Geo-Encryption Algorithm

***Abstract* - Data Security is one of the biggest concerns of the free world. Cyber-warfare and cyber-attacks are now more dangerous for a country or an organisation than any other form of attacks. Data protection has now become important more than ever. With the recent advances in the field of supercomputers and the attacks that are being made to crack the existing algorithms like the Advanced Encryption Algorithm (AES), this paper proposes a new algorithm that will randomise the data to arbitrary points on a 2D plane and generate a random key for each point. The algorithm proposed is fast and more secure than the currently used encryption algorithms.**

1.1 Cryptography

Cryptography is the study of such practices or algorithms that will allow communication between two parties in a secure manner, such that any intermediary malicious party isn’t able to trace or get their hands on the original data that was being communicated. Cryptologists design various types of algorithms keeping data confidentiality, data integrity, authentication and non-repudiation in their mind. A cryptographic system comprises of a sender who has a data (Plain text) that is encrypted on the senders’ side into an unreadable format (Cipher text) and then sent to the intended receiver who uses an algorithm or a unique key to decrypt the cipher text to the plain text and hence retrieving the original data back. Professionals have the skill in production cryptanalysis but innovation in the design of new types of cryptographic systems has been amateurs’ forte (Diffie W. et al. 1976). On that note the paper proposes a new Geo – Encryption algorithm. It is a symmetric block algorithm that can be used to encrypt data to an incomprehensible format using a key and then decrypt the data using the same key to get our original data back. We also use

**2. Introduction**

2.1 Symmetric/Asymmetric Encryption

Symmetric encryptions are supposed to be fast and secure as long as the key remains a secret. The receiver and the sender both have the same key to encrypt and decrypt the data. The issue with symmetric algorithms is that the parties must exchange the keys beforehand in a secure manner such that the keys are not intercepted. The security of such algorithms depends upon the key; therefore, it is advisable to use multiple keys. Algorithms also add an additional functionality of key scheduling to avoid keeping the keys consistent and this makes transmitting all the keys securely, important. Whereas, in Asymmetric encryption, the communicating parties use two different keys for encryption and decryption. Encryption of data is done using the Public key of the receiver and the receiver uses its Private key to decrypt the data. This Private key needs to be kept as the secret key of the receiver.

2.2 Block and Stream Cipher

In Block cipher, Algorithms take a large block of the text to be encode, typically 64 to 128 bits and encode it using a Key. This same key is used to encode the other parts of the text.

In Stream Cipher, Algorithms take a relatively small blocks of one bit or one byte long and encode them with a Key and many previous blocks. The algorithm uses different key for each bit/byte encoding.

2.3 Areas of Application

2.3.1 Encrypting Data Files or Data Streams

All sensitive data files of any format can be encrypted using the algorithm. All files are read and encrypted in bytes of 8 blocks each. A continuous data stream can be also encrypted by the algorithm efficiently. Hence, video encoding is possible

2.3.2 Random Bit generator

2.3.3 Packet Encryption – ATM PIN, Messaging

2.5 Design Decisions

No weak keys as a design goal

**3. Literature Survey:**

3.1 Elliptic Curve Cryptography (ECC)

<https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/>

<https://blog.goodaudience.com/very-basic-elliptic-curve-cryptography-16c4f6c349ed>

<https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/>

ECC is an asymmetric algorithm that has a public and private key with both the communicating parties to encrypt and decrypt. A 256 bit key in ECC is more secure than 3072-bit key size RSA. The Trapdoor function of ECC is very efficient. The private key is formed by the number of hops that would take to reach form a randomly chosen starting point on the curve to an another randomly chosen ending point whereas, the public key is formed using the starting point and ending point. It is quite impossible to know the hop numbers by just knowing the starting point and the ending point and that makes this algorithm quite beautiful. Still, there have been some questions and uncertainties regarding the algorithm that can’t be ignored completely. A Dual Elliptic Curve Deterministic Random Bit Generator, which was being used by the RSA company as well has been found to be with a backdoor which can allow attackers to predict the random sequence based on a secret key. There are many bad curves as well that will generate unprotected values. Another drawback of the algorithm is that it requires a good source of entropy. It is suitable for only smartcards, tokens and wireless and communication devices.

