IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 1, Ver. II (Jan.-Feb. 2017), PP 75-83
www.iosrjournals.org

Design Package to Build and Evaluate Encryption Algorithms

Prof. Dr. Salim Ali Abbas,Ali Jumaa Hashim
(Department of Computer Science, College of Education/ Al-Mustansiriya University, Iraq)

Abstract : *Building secure stream cipher algorithm in fast and comfortable manner is consider point of interest for most of those interested in building encryption algorithms to conceal their information and prevent detection it by unauthorized persons. The main goal of this paper is to design a package to implement and evaluate most of stream cipher algorithms in fast, convenient manner. The package provides the ability to the designer to design stream cipher algorithms either visually or by C# programming language. The proposed package provides number of the components required to build stream cipher algorithm, while providing the potentiality of linking these components to enable the exchange of data among them when running the algorithm, it has also been providing the potentiality of testing the key produced by designed algorithm. Testing performed by applying fourteen statistical tests that provided by NIST and all these fourteen tests executed in parallel to reduce theirexecution time. It also provides the possibility to use the designed stream cipher algorithm to encrypt any plaintext and decrypt any cipher text.*

Keywords: *Design,Evaluating,NIST statistical tests, Package, Stream cipher algorithm*

I. Introduction

In the information age, where digital communication techniques and digital storage device have emerged, and because of these technological developments, the need to find a way to protect information from theft and prevent disclosure of information of unauthorized people has emerged [1]. Cryptography is deals with design and analysis of mathematical techniques that allowsafe communications in the existence of wicked opponents [2]. Encryption is one of the technique by which can hide information and turn it into another form that is not understand. Encryption is a service that achieve the confidentiality of data, given the importance of encryption to provide confidentiality of data, so building encryption algorithm, implemented and tested is of great importance for specialists and those that interested in the field of encryption [3].

Encryption algorithm is classified into two types according to the number of keys used for encryption, the first type is symmetric encryption, in this type used one secret key that shared between the communication parties (the sender and the recipient). The second type is asymmetric encryption, in this type used two keys, the first public and open for everyone, and the second is kept secret, and in this type one key cannot be inferred from the other keys [4]. Asymmetric encryption is slow because it's based on mathematical theories, which is impractical in applications that require live and direct processing. In symmetric encryption, there are two types of algorithms, Block algorithm and Stream algorithm, in Block encryption algorithm the plaintext dividing into blocks of bits. Famous Block algorithms are AES and DES [5]. Stream cipher is the encryption system developed by Claude Shannon after studying encryption system one-time pad, and developed stream cipher as a solution to the problems of encryption one-time pad system. In encryption system, one-time pad the key must be completely random and its length is equal to the length of the message to be encrypted and uses one time, because of the impracticability of one-time pad system so stream encryption system consider acceptable solution to these problems. In the stream cipher algorithm, pseudo-random key is generated using one of pseudo-random generator [6]. Encryption is done by dividing the clear text into bits and then every bit is encrypted individually. Stream algorithm relies entirely on security of pseudorandom key stream. To design stream algorithm, the designer can use several components and methods, the LFSR is most popular components used in the constructing of stream algorithm [7].

II. The Proposed Package Overview

The proposed package consists of three major parts. The first part is building any stream cipher algorithms (whose components are available in the proposed package) visually or by using programming language code. The second part is evaluating the keystream produced the designed stream cipher algorithms using fourteen statistical tests (with the exception of a discrete fourier transform test)that provided by NIST to measure the randomness proprieties of the key stream. The third part is encrypting and/or decrypting any text message. This paper describes the steps which are taken to design the proposed package.

III. Building Stream Cipher Algorithm

To build stream cipher algorithms, there are two approach. The first one is building stream cipher visually through provide the required components and provide the potentiality to link these components to generate sequence of bits. The second is building stream cipher algorithm through C# programming language. The proposed package provides the two approach. The following sections will discuss how provide these two approaches for constructing stream cipher algorithm in the proposed package.

3.1 Building Stream Cipher Visually:

To build stream cipher algorithm there are number of components that package provides. The common component is a LSFR, and also there are number of other components such as logical gates (XOR, AND, OR, NOR, NOT, NAND, JK), Table, Inner product, police and other components, to constructing stream cipher algorithm the designer has to drag the required components from the stream component list and drop it on area that allocated to building stream cipher algorithm. Fig1 shown the structure of proposed package.

