IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 04, Issue 04 (April. 2014), ||V3|| PP 39-53

Mchf Algorithm for Multilevel Cloud Security

Bijoy Boban[1], Ankur Sodhi[2]
[1](School of Computer Science, Dept. of Science and Technology, Lovely Professional University)
[2](Asst. Professor, Lovely Faculty of technology, Lovely Professional University)

Abstract: **-** In this research work I introduce a new system named meiosis based ciphering and hash function generation (MCHF) algorithm for encryption, cryptographic hash function generation, pseudo random number generation, 3D login system and optimisation for various security levels of cloud computing using the concepts of meiosis rendered from molecular biology and willing to give light to a sector in computer science called Digital Cell Emulation Technique which can further be used for other purposes for better results than one which I included in this research work; here I render the concept of meiotic cell division process undergone during fertilization which is the real reason (chromosomal cross over) for dissimilarity of an individual (Genetically) to another. I consider the latest finding of chaotic movement of DNA during meiosis and the motion that takes place in the fluid is theoretically implemented using 3D Navier Stokes equation which is meant for calculating the movement of Newtonian fluids like DNA in 3D space, which I render to emulate the cross over during metaphase of meiosis. Initial encryption is carried out which imitates the cross over and then the hash code generation that imitates the nucleotide formation or DNA shrinking to form chromosomes. This algorithm ensures all the requirement standards for a good hash code algorithm with secure encryption using 1024 bit key and can be used for pseudo code generator as it maintain a good proportion between the no. of 0's and 1's in the binary output. Further the hash code will be implemented for 3D login system to 100% override brute force attack as the process of login will be 100% human interaction based. Then I will be demonstrating the optimisation pseudo code generation and will be comparing it with the existing algorithm to show the betterment in time, throughput and power with statistical result created dynamically during each execution.

Keywords: **-** *Cloud security, Genetic algorithm, Meiosis based cryptography, Navier-stokes equation.*

I. INTRODUCTION

Meiosis based Cryptographic encryption and hash function generator is a research topic which is not as explored as an individual event to be emulated although we have the genetic algorithm to be used for cryptographic purposes. The reason why this topic is of much importance is that even taught we have many optimisation algorithms we rely on genetic algorithm as it is more precise and is a pure emulation of human mutation. So the sole purpose of this paper is to develop and show an encryption technique as well as a hash function generation technique which can be implemented in cloud environment to provide security which is hard to break without the authentic user intervention. The reason why the organisms that live on earth is intelligent and different from one another stays in the cells from which we all are born. The organic bimolecular reactions create highly proficient, complex, self-correcting and self-directing processes such as meiotic replication, transcription and translation [1]. These processes, along with mutations form the basis of evolution, there by forming the foundation for the building of an intelligent life. These concepts form the basis of technological advancements for emulating computational intelligence in many engineering applications [2]. The prime reason for the dissimilarity of all the individuals who exist in this planet with different parental origin is chiasma formation during meiosis process of fertilization where the nucleotide strands from the mother and father undergoes cross over for the generation of 4 diploid cells. If we can undergo the same principles of meiosis in data with a virtual construction of A-DNA from the data bits following the conventions of Cryptography and rendering of state at which the meiosis takes place, which include the centroidal based movement of the chromosome in the nucleoplasm which we can emulate with the help of 3D Chaotic flow states for Accelerated DNA replication which uses 3D Navier-Stokes Equation in R^3 Euclidian space [3], then we can generate a highly secure Hash function that satisfies all the constrains of hash code and pseudo code which can be used at various levels of security in cloud. Security is a complex property and difficult to design or optimize. The existence of so many methods of attack makes the protection of an information system very complicated. The secrecy of transmitted information using encipherment and also with authentication of information verifying the identity of people, preventing the stealing of information and controlling access to both data and software have become such vital issues today, that data security is highly essential. In cryptography, the message is transformed so as to be unintelligible even though its existence is apparent [7]. Prior encryption with the help of chiasma formation is carried out before the hash code generation, the encryption consist of 'n' rounds which will be selected so that it support the non-retrieval or close encounter of the cipher code back to the plain code

when implementing the encryption in 256 bit ASCII (we are starting the test with 128 rounds). Padding is done at every node with each node having a size of 1024 bits with 896 bit of data and 128 bit of padding, the first 8 bits of all the prime numbers from 3 to 1000 is used for the padding. Last 128 bit shows the length of the original message. The padding will be done on the basis of the emulation of the conditions for PCR during meiosis considering the onset of flow and transition to convective turbulence as determined by the dimensionless Rayleigh number: ($Ra = [g\beta(T2–T1)h3]/v\alpha$ which explains the chaotic advection in micro-scale flow geometries which can be harnessed to greatly enhance the rate of thermally activated PCR [2]. For the encryption technique we use SHA512 like data handling with the ASCII intake similar to elliptical curve encryption which can be called an elliptical- modified version to the RC4 swapping. One of the main issues involved in cryptography is the existence of a global key [8]. A global key of appropriate length is chosen for the encryption process from which, at each stage, a sub-key is derived.

We use the Navier- Strokes Equation to emulate the cross over encryption, in Mathematics the Navier–Stokes equations are a system of nonlinear partial differential equations for abstract vector fields of any size, in physics and engineering, they are a system of equations that models the motion of liquids or non-rarefied gases using continuum mechanics. The equations are a statement of Newton's second law, with the forces modelled according to those in a viscous Newtonian fluid as the sum of contributions by pressure, viscous Stress and an external body force. Rheology is the study of the flow of matter, primarily in the liquid state, but also as 'soft solids' or solids under conditions in which they respond with plastic flow rather than deforming elastically in response to an applied force [3]. It applies to substances which have a complex molecular structure, such as mud, sludge, suspensions, polymers and other glass formers (e.g. silicates), as well as many foods and additives, bodily fluids (e.g. blood) and other biological materials. Newtonian fluids can be characterized by a single coefficient of viscosity for a specific temperature. Although this viscosity will change with temperature, it does not change with the flow rate or strain rate. Only a small group of fluids exhibit such constant viscosity, and they are known as Newtonian fluids. So we consider DNA as a Newtonian fluid. So it proves that we can use Navier- Strokes equation on DNA movement and we consider DNA as a Continuum Particle. The reason behind using a biological phenomenon can be well explained when considering evolutionary algorithms, their importance and success.

II. METHODOLOGY

According to the current scenario in networking we use quantum cryptographic techniques when we are using fibre optic cables to transmit data especially when we are using high speed Giga byte Ethernet switching technology. So quantum cryptography has a prime role in encrypted data transmission than encryption at the user end for confidentiality. Even NASA has banned encryption inside their vicinity. When we moved on to the cloud we were hit with the issues of security again, the prime reason was the heterogeneity and chance of open wormholes in the architecture where data is exposed. So from research I have found that the 3 prime weak spot of attack in cloud is the login area, database transactions, and data storage. What if we has a technology that can 100% avoid brute force, then it has to be fully depending on user interaction for login, thus the method of using hash code to save password and further retrieval for authentication from a 3D login system developed in java 3D came into my consideration, so that there is no loophole for any black hat to embed a brute force. Then further securing the database using a 1024 bit key hash code was the next step to avoid SQL injection attack on database, even if anyone can access the database illegally, he won't be able to read the data at any point of time as it uses a 1024 bit key hash, which takes approximately 2^{1023} operation of brute force which takes 1.797693134862315907729305190789 * e^{308} keys at 2.4 GHz processor at 80% processor capacity for billions of years to hack the code. For securing the data we will be using hash codes as explained, but how secure can a hash code be when handling millions of users in the cloud. In this case we are in need of a better hash code generation algorithm. Here we are coming up with the same reason why genetic algorithm is better than any optimisation algorithm, the phenomenon of emulating mutation i.e. meiosis, this is what that we are using to make meiosis based cryptographic hash function generation algorithm. Securing and maintaining the integrity of authorised data in cloud level is handled by the techniques involving hash codes added as digest to the document. In this research we are introducing a technique involving emulation of meiosis in DNA based cryptography. For the emulation we consider the states at which the meiosis takes place, as it's a motion of chromosomes in a Newtonian fluid in 3 dimensional spaces, we consider the Navier Stokes equation to emulate the meiosis and the transposition of bits according to the equation. Then a conflict that could arise might be the use of 1204 bit key telling that it's too long; not at all, it's the same reason we moved from SHA128 to SHA256 to SHA386 to SHA512 that made us to move to a 1024 bit key length, The processor speed of super computers are increasing day by day and increased to 33 petaflops as per 2014, and storing a 1024 bit key in cloud is not an issue now as memory is not a tradeoffs factor in 2014. And as sectors of 1024 bits are used in cloud, there won't be any fragmentation issues in cloud storage is another advantage. So making it 1024 bit key can make even the super computers to take a really long time to break the security of the code using brute force. The

