ISSN: 0067-2904

Generate Random Arabic Characters and Numbers for CAPTCHA

Mariam Taha Sulaiman*, Nidaa Flaih Hassan
Department of Computer Science, Technology University, Baghdad, Iraq.

Abstract

Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA**)** is a program where its goal is to check the user identity if it is a human or web program by creating tests that is easy to human but difficult to computer programs. In this paper, a mixed Arabic CAPTCHA schema is proposed to generate Arabic characters and numbers using generators that combines more than one Linear Feedback Shift Registers(LFSRs) via a non-linear function to produce the binary sequence. This random binary sequence is translated to be Arabic characters and numbers to be used for Arabic CAPTCHA, to ensure the randomness, each generator output is analyzed via randomness analysis using National Institute of Standards and Technology (NIST) statistical test suite.

توليد حروف عربية وارقام عشوائية للكابتشا

مريم طه سليمان*، نداء فليح حسن
قسم علوم الحاسبات، الجامعة التكنولوجيه، بغداد، العراق.

الخلاصة

الكابتشا "اختبار تورينج العام الآلي تماما للتمييز بين برامج الحاسوب والبشر" هو برنامج يستخدم للتحقق فيما اذا كان المستخدم انسان او برنامج حاسوبي وذلك من خلال توليد اختبارات تكون سهلة على الانسان وصعبة على برامج الحاسوب. في هذا البحث تم اقتراح مخطط جديدة للكابتشا العربية لتوليد رموز عربية وارقام باستخدام المولدات التي تدمج اكثر من (LFSRs) عن طريق تطبيق معادلة غير خطية لتوليد متسلسلة عشوائية، هذه المتسلسلة الثنائية العشوائية يتم تحويلها الى رموز عربية وارقام ليتم استخدامها لنظام الكابتشا العربية، ولقياس كفاءة أداء هذه المتسلسلة العشوائية، يتم اختبار نتيجة كل مولد عن طريق تحليل العشوائية باستخدام حزمة الاختبار الاحصائية للمعهد الوطني للمقاييس والتكنولوجيا (NIST).

Introduction

Completely Automated public Turing test to tell Computers and Humans Apart schema is abbreviation of the word CAPTCHA, which is utilized to differentiate between users of human and computer programs. CAPTCHA should be easily solved by human but not by a bot, CAPTCHA has the following characteristics [1]:

1 . A machine is a judge not a human.

2 . The aim is that substantially all the human users will be identified and pass the test and no computer will pass.

*Email: mariamtaha201420@yahoo.com

The increase of accessible services that can be reached publicly on the Web is useful for the society in general, however, unluckily new and unusual abuses have been called by creating programs to rob services and to make fake dealings [2]. Therefore, the CAPTCHA has some applications that prevent these abuses [3]:

1. Alleviating Comment Spam.
2. Online Polling.
3. Web Registration.

The CAPTCHA system prohibits different websites to avert various bots from attacking network resources. A CAPTCHA system with good quality should have the following features [4]:

1 . The content must be understandable by the human.
2 . Quicker and few time consumption.
3 . Appropriate for all form of bots hurt.

The motivation for generating Arabic CAPTCHA schema is that there are a few of Arabic CAPTCHA schema; so it could be helpful for Arabian talking users. Also, many of the current English CAPTCHAs schemas have few attendant problems so, they can't guarantee the websites safety.

In this paper, a new CAPTCHA schema is proposed to generate Arabic letters and numbers using combination of generators, these generators combined more than one LFSRs via a non-linear function to produce one binary sequence, then this sequence is translated to letters to generate them in random form for CAPTCHA schema. In the rest of paper, Types of CAPTCHA are described in section 2 , pseudo-random linear feedback shift register (LFSR) generator is given in section 3. In section 4, testing of randomness statistical tests (NIST) are discussed. The proposed schema is described in section 5, and the conclusion is presented in section 6.

The Related Work

In 1997, Moni Naor [5] made the first reference to the automated turing tests which appeared in unpublished handwriting, this handwriting involves some crucial concepts and obviousness; however, it didn't give suggestion and a proper definition for the robotic turing test. In 1998, The first practical Automated Turing Test sample was established by Altavista[6] to prohibit web bots from signing up in websites automatically, the complication of the system was depend on reading a somewhat distorted characters printed on an image and it operated very well during the practice, but this system was attacked by OCR(optical character recognition).