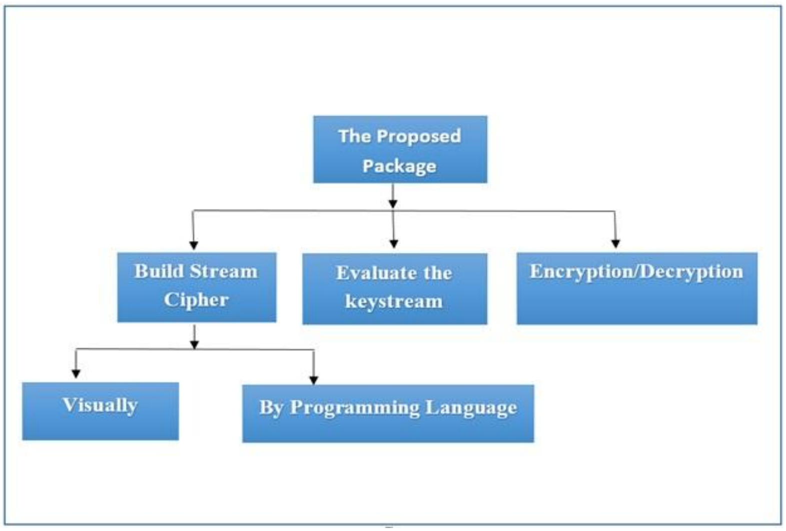


Fig.1: Structure of the proposed package.

3.1.1 Design of the Main Interface

The main interface of the proposed package consists of different parts as shown in fig2.The interface consists of four parts, the buttons bar, design area, stream cipher components bar, and information box.

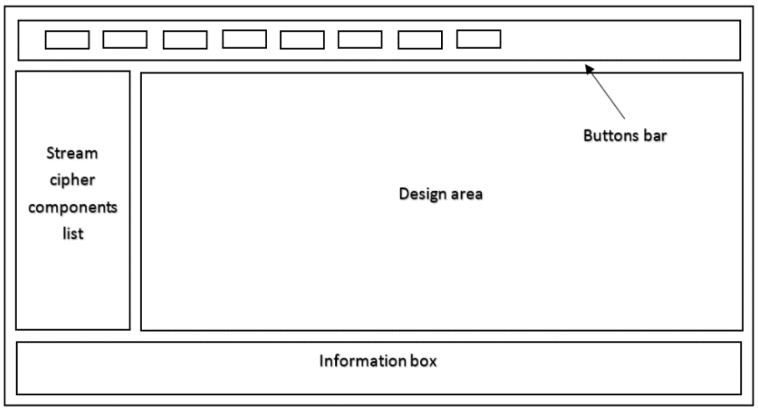


Fig.2: Main interface.

Following is descriptions of each part:

1- Buttons bar has following buttons:
  a) Button for generate one bit from the designed keystream generator in each click.
  b) Button for generate sequence of bits in each click, length of desired sequence enters in the textbox.
  c) Button for open the window of NIST statistical tests that used in evaluate any sequence of binary bit that produce from keystream generator that design in the package or outside of package.
  d) Button for open the window of design stream cipher algorithms using C# programming language.

e) Two buttons to save and load designed stream cipher algorithm.
f) Button for delete any selected stream cipher components.
g) Text box used to enter the following:
   1) The length of LFSR and table components.
   2) To enter the content of LFSR and table. Text box also used to enter a number that represent the length of sequence to be generated from the designed keystream generator in the design area.

2- Design area
It is area allocated to design keystream generator visually through drag any desired components from components list and drop them on the design area,
3- Stream cipher components
It is a list of stream cipher components that used in constructed keystream generator.
4- Information box
This text box used to show information about stream cipher components that exist in design area.

3.1.2 Design of Stream Cipher Components

To construct stream cipher algorithm there are number of components, the common component is the shift register. Every component further decomposes into three subcomponents, the first is the input, second is processing, the third is output. Each subcomponent will represent in the package by different class. Fig.3 shown the visual representation of the stream cipher component.

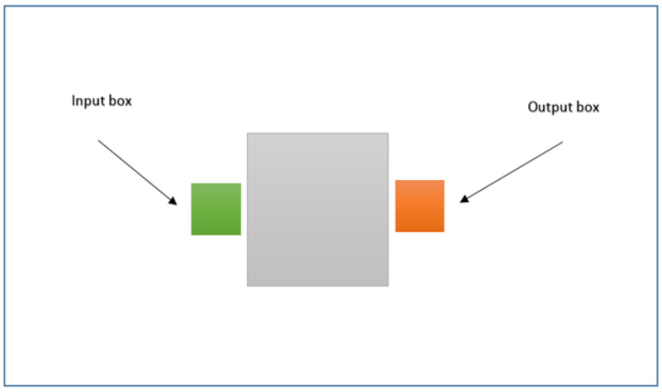


Fig.3:Generaldesign of stream cipher component.

Each component either accept one input or more than one input and has one output. Input box and output box will represent in package by separate class, each class has the following proprieties and function:
1- Proprieties called state, store either 1 or 0.
2- A mechanism to inform other component object that associated with when its state value changed.

3.1.3 Description of the Mechanism to Connect SCA Components

After putting all the stream cipher components on the design area in the main interface, the designer start to make connections among the components to allow of passing data among connected components. Fig.4 shown the scenario of linking two components(Comp1 and Comp2), each has one input and one output.