implementation, demonstration, throughput, power analysis and application level solution to the security issues in cloud will be done using j2SE. We have gone through the process that is required as general for the implementation of the proposed algorithm and have come up with an algorithm as shown below.

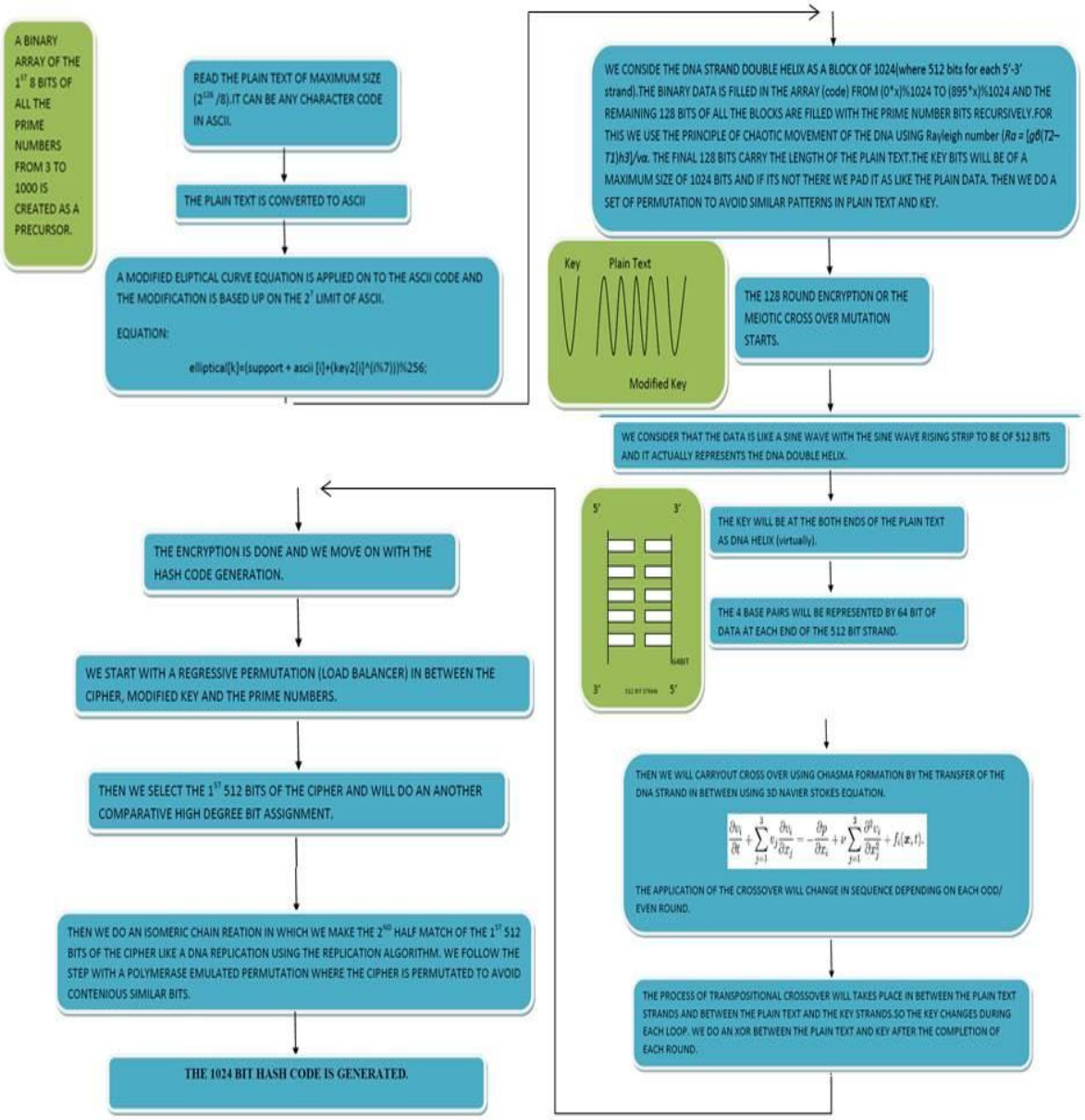


Fig 1: Proposed sample algorithm.

Considering the algorithm we are going to generate the code part to implement the execution part of each phase in the given algorithm.

2.1 CROSSOVER PHASE.

The code for chiasma swap is as in Fig 2.

```
for(i=512;i<(y-512);i+=1024)
{
    for(j=i,k=i+1023;j<(i+512),k>((i+1023)-512);j+=128,k-=128)
    {
        for(l=j,m=k;l<(j+64),m>(k-64);l++,m--)
        {
            temp=node[l];
            node[l]=node[m];
            node[m]=temp;
        }
    }
}
```

Fig 2. Chiasma formation

1. We consider the entire data bits like a sine wave, in which each 512 bit is one entire DNA strand
2. Traversal is from the 512th bit to the last bit minus 512 bit, because its reserved for the cross over to mutate the key.
3. Each 512 bit strand is further divided into 64 bits of segment which represent the base pairs A, T, C, and G.
4. One segment of 64 bit is crossovered with its adjacent segment of another DNA segment, but not all segments are crossovered, in the odd round even segment numbers are crossed and vice versa.

2.2 KEY CROSSOVER PHASE

We avoid the 1st and last 512 bits of the data bit as we do the chiasma of those bits with the key bits and it modifies the key as shown in Fig 3.

```
for(i=0,j=512;i<512,j<1024;i+=128,j+=128)
{
    for(k=i,l=j;k<(i+64),l<(j+64);k++,l++)
    {
        temp=node[k];
        node[k]=knode[l];
        knode[l]=temp;
    }
}
//left end done
//key modified
for(i=(y-1),j=0;i>((y-1)-512),j<512;i-=128,j+=128)
{
    for(k=i,l=j;k>(i-64),l<(j+64);k--,l++)
    {
        temp=node[k];
        node[k]=knode[l];
        knode[l]=temp;
    }
}
//right end done
```

Fig 3. Chiasma modifies the key.

1. Similar to the cross over phases in the data bits, we do crossover on the keys with the 0-511 bits and the last 512 bits of data with the keys at both ends of 1024 bits each.

