IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-2 E-ISSN: 2347-2693

OTP Generation Algorithm: A Rubik's Cube Principle Implementation

Bose S.[1*], and Roy Chowdhury D.[2]

[1*] Department of Computer Science & Application, Salesian College, University of North Bengal, Siliguri, INDIA
[2]Department of Computer Science & Application, University of North Bengal, Siliguri, INDIA

[*]*Corresponding Author: bosesubhajit87@gmail.com, Tel.: +91-9832356650*

Available online at: www.ijcseonline.org

Received: 03/Jan/2017 Revised: 10/Jan/2017 Accepted: 04/Feb/2017 Published: 28/Feb/2017

***Abstract*— A One-Time Password (OTP) is an auto - generated, string of characters (password) that validates the user to carry out a single transaction or session on digital devices like a Computer, Smartphone, Tablet etc. Various unique techniques underwent implementation time and again for producing an optimal and efficient OTP. In most of the techniques, the OTP generated is of a shorter length and comprised of only letters of English alphabet (a – z, A - Z), digits (0 – 9) and characters like @ etc. In this paper a novel approach for generating One-Time Password (OTP) has been proposed using Rubik's cube principle based on a 4 × 4 cube in which each box of the cube is labeled with characters present on the keyboard such that when the layers of the cube are scrambled in various ways, it creates a 16 character OTP. This technique using Rubik's cube have never been applied before to generate an OTP.**

Keywords- One TimePassword(OTP); Hash Function; Random Function; Replay Attacks.

I. INTRODUCTION

Security on the Internet is an issue of major concern. The Internet is an unsafe medium where exchanging information might lead to intrusion or fraudulent activities by web mechanisms such as phishing. Different techniques had been implemented to protect information which passes through this insecure channel, one of them being Cryptography .

Consequently, the digital lock and keys (user ids and passwords) that users keep with them are no longer completely safe as their user ids and passwords can easily get stolen through programs such as spyware or through phishing. This is the reason for which, while doing online financial transaction, we are being asked to generate an OTP which is being sent to the user's cell phone to validate the person who is doing this transaction to be the authentic user of the debit/credit card that is being used in the transaction and not somebody else who is impersonating the actual user.

Hence, a 2-Step Verification procedure, which is being provided by well-known websites such as Google, should be implemented by every user to add one more layer of security to their account.

First, through something they have (their account password) and second, through something they receive (an OTP that is being sent to their cell phone). The reason is that if a password is compromised, the OTP would still have to be broken as well to gain access.

OTPs cater to the numerous flaws that are related with the traditional password oriented verification system. To implement the OTP, a user must have a phone apart from the user id and password or pin that is known to the user, as the OTP will be sent to the phone.

The best thing about OTPs is, compared to static passwords; they are not defenceless against replay attacks. Another important feature that an OTP provides is, if a person maintains the same password on multiple systems or accounts then that person is not defenceless on all of them, if the password is stolen by a hacker.

OTP generation algorithms basically employ randomness and hash functions so that it becomes difficult for an attacker to predict consecutive successor OTPs by observing previous ones.
The various ways by which OTPs can be generated are given below:

- Using a **time-synchronization** mechanism between the authentication server and the client providing the password
- Using a mathematical **algorithm** to create an OTP **based on the previous OTP**
  Using a mathematical **algorithm** where the new password is **based on a challenge and/or a counter**.

The remaining portion of the paper is organized as follows:
In the next section (Section - II) some of the contemporary algorithms which had been developed as related research are described briefly. In Section – III the steps that are to be taken

before carrying out the proposed method are described. In Section – IV, terminologies related to the proposed method have been detailed. Section – V covers the OTP generation algorithm based on Rubik's cube principle. Experimental results based on the proposed method are presented in Section - VI. Finally, the paper concludes in Section – VII.

