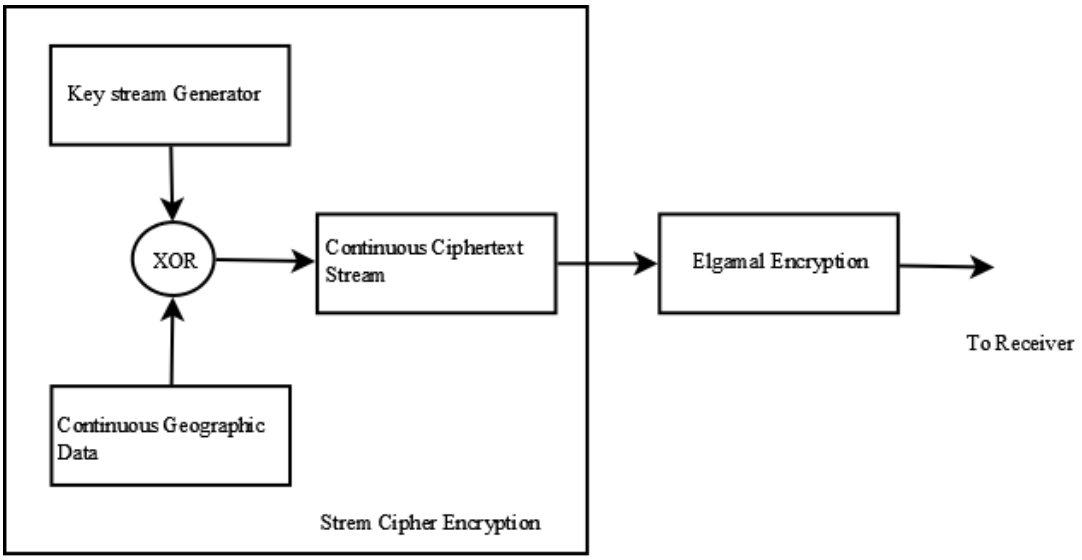

Figure 8: Encryption process for continuous geo-location data

Figure 8 mainly shows how continuous location data is encrypted, detailed steps in encryption are given below,

1. Continuous location data (x=latitude, y= longitude) is converted into binary information which is 64 bit, where first 32 bit are latitude and last 32 bit longitude. These 32 bit are further subdivided as first 8 bit numerical part of latitude and next 24

bit are fractional part of latitude and for longitude first 8 bit numerical part of longitude and next 24 bit are fractional part of longitude. As an example consider (latitude=19.0222, longitude=72.8561) now this (lat, long) information is converted into binary as latitude= **00010011**00000101101010111011100000 and longitude= **01001000**11011011001010010110000O.

2. Now at each clock pulse given to LFSR each LFSR generates a pseudorandom bit (because generating truly random bit is practically impossible discussed in section 5). Let $LFSR_1$ generates $x_1=0$, $LFSR_2$ generates $x_2=1$ and $LFSR_3$ generates $x_3=0$. Then key generated by Geffe generator is $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \oplus (x_2 \wedge x_3) \oplus x_3=0$. Feeding fast clock pulse to Geffe generator, it will generate 64pseudorandom bits in negligible time. Keystream generator will continue to generate keystream infinitely (it will stop only when user stops sharing location).

3. After particular interval, 64 (latitude||longitude) bits are XOR with 64 keystream bits generated by Geffegenerator and ciphertext stream is generated in slots of 64 bits. Consider the same example, latitude||longitude is **00010011**00000101101011101110000**01001000**11011011001010010110000O. Let 64 bit keystream generated by Geffe generator be,
1010101010110100000010101010101010010111010101000010101001010101. Thus ciphertext stream of 64 bit is
1011100110110001101001000100101**11101101**01110001001111000100101O

4. Now this ciphertext stream of 64 bit is converted to decimal, (185.69391310214996337891, 237.4423). This is represented on map as encrypted location, however latitude cannot be greater than 90 thus mod 90 operation is performed on encrypted latitude and longitude cannot be more than 180 thus mod 180 operation is performed on encrypted longitude.Thus encrypted latitude, longitude represented on map will be(5.69391310215,57.4423).

5. Now this encrypted latitude, longitude is sent using Elgamal variant of elliptic curve cryptography where elliptic curve is chosen of the form given in section 1.5, elliptic curve points are calculated and generator ‘α’ is also calculated.