After the sequence of encryption we move on to the hash code, there we use the Navier- stokes equation;
Here we assume, Let v(x, t) be a 3-dimensional vector, the velocity of the fluid, and let p(x, t) be the pressure of the fluid which we take as random prime numbers. The Navier–Stokes equations are supported by:

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\nabla p + \nu \Delta \mathbf{v} + \mathbf{f}(\boldsymbol{x}, t)$$

where v>0 is the kinematic viscosity, f(x, t) the external force,▼ is the gradient operator and Δ is the Laplacian operator, which is also denoted by ▼.▼ . Note that this is a vector equation, i.e. it has three scalar equations. Writing down the coordinates of the velocity and the external force as:

$$\mathbf{v}(\boldsymbol{x}, t) = \big(v_1(\boldsymbol{x}, t),\ v_2(\boldsymbol{x}, t),\ v_3(\boldsymbol{x}, t)\big)\,,$$

$$\mathbf{f}(\boldsymbol{x}, t) = \big(f_1(\boldsymbol{x}, t),\ f_2(\boldsymbol{x}, t),\ f_3(\boldsymbol{x}, t)\big)$$

then for each i=1,2,3 there is the corresponding scalar Navier–Stokes equation is:

$$\frac{\partial v_i}{\partial t} + \sum_{j=1}^{3} v_j \frac{\partial v_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \nu \sum_{j=1}^{3} \frac{\partial^2 v_i}{\partial x_j^2} + f_i(\boldsymbol{x}, t).$$

This Equation is then used to initiate the hash code generation, after that we will make an isomer for the 1st 512 DNA bit sequence so as to make is satisfy the constrains of good pseudo code, then we permute the code with the prime or the key according to the polynomial equation based condition like a polymerase, and then the hash code is generated after performing 54 rounds of the above steps as we use 54 prime numbers. Actually the polymerase round alter the equality formed during isomer round in the totality of 0's and 1's in the binary sequence.

```
y=1024;
for(i=0,j=(y/3);i<y;i++,j=(j+((i^7)%1024))%1024)
{
     if(node[i]!=knode[j])
     {
          temp=node[(i*j)%y];
          node[(i*j)%y]=node[i];
          node[i]=temp;
     }
     if(node[(i+j)%y]!=prime[i%128][j%8])
     {
          temp=prime[i%128][j%8];
          node[(i+j)%y]=node[i];
          node[i]=temp;
     }
     if(i==y-1)
          break;
}//polymerase made
```

Fig 4. Polymerase formation.

After this step the 1024 bit hash code will be shown which when go through can see the equality in the number of 0's and 1's and the satisfaction of all the constrains of a good hash code generator which is low in susceptibility towards regeneration of data.

III. IMPLIMENTATION AND RESULTS

The code for meiosis based cryptographic encryption and hash function generation(MCHF) has been made in C and using JNI header it has been ported as .dll file and is used as an interface in Java swing application development to show the working of the algorithm. Now we will be going through each phases of the application starting with the front start window. From this we will be moving on to the application window where the multiple options will be shown as in Fig 5.

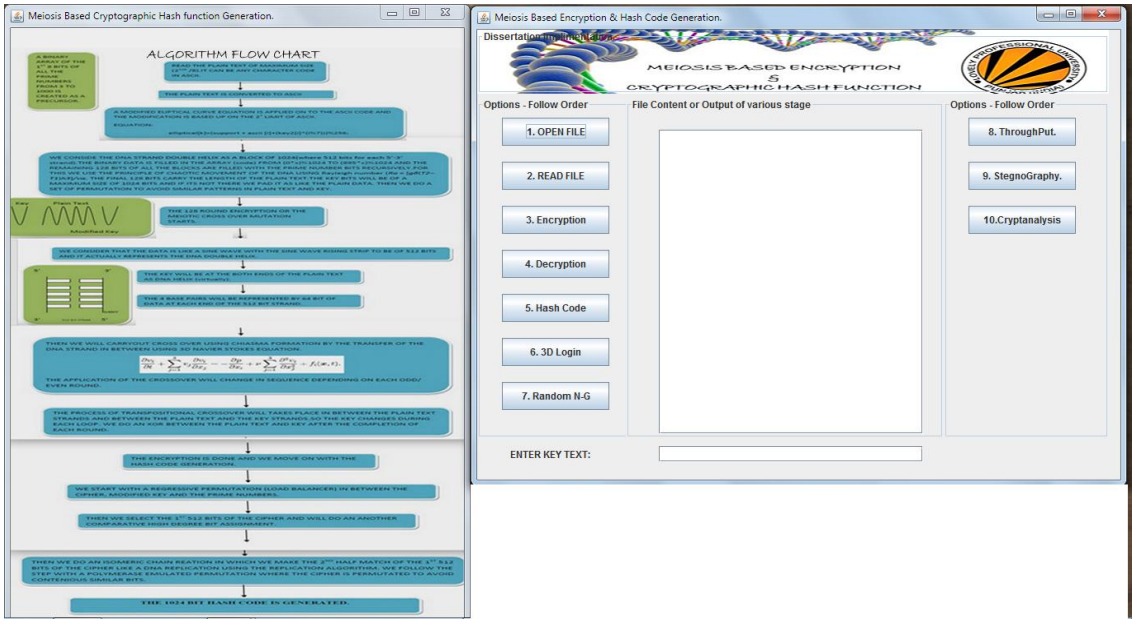


Figure 5: Option Window with Algorithm chart.

Here there are multiple options given such as

1. OPEN FILE option to select the file to be encrypted and hashed and to add digital signature.
2. READ FILE option to read the content of the file.
3. Encryption option to encrypt the data using a key of greater than size 10 which will be taken in the text box below the read window.
4. Decryption option to decrypt the data using the same key.
5. Hash Code option to generate the hash code of the file and show the digital signature as well as a comparative analysis between the time complexities of each different algorithm used for hashing with respect to MCHF.
6. 3D Login option to test the accessibility of cloud servers using a user interacted 3D login system to avoid 100% brute force attack and DDOS attack to the server. The MCHF is used to encrypt the 3D login output

when user touches or clicks each objects in the Java 3D interface. There after the output from encryption is hashed and saved in the database so that the SQL injection attacks can be avoided permanently.

a. So the main security issues in the cloud
   i. Login security
   ii. Database security

is avoided when using this application.

   b. User friendly logic method.
   c. Secure 1024 bit key based hashing using DNA emulation.

7. Random Number Generation option to generate random numbers from the generated hash as the initial entry and to find optimal output using special tools.
8. Throughput analysis option is used to take different file of various formats like .c, .docx, .pdf, .doc, .pptx etc of various file size from 10kb to 10Mb to test the throughput (file size/delay), Power factor (throughput/delay) and the improvement factor (power of other algorithms/power of MCHF) of our algorithm MCHF and other algorithms such as MD4, MD5, SHA, SHA1, SHA128, SHA256, SHA384 and SHA512.
9. Stegnography option is to use the MCHF algorithm to encrypt the data taken from a table entry and to store the value using a key in a png file of small size.
10. Cryptanalysis option will be used to do a cryptanalysis on the encrypted output as well as the has code to test and to verify the security reliability of the MCHF algorithm.

3.1 IMPLEMENTATION PHASES

Here we read a file of any format as shown in Fig 6, I selected a file of c format to be signed and encrypted and to generate the hash code.

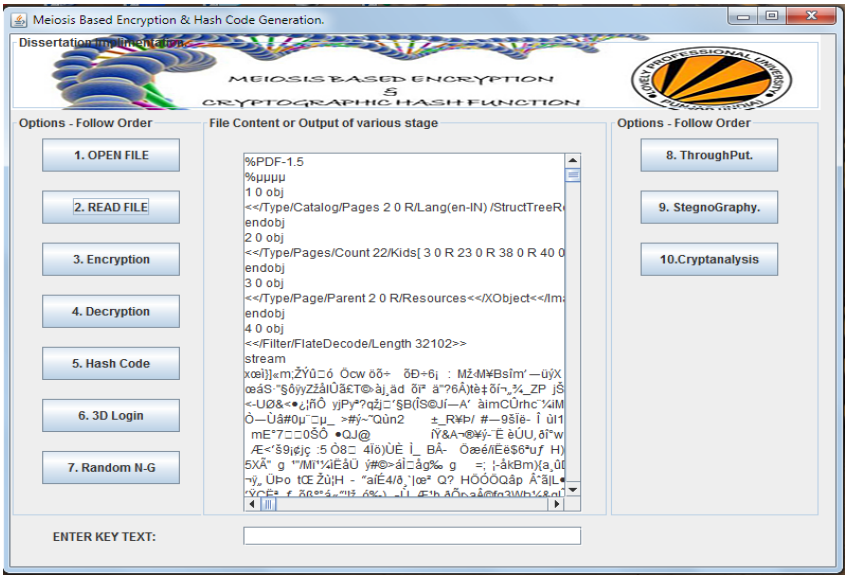


Figure 6: File content is read.