II. RELATED RESEARCH WORK

Till now, there have been quite a lot of research activities in the field of Internet Security through OTP generation. Many algorithms had been developed, some completely new while the others are based on existing ones. Several algorithms are available in literature.

Tamanna Saini [1] proposed an OTP creation method by using Genetic Algorithm with Elliptic Curve Cryptography wherein when a user logs in using his/her username and password, the Server after validating the user encrypts the username and password and then sends both the username and password as input to an OTP generator which selects two alphabets from the encrypted data and then using genetic algorithm and elliptic curve cryptography generates the required OTP.

Yun Huang, Zheng Huang and Haoran Zhao [2] proposed TSOTP: a new effective simple OTP method that generates a unique passcode for each use. The calculation uses both time stamps and sequence numbers. In their OTP prototype, they first calculate an OTP based on time stamps to avoid high computation cost on the claimant side and to detect forced delay attacks. Due to message transit time, processing time and clock drift, an acceptance window is used by the verifier.

Himika Parmar, Nancy Nainan and Sumaiya Thaseen [3] proposed an authentication service that is image based and which eliminates the need for text passwords. Using the instant messaging service available in internet, users will obtain the One-Time Password (OTP) after image authentication. This OTP then can be used by users to access their personal accounts. The image based authentication method relies on the user's ability to recognize pre-chosen categories from a grid of pictures. This paper integrates Image based authentication and HMAC (Hash Message Authentication Code) based one-time password to achieve high level of security in authenticating the user over the internet. These algorithms are very economical to implement provided they are time synchronized with the user.

Sonal Fatangare and Archana Lomte [4] proposed a system in which an OTP user authentication protocol leverages a user's cell phone and short message service to resist password stealing and password reuse attacks. Through their system, users will have to only remember a long term password for login on all websites. It uses one time password strategy. Their Protocol is efficient and affordable compared to the conventional web authentication mechanism. Their design principle is to remove the negative influence of human factor as much as possible. It only requires each participating website possess a unique phone number and involves a registration and a recovery phase.

Neha Vishwakarma and Kopal Gangrade [5] proposed an image based time synchronized OTP generation method system that uses random image and text based OTP generation with SHA-512 algorithm and encryption by ECC method to produce a secured two factor, one time password.

III. PREREQUISITE TO THE PROPOSED METHOD

To implement the proposed method we need a 4 × 4 Rubik's cube which has to be tagged with printable characters present on a keyboard on each of its side faces.
The printable characters include the following:
10 Digits: 0123456789
26 Lower case letters: abcdefghijklmnopqrstuvwxyz
26 Upper case letters: ABCDEFGHIJKLMNOPQRSTUV WXYZ
34 special characters: `~!@#$%^&*()-_=+[]\{}|;':",./<>?
The following diagrams explain the tagging process:

1. We can start tagging with any side of the cube. Let us choose the side having green stickers on it and name it as Side - G. We begin tagging with the uppercase letters by allotting ABCD in the 1st row, EFGH in the 2nd row, IJKL in the 3rd row and so on, as depicted in Fig. 1.

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

Figure 1. Side – G

2. As all the 26 uppercase letters cannot be accommodated on one side face, we continue the tagging process by allotting the remaining uppercase letters on the side on its right edge, that is, the side having red stickers on it. We name this side as Side – R. After allotting the remaining uppercase letters, we are still left with 6 untagged boxes. To fill these boxes, we now begin tagging with the lowercase letters as depicted in Fig. 2.

