International Journal of Computer Applications (0975 – 8887)
Volume 55– No.6, October 2012

Dynamic Mobile Token for Web Security using MD5 and One Time Password Method

I Gede N. Agung Jayarana
Department of Information Technology Udayana University, Bali, Indonesia

A. A. Kt. Agung Cahyawan
Department of Information Technology Udayana University, Bali, Indonesia

Gusti Made Arya Sasmita
Department of Information Technology Udayana University, Bali, Indonesia

ABSTRACT

Nowadays, websites have been converted from just a display of information into an online transaction in the form of goods, services, or money. With that development, the security of the website is also need to be tightened, not just rely on usernames and passwords but also to the dynamic code of the mobile token which is difficult to be cracked [1].

Dynamic mobile token is an application which is planted in the mobile phone to generate a code that was formed by the method of one time password and can only be used for one login session or transaction. Each mobile token has a value in the "secret" variable that makes it unique or different from the others, to separate one user access transactions or personal page.

General Terms

Website Security, Mobile.

Keywords

One Time Password, MD5 Algorithm, Website Security, Online Transaction Security, Mobile Token.

1. INTRODUCTION

One time password (OTP) is a password that is only valid for a single login session or transaction [13]. OTP is widely used as a password that is not planted in the database, but only as a single use password and immediately forfeited. The benefit of the OTP is located on the different application with a static password which is planted in the database. The use of encrypted static passwords are also not immune from the attack by using a key logger [2] or sort of it, because if an attacker managed to get the main password and OTP password still login and transactions will not be processed because the password is no longer valid. Code generation as encryption is using Message-Digest Algorithm 5 (MD5) which are widely used with 128-bit hash value, this algorithm has been widely used for security applications, password encryption, and integrity test of a file [3, 14].

The application of Dynamic Mobile Token uses three codes consisting of epoch time as the key of one time password, the value of the “secret” variable in which each user has a different value so that when it degenerate at the same time, it will result in different value, and 4 digit random value between 1000 and 9999 resulting from the website. These three values are then combined and encrypted with md5 algorithm to generate the output of the value of 128 bits or 32 hexadecimal numbers. Only first 6 digits of the hexadecimal number are used from the result of the output.

The level security of the password is good enough to double the security in an account and the login process, because each password OTP is only valid once and if you do any mistake, the code generated from the website will change anyway. The time period of the password’s life span is 180 seconds, the time to break the OTP password in ratio is 16^6 = 16,777,216 possibilities in a single input of passwords.

2. MOBILE TOKEN CONCEPT

Dynamic Mobile Token has a concept to secure online transactions. Mobile Token becomes an additional factor in the authentication process, to prove that the user who do the login session or transaction process is a legitimate user.

2.1 Authentication Method

The aim of authentication is to prove that the accessing user is the real user. There are many methods that can prove it, but for authentication methods can be seen in the three categories of methods:

1. **Something You Know**
   It is the most common authentication method. This method is relying on the confidentiality of information, such as passwords and PIN. This method assumes that no one knows the secret unless the user itself.

2. **Something You Have**
   This is usually an additional factor to create a more secure authentication. This method relies on items which usually are unique, for examples, the magnetic card/smartcard, hardware tokens, USB tokens, and else. This method assumes that no one has the hardware unless the user itself.

3. **Something You Are**
   This is the most rarely used method because of technology and the human factor as well. This method relies on the uniqueness of the body parts that is not exist in others such as fingerprint, voice, retina or fingerprint. This method assumes that the parts of the body such as fingerprints and retina are different with others.

2.2 Password Mode

Dynamic Mobile Token there are two mode used [4, 12] :

1. **Challenge/Response Mode (C/R)** [9]
   This mode is most often used when doing transaction. In this mode the server provides a challenge in the form of a series of numbers. That number must be entered into the Mobile Token to get an answer (response). Then the user enters the number that appears on its own Mobile Token into text box on the website. Mobile Token will issue a different code though with the same code

challenge. Periodically depending on the time when we answer the challenge in a token.

2. **Self Generated Mode (Response Only)**
   In this mode the server does not give any kind of value (challenge). Mobile Token users can directly issue a series of combination of numbers and letters without having to enter the challenge. As the mode C/R, Mobile Token also issued different codes periodically depending on the time when the token is ordered to produce self-generated code.

2.3 Security Level

At the actual level of security in C/R and Self Generated (Response Only) mode is nothing but the password as well. However, it is different from the password used to login, passwords from Mobile Token has limitation for security reasons, namely:

1. **May only be used one time**
   Once a password is used, the same password can no longer be used for the second time. With this way, there is no point to intercept the degenerated passwords of Mobile Token because the password cannot be used again. However, if the password is managed to be intercepted so it never gets to the server, it is still a valuable password as the server password has not been used.

2. **May only be used within a limited time span**
   Mobile Token generated passwords have a very limited life, probably between 2-3 minutes. When the age expires, the password cannot be used anymore, although it has never been used. Time is a very critical element in this system of Mobile Token

3. **May only be used in the narrow context**
   If the password / PIN used for - login is a free context password, in the sense that with the password only, it can do many things, from seeing balances, check transaction and else. But the token generated password can only be used in a narrow context, for example, the password that is used to buy a credit to the number 08123456789, is cannot be used to transfer funds.

3. GENERAL DESIGN SYSTEM

3.1 Authentication Process

Such as passwords in general, on condition that authentication successful is the password that is sent to the client is the same passwords stored on the server.

With security reasons rarely server stores user passwords in plaintext form. Commonly, server stores user passwords in hash form so it cannot be returned in plaintext form. So successful authentication requirements can be interpreted as the result of the calculation hash of the password sent by the client must be the same with the hash value stored in the server (see Figure 1).