The read file content which is displayed in the test field will act as the entry to the encryption process as shown in Fig 6. Then a key of size greater than 10 is provided for encryption. Key size is set as 10 to make the encryption strong as it will generate a key bit of (10+size)*8 bit which is greater than 80 bits user key and the remaining will only be the padding. The encryption and decryption is using the MCHF encryption phase as shown in Fig 7 and further the same key can be used to decrypt the data, if the key is different, then the decryption output won't be displayed in the text field.

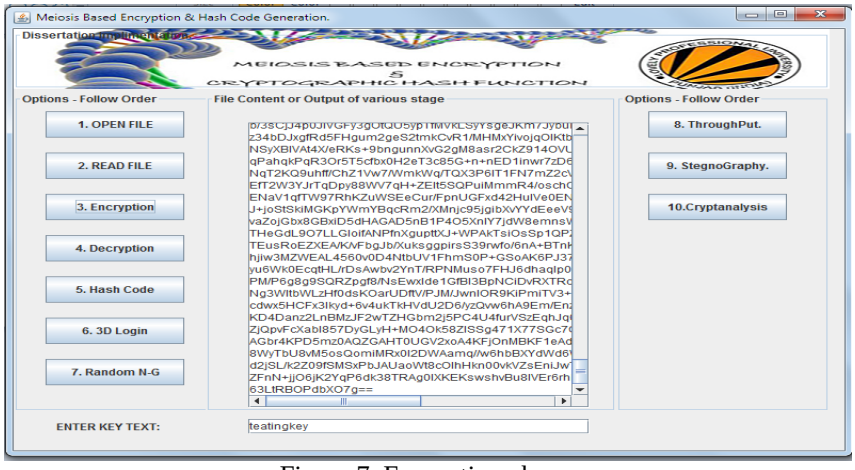


Figure 7: Encryption phase.

After the encryption the data which is generated as a multiple of 1024 will be fed into the hash code generator and at the same time the digital signature of the file will be created and it will be appended in the data which is shown at that time and in another phase the genuinely of the digital signature is checked by recreating the digest, here we use a java generated public key file to assist in public key exchange during digital signature. At the same time multiple hash code on the same data is generated using

1. MCHF
2. MD5
3. MD4
4. SHA
5. SHA1
6. SHA256
7. SHA384
8. SHA512

And the start time and end time of response from each algorithm in nano seconds is retrieved and then the difference is shown in micro seconds.

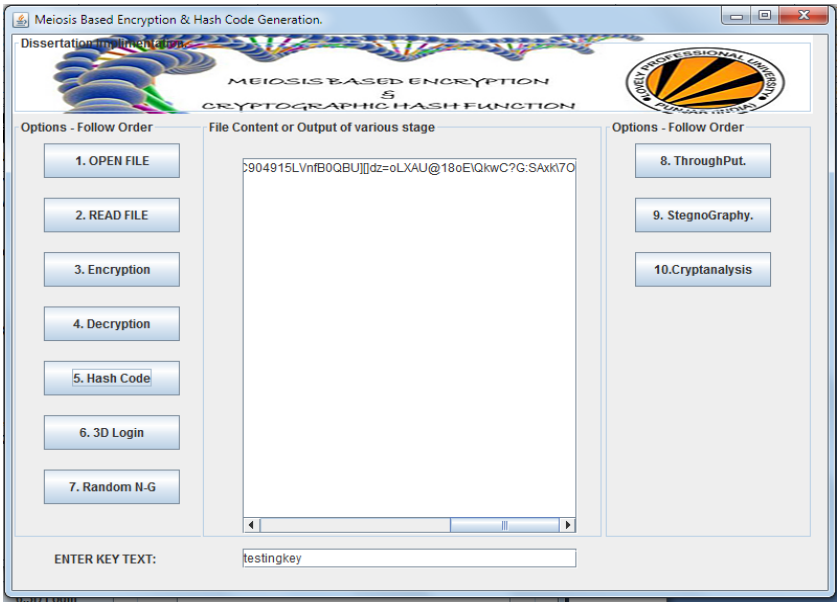
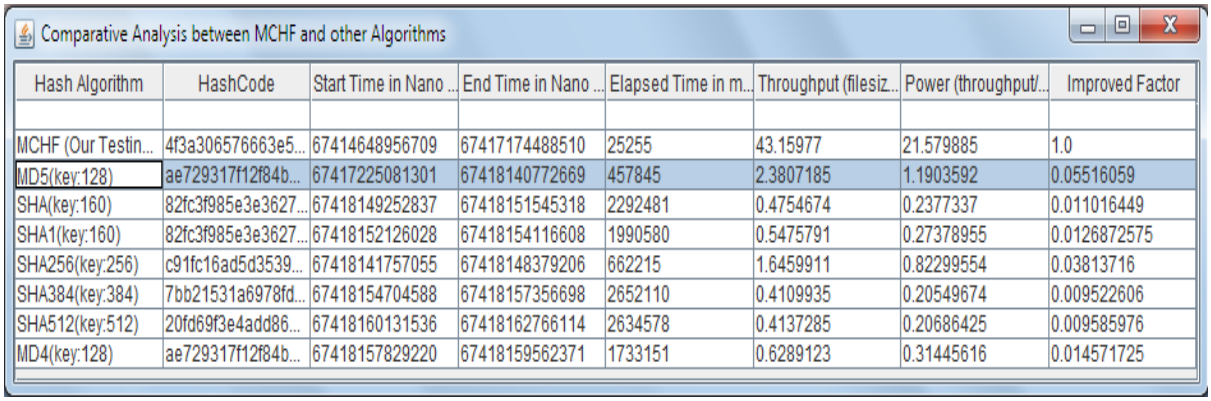


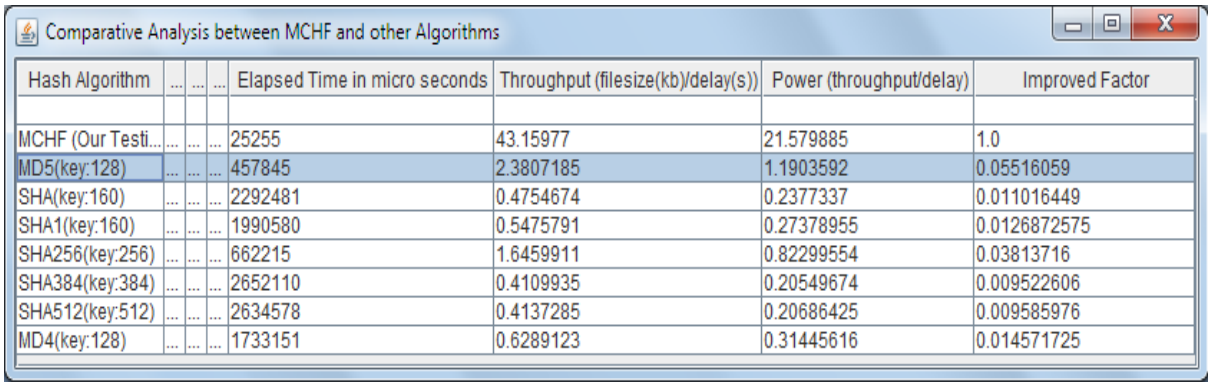
Figure 8: Hash code of the data (generated after ciphering)

Figure 9: Digital Signature of the file content.



