CRACKING THE ENCRYPTION KEY FROM AUTO GENERATED CIPHER

J Senthil[1], S Arumugam[2], S Margret Anouncia[3] Abhinav Kapoor[4]

[1] SCSE,VIT University, Vellore-14, India

[2] Nandha College of Technology, Erode-52, India

[3] SCSE, VIT University, Vellore-14, India

[4] SCSE, VIT University, Vellore-14, India

ABSTRACT:

Generation of random strings from a random generator has been going on since decades. But, predicting the nature and pattern of strings beforehand has been the issue under the realm of Artificial Intelligence. We have come up with an algorithm to predict the characters of the key used in encrypting the plaintext by analysing the previous set of random cipher texts. The result will be truly exact if the algorithm is implemented on a machine which has no other previous random sequence generator implemented on its background.

Keywords: Random Strings, Pseudo random number, Prediction algorithm.

INTRODUCTION:

The algorithm 'Pseudo Random Number Generator' generates random numbers from a seed. A random number generator (often abbreviated as RNG) is a computational or physical device designed to generate a sequence of numbers or symbols that lack any pattern, i.e. appear random. A pseudorandom number generator (PRNG), also known as a deterministic random bit generator (DRBG), is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. The sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the PRNG's state**.** Although sequences that are closer to truly random can be generated using hardware random number generators, *pseudorandom* numbers are important in practice for simulations (e.g., of physical systems with the Monte Carlo method), and are central in the practice of cryptography and procedural generation. Common classes of these algorithms are linear congruent generators, lagged Fibonacci generators, linear feedback shift registers, feedback with carry shift registers, and generalised feedback shift registers. Recent instances of pseudorandom algorithms include Blum Blum Shub, Fortuna, and the Mersenne Twister.

Careful mathematical analysis is required to have any confidence a PRNG generates numbers that are sufficiently "random" to suit the intended use. Robert R. Coveyou of Oak Ridge National Laboratory once titled an article, "The generation of random numbers is too important to be left to chance." As John von Neumannjoked, "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."

In many applications randomness have led to the development of several different methods for generating random data. Because of the mechanical nature of these techniques, generating large amounts of sufficiently random numbers (important in statistics) required a lot of work and/or time.

The output random numbers acts as a seed and hence, a long string of random numbers is generated. Our suggested algorithm is based on the concepts of set theory, logical reasoning and conditional probability .We have used conditional probability in context of making the best possible choice by neglecting the least probable choices at a particular instance of time.

NEW APPROACH:

Algorithm:

Consider a random string generator which generates strings of length n. Obtain n number of strings of length n through n trials.

1. Consider the characters at positions 1,n+1,2n+1,3n+1,4n+1,..n*n +1.

Count the number of occurrences of each character while analyzing the characters at above mentioned positions.

2.Make a state consisting of a set of week states and a set of strong states.

Weak states are the characters with the least count and the strong states are the characters with the highest count of occurrence.

State= ((Set of weak states),(Set of strong states))

3. Repeat steps 1 and 2 for characters at position i to j where i=2 to n and j=2i+k where k=j and j varies from 2,3,…,n.

4. A set consisting of n states will be obtained.

5. Analyze all the states in the set. If in a particular state, there are highest number of weaker states, then the corresponding set of stronger states are the next most probable character /characters.

6. If there are more than one states in the set such that the number of weaker states in two separate states of set are same, then select the corresponding stronger states from that particular states and remove all other characters from the set.

7.Repeat steps 5 and 6 until any one of following conditions satisfy:

(i)A state such as (λ,{x}) is obtained where x is empty string and hence {x} will be next most probable character.

(ii)If the (i) condition is not obtained but more than one similar states are obtained, then corresponding set of strong states in that particular state will be the next probable character.

(iii)If the conditions (i) and (ii) do not satisfy, then all the characters obtained at last will be equally probable as these will belong to undistinguishable states.

For instance:
Consider 4 strings of length 4.