3.2 AES (Advanced Encryption Standard)

AES is a symmetric block cipher which has a block size of 128 bits. Variants of AES have 10 rounds for 128 bit key size, 12 rounds for 192-bit key size and 14 rounds for 256 bit key size, for encryption-decryption; where each round has four similar steps viz., Substitute Bytes, Shift Rows, Mix Column and Add Round Key. The last step involves the key size factor and thus makes it different from other AES variants.

3.3 3DES (Three Data Encryption Standard)

DES (Data Encryption Standard) was developed in 1975 at IBM. It mainly had two issues of being not strong enough and being slow. 3DES was a modified and improved version of the DES algorithm. With the advancement of the cheap digital hardware that had brought down the design limitations of the mechanical computing and the cost of making them had brought DES to a shrinking state. With just a key size of 58 bits, the algorithm was not strong enough to resist Brute-Force attacks in the late 1990s. The Electronic Frontier Foundation (EFF) decrypted a DES-encoded message in 56 hours and by the next year itself it was able to reduce its decryption time to 22 hours by harnessing the power of thousands of networked computers. Cryptologists applied DES encryption algorithm three times to each data block to overcome this issue. It was also proposed by IBM in 1978 that DES should be substituted by 3DES. 3DES uses 48 rounds in its computation and has a Block size of 64 bits. Though it is stronger than DES surely, but it made an already slow algorithm slower.

3.4 Blowfish

Blowfish is a 16-round Feistel cipher, a symmetric 64-bit block cipher that has a key size range from 32 to 448 bits. This algorithm was developed by Bruce Schneier who made it publicly available for all to use and test. It expands the key of 448bits to several subkey arrays of 4168 bytes. The algorithm was designed for performance-constrained environments like in embedded systems. The computation time increases linearly for the algorithm as the message to encrypt increase. The algorithm is much faster than DES.

**4. Building Blocks**

Boolean operations like AND, OR and XOR are analogous to set operations. AND mathematically represents intersection, OR represents union and XOR - the difference between the inputs. This is called the isomorphic nature of XOR which allows to toggle between the inputs.

If C = (A ⊕ B)

then, A = (B ⊕ C) = B ⊕ (A ⊕ B)

and B = (A ⊕ C) = A ⊕ (A ⊕ B)

So, we can get the value of A if we have the values of B and (A ⊕ B).

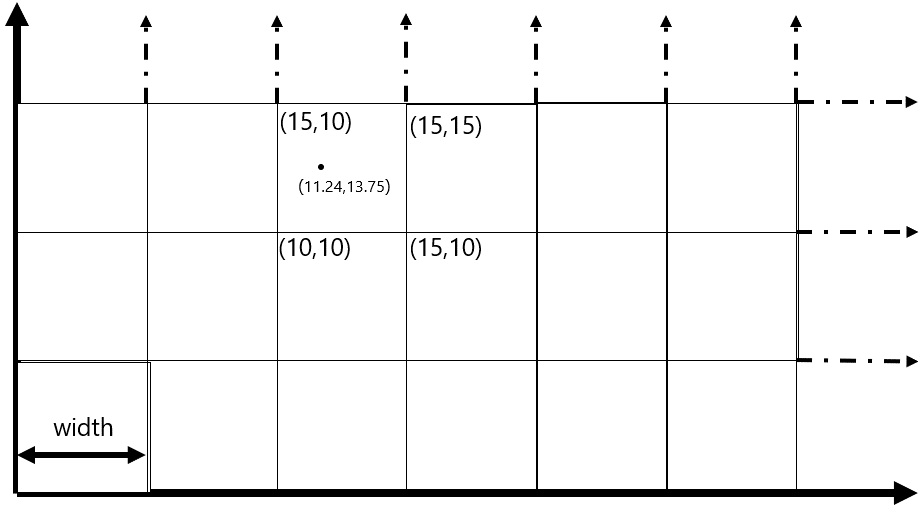


Figure 1: Cells of the Grid

**5. Description of the Algorithm**

*Configurations –*

This algorithm needs the encrypting party and the decrypting party to agree upon some configuration options, as below.

Block size (*l*): The length of the block in bits, to be treated as one entity while encrypting.

Block Value (V): Value of *l*-bit block on radix 10.

Width (w): The width of the cell.

Key (K): Random bit array of size k.

Key Length (k): The length of key in bits.

Code-Book (CB): A 2-D array index on the x-min and y-min of each cell and stores the key associated with the cell.