Hash Algorithm	HashCode	Start Time in Nano ...	End Time in Nano ...	Elapsed Time in m...	Throughput (filesiz...	Power (throughput/...	Improved Factor
MCHF (Our Testin...	4f3a306576663e5...	67414648956709	67417174488510	25255	43.15977	21.579885	1.0
MD5(key:128)	ae729317f12f84b...	67417225081301	67418140772669	457845	2.3807185	1.1903592	0.05516059
SHA(key:160)	82fc3f985e3e3627...	67418149252837	67418151545318	2292481	0.4754674	0.2377337	0.011016449
SHA1(key:160)	82fc3f985e3e3627...	67418152126028	67418154116608	1990580	0.5475791	0.27378955	0.0126872575
SHA256(key:256)	c91fc16ad5d3539...	67418141757055	67418148379206	662215	1.6459911	0.82299554	0.03813716
SHA384(key:384)	7bb21531a6978fd...	67418154704588	67418157356698	2652110	0.4109935	0.20549674	0.009522606
SHA512(key:512)	20fd69f3e4add86...	67418160131536	67418162766114	2634578	0.4137285	0.20686425	0.009585976
MD4(key:128)	ae729317f12f84b...	67418157829220	67418159562371	1733151	0.6289123	0.31445616	0.014571725

Figure 10: Comparative analysis between different hash code algorithms.



Hash Algorithm	...	...	...	Elapsed Time in micro seconds	Throughput (filesize(kb)/delay(s))	Power (throughput/delay)	Improved Factor
MCHF (Our Testi...	...	...	...	25255	43.15977	21.579885	1.0
MD5(key:128)	...	...	...	457845	2.3807185	1.1903592	0.05516059
SHA(key:160)	...	...	...	2292481	0.4754674	0.2377337	0.011016449
SHA1(key:160)	...	...	...	1990580	0.5475791	0.27378955	0.0126872575
SHA256(key:256)	...	...	...	662215	1.6459911	0.82299554	0.03813716
SHA384(key:384)	...	...	...	2652110	0.4109935	0.20549674	0.009522606
SHA512(key:512)	...	...	...	2634578	0.4137285	0.20686425	0.009585976
MD4(key:128)	...	...	...	1733151	0.6289123	0.31445616	0.014571725

Figure 11: Time complexity analysis and other QoS factors.

From the output of hash code, we generated the time complexity test for different algorithms which has different key size as shown in Fig 10 and Fig 11. As we can see our algorithm is 3rd in the list, which shows that the trial 128 loop even gives a better time complexity, testing of loop can be further improved by reducing the no. of loops in the hash code phase.

Now let us move on to the application of the hash code in 3D login and random number generation and optimisation.

In this phase the hash code generates a random plot ‘a’, ’c’ and ‘m’ value and is forwarded to the RNG to generate the graph and to generate the plot values as shown in Fig 12. Further we will go for the analysis phase using Data Tool developed by MIT and we will b using the function to transfer our data to the Data Tool to plot the graph so as to find the min and max values to find the optimal plots as shown in Fig 13.

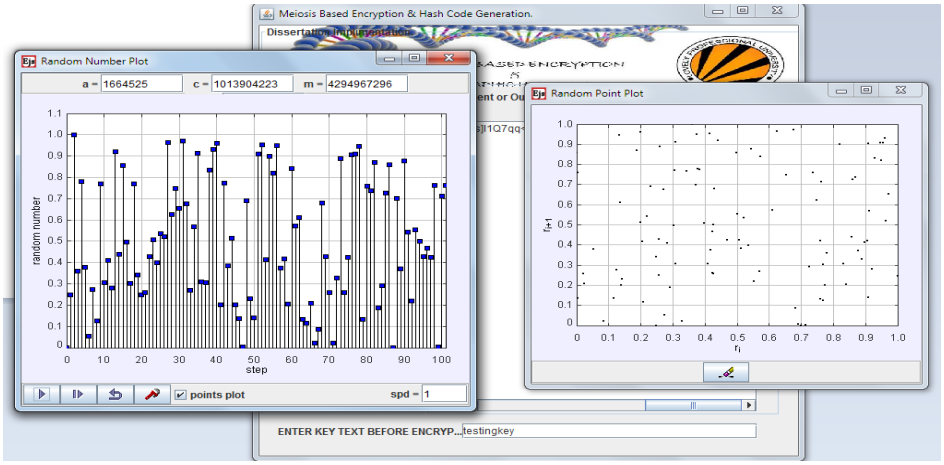


Figure 12: Random number generation.

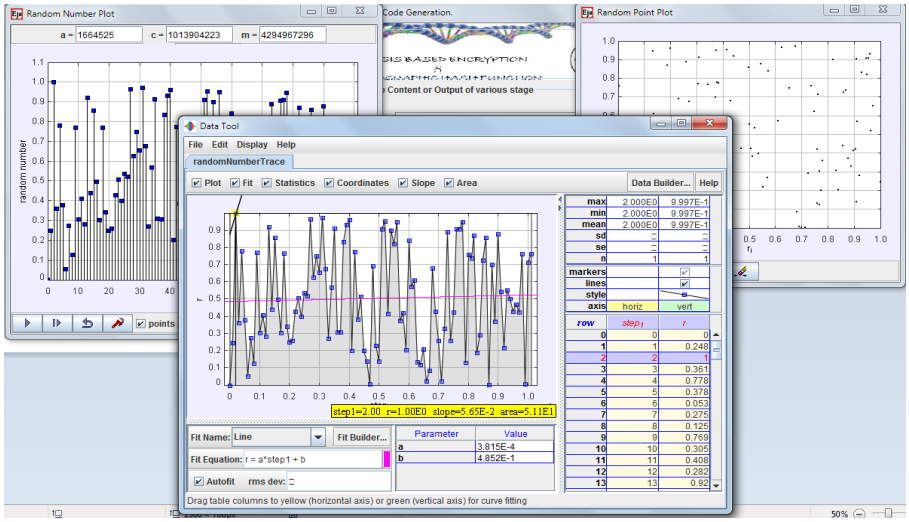


Figure 13: Data Tool plots the generated random point.

Here the Data Tool will plot the random points and the global maxima are shown as the 2nd value in the table which is highlighted in Fig 14.

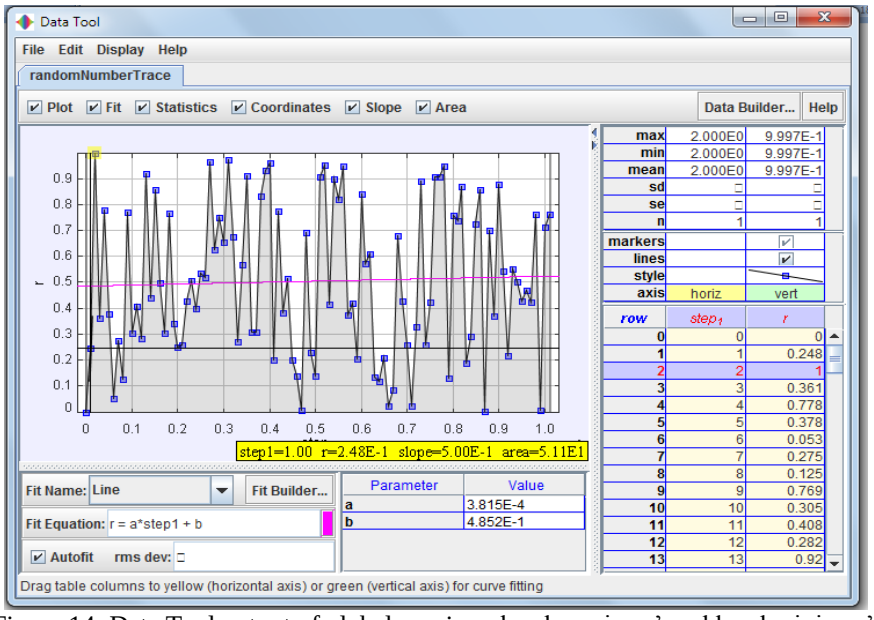


Figure 14: Data Tool output of global maxima, local maximas' and local minimas'.