In 2007, Elson et al. [7], suggested a way where the user must choose the cat images among set of cat and dog images, for the set of images, the database was used contained more than 3 million images.

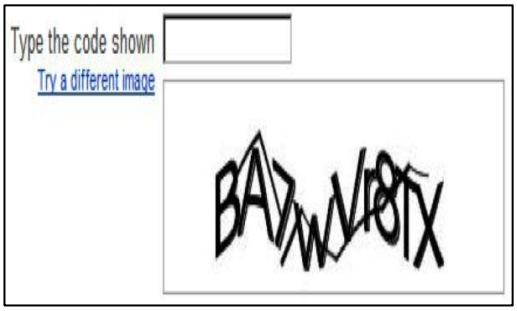
The commonly CAPTCHA employs a Latin text which has several attendant fragility in it. CAPTCHA execution that utilizes other languages has been neglected by the research society except the CAPTCHA that employs the Persian language.

In 2009, Gupta et al.[8], suggested the way of putting a numbers to text CAPTCHAs, which are embedded within the text poses the OCR. In 2013, Bilal Khan et al. [9] proposed CAPTCHA based on the Arabic language, it is considered as a big advancement in a web security area, Arabic based CAPTCHA is produced as an image which is distorted with various techniques in order to become harder to break it by OCR.

Types of CAPTCHAs

CAPTCHA is generally characterized into four types

1. Text based CAPTCHAs: Text-based CAPTCHA is appearing in deformed form which involves letters that are case- insensitive and numbers. This type of CAPTCHA can be found in known websites like Yahoo, Hotmail, Gmail, and YouTube. Gimpy, Ez-Gimpy, Baffle-Text and MSN-CAPTCHA are types of Text-based CAPTCHA [4], [10]. Figure-1 shows examples of text CAPTCHA.



a- typical text based CAPTCHA

b- Example of Gimpy CAPTCHA

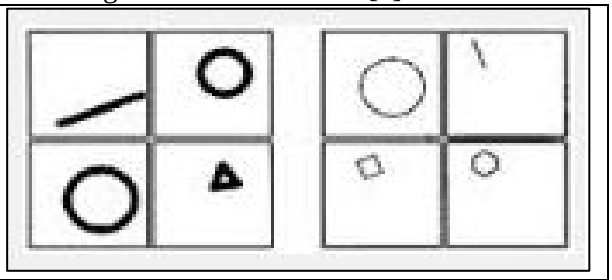
c- Example of Ez-Gimpy CAPTCHA

d- Example of Baffle-Text CAPTCHA

Figure 1- A typical Text based CAPTCHAs [1], [10].

2. Image based CAPTCHAs: Image based CAPTCHAs were made as alternatives to the text based CAPTCHA [3], so the image recognition is a task performed by the users. Examples on this type are: Pix and Bongo CAPTCHA. Figure-2 shows example of image based CAPTCHAs [4].

a- Example of Pix CAPTCHA



b- Example of Bongo CAPTCHA

Figure2- Image based CAPTCHAs [4]

3. Video based CAPTCHAs: In video based **CAPTCHAs** a moving object is given to the user and is requested to accomplish a particular task [3]. Figure-3 shows a frame from video based CAPTCHA.

Figure 3-An Example of Video-based CAPTCHA [4]

Audio-Based CAPTCHAs: These systems depend on speech recognition by the user; this type of CAPTCHA is designed as an alternative for people who are visually impaired. It consists of downloadable voice clips. The user should listen to the clip and then writes the word [4], [3].

Pseudo-Random Linear Feedback Shift Register Generator (PRLFSRG)

Linear feedback shift registers are commonly used in bit generators since, they are appropriate for hardware implementation, creating sequences with large periods and are easily analyzed using algebraic techniques. However, the disadvantage is that the output sequences of LFSRs are readily predictable [11].

In feedback shift register as shown in Figure-4, the flip-flops $F0, \dots, Fn-1$ are numbered. At each step Fi takes the value of $Fi-1$ for $i > 0$ and the update to the *F0 is* depending on the feedback function *f*.