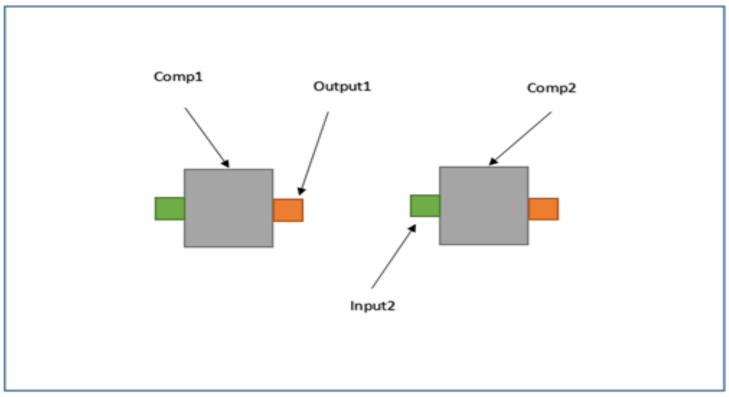


Fig.4: Example of connecting two components.

To make connection between Output1 object that belong to comp1 and Input2 object that belong to Comp2, the designer first clicks on the Output1 of Comp1, then the designer clicks on the Input2 object of Comp2, after that the connection line will appear connecting the two components and it is imply the success of connection.

3.1.4 Design of Stream Cipher Algorithm Components

There are different stream cipher components that can represent visually in the package. In following sections discussed the design of each component, how processing data that input to it, and how inform other components that linked to of changing in its state.

1- Shift Register Component

Fig.5 shows the design of the shift register.LFSR consists of cells that store each one only 1 or 0, each cell has two output box that used in link the cell to other components, except first cell that contain two output box and one input box, the input box used to feed new bit to the register. The register also has clock box as pointed in fig.5, it is used to control the shifting operation in the LFSR such that if the state of clock box is zero then there is no shifting, otherwise, if the state is equal to one, the shift operation is allowed.

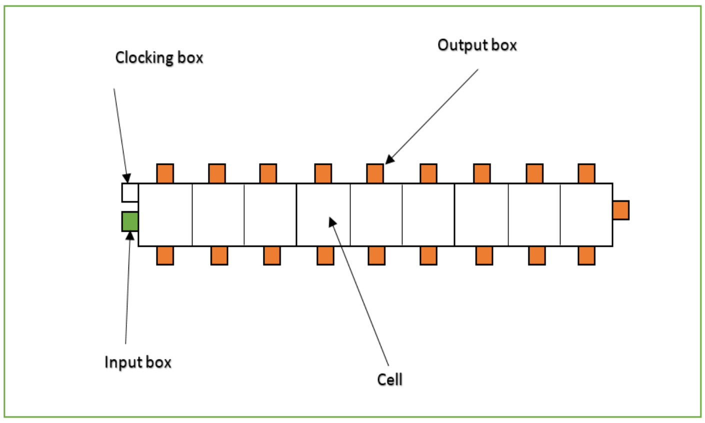


Fig.5: Design of LFSR.

2- Police Component

Police component consists of three input and one output components. Fig.6 shown the visual representation of the police in the package:

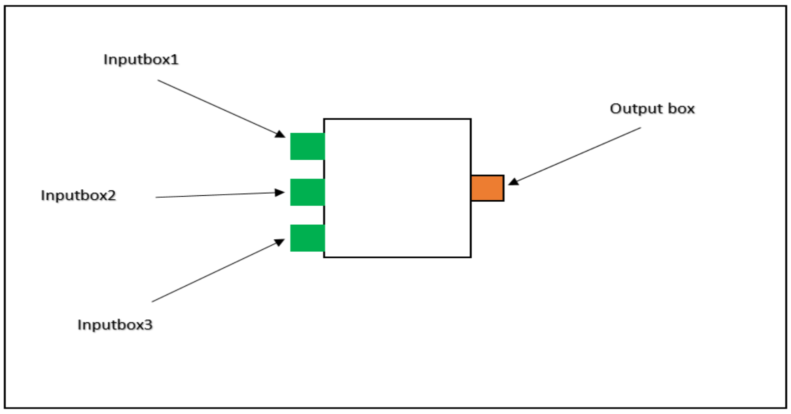


Fig.6:Design of police component.

As in fig.6, if the state of Inputbox2 is equal to 0, the police output will be the state of the Inputbox1, otherwise if the Inputbox2 is equal to 1, the police output will be the state of the Inputbox3.

3- The logical gates

All logical gates consist of two input and one output except "NOT gate" which contain only one input. The proposed package provides gates: XOR, OR, AND, NOR, NAND, NOT. Fig.7 shows visual representation of the logical gate.