Q	R	S	T
U	V	W	X
Y	Z	a	b
c	d	e	f

Figure 2. Side – R

3. Again, as all the 26 lowercase letters cannot be accommodated on Side - R, we continue the tagging

process by allotting the remaining lowercase letters on the side on its right edge, that is, the side having blue stickers on it. We name this side as Side – B as depicted in Fig. 3.

g	h	i	j
k	l	m	n
o	p	q	r
s	t	u	v

Figure 3. Side – B

4. After tagging Side – B with the remaining lowercase letters, we found that the last 4 letters – w, x, y and z could not be tagged on Side – B. So, we continued the tagging process by allotting the remaining 4 lowercase letters on the side on its right edge, that is, the side having orange stickers on it. We name this side as Side – O. After allotting the remaining lowercase letters, we are still left with 12 untagged boxes. To fill these boxes, we now begin tagging with the digits 0 to 9, followed by the special characters as depicted in Fig. 4.

w	x	y	z
0	1	2	3
4	5	6	7
8	9	`	~

Figure 4. Side - O

5. Again, as all the special characters cannot be accommodated on Side - O, we continue the tagging process by allotting the remaining special characters on the side on its upper edge, that is, the side having white stickers on it. We name this side as Side – W as depicted in Fig. 5.

!	@	#	$
%	^	&	*
(	)	-	_
+	=	[	]

Figure 5. Side – W

6. After tagging Side – W with the remaining special characters we found that all could not be accommodated there. So we continued the tagging process by allotting the remaining special characters on its opposite side, that is, the side having yellow stickers on it. We name this side as Side – Y as depicted in Fig. 6.

\	{	}	|
;	:	,	.
“	”	‘	’
<	>	/	?

Figure 6. Side - Y

A very important point to be noted from the above tagging process is, we can begin tagging from any side of a Rubik’s cube as mentioned earlier and that also with any set of characters (uppercase letters or lowercase letters or digits or special characters). Once we finish off tagging characters on one side, it is up to us to decide as to which side we should continue with, that is, the side on the left, right, top or bottom edge of the current side face of the cube, for tagging the remaining characters of a particular character set.

Also, once we finish off tagging characters of a particular character set and we are left with untagged boxes on that side face of the cube, it is again up to us to decide which character set, next, we should begin with tagging again. For instance, when we completed tagging the uppercase letters we found that there were 6 untagged boxes on Side – R, that time we had the option of taking up either lowercase letters or digits or special characters. We took up lowercase letters for tagging those boxes.

IV. TERMINOLOGIES FOR PROPOSED METHOD

Layers: Each row or column on any side face of a Rubik’s cube represents a layer.

Boxes: We will designate each and every square on all the side faces of a Rubik’s cube as boxes. The figure below illustrates it.

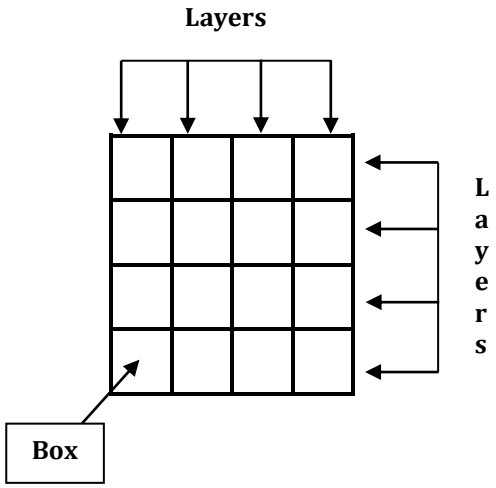

Figure 6. A side view of a Rubik’s Cube

V. PROPOSED METHOD

The study will need several random functions to be implemented on various parameters to determine how to scramble a Rubik’s cube to produce a 16 character OTP which will then be sent to users so that they can login to their accounts.

The parameters are:

Side Face: There are six side faces in a Rubik's cube. We need to determine the base side on which the scrambling will be done to produce the 16 character OTP.

Let R_{SF} be a random function which will determine the base side face by generating randomly an integer value, in the range from 1 to 6, where SF stands for Side Face. The values and the corresponding side faces they represent are given below:

- ✓ **1: Side – G (Green)**
- ✓ **2: Side – R (Red)**
- ✓ **3: Side – B (Blue)**
- ✓ **4: Side – O (Orange)**
- ✓ **5: Side – W (White)**
- ✓ **6: Side – Y (Yellow)**

Layers: Once the base side face is selected, we have to find out how many layers have to be rotated considering both the horizontal and the vertical layers. Again, for those layers that will rotate, we also have to find out, how many times each of these layers will rotate and in which direction.