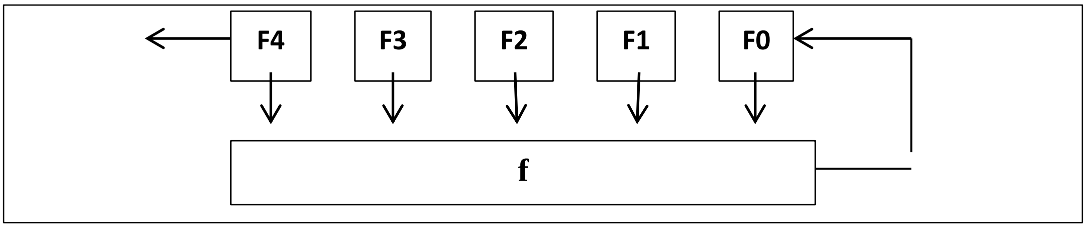


Figure 4- A feedback shift register [12].

Mathematically, the sequence (a_i) i ϵ n produced from shift register is only the sequence that meet the n-expression recursion. Equation (1) shows the feedback function of the shift register to produce a new bit that will be put in the beginning of the sequence[12]:

$$a_{i+n} = f(ai, \dots\dots, a_{i+n-1}) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad (1)$$

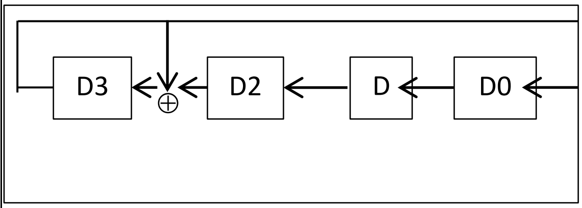
Where:

a : the resulted bit from the feedback function.

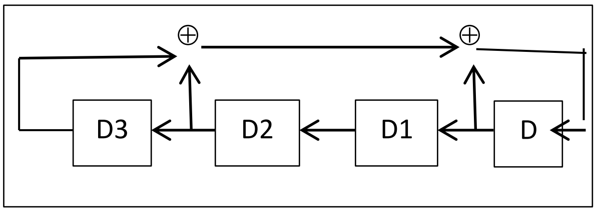
f : the feedback function that is applied on the initial seed of shift register recursively.

n : seed length.

A shift register is called linear if the feedback function is linear [12]. Figure-5 shows two examples of LFSR.



(a): Internal EOR type.



(b): External EOR Type.

Figure 5-LFSRs Example [13].

Both wo Figures deal with D-type flip flops and linear logic ingredients (Exclusive-or (xor)) that recognize LFSRs. Figure-6 describes LFSR that simulates 9 shifts of Figure-4 [13].