Set1	Set2	Set3	Set4	
3	4	3	1	->string 1
1	1	2	1	->string 2
4	2	3	2	->string 3
1	4	3	1	->string 4

Now for the set 1, the state is ((3,4),1) where (3,4) is the set of weak states and 1 is the strong state.
For set 2, the state is ((1,2),4)
For set 3, the state is (2,3)
For set 4, the state is (2,1)
Now, consider set S such that
S={((3,4),1),((1,2),4),(2,3),(2,1)}
Applying step 5,
The states 1 and 4 are most probable stronger states.
So, apply step 6.

Remove all the weaker states i.e., 2 and 3.Hence,the set S becomes:
S={(4,1),(1,4),(λ,1)}
Since condition (i) is satisfied by the state (λ,1) where λ is empty string.
Hence 1 is the next most probable output character which will be presented as a result of the algorithm running on any machine which can be verified by analyzing the previous patterns.

CONCLUSION AND FUTURE WORK:

The above algorithm works for strings of any length, but it requires n number of strings of length n for its exact result. We see the future design of a DFA and a NFA machine based on this algorithm which can bring great advancements, in fields of 'Artificial Intelligence' and 'Network Security'. This algorithm will make computer intelligent enough to predict the next result.

APPENDIX
IMPLEMENTATION:
The Code to generate random strings has been implemented in J2SE.

```
import java.util.*;
import java.awt.*;
public class str {
Random generator;
public dice(int sides)
{
generator=new Random();
faces=sides;
}
public  int game(int faces)
{
return (1+ generator.nextInt(faces));
}
private int faces;
}
```

The mainclass is:

```
import java.util.*;
import java.awt.*;
public class mainclass {
public static void main(String[] args)
{
        int sides;
Scanner a=new Scanner(System.in);
System.out.println("Enter the number of sides");
sides=a.nextInt();
dice  x=new dice  (sides);
System.out.println("Enter your number of tries");
int tries=a.nextInt();
for(int i=1;i<=tries;i++)
{
System.out.println(x.game(sides));
}
}
}
```

REFERENCES:

Aiello, W., Rajagopalan, S., and Venkatesan, R. . Design of practical and provably good random number generators. Journal of Algorithms, 29(2):358–389.

Blum, L., Blum, M., and Schub, M. A simple unpredictable pseudorandom number generator. SIAM Journal on Computing, 15(2):364–383.

Bratley, P., Fox, B. L., and Schrage, L. E. (1987). A Guide to Simulation. Springer-Verlag, New York, second edition.

Brown, M. and Solomon, H. On combining pseudorandom number generators. Annals of Statistics, 1:691–695.

Chen, H. C. and Asau, Y. On generating random variates from an empirical distribution. AIEE Transactions, 6:163–166.

Cheng, R. C. H. Random variate generation. In Banks, J., editor, Handbook of Simulation, pages 139–172. Wiley. chapter 5.

Collings, B. J. Compound random number generators. Journal of the American Statistical Association, 82(398):525–527.

Conway, J. H. and Sloane, N. J. A. Sphere Packings, Lattices and Groups. Grundlehren der Mathematischen Wissenschaften 290. Springer- Verlag, New York, 3rd edition.

Couture, R. and L'Ecuyer, P. On the lattice structure of certain linear congruential sequences related to AWC/SWB generators. Mathematics of Computation, 62(206):798–808.

Couture, R. and L'Ecuyer, P. Orbits and lattices for linear random number generators with composite moduli. Mathematics of Computation,65(213):189–201.

First Author: J.Senthil, obtained bachelors degree in Computer Engineering from Kongu Engineering College and Masters degree from Illinois Institute of Technology, Chicago, USA in Computer Engineering with Honors. Research interest is in Software Automation and in Pervasive computing. Currently working in VIT, as Assistant Professor Sr.

Second Author: Dr.S.Arumugam, CEO of Nandha Educational Institution.

Third Author: Dr.S.Margret Anouncia, Director, SCSE, VIT University, Vellore.

Forth Author: Abhinav Kapoor,SCSE,VIT University, Vellore