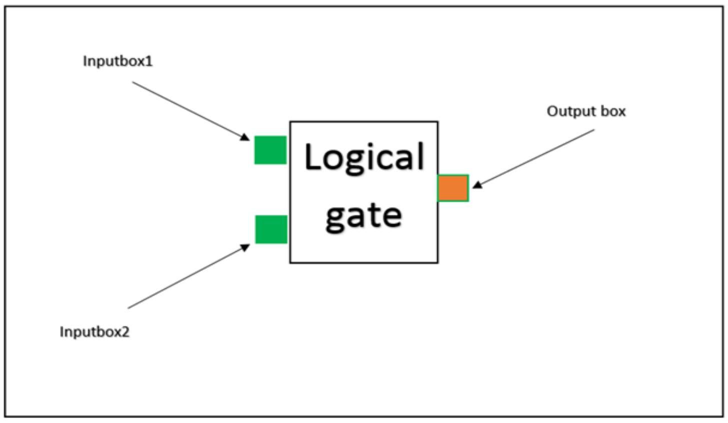


Fig.7: Design of logical gate component.

The logical gate as shown in fig7. For example, if the logical gate is XOR, if the state of Inputbox1 and the state of Inputbox2 are both equal 1 or 0, then the XOR gate will output 0, otherwise, if the state of Inputbox1 and the state of Inputbox2 are different value, then the XOR gate will output 1.

5-Table

Fig.8 shown the table component in the package.The table can accept more than one input. Fig.9 shown the scenario of connecting three logical gates to the table.

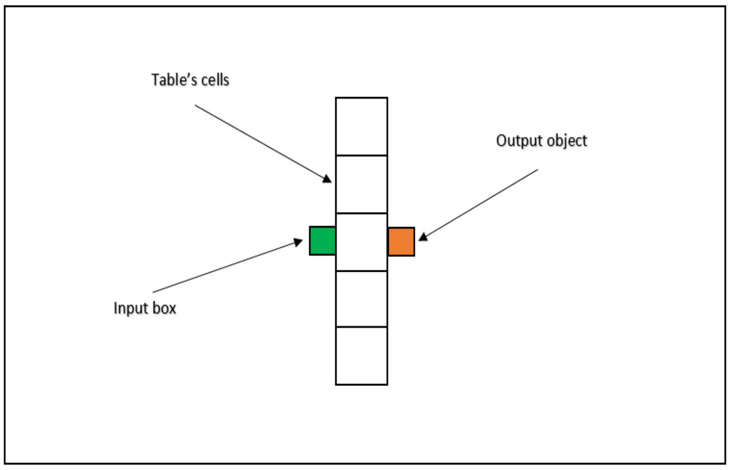


Fig.8: Design of table component.

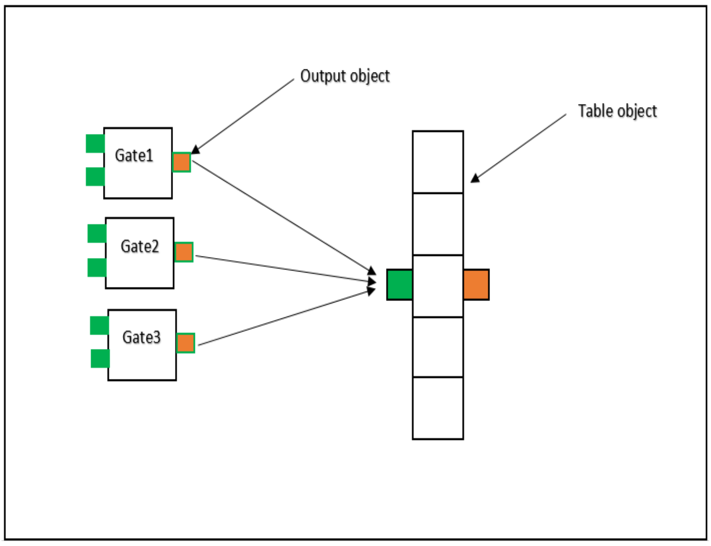


Fig.9: Example of connecting table.

Each output object that connected to table has variable called state store the output of gate. If the designer first connects gate1 to table then connect gate2 and last gate3, if states of the of output objects of all gates are 0,1,1 respectively, then the table convert all states of the that gates to the decimal number and output the value of cell that its index is equal to the calculated decimal number.

3.2 Building Stream Cipher Algorithms by C# Programming Language

Fig.10 shows the package's interface of build stream cipher algorithm by C# programming language.

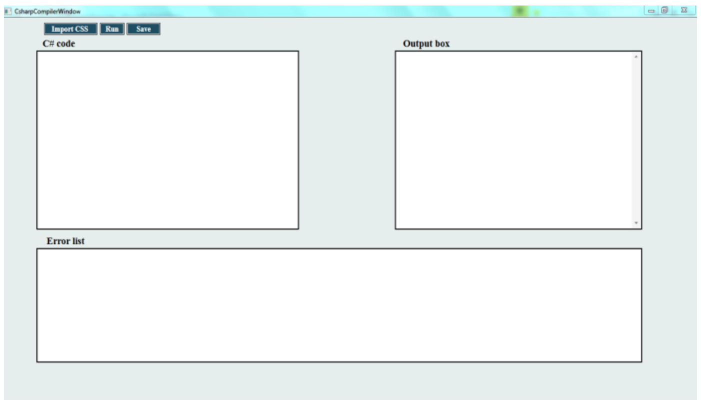


Fig.10:C# compiler interface.