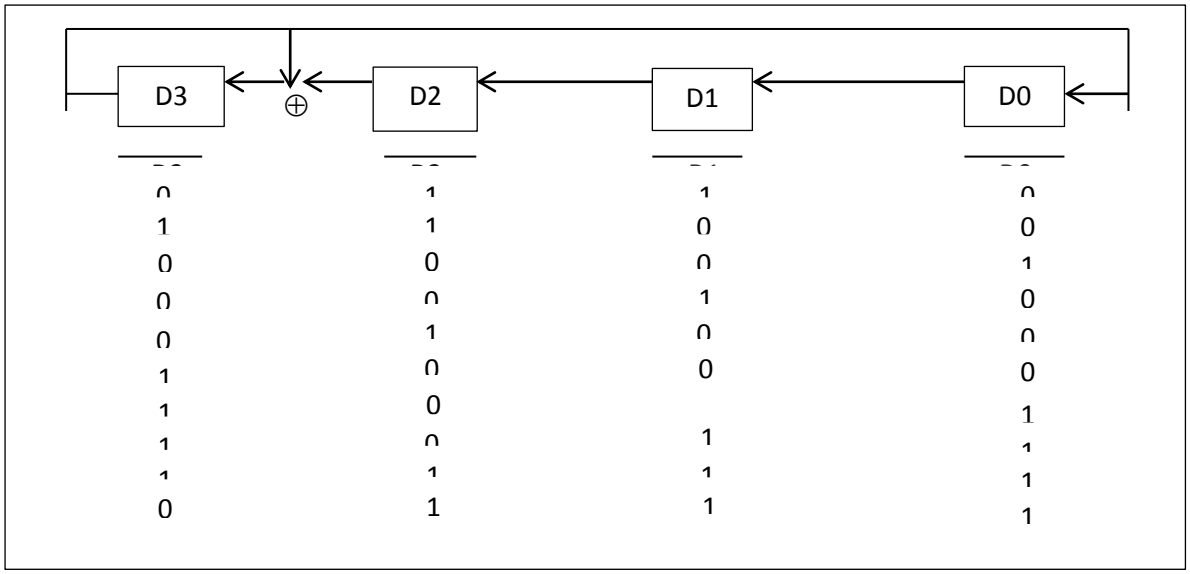
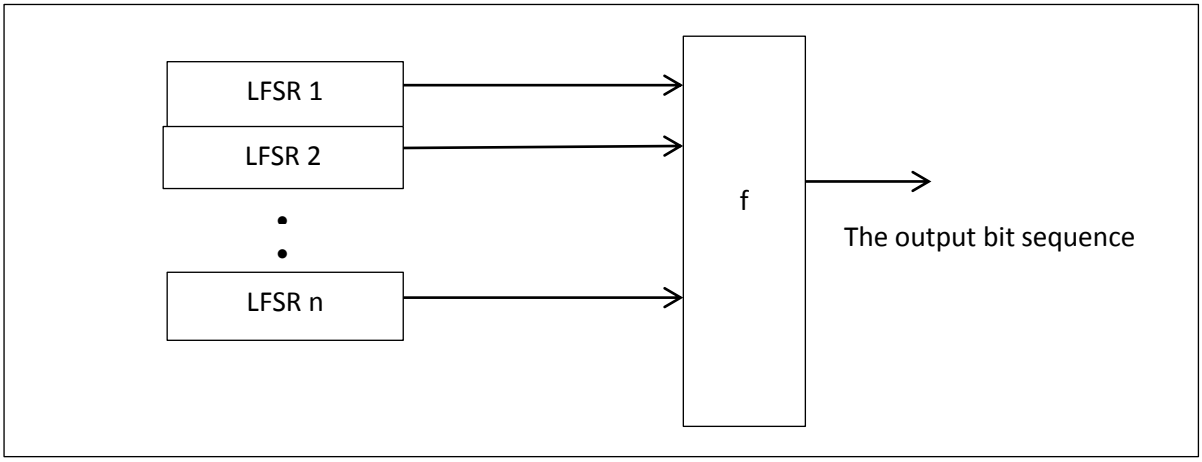


Figure 6- LFSR that simulates 9 shifts of Figure-10 with the initial seed "0110"[13] .

Nonlinear combination generators

One common way to break the inherit of linearity in LFSRs is to utilize many LFSRs. the output binary sequences of LFSRs are collected via a nonlinear function (f) and produce one output binary sequence. Such generators are called nonlinear combination generators and the function(f) is called a combining function[11]. Figure-7 illustrates nonlinear combination generator.



Hadmard generator

consists of combining two linear feedback shift register of (L1*L2) bit length with the following nonlinear function as shown the Figure-8 [12]:

$S = (X1\ and\ X2)$ ……(2)

Where:

X1 : the resulted binary sequence from the first LFSR.

X2 : the resulted binary sequence from the second LFSR.

S : the resulted binary sequence.

L1 : the length of the first LFSR.

L2 : the length of the second LFSR.

and the maximum length of bits for this generator is estimated by the following equation:

Max_Length = $(2^{L1}-1)(2^{L2}-1)$ ……(3)

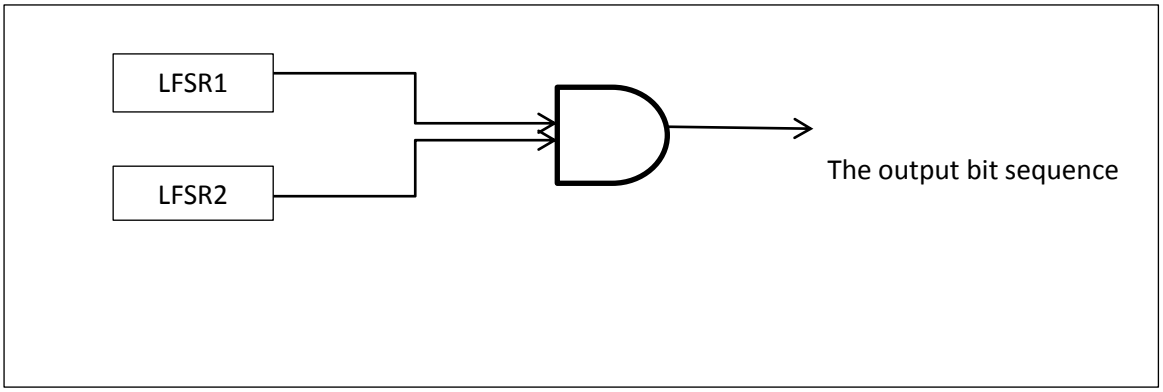


Figure 8- Hadmard Generator [12]