6. Now party A wants to send message (185.69391310214996337891, 237.4423) to party B. B's public key is taken from public key directory (α,β,P) and 'a' is secret key of party B. here 'α' is generator, 'β=a*α' so that discrete log problem is hard and 'P' is prime number. Alice sends this message in four slots, in first slot it sends 185. Now party A choose some random number 'k' and calculate $y_1=(k*\alpha)$ and $y_2=(185+ k*\beta)$, and send y_1,y_2 to party B (here 185 is numerical part of latitude).Similar process is repeated by party A for sending 6939 (fractional part of latitude), 237 (numerical part of longitude), 4423 (fractional part of longitude)to party B.

7. Above 6 steps are continuously repeated till the sender or receiver does not close the connection, Recovering back the location sent by party A to party B is discussed in decryption process in section 3.5.2

3.5.2Decryption of continuous location data using stream cipher and elliptic curve cryptography

Decryption of continuous location data using stream cipher and elliptic curve cryptography is discussed in section and shown in figure 9,

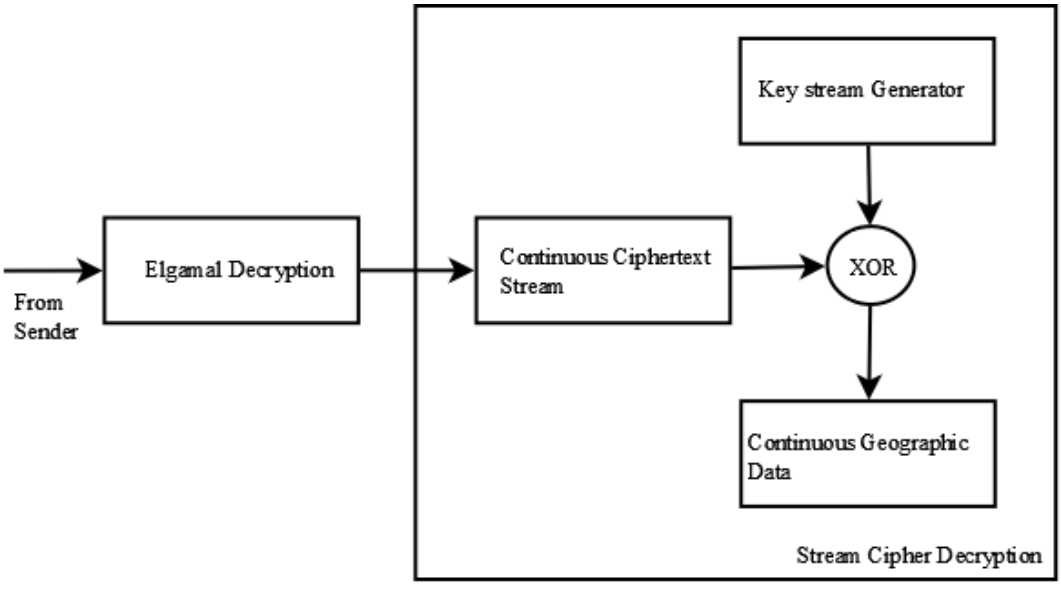

Figure 9: Decryption process for continuous geo-location data

Figure 9 mainly shows decryption process for continuous location data, details are discussed below

1. For each message (y_1,y_2) received from party A, party B calculates y_2-ay_1, after calculating this four times party B gets numerical part of latitude, fractional part of

latitude, numerical part of longitude and fractional part of longitude. Which becomes (latitude, longitude) (185.6939, 237.4423).

2. However this is encrypted using stream cipher, thus it is decrypted using stream cipher by XOR operation (since XOR is invertible function). Above latitude, longitude is converted to binary using same conventions used for encryption (64 bit, where first 32 bit are latitude and last 32 bit longitude. These 32 bit are further subdivided as first 8 bit numerical part of latitude and next 24 bit are fractional part of latitude and for longitude first 8 bit numerical part of longitude and next 24 bit are fractional part of longitude).