So the algorithm can be used for optimization purpose and it will be using stimulated annealing to find the global maxima out of all local maximas'.

Now we move on to the use of 3D log in system for cloud. Here the login is based up on pure user interaction in the 3D environment. There will be different type of objects in the environment as shown in Fig 15, the sequence of the objects that the user clicks will be his password and the data will be taken as a com object as shown in Fig 16 and then it will be encrypted and a hash code will be generated as stored in database.

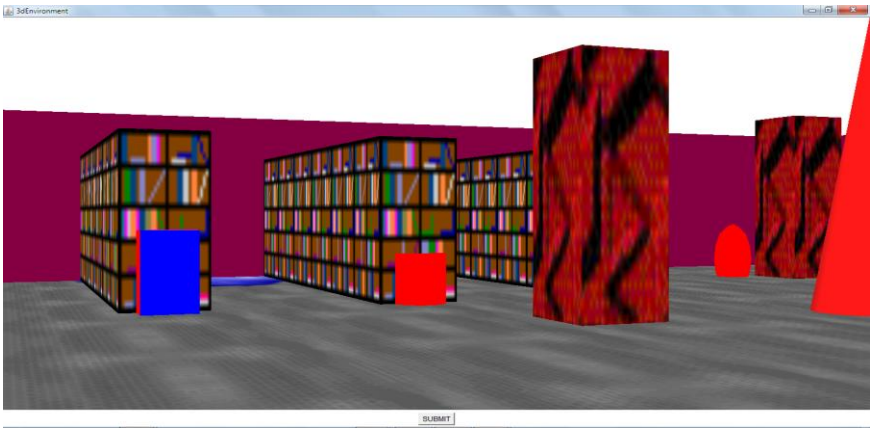


Figure 15: 3D Login interface

Figure 16: Data generated in Command Prompt when the user access each objects.

So the system can avoid 100 % brute force attack on login and DDOS in the login and SQL injection at the backend. This is the working of Meiosis based cryptographic algorithm and its applications created in java swing and java 3D environment and the coding is done in JVM for hashing using JNI interface so that the C program can directly access the JVM as a .dll file.

Now let us consider the stegnographic application of the encryption. For this purpose I have created an interphase in java swing to show a set of tables in which the user can provide his/ her credentials which will be stored using a key as a PNG file as shown in the Figure 17:

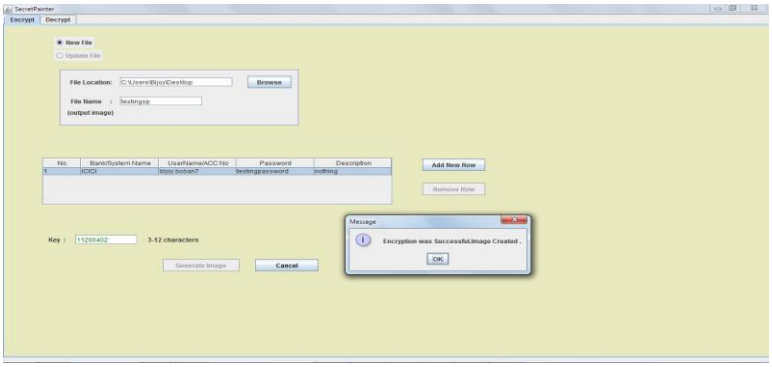


Figure 17: Stegnographic application of MCHF algorithm (Frame work of stegnographic application rendered from secret painter open source code)

When the generate image button is pressed the file will be created in its given location, the application closes by itself to show that no data is stored in the application interface as soon as the image is generated. Now let us move on to the decryption phase.

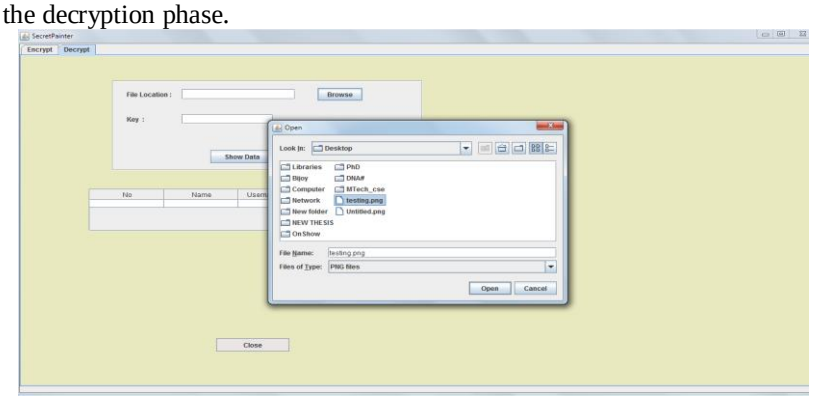


Figure 18: Decryption of the data from the image.

After selecting the image for decryption the key has to be given and if the key is wrong the data won't be shown.

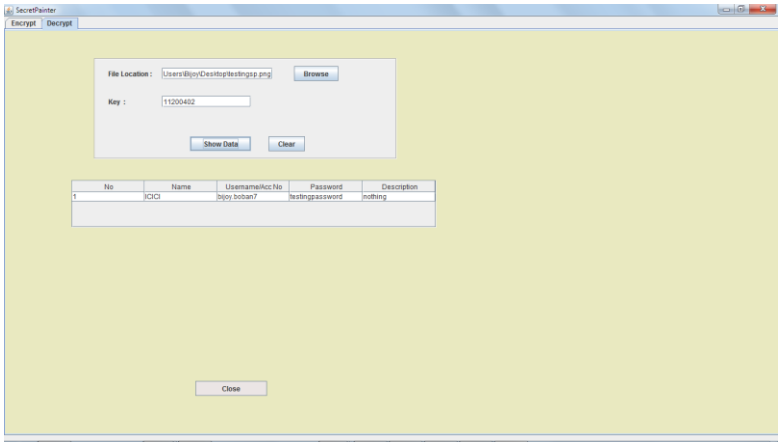


Figure 19: Decrypted data

IV. CRYPTANALYSIS AND BRUTEFORCE TESTING

Now let us move on to cryptanalysis, for the purpose of cryptanalysis we will be using CrypTool where we pass the hash code for various cryptanalysis techniques to find the possibilities to crack the code.

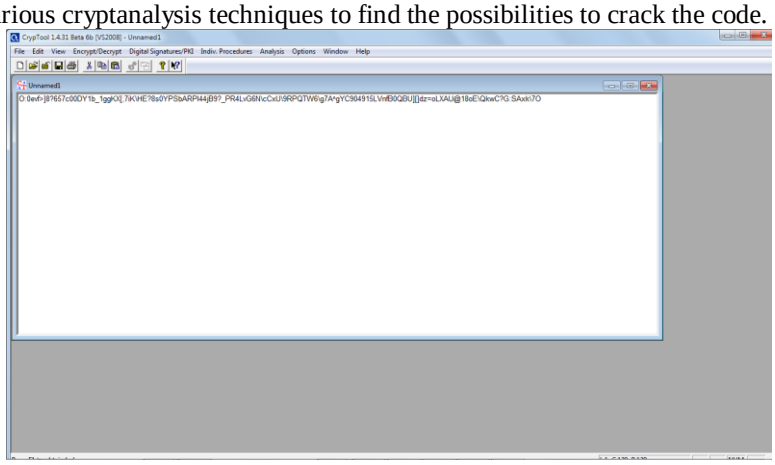


Figure 20: Generated hash code in cryptool console.

Now we select the hash analysis window which is used to search whether another file containing the fake message can generate a hash code similar to the data in the original message for which we generated the hash code.