Geff generator

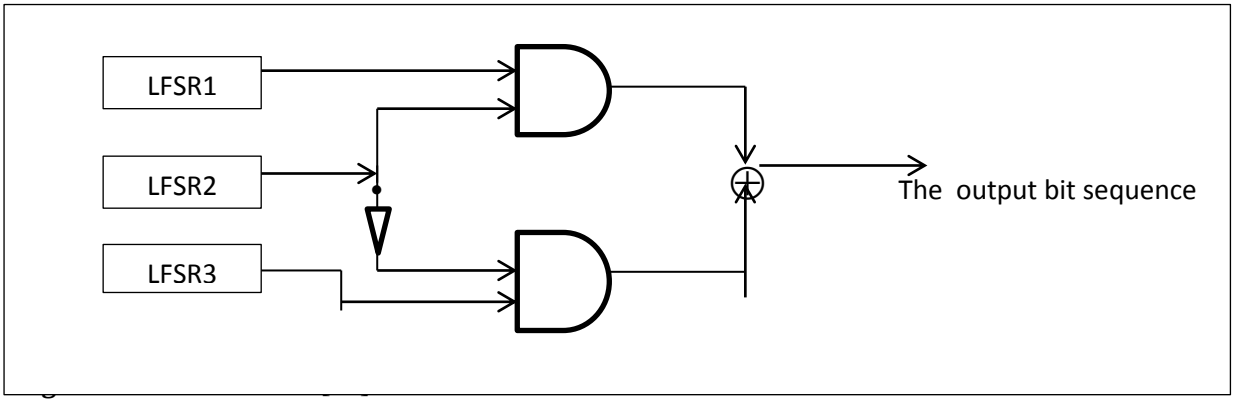
Non-Linear Geff generator which consists of combining three Linear Feedback Shift Registers (LFSR's) of (L1*L2*L3) bit length, with the following non-linear function as shown the Figure-9 [12]:

$S = (X1\ and\ X2) xor\ (not(X2) and\ X3)$ ……(4)

Where:

X1: the resulted binary sequence from the first LFSR.
X2: the resulted binary sequence from the second LFSR.
X3: the resulted binary sequence from the third LFSR.
S : the resulted binary sequence.
L1 : the length of the first LFSR.
L2 : the length of the second LFSR.
L3 : the length of the third LFSR.
and the maximum length of bits for this generator is estimated by the following equation :

Max_Length = $(2^{L1}-1)(2^{L2}-1)(2^{L3}-1)$ ……(5)



Testing of Randomness by NIST Statistical Tests

The NIST (National Institute of Standards and Technology) Test Suite is a statistical bundle involving 15 tests where the function of each test is to test the randomness of a particular binary sequence of any length resulted by random or pseudorandom number generators.

A parameter 'n' parameterized all tests which refer to the length (in bits) of the processed binary sequence. Another parameter is also used to parameterize these tests and is referred to by m or M. m parameter is established for detecting the existence of many m-bit patterns in a bit stream while the M parameter Study the distribution of a particular characteristic across n/M portions of a specified sequence[12] . These 15 tests are[14]:

1. The Frequency (Monobit) Test.
2. Frequency Test within a Block.
3. The Runs Test.
4. Tests for the Longest-Run-of-Ones in a Block.

5. The Binary Matrix Rank Test.
6. The Discrete Fourier Transform (Spectral) Test.
7. The Non-overlapping Template Matching Test.
8. The Overlapping Template Matching Test.
9. Maurer's "Universal Statistical" Test.
10. The Linear Complexity Test.
11. The Serial Test.
12. The Approximate Entropy Test.
13. The Cumulative Sums Test.
14. The Random Excursions Test.
15. The Random Excursions Variant Test.

The Proposed Schema

In this paper an Arabic letters and numbers is generated for CAPTCHA using combination generators that collects more than one LFSRs where each LFSR has its maximum length. The binary sequence generated from LFSRs is converted in the Arabic letters and numbers.