Block Value Mapper Bits (BVMB): It will be half the size of the key in bits.

Key Mapper Bits (KMB): This will decide the key.

*Encryption -*

The most used algorithms today such as, AES, Elliptic Curve, 3DES, etc all have a basic idea of replacing a character by another and/or use prime numbers in computation process. This makes these algorithms computationally heavy and time taking. The idea proposed here is different in these aspects, i.e. neither does it map a character to another, nor does it use prime numbers.

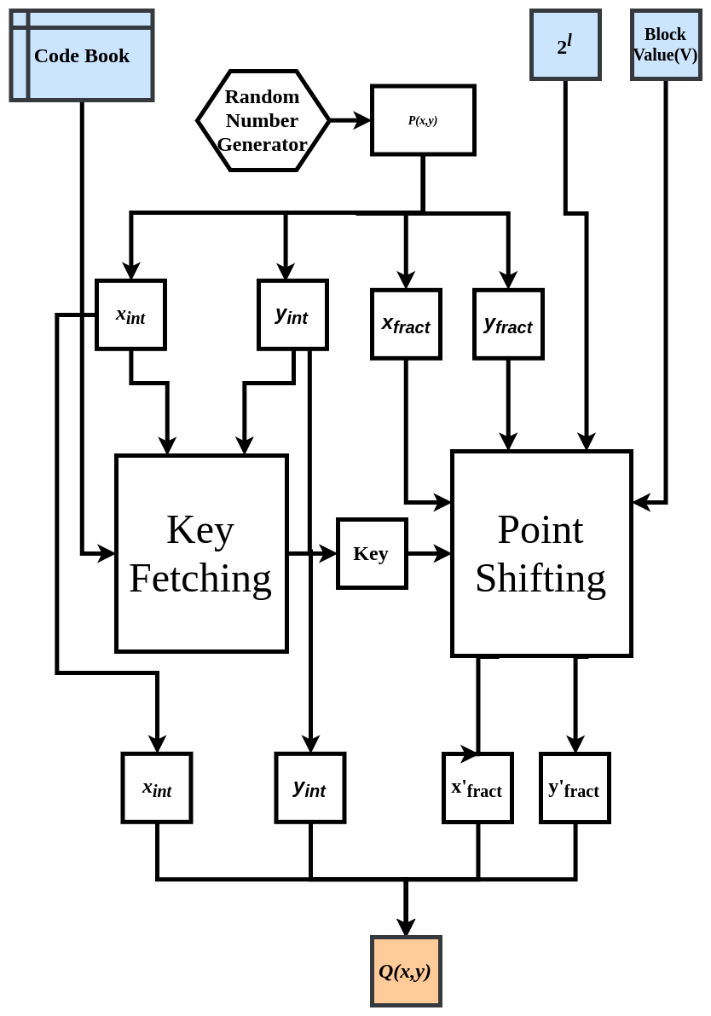


Figure 2: Encryption

If the encryption function is represented mathematically, one gets the bellow equation.

f(b): B→G

where,

B: a set of blocks of size *l.*

G: a plane along X-Y axis.

Figure 3: Encryption

When describing B, a set of blocks of a given block size infers that B is a set of bit arrays, each of length *l*. For example, if *l* =32, B is a set that comprises all the values from 0 to 232 in binary, formatted to 32bits with leading zeroes. G can be viewed as a plane with grid as shown in Fig 1. Each cell of this grid has associated with itself a random key. The key length (k) in bits is twice the size of block (*l*). All the points in each cell will only use the associated key of that cell while encryption and decryption.

k = 2 \* *l*

Where, k: Key length *l*: Block size

The Point (P) which would be represented using (*x, y*) would have four values. The integer part of *x* and *y*, and the fractional part of *x* and *y*. As the algorithm maps a block to a point P (*x*, *y*), the encryption process can simply be described as “To find a random point that maps to the desired block value”. To achieve this, a random point is selected and then shifted such that it represents the desired block value. As the randomly selected point decides the key value it is necessary that the point does not shift to another cell, and in-process change the key. So, while shifting it is noted that only the fractional part changes leaving the integer part intact. Due to this the fractional part is ignored while fetching the key from the Code-Book and integer part is ignored while shifting. As the key fetching and the point shifting depends upon distinct parts of *x* and *y*, point P can be represent as,