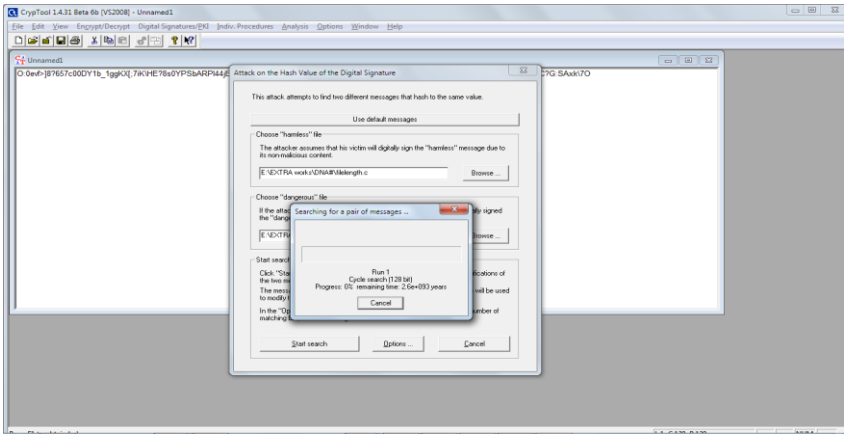


Figure 21: Console showing the chance of faking the hash code with a fake message.

It shows in the console that the process takes $2.6*e^{93}$ years to crack the hash code for a 2.4GHz processor. Which shows that our hash code is safe from attack in the sense that the fastest super computer of speed 33.5 petaflops (36833639530496Hz) takes $7.05876 * e^{79}$ sec which is $2.238 * e^{72}$ years.
Now we will be trying brute force attack on the output from encryption phase using cryptool. For this we will be passing the encrypted value and will be testing brute force.

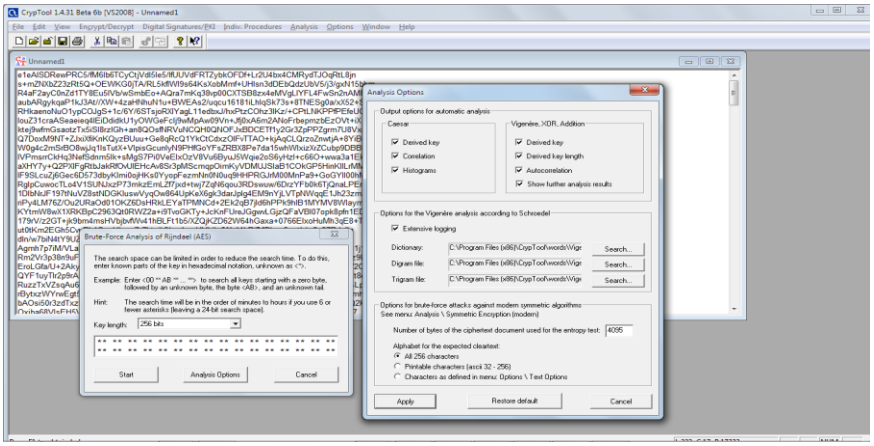


Figure 22: Brute force set up to crack the key up to 256 bits.

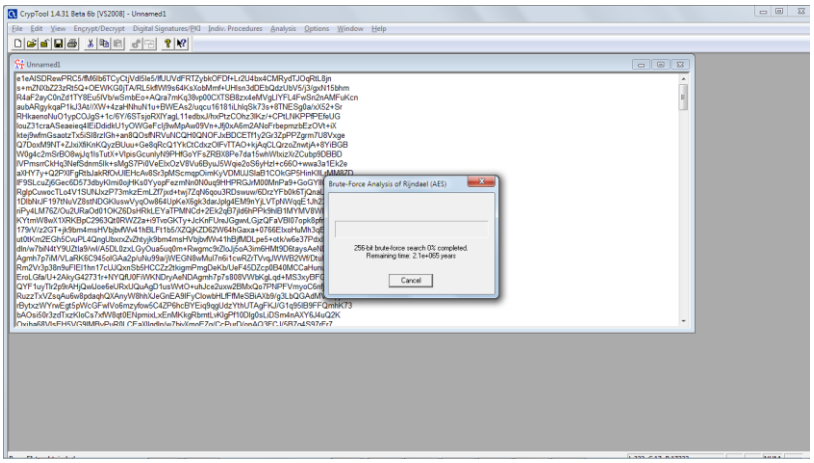


Figure 23: Brute force for attack execution

In the output we can see that the process takes $2.1*e^{65}$ years to complete the task to decrypt the cipher up to a key length of 256 bits so for 1024 bit key it will be approximately $2.1*e^{511}$ years. And the test decryption output is

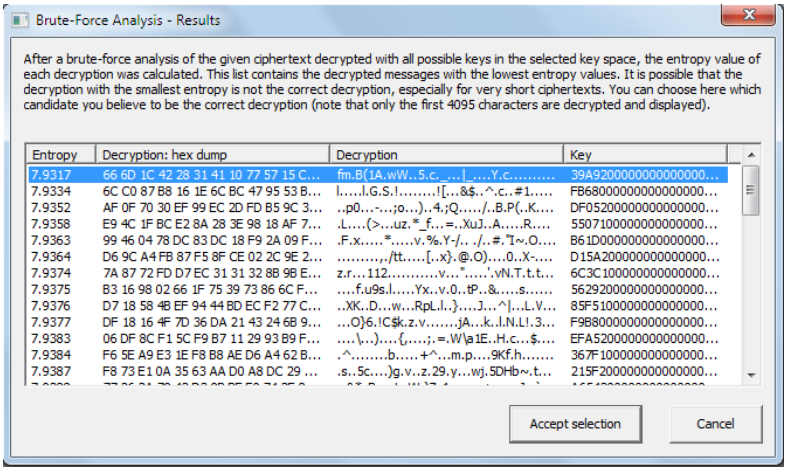


Figure 24: Brute force output after temporary retrieval.

Now we will be considering the 3D visualisation of the hash output to see the equal distribution of variables in the output to show that the function is strong because the formation of pattern shows weak points when there are more digrams trigrams and n-grams.

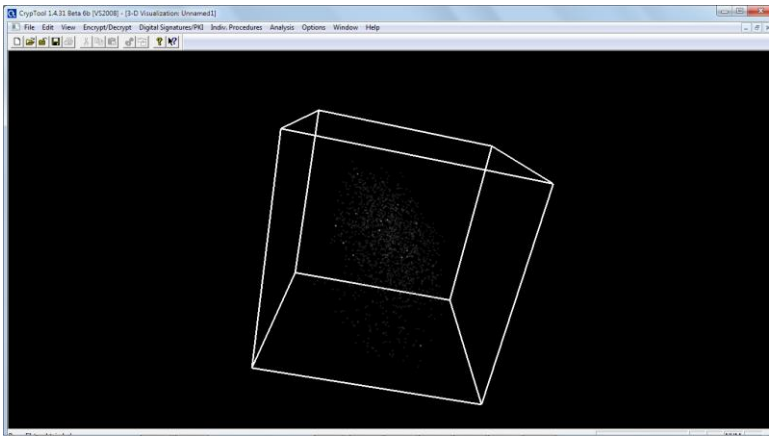


Figure 25: 3D output which shows the distribution of alphabets in the cipher output (no patters).

The floating point output shows equality in all different characters per 64 byte blocks throughout the distribution to show that there is no specific flaws due to patterns which help in cryptanalysis.