For this to happen ,we must first label the layers as HL1, HL2, HL3, HL4 for the horizontal layers and VL1, VL2, VL3, VL4 for the vertical layers as depicted in Fig. 7.

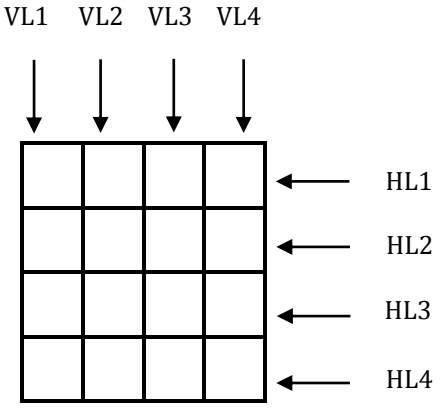

Figure 7. Labelling of layers of base side face

As there are 4 layers in each direction, we will iterate a random function, say R_{LM}, inside a loop 4 times with the layer position to generate an integer value, in the range from 0 to 1 where 1 denotes that the layer will rotate and 0 denotes that the layer will not rotate and also, a counter to keep count of how many layers will rotate. If at the end of the 3rd iteration, the counter has value 3, we will come out of the loop else continue with the iteration. This is done so that, the counter value never exceeds 3 as we would like to retain some of the characters of base side face before the scrambling took place. This process will repeat twice once for the horizontal layers and once for the vertical layers. Now, a situation arises where we have to decide which loop to start with, the loop for the horizontal layers or the loop for the vertical layers. So, we will use a random function R_{HV} to determine the layer direction by generating an integer value, in the range from 1 to 2 where 1 denotes horizontal layer and 2 denotes vertical layer. After the iterations with either of the horizontal or the vertical layer is over, we will reset the counter value to 0 for the next set of iterations with the other direction layers.

Also, inside the loop whenever R_{LM} generates 1 for a particular layer in a particular direction we should know which way it will rotate and how many times it will rotate. For instance, if the layer is a horizontal layer and R_{LM} generates 1 for it then, we have to find out which way it will rotate, that is, left to right or right to left and that also how many times. Similarly, if the layer is a vertical layer and R_{LM} generates 1 for it then, we have to find out which way it will rotate, that is, top to bottom or bottom to top and that also how many times.

Considering the above situation, we need 3 random functions to solve the problem. Let the random functions be denoted by R_{DLR}, R_{DTB} and R_{LMF} where, DLR stands for Direction Left Right, DTB stands for Direction Top Bottom and LMF stands for Layer Movement Frequency. When we will work with the horizontal layers, the random functions R_{DLR} and R_{LMF} will be used where R_{DLR} will determine whether a layer will rotate from left to right or right to left and R_{LMF} will determine, how many times a layer will rotate in a particular direction. Similarly, when we will work with the vertical layers, the random functions R_{DTB} and R_{LMF} will be used where R_{DTB} will determine whether a layer will rotate from top to bottom or bottom to top and R_{LMF} will determine, how many times a layer will rotate in a particular direction.

The values and the directions represented by the random functions R_{DLR} and R_{DTB} are as given below:

For R_{DLR} ,

- ✓ **1: Left to Right**
- ✓ **2: Right to Left**

For R_{DTB} ,

- ✓ **1: Top to Bottom**
- ✓ **2: Bottom to Top**

As far as, the function R_{LMF} is concerned, it will randomly generate integer values in the range from 1 to 3 to determine, how many times a layer will rotate in a particular direction.

The algorithm for the above proposed method is as follows:

Step 1: START

Step 2: Select a side face of a Rubik's cube to

generate an integer value using random function $\mathbf{R_{SF}}$.