*P (xint, xfract, yint, yfract)*

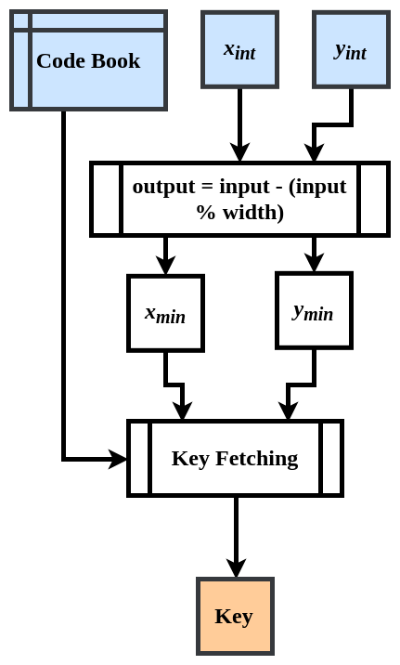
**

Figure 4: Fetching the Key

*Key Fetching –*

To fetch the key, we need to find the cell in which the randomly selected point lies. As the grid is of uniform width (w) and the Code-Book is indexed on the minima of each cell, the key can be located as,

K = CB [*xint* – (*xint* mod w)] [*yint* – (*yint* mod w)]

Now this key is used in encrypting the upcoming block of the data. The fractional part of the x and y are also random points which are concatenated together and then a XOR operation is performed on them with the key. Length of α is going to be same as that of the key.

α = (fract(x) ● fract(y)) ⊕ Key

The α value needs to be adjusted such that it represents the block value to be encoded. So, on adding the offset to β and again operating a XOR operation on it with the key,

β = α % 2*l*

offset = β – Block Value

β*'* = β + offset

α*'* = β*'* ⊕ Key

it gives a value α', which is then split into two values. The first half represents the new fract(x) and the second half represents the new fract(y). These bits have the encoded data in them.

The new fract(x) (*x'fract*) and fract(y) (*y'fract)* values are now added to the integer parts of x and y that were selected randomly in the very beginning of the algorithm, thus retaining the key location on the grid and hiding the intended data.

*P(xint,xfract,yint,yfract)****→****Q(xint,x'fract,yint,y'fract)*

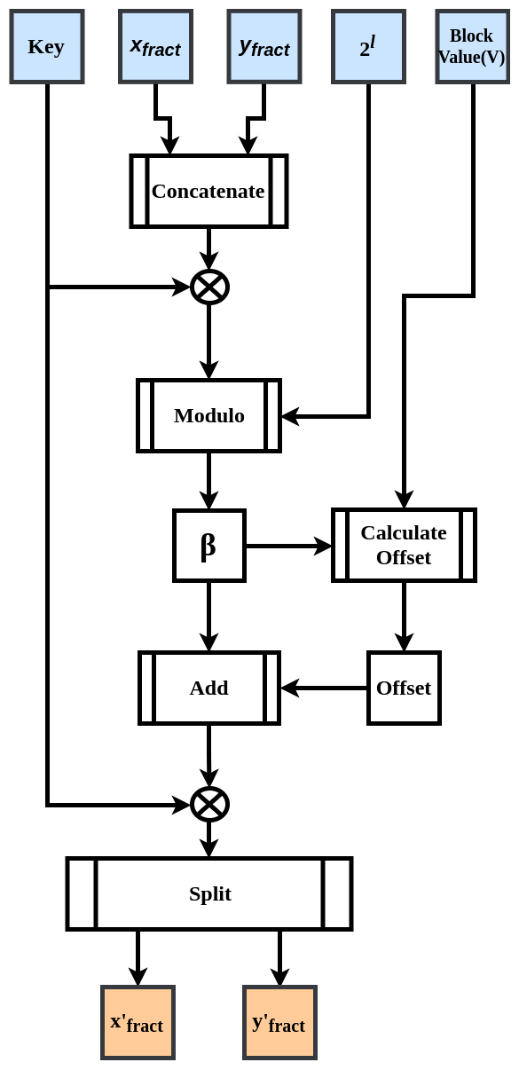


Figure 5: Shifting the Point P to Point Q

Each block is written serially in its encrypted format in a binary file which can then be shared over the network to the intended receiver (decrypting party).