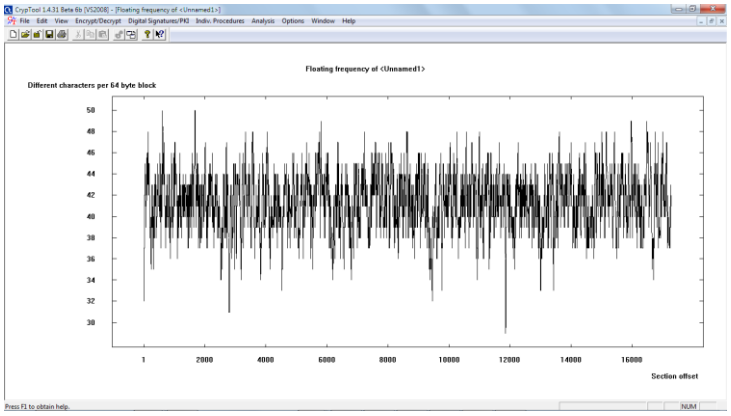


Figure 26: Floating point graphic representation of byte distribution

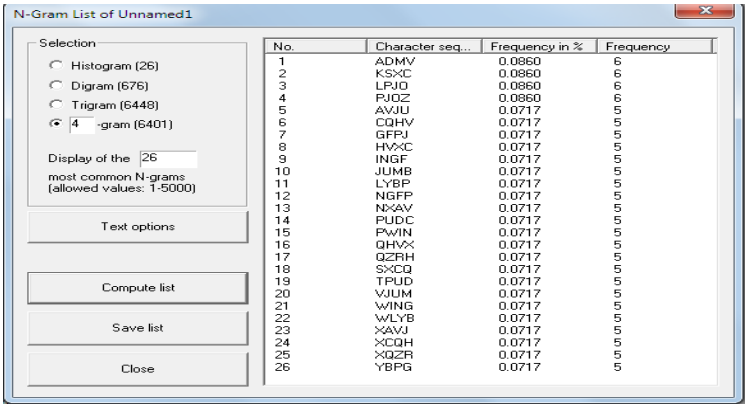


Figure 27: N-grams in the encrypted data as well as digrams and trigrams

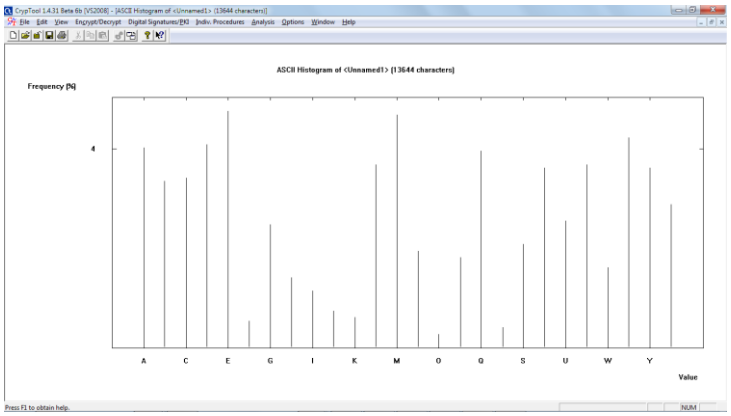


Figure 28: Histogram pf encrypted data.

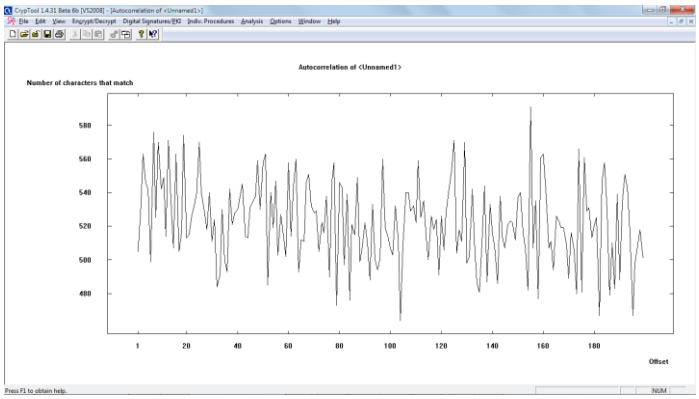


Figure 29: Auto correction to check for cryptanalysis loop holes

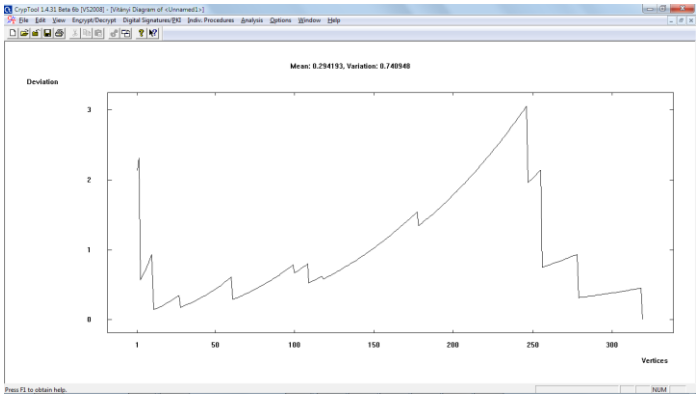


Figure 30: mean deviation and standard deviation as well as variance of the cipher

V. CONCLUSION

Whenever we go through security in cloud, a new loop hole or a backdoor opens for intrusion, mechanisms such as encryption and security model always played the way to improving the security in cloud and still the weakness that there is always a path for a black hat to go through the front door like a user still remains an issue, as demonstrated above

- A pure 100% human interactive login system at the front end
- A hash code using 1024 bit key which is used for data base
- A hash code based splay tree algorithm to randomly change the hash code value every 30 minutes which refers to the memory location of the file

All these 3 techniques at 3 most vulnerable points on cloud security architecture can avoid more than 90% attack on cloud users and its authenticity [9]. The implementation as well as the performance of the proposed and shows security mechanism using hash code will be easy on cloud as the cloud provides homogeneity in its deployment as it can be implemented at different deployment models even if its private cloud or public cloud, the performance of the system will be progressively immune to attacks. The future scope of this MCHF algorithm is the deployment of this security implementations as a cloud architecture or a security model for cloud as security is of prime importance in cloud computing.

VI. ACKNOWLEDGEMENTS

The topic was inter disciplinary, which included not just cryptography but also the basics of fertilization and mutation, fluid mechanics, continuum mechanics etc, I thank Asst. Prof Ankur Sodhi for the help he provided in sorting out the problem and for the other technical and advisory support. I extend my thanks to the department of Computer science in Lovely Professional University.

REFERENCES

Journal Papers:

[1] Bijoy Boban, Meiosis Based Cryptographic Hash Function Generation with the Influence of 3D Navier Stokes Equation on Chromosomal Chaotic Movement in Nucleoplasm, international journal for advanced research in computer science, Vol-4, No. 8, May- June 2013.

[2] R. Siddharth and Rajeev Kumar, Formulation of an Encryption Algorithm on the Basis of Molecular Genetics and Image Patterns by M. Prashant, Department of Computer Science & Information Systems, Birla Institute of Technology & Science, Pilani – 333 031, India.

[3] Aashish Priye, Radha Muddu, Yassin A. Hassan, and Victor M. Ugaz, Selecting 3D Chaotic flow states for Accelerated DNA replication in MICRO-SCALE convective PCR, Department of Chemical Engineering, Texas A&M University, College Station, TX 77843, USA.

[4] Thomas Y. Hou[1], Zhen Lei[2], On the Partial Regularity of a 3D Model of the Navier-Stokes Equations, 1. Applied and Compute. Math, Caltech, Pasadena, CA 91125, USA. 2. School of Mathematical Sciences, Fudan University, Shanghai 200433, P.R. China.

[5] Ashish Gehani, Thomas LaBean, and John Reif, DNA-Based Cryptography, Department of Computer Science, Duke University.

[6] H. Z. Hsu and R. C. T. Lee, DNA Based Encryption Methods , Department of Computer Science National Chi Nan University, Puli, Nantou Hsieh, Taiwan 545.

[7] k.s tang, k.f man, s. kwong, q. he, Genetic Algorithm and their applications, IEEE signal processing magazine, November 1996.

Books:

[8] Herskowitz,H. I., Basic *Principles of Molecular Genetics*, Thomas Nelson, England, 1968.

[9] Bijoy Boban, Comparative *analysis between Grid and Cloud computing*, Grin Publications, California, 2012.