The algorithm for generating random Arabic letters and numbers for Hadmard generator are illustrated in algorithm 1:

Algorithm (1): Generate Random Arabic Letters and Numbers (Hadmard Generator)
Input : S1: the first initial seed //consist of mixing of letters and numbers.
S2: the second initial seed // consist of mixing of letters and numbers.
No_Sym: number of the required characters.
Output: Pseudorandom mixing of Arabic letters and digits.

Step1: For each LFSR seed with its feedback function do the following:
Convert the seed to the binary form.
For N= 1 to No_Sym ∗11 do
Shift the binary sequence of seed twice as the following:
For M= 1 to 2 do
$x_{n+5}=x_n \oplus x_{n+2} \oplus x_{n+3}$ where n= {0,1,2,3,……..} // apply a feedback function.
Shift one bit right
Set the resulted bit to the beginning of sequence.
End For
End For

Step2: Apply a non-linear function of the Hadmard generator according to equation (2).

Step3: Convert the resulted binary sequence to the character form .

Step4: end.

The above algorithm can be also applied to the Geff generator, where the input is three initial seed instead of two and the non-linear function of the Geff generator is applied according to equation (4).

Results and Discussion

The randomness of both generators were analyzed and the results showed that Hadmard generator has less randomness compared with Geff generator since Hadmard generator consists of two LFSRs while Geff Generator consists of three LFSRs. Also the non-linear function of the Hadmard Generator is less complex than of the Geff Generator.

Table-1 shows the results of 15 tests of **NIST** statistical test suite for Hadmard generator, and Table-2 shows the results of 15 tests of NIST statistical test suite for Geff generator.

Table 1- results of 15 randomness tests for Hadmard Generator

Test Name	*Parameters*	*P-value*	*Result*
Frequency Test	**n=2048**	**0.0**	***Not pass***
Frequency within Block Test	**n=1024** **M=24**	**2.0639872719298109E-45**	***Not pass***
Run Test	**n=120**	**0.0**	***Not pass***
Longest Run of Ones in a Block Test	**n=128** **M=8** **K=3**	**0.01464845328832301**	***Pass***
Binary Matrix Rank Test	**n= 41984** **M=32** **Q=32**	**0.000000308455037214158**	***Not pass***
Discrete Fourier Transform Test	**n=1000**	**0.56165763407334479**	***Pass***
Non-overlapping Template Matching Test	**n=100** **M=10** **m=9** **B=000000001**	**0.00000009014051694138**	***Not pass***
Overlapping Template Matching Test	**n=1000000** **M = 100000** **m=9** **B =111111111**	**0.999458898920335**	***Pass***
Maurer's "Universal Statistical" Test	**n=387840** **L=6** **Q=640**	**0.0**	***Not pass***
Linear Complexity Test	**n=1000000,** **M=500**	**4.0841509412911519E-63**	***Not pass***
Serial Test	**n=100** **m=4**	**8.4342830469510429E-33** **0.0000000000925838390787**	***Not pass*** ***Not pass***
Approximate Entropy Test	**n=100** **m=3**	**0.00000000000420805142888**	***Not pass***
Cumulative Sums Test	**n= 1000**	**0.011412078030893613**	***Pass***
Random Excursions Test	**n=1000000**	**0.00001111789670699629 7** **0.005356538842106036** **0.98837260158096363** **0.962581903015739** **0.70001667194455353** **0.98837260158096363** **0.99532694853562764** **0.99789963674605608**	***Not pass*** ***Not pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass***
Random Excursions Variant Test	**n=1000000**	**0.9034788300777421** **0.89727889046929865** **0.88970688113444185** **0.88016842176554122** **0.867632329699021** **0.8501067716269014** **0.82306335579544831** **0.7728301262532622** **0.61707507455201238** **0.999999999** **0.7728301262532622** **0.6547208997400692** **0.70545710121524141** **0.73888281604994976**	***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass***

		0.76302473736783194 **0.78151142383247718** **0.79625353148395284** **0.80836525895619271**	***Pass*** ***Pass*** ***Pass*** ***Pass***