3. (latitude=185.6939, longitude=237.4423), which is cipher text encrypted by stream cipher, it is converted into binary as **10111001**10110001101001000100101010**11101101**01110001001111000100101O,
LFSR's at receiver side are having same IV and same polynomial thus they will generate same pseudorandom keystream as generated by party A for encryption, keystream is
10101010101101000000101010101010100101101010100001010100101010.
After performing XOR operation user gets latitude and longitude as,
0001001100000101101011101110000**01001000**11011011001010010110000O

4. This information is converted to decimal to get actual latitude, longitude as (latitude=19.0222, longitude=72.8561). This location is shown in map as decrypted location in implementation section.

5. Above steps are repeated continuously till either of party closes the connection. Above work is implemented in GIS software which is discussed in section 4.

4. Implementation and Results

Implemented of encryption of continuous geographic location using stream cipher and ECC in GIS software is discussed in this section.

4.1 Encryption of geographic location

Instantaneous location is converted into binary (64 bit) and XOR with pseudorandom keystream (64 bit) and encrypted location is calculated. In order that latitude does not go beyond 90, mod 90 operation is performed. Longitude should not go beyond 180 thus mod 180 is performed.Encrypted location is shown in figure 10.

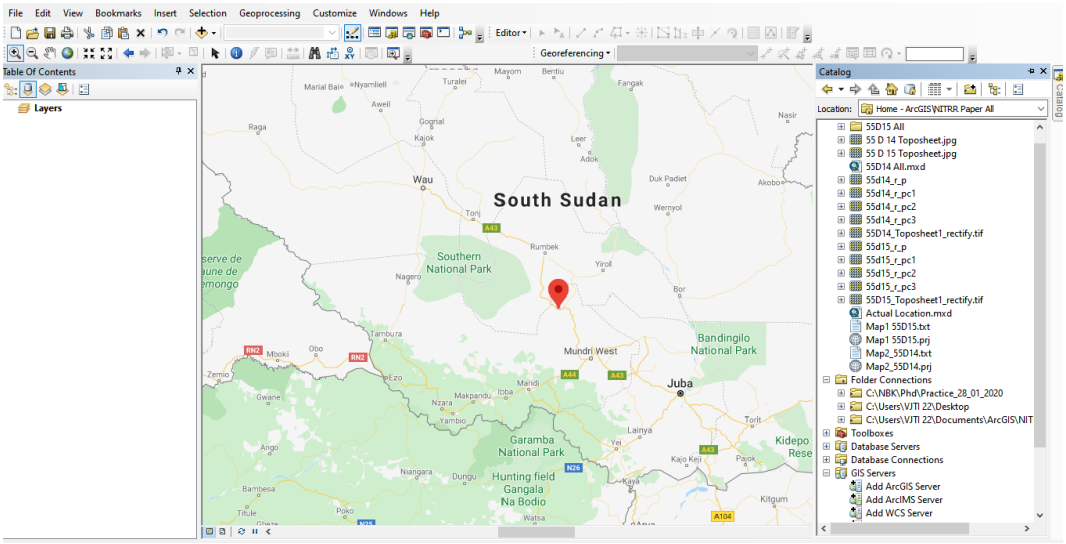

Figure 10: Encrypted continuous geo-location

Figure 10 mainly shows location encrypted using stream cipher based encryption and latitude and longitude changed significantly. From this information it appears that person is at completely different location.

4.2 Decryption of geographic location

Encrypted continuous location is decrypted using stream cipher and ECC and decrypted location is shown in figure 11,