The designer can enter the C# code of any stream cipher algorithm in C# code box or can import the C# code by click on the import button in button bar. After enter C# code, the designer has to click on the Run button in button bar to start compiling and execute of entered code. After execute the entered C# code, the output of C# code will appear in output box. Any syntax error in the entered code this will show error message in the error box as pointing in fig.10.

4. Design Encryption/Decryption Interface

Fig.11 shows the package's interface of encryption and decryption.In the interface, there are number of buttons that perform following actions:

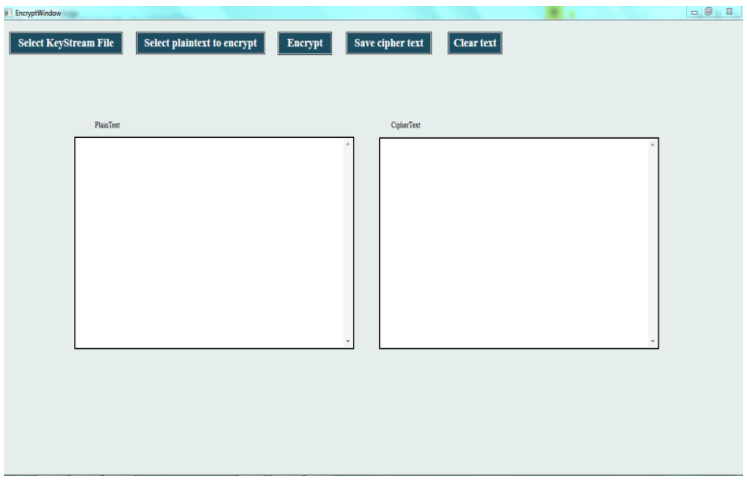


Fig.11: Encryption/decryption interface.

1- Import any text to encrypt it or decrypt.
2- Import any key file to use it in encryption/decryption process.
3- Clear all text in text boxes.

When the designer clicks on the encrypt button, the following actions will perform:

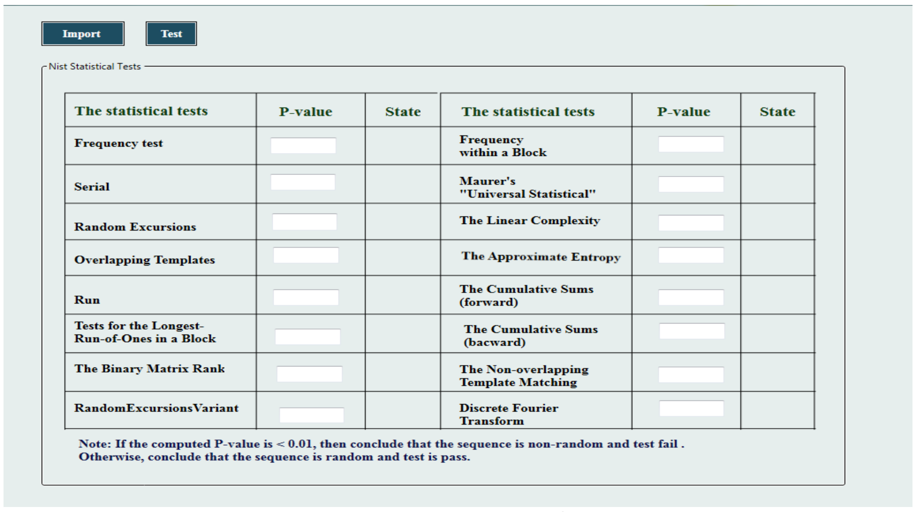
1- Convert the plain text and key stream characters to a sequence of byte.
2- Mix the key bytes with plaintext bytes by using XOR function, the result represents the cipher text. The cipher text stored in new array.
3- The new sequence that produce from mix plaintext with key stream will appear in Textbox2 in the encryption /decryption interface.

To decrypt the cipher text, the designer has to do the following

1-Import the cipher text file through click on import Plain/cipher text button.
2-Import the key stream used in produce the cipher text.
3-Click on the Encrypt/Decrypt button to start mixing the cipher bytes with key bytes, the result is the plaintext and will appear in the Textbox2.

IV. Design of Evaluation Interface

Fig.12 shown the structure of the evaluation interface. The interface shown all fifteen NIST statistical tests with its P-values and the state of the test whether it's pass or fail.



The statistical tests	P-value	State	The statistical tests	P-value	State
Frequency test			Frequency within a Block		
Serial			Maurer's "Universal Statistical"		
Random Excursions			The Linear Complexity		
Overlapping Templates			The Approximate Entropy		
Run			The Cumulative Sums (forward)		
Tests for the Longest-Run-of-Ones in a Block			The Cumulative Sums (bacward)		
The Binary Matrix Rank			The Non-overlapping Template Matching		
RandomExcursionsVariant			Discrete Fourier Transform		

Note: If the computed P-value is < 0.01, then conclude that the sequence is non-random and test fail . Otherwise, conclude that the sequence is random and test is pass.