Table 2-results of 15 randomness tests for Geff Generator

Test Name	***Parameters***	***P-value***	***Result***
Frequency Test	**n=2048**	**0.59588305587547974**	***Pass***
Frequency within Block Test	**n=1024** **M=24**	**0.042574280677609183**	***Pass***
Run Test	**n=120**	**0.000000000387393006562319 9**	***Not pass***
Longest Run of Ones in a Block Test	**n=128** **M=8** **K=3**	**0.34517346833190854**	***Pass***
Binary Matrix Rank Test	**n= 41984** **M=32** **Q=32**	**0.00011392992039678925**	***Not pass***
Discrete Fourier Transform Test	**n=1000**	**0.081658435957491671**	***Pass***
Non-overlapping Template Matching Test	**n=100** **M=10** **m=9** **B=000000001**	**0.999999999999991**	***Pass***
Overlapping Template Matching Test	**n=1000000** **M = 100000** **m=9** **B =111111111**	**0.99231221261874236**	***Pass***
Maurer's "Universal Statistical" Test	**n=387840** **L=6** **Q=640**	**0.0**	***Not pass***
Linear Complexity Test	**n=1000000,** **M=500**	**0.0000372618287389854 27**	***Not pass***
Serial Test	**n=100** **m=4**	**0.0000000000022733674947 6** **0.0000085838116867496387**	***Not pass*** ***Not pass***
Approximate Entropy Test	**n=100** **m=3**	**0.0000000133658577870193 7**	***Not pass***
Cumulative Sums Test	**n= 1000**	**0.4115109937213085**	***Pass***
Random Excursions Test	**n=1000000**	**0.0056619077387473** **0.0010311071692925963** **0.0014068972466118319** **0.18319958540651377** **0.0014065357774440772** **0.0045508377340672864** **0.000007804519757542395** **0.00000046357882553574**	***Not pass*** ***Not pass*** ***Not pass*** ***Pass*** ***Not pass*** ***Not pass*** ***Not pass*** ***Not pass***
Random Excursions Variant Test	**n=1000000**	**0.255290613059326** **0.29560006577759812** **0.344097645926986** **0.40333954631811753** **0.47728920158360388** **0.68701344031731193** **0.999999999** **0.90203456854890085**	***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass*** ***Pass***

		0.66981543226924267	***Pass***
		0.83117047808536737	***Pass***
		0.80554069520404659	***Pass***
		0.63355349763355051	***Pass***
		0.57270229278806206	***Pass***
		0.522431165077713	***Pass***
		0.56289823005204886	***Pass***
		0.5946004961621425	***Pass***
		0.74118164220184	***Pass***
		0.87672171783646025	***Pass***

As shown in the above tables, Hadmard Generator passes 5 tests which are: Longest run of ones in a block test, Discrete Fourier transform test, Overlapping template matching test, Cumulative sums test and Random Excursions Variant Test.

While Geff Generator passes 8 tests which are: Frequency test, Frequency within block test, Longest run of ones in a block test , Discrete Fourier transform test , Non-overlapping template matching test , Overlapping template matching test , Cumulative sums test and Random Excursions Variant Test.

The output of the proposed generators are converted to characters and numbers form, and this is done by taking every 11 bits (since, the maximum representation required for Arabic characters in the binary form is 11 bits), mapping these 11 bits to numbers in decimal representation which is considered as ASCII-code , then represent the ASCII- code in the form of symbol.

The output symbol should be in the range of Arabic characters and numbers as stated before, so the representing of resulted decimal number in the range of Arabic characters and numbers ASCII –code is done by applying the following steps:

1. Subtract the required number from the minimum number of the range.

2. Take the modulus of the resulted number from step 1 and the difference between the maximum and minimum number of the range.

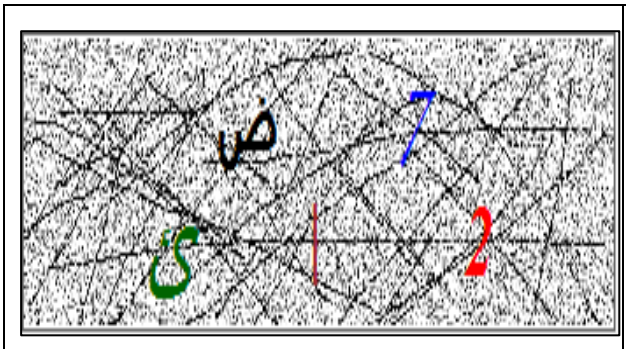
3. Check the resulted number from step 2 if it is negative, then add it with the difference between the maximum and minimum number of the range and continue until it becomes a positive number.