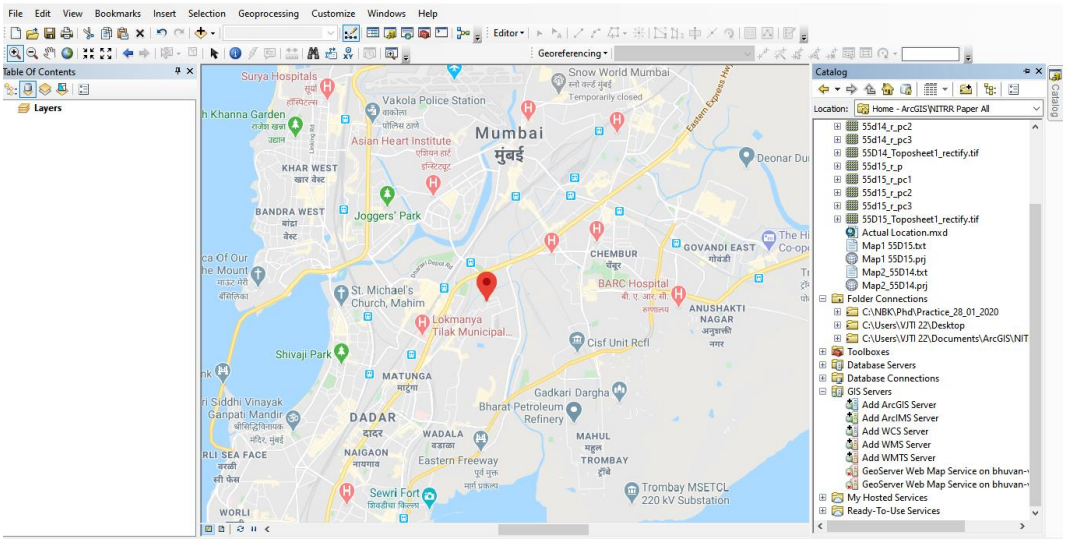

Figure 11: Decrypted continuous geo location

Figure 11 mainly shows decrypted location using stream cipher and elliptic curve cryptography.

5. Security Analysis

5.1 General properties

Non repudiation: Parties should not deny request messages or data messages sent by them in order to prevent this care has been taken in authentication protocol. While giving authentication request user encrypts information using recivers public key as well has his own private key which is also called digital signature of sender

$E_{PUBb}(ID_A||BU_A||loc_A||Ts_A||\mathbf{E_{Ka}(H(ID_A||BU_A||loc_A||Ts_A)}))$ thus party cannot deny the messages sent by them.

*Confidentiality:*Messages sent or receives form beginning are encrypted using receivers public key thus it can only be decrypted by receivers private key (which is secret to the receiver), $\mathbf{E_{PUBb}(}ID_A||BU_A||loc_A||Ts_A||E_{Ka}(H(ID_A||BU_A||loc_A||Ts_A))\mathbf{)}$

Thus no person can view contents of message except receiver.

Authentication: Only authenticated parties can request location or send location, secure authentication protocol is proposed based on public key cryptography and digital signature.

Thus it can be concluded proposed work holds all general properties of security.

5.2Analysis of stream cipher

According to Shannon's notion of perfect secure cipher, probability of getting plaintext given ciphertext is same as probability of plaintext. Which means intruder gets no advantage of knowing ciphertext,

Probablity(Plaintext|Ciphertext) = Probability (Plaintext), which means that cipher is perfectly secure cipher.This can be achieved only when keystream is truly random P (key=0) =P (key=1) =0.5.

Now LFSR will generate keystream which will have period of 2^n-1 where 'n' is size of LFSR and it will generate same period again and again. However we have used Geffe generator which has 3 LFSR's and function to combine the output of all LFSR. Thus this increases period of LFSR and increases the randomness in keybit generated by Geffe generator. Generating a truly random bit is not possible in cryptography however combining more LFSR's increases randomness to large extent thus pseudorandom bit is generated and according to Shannon's notion perfect secure cipher is implemented.

5.3 Analysis of elliptic curve cryptography

Elliptic curve cryptography is most secure and is part of most cryptosystems used today. Security of algorithm based on ECC relies mainly on solving discrete logarithm problem, when this problem is combined with elliptic curves it becomes elliptic curve discrete logarithm problem (ECDLP) which is more difficult and till date there is no solution to these

problems. In order to break the security of proposed work attacker has to solve ECDLP. Till date there is no polynomial time algorithm to solve ECDLP. Also to there is no polynomial time algorithm to solve elliptic curve computational Diffie Hellman problem (ECCDHP), elliptic curve decision Diffie Hellman problem (ECDDHP).

5.4 Attack

This can be proved that all popular attacks are defended by proposed work, there are only two ways for attacker to attack proposed system and get location information

1. *Attack on stream cipher:* Intruder should generate same keystream generated by Geffe generator and XOR it with each 64 bits of ciphertext and get back location in binary. Now this binary information can be converted to decimal and intruder gets location in decimal. However this is not so easy, generating same keystream as generated by Geffe generator, intruder needs exactly same initialization vector and corresponding polynomial which are shared using elliptic curve Diffie Hellman key exchange as shown in section 3.4. Thus to get exactly same initialization vector and same polynomial intruder has to solve ECCDHP and ECDDHP and there is no algorithm till date which will solve this problems in polynomial time.