*Decryption –*

The decryption process starts with the encrypted point (Q) as input. For the key, the same process is applied as in the encryption process. As the integer portion of the Point P does not change while encryption, the key fetching for decryption can be done in the same way as in the encryption process. Once the key is fetched, the block value can be retrieved from the fractional portions of Point (Q) by backtracking.

α = (fract(x) ● fract(y)) ⊕ Key

The key to be used for this block would be found using the Code-Book and the integer parts of the coordinates of that block. The fractional parts of the coordinates x and y would be concatenated and then a XOR operation would be applied on them using the specific key, thus giving us the data that the data (Plain text).

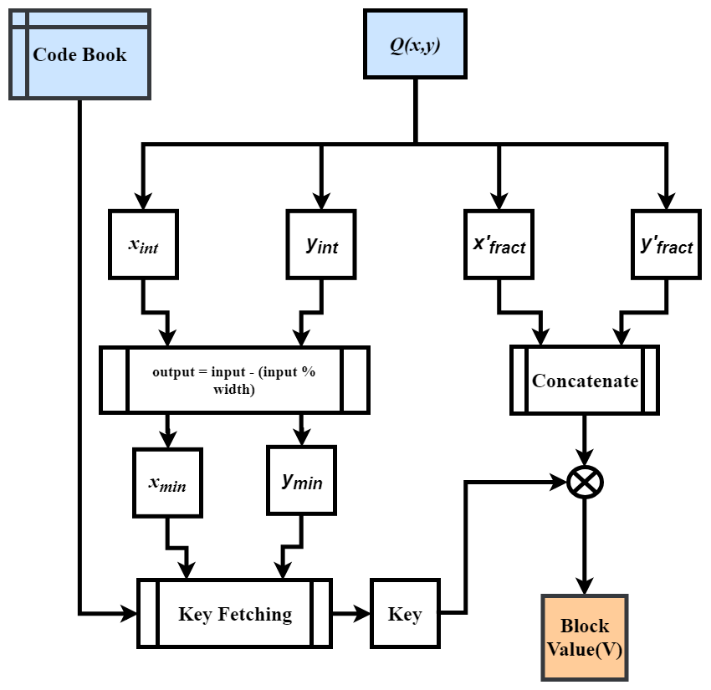


Figure : Decryption Process

**6. Discussion**

The algorithm is quite efficient in terms of speed and accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| File Size (KB) | No. of rounds | Total Time Taken | Performance (KB/s) |
| 1 | 100000 | 25.3378 | 3946.67256 |
| 10 | 10000 | 21.518 | 4647.272051 |
| 100 | 1000 | 20.6311 | 4847.051296 |
| 1000 | 100 | 20.2276 | 4943.740236 |
| 10000 | 10 | 21.1668 | 4724.379689 |
| 100000 | 1 | 22.5895 | 4426.835477 |

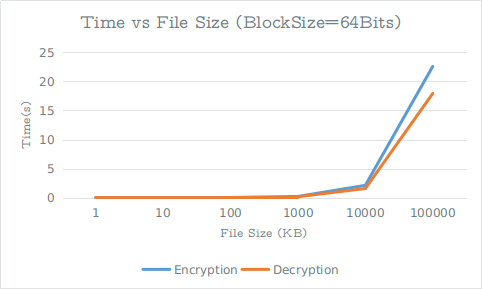
Benchmarking – Handling Attack types

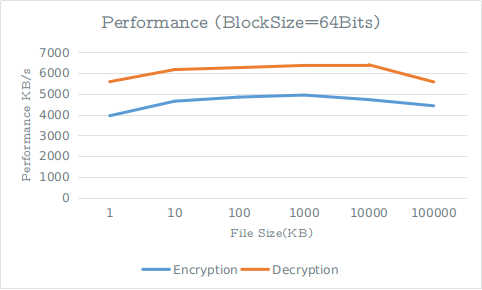
How is the Algorithm resistant to brute force attack?

A brute force attack tries all the possible keys and looks for the one which gives the decrypted data. The length of the key determines the number of possible keys and hence the feasibility of the attack. Algorithms like AES increase the size of the key to escape these types of attacks. In Geo-Encryption algorithm the keys are based on the cells which are being selected up randomly and therefore it is not certain that the characters will have the same key. Even if the same characters are encrypted again and again, they would be selecting a random key based on the sectors each time.

https://www.eetimes.com/document.asp?doc\_id=1279619#

Comparison with other Algorithms





**7. References**