4. Add the nonnegative value with the maximum number of the range.

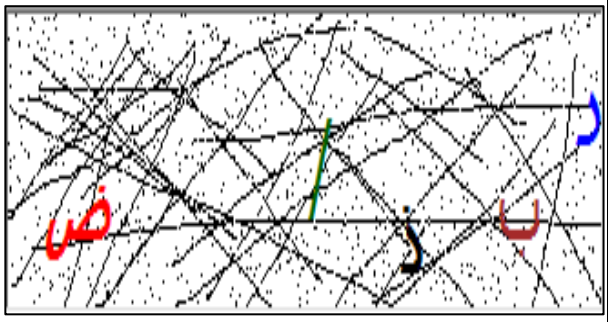
The final result for CAPTCHA schema is a stream of random Arabic characters, numbers or mix of them which are displayed on the selected background image.

A distortion and some CAPTCHA effects are added to background image to complicate the characters recognition process by the automated programs; Figure-10 shows an image of random Arabic letters and numbers with some types of noise:

.



a. Gaussian Noise



b. Salt and Pepper Noise

Figure 10- Arabic CAPTCHA

Conclusion

CAPTCHA is a system that keeps website services from attacks by making tests; it's easy for human but difficult for computer programs to pass. CAPTCHA has many types and the most common types are text, image and video based CAPTCHA. In this paper, an Arabic characters and numbers are generated randomly for Arabic CAPTCHA schema by using two types of generator which are Hadmard generator and Geff generator. The randomness of both generators is tested , its conclude that Hadmard generator has less randomness compared with Geff generator since Hadmard generator consists of two LFSRs while Geff Generator consists of three LFSRs, also the non-linear function of the Hadmard Generator is less complex than of the Geff Generator.

References

1. Kulkarni, S. and Fadewar, H. S. **2013**. CAPTCHA Based Web Security. *International Journal of Advanced Research in Computer Science and Software Engineering*. **3**(11):154-158.
2. Mathew, G.T. **2010**. CAPTCHA Seminar Report. Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology. Computer Science Engineering of Cochin University of Science and Technology.
3. RAGAVI, V. and GEETHA, G. **2014**. Captcha And Its Applications. *Journal of Computer Science Engineering and Information Technology Research* (JCSEITR). **4**(1):11-16.
4. Kaur, K. and Behal, S. **2014**. CAPTCHA and Its Techniques. *International Journal of Computer science and Information Technologies*. **5**(5): 6341-6344.
5. Naor, M. **1996**. Verification of a human in the loop or Identification via the Turing Test. 13^{th} ed. Dept. of Applied mathematics and computer science. Weizmann institute of science. Rechovot 76100.
6. Lillibridge, M.D., Abadi, M., Bharat, K. and Broder, A. **2001.** Method for selectively restricting access to computer systems. United states patent. US Patent 6,195,698 B1. Applied April 1998.
7. Elson, J., Douceur, J. and Saul, J. **2007**. Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization**.** in Proceedings of the 14th ACM Conference on Computer and Communications Security. Virginia USA. Pp: 366-374.
8. Gupta, A., Jain, A., Raj, A. and Jain, A. **2009**. Sequenced Tagged CAPTCHA: Generation and its Analysis. in Proceedings of International Advance Computing Conference. India. pp. 1286-1291.
9. Khan, B., Alghathbar, K. , Khan, M.K., Alkelabi, A. and Alajaji, A. **2013**. Cyber Security Using Arabic CAPTCHA Scheme**.** *The International Arab Journal of Information Technology*. **10**(1): 76-84.
10. remanand, V.P., Meiappane, A. and Arulalan, V. **2015.** Survey on CAPTCHA and its Techniques for BOT Protection. *International Journal of Computer Applications* (0975 – 8887). **109**(5): 1-4.
11. Menezes, A., Oorschot, P.V and Vanstone, S.A. **1996**. *Stream Ciphers. Handbook of Applied Cryptography*. CRC Press.pp.191-222.
12. klein, A**. 2013**. *Stream Ciphers*. Dept. of Pure Mathem & Computer Algebra, London. New York. state university of Ghent, Ghent, Belgium.
13. saluja, K.K.**1987**. Linear Feedback Shift Registers Theory and Applications. Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, Wisconsin 53706, Revised October 1988, Updated 1991.
14. Rukhin, A. , Soto, J. , Nechvatal, J. , Smid, M. , Barker, E. , Leigh, S. , Levenson, M. , Vangel, M. , Banks, D. , Heckert, A. , Dray, J. , Vo, S. **2010**. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. National Institute of Standards and Technology. Technology Administration, U.S. Department of Commerce.