Step 3: Generate an integer value using random function $\mathbf{R_{HV}}$ and store it in **RHV**.

Step 4: Initialize **count** with **0**.

Step 5: **If RHV = 1, then:**
Step 5a: Repeat Steps i, ii, iii **for I = 1 to 4 by 1:**
Step i: Generate an integer value using random function $\mathbf{R_{LM}}$ for **Layer I** and assign it to **RLM**.

Step ii: **If RLM = 1, then:**
Generating an integer value using random function $\mathbf{R_{DLR}}$ and assign it to **RDLR**.

If RDLR = 1, then:
Generate an integer value using random function $\mathbf{R_{LMF}}$ and assign it to **RLMF**.
Rotate the **Layer I RLMF** times from left to right.
Else
Generate an integer value using random function $\mathbf{R_{LMF}}$ and assign it to **RLMF**.
Rotate the **Layer I RLMF** times from right to left.
End If

Increment **count** by **1**.
End If

Step iii: **If count = 3, then:**
Break out of the Loop
End If
End For

Step 5b: Reset **count** to **0**

Step 5c: Repeat Steps i, ii, iii **for J = 1 to 4 by 1:**
Step i: Generate an integer value using random function $\mathbf{R_{LM}}$ for **Layer J** and assign it to **RLM**.

Step ii: **If RLM = 1, then:**
Generating an integer value using random function $\mathbf{R_{DLR}}$ and assign it to **RDTB**.

If RDTB = 1, then:
Generate an integer value using random function $\mathbf{R_{LMF}}$ and assign it to **RLMF**.
Rotate the **Layer J RLMF** times from top to bottom.
Else
Generate an integer value using random function $\mathbf{R_{LMF}}$ and assign it to **RLMF**.
Rotate the **Layer J RLMF** times from bottom to top.
End If

Increment **count** by **1**.
End If

Step iii: **If count = 3, then:**
Break out of the Loop
End If
End For

Else If

Step 5a: Repeat Steps i, ii, iii **for J = 1 to 4 by 1:**
Step i: Generate an integer value using random function $\mathbf{R_{LM}}$ for **Layer J** and assign it to **RLM**.

Step ii: **If RLM = 1, then:**
Generate an integer value using random function $\mathbf{R_{DLR}}$ and assign it to **RDTB**.

If RDTB = 1, then:
Generate an integer value using random function $\mathbf{R_{LMF}}$ and assign it to **RLMF.**
Rotate the **Layer J RLMF** times from top to bottom.
Else
Generate an integer value using random function $\mathbf{R_{LMF}}$ and assign it to **RLMF**.
Rotate the **Layer J RLMF** times from bottom to top.
End If

Increment **count** by **1**.
End If

Step iii: **If count = 3, then:**
Break out of the Loop
End If
End For

Step 5b: Reset **count** to **0**

Step 5c: Repeat Steps i, ii, iii **for I = 1 to 4 by 1:**
Step i: Generate an integer value using random function $\mathbf{R_{LM}}$ for **Layer I** and assign it to **RLM**.

Step ii: **If RLM = 1, then:**
Generate an integer value using random function $\mathbf{R_{DLR}}$ and assign it to **RDLR**.

If RDLR = 1, then:
Generate an integer value using random function $\mathbf{R_{LMF}}$ and assign it to **RLMF**.

Rotate the **Layer I RLMF** times from left to right.
Else
Generate an integer value using random function $\mathbf{R_{LMF}}$ and assign it to **RLMF**.
Rotate the **Layer I RLMF** times from right to left.
End If

Increment **count** by **1**.
End If

Step iii: **If count = 3, then:**
Break out of the Loop
End If
End For
End If

Step 6: Generate OTP by writing the characters on the base side face of the cube in row-order format. That is, **HL1** followed by **HL2** then **HL3** and finally **HL4**.