Fig.12: Evaluation interface.

The evaluation interface contain table that shown information about 14 statistical tests as shown in fig.12.
The buttons bar contains two buttons:
1- Button for import the sequence of binary bit that to be test
2- Button for start of applying 14 NIST statistical tests.

V. Design Number of Keystream Generators and Evaluate the Produced Key

Following is implementation and evaluation of six keystream generators that have been built by the proposed package.

6.1 Building Geffe Generator and Evaluate the Produced key

Fig.13 shows the components of geffe and connection lines among them as executed in the package. The designed geffe generator has been tested and fig.14 shown the results of apply fourteen NIST statistical tests.The designed geffe generator consists of three LFSR, two AND gate, one Not gate, one XOR function. All LFSRs are initialized with value "fff".

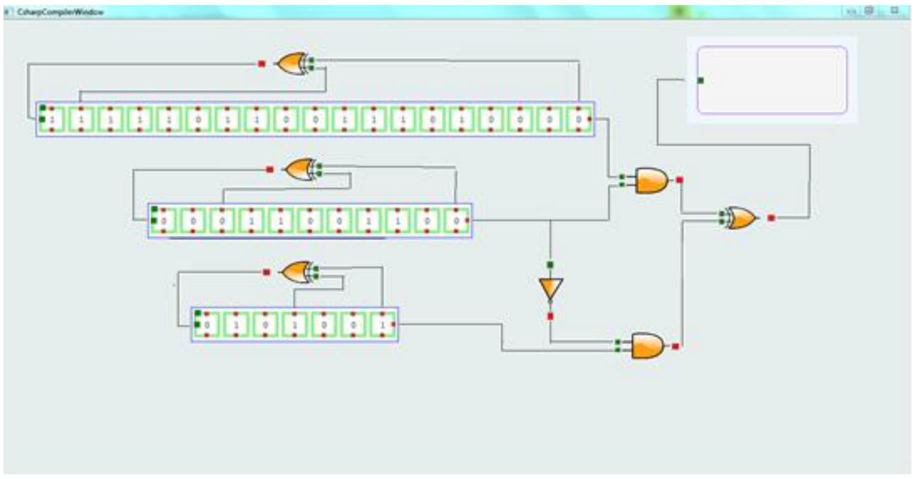
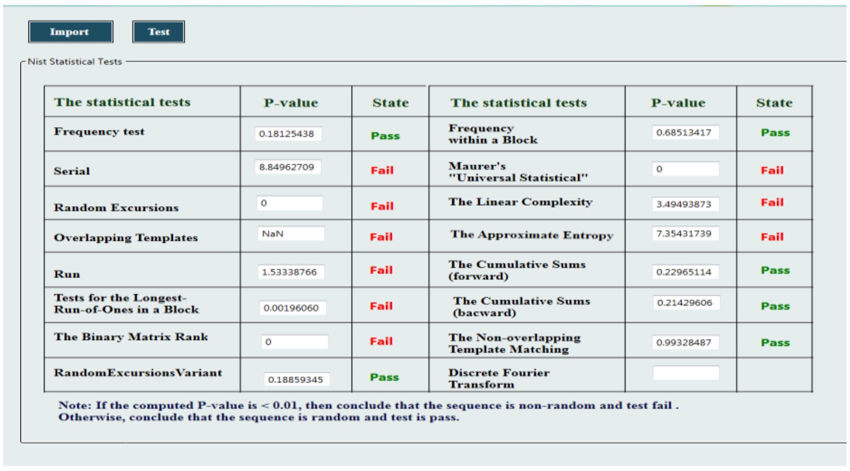


Fig.13: Designed geffe algorithm.



The statistical tests	P-value	State	The statistical tests	P-value	State
Frequency test	0.18125438	Pass	Frequency within a Block	0.68513417	Pass
Serial	8.84962709	Fail	Maurer's "Universal Statistical"	0	Fail
Random Excursions	0	Fail	The Linear Complexity	3.49493873	Fail
Overlapping Templates	NaN	Fail	The Approximate Entropy	7.35431739	Fail
Run	1.53338766	Fail	The Cumulative Sums (forward)	0.22965114	Pass
Tests for the Longest-Run-of-Ones in a Block	0.00196060	Fail	The Cumulative Sums (bacward)	0.21429606	Pass
The Binary Matrix Rank	0	Fail	The Non-overlapping Template Matching	0.99328487	Pass
RandomExcursionsVariant	0.18859345	Pass	Discrete Fourier Transform		

Note: If the computed P-value is < 0.01, then conclude that the sequence is non-random and test fail .
Otherwise, conclude that the sequence is random and test is pass.