2. *Attack on ECC:* As discussed in section 3.5, ‘α’ is generator of cyclic group and ‘β=a*α’, now both ‘α’ and ‘β’ are public and ‘a’ is secret key. Intruder can break complete cryptosystem if he is able to find ‘a’ from ‘β’ and ‘α’. Actually getting ‘a’ from β’ and ‘α’ is elliptic curve discrete logarithm problem and since there is no solution to this problem in polynomial time, intruder cannot break security of proposed work

There are only two ways to break the system either attack on stream cipher or attack on ECC, both requires intruder to solve mathematical hard problems discussed above.

6. Conclusion

In today’s era sharing continuous location becomes compulsory most of the times, with this reason protection of user’s location also becomes necessary. Secure protocols for authentication, agreement of IV, agreement of polynomial, encryption-decryption are proposed using stream cipher and ECC. Proposed work was implemented in GIS software and encrypted and decrypted location was projected on map. Security analysis was done and it was found that system is secure against all popular attacks and there are only two possible way to bring system down and get continuous location data, however, in both ways user has to solve mathematical hard problem which do not have any solution in polynomial time till date.

References

1. Murase, Makoto. "Linear feedback shift register." U.S. Patent No. 5,090,035. 18 Feb. 1992.
2. Philip, R. Geffe. "How to protect data with ciphers that are really hard to break." *Electronics* 46.1 (1973): 99-101.
3. Diffie, Whitfield. "New direction in cryptography." *IEEE Trans. Inform. Theory* 22 (1976): 472-492.
4. Koblitz, Neal. "Elliptic curve cryptosystems." *Mathematics of computation* 48.177 (1987): 203-209.
5. Li, Jingwei, et al. "MobiShare+: Security Improved System for Location Sharing in Mobile Online Social Networks." *J. Internet Serv. Inf. Secur.* 4.1 (2014): 25-36.
6. Yi, Shanhe, Zhengrui Qin, and Qun Li. "Security and privacy issues of fog computing: A survey." *International conference on wireless algorithms, systems, and applications*. Springer, Cham, 2015.
7. Si, Haiping, et al. "IoT information sharing security mechanism based on blockchain technology." *Future Generation Computer Systems* 101 (2019): 1028-1040.
8. Ji, Yaxian, et al. "BMPLS: Blockchain-based multi-level privacy-preserving location sharing scheme for telecare medical information systems." *Journal of medical systems* 42.8 (2018): 147.
9. Xiao, Xi, et al. "CenLocShare: a centralized privacy-preserving location-sharing system for mobile online social networks." *Future Generation Computer Systems* 86 (2018): 863-872.
10. Zhang, Yanchao, et al. "Location-based compromise-tolerant security mechanisms for wireless sensor networks." *IEEE Journal on selected areas in communications* 24.2 (2006): 247-260.
11. Scipioni, Marcello Paolo, and Marc Langheinrich. "I'm here! privacy challenges in mobile location sharing." *IWSSI/SPMU* (2010).
12. Li, Nan, and Guanling Chen. "Sharing location in online social networks." *IEEE network* 24.5 (2010): 20-25.
13. Damiani, Maria Luisa, Claudio Silvestri, and Elisa Bertino. "Fine-grained cloaking of sensitive positions in location-sharing applications." *IEEE Pervasive Computing* 10.4 (2011): 64-72.

14. Sun, Gang, et al. "User-defined privacy location-sharing system in mobile online social networks." *Journal of Network and Computer Applications* 86 (2017): 34-45.
15. Li, Huaxin, et al. "Privacy leakage of location sharing in mobile social networks: Attacks and defense." *IEEE Transactions on Dependable and Secure Computing* 15.4 (2016): 646-660.
16. Schlegel, Roman, et al. "Privacy-preserving location sharing services for social networks." *IEEE Transactions on Services Computing* 10.5 (2016): 811-825.
17. Liao, Hsien-Chou, and Yun-Hsiang Chao. "A new data encryption algorithm based on the location of mobile users." *Information Technology Journal* 7.1 (2008): 63-69.
18. Khandare, Nikhil B. "PPSTS: Privacy Preservation in Geographical Data by Spatio-Temporal Shifting Using Elliptic Curve Cryptography." *Wireless Personal Communications* (2020): 1-19.