Step 7: STOP

VI. EXPERIMENTAL RESULTS

The proposed algorithm was tested to find out what type of OTP it generates. The Table I given below gives a summarized result of the testing:

TABLE I. RESULTS OF THE STUDY

RSF	BASE SIDE	R_{HV}	LAYERS	R_{LM}	R_{DLR}	R_{DTB}	R_{LMF}	GENERATED OTP
1	GREEN	1	HL1	1	1	NA	2	ghijEFGH4567wxyz
			HL2	0	NA		NA	
			HL3	1	2		3	
			HL4	1	1		1	
			VL1	0	NA	NA	NA	
			VL2	0		NA	NA	
			VL3	0		NA	NA	
			VL4	0		NA	NA	
2	RED	2	VL1	1	NA	1	2	gh#Dkl&Hop-Lst[P
			VL2	1		2	2	
			VL3	1		2	3	
			VL4	0		NA	NA	
			HL1	0	NA	NA	NA	
			HL2	0	NA		NA	
			HL3	0	NA		NA	
			HL4	0	NA		NA	

VII. CONCLUSION & FUTURE WORK

This paper proposes a novel approach for generating One-Time Password (OTP) using Rubik's cube principle on a 4×4 cube in which the layers of the cube are scrambled in various ways to produce a 16 character OTP. This technique using Rubik's cube have never been applied before to generate an OTP. Experimental results reveal that the generated 16 character OTP is quite effective.

Future improvements to the proposed method will focus on reducing the size of the OTP and optimizing the algorithm to increase its efficiency.

REFERENCES

[1] Saini T., "One Time Password Generator System", International Journal of Advanced Research in Computer Science and Software Engineering, Vol.**4(3)**, pp.**781–785**, March **2014.**

[2] Huang Y., Huang Z., Zhao H., Lai X., "A new One-time Password Method", In the Proceedings of 2013 International Conference on Electronic Engineering and Computer Science , pp.**32–37**, **2013**.

[3] Parmar H., Nainan N. and Thaseen S., "GENERATION OF SECURE ONE-TIME PASSWORD BASED ON IMAGE AUTHENTICATION", In the Proceedings of Academy & Industrial Research Collaboration Center - Computer Science Conference Proceedings, pp.**195- 206**, **2012**.

[4] Fatangare S., Prof. Lomte A., "Robust OTP Generation Using Secure Authentication Protocol", International Journal of COMPUTER TECHNOLOGY AND APPLICATIONS, Vol.**5(2)**, pp.**546-552**, March – April **2014.**

[5] Vishwakarma N., Gangrade K., "Secure Image Based One Time Password", International Journal of Science and Research, Vol.**5(11)**, pp.**680–683**, November **2016.**

Authors Profile

Mr. Subhajit Bose completed Bachelor of Computer Application from Information Technology Centre, University of North Bengal in 2008 and Master of Computer Application from Department of Computer Science & Application, University of North Bengal in 2011. He is currently working as Assistant Professor in Department of Computer Science & Application, Salesian College. His main research work focuses on Artificial Intelligence, Data Mining, Cryptography algorithms. He has 5 and half years of teaching experience.
email: bosesubhajit87@gmail.com

Dr. Dilip Roy Chowdhury is working as Asst. Professor in the Department of Computer Science and Application at the University of North Bengal. Before that, he has served Dept. of Computer Science & Application, Gyan Jyoti College, as HOD and took other administrative responsibilities along with regular teaching job. Dr. Roy Chowdhury's main research interests lies with the design and implementation of Expert System Development using Artificial Intelligence and Soft Computing techniques. Besides, he is continuing his research in the fields of Artificial Neural Networking, Data Mining, Rough Set Computing, Knowledgebase Design and Information Retrieval. He has more than 40 research publications in various national and international journals of repute. He is associated with various national and international journals of repute as a member of reviewing committee and editorial committee.email: diliproychowdhury@gmail.com

.