Fig.14: Results of test geffe's key.

The figure shown that only 6 tests are passed from total of 14 tests.

6.2 Design New Keystream Generator and Evaluate the Produced Key

Some information about the new designed generator:

1- Number of used LFSRs are 6; LFSRs have lengths: (11, 4, 7, 19, 5, 13).
2- Other components used are: Police, JK flip flop, Table, XOR gate.
3- Length of Table component is 256, and initialize with value
" abcdefghijklmnopqrstuvwxyzabcde".

Fig.15 shows the designed key stream generator.

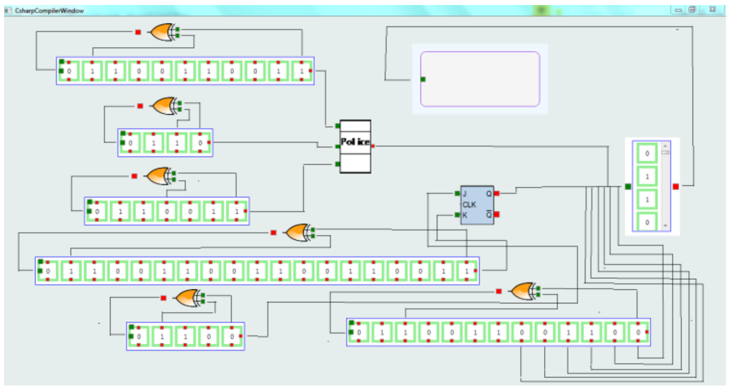
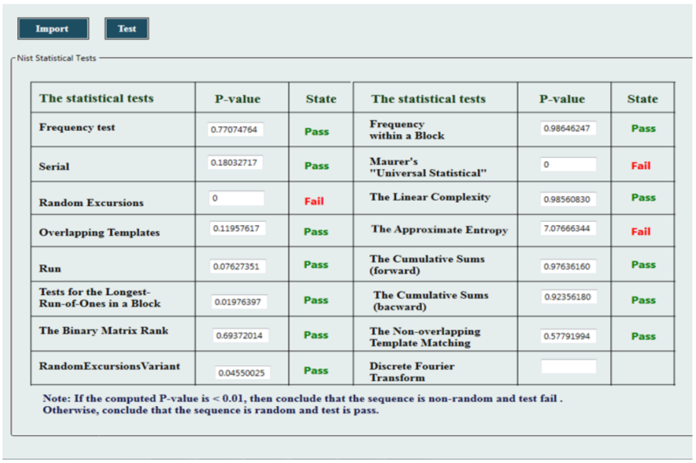


Fig.15: New keystream generator.

Fig.16 shown the results of apply 14 NIST statistical tests to the bit sequence of length 1425 produced from the designed generator.



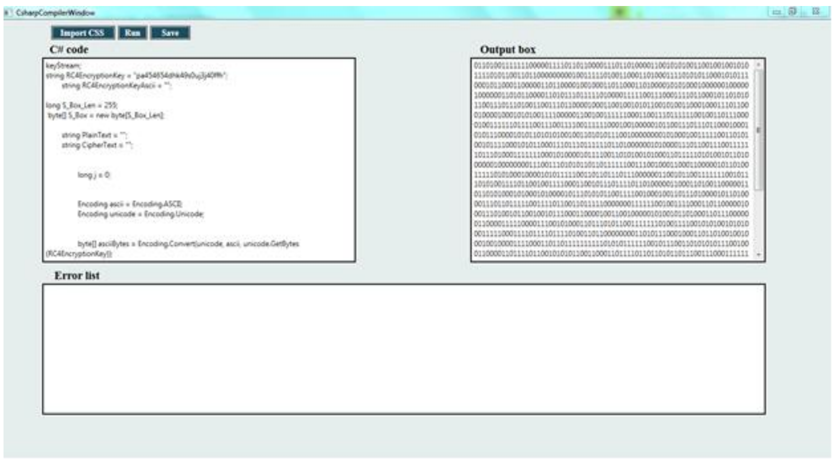
The statistical tests	P-value	State	The statistical tests	P-value	State
Frequency test	0.77074764	Pass	Frequency within a Block	0.98646247	Pass
Serial	0.18032717	Pass	Maurer's "Universal Statistical"	0	Fail
Random Excursions	0	Fail	The Linear Complexity	0.98560830	Pass
Overlapping Templates	0.11957617	Pass	The Approximate Entropy	7.07666344	Fail
Run	0.07627351	Pass	The Cumulative Sums (forward)	0.97636160	Pass
Tests for the Longest-Run-of-Ones in a Block	0.01976397	Pass	The Cumulative Sums (bacward)	0.92356180	Pass
The Binary Matrix Rank	0.69372014	Pass	The Non-overlapping Template Matching	0.57791994	Pass
RandomExcursionsVariant	0.04550025	Pass	Discrete Fourier Transform		

Note: If the computed P-value is < 0.01, then conclude that the sequence is non-random and test fail .
Otherwise, conclude that the sequence is random and test is pass.

Fig.16: Results of testing the key of new keystream generator.

6.3 Implement and Evaluate RC4 Stream Cipher Algorithm

Fig.17 shows build stream cipher algorithm by the proposed package, the algorithm is RC4 and has been built by enter C# code of RC4.

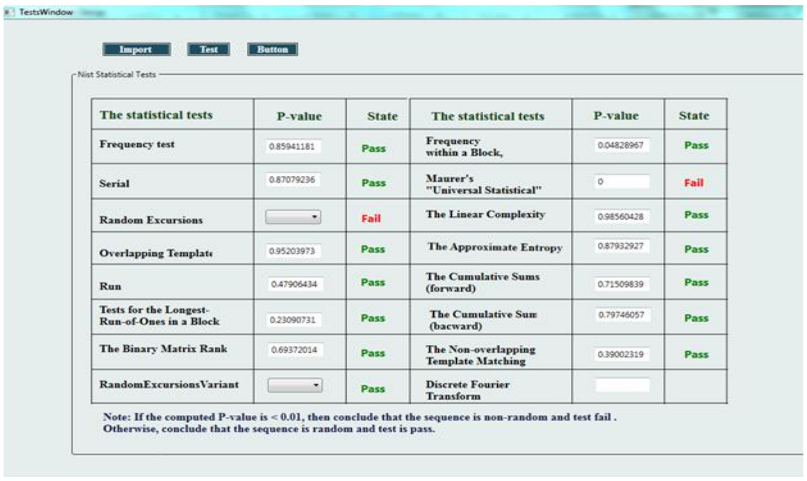


```
keyStream;
string RC4EncryptionKey = "pa454654dhk49s0uj3j40fh";
      string RC4EncryptionKeyAscii = "";

long S_Box_Len = 255;
 byte[] S_Box = new byte[S_Box_Len];

      string PlainText = "";
      string CipherText = "";


            long j = 0;


            Encoding ascii = Encoding.ASCII;
            Encoding unicode = Encoding.Unicode;


            byte[] asciiBytes = Encoding.Convert(unicode, ascii, unicode.GetBytes
(RC4EncryptionKey));
```

```
[illegible]
```

Fig.17: The implementation of RC4.

After save the key produced by the code of RC4 in text file, the key should save as binary sequence of '1' and '0'. Fig.18 shown the result of apply 14 statistical tests on the produced key of RC4. Length of tested key sequence is 2040 bits.



The statistical tests	P-value	State	The statistical tests	P-value	State
Frequency test	0.85941181	Pass	Frequency within a Block,	0.04828967	Pass
Serial	0.87079236	Pass	Maurer's "Universal Statistical"	0	Fail
Random Excursions		Fail	The Linear Complexity	0.98560428	Pass
Overlapping Template	0.95203973	Pass	The Approximate Entropy	0.87932927	Pass
Run	0.47906434	Pass	The Cumulative Sums (forward)	0.71509839	Pass
Tests for the Longest-Run-of-Ones in a Block	0.23090731	Pass	The Cumulative Sum (bacward)	0.79746057	Pass
The Binary Matrix Rank	0.69372014	Pass	The Non-overlapping Template Matching	0.39002319	Pass
RandomExcursionsVariant		Pass	Discrete Fourier Transform		

Note: If the computed P-value is < 0.01, then conclude that the sequence is non-random and test fail . Otherwise, conclude that the sequence is random and test is pass.

Fig.18: Results of test RC4's key.

The figure above shows 13 tests from a total of 14 tests are passed and only 1 tests are fail.

VI. Conclusion

We have developed package which capable of design any stream cipher algorithm either visually from the existing components that provided by the proposed package or by enter the C# code of stream cipher algorithm. Also, the proposed package provided to the designer the ability to evaluate the keystream which produced by the designed algorithm, the evaluation is done by apply fourteen statistical tests which provided by NIST, all these fourteen tests is executed in parallel in the package to speed up the execution time of these tests.

References

[1] Rainer A. Rueppel, *Analysis and Design of Stream Ciphers* (Berlin, Springer, 1986).
[2] Darrel Hankerson, Alfred Menezes, Scott Vanstone,*Guide to Elliptic Curve Cryptography* (New York, Springer, 2004).
[3] William Stallings, *Cryptography and Network Security Principles and Practice* (Boston, Pearson, 2011).
[4] Richard R. Brooks, *Introduction to Computer and Network Security* (Boca Raton, CRC Press, 2014).
[5] Christof Paar and Jan Pelzl, *Understanding Cryptography a Textbook for Students and Practitioners* (Berlin, Springer, 2010).
[6] M.J.B. Robshaw, *Stream Ciphers RSA Laboratories Technical Report TR-701*(Redwood City, CA, RSA Laboratories, 1995).
[7] Andreas Klein, *Stream Cipher* (London, Springer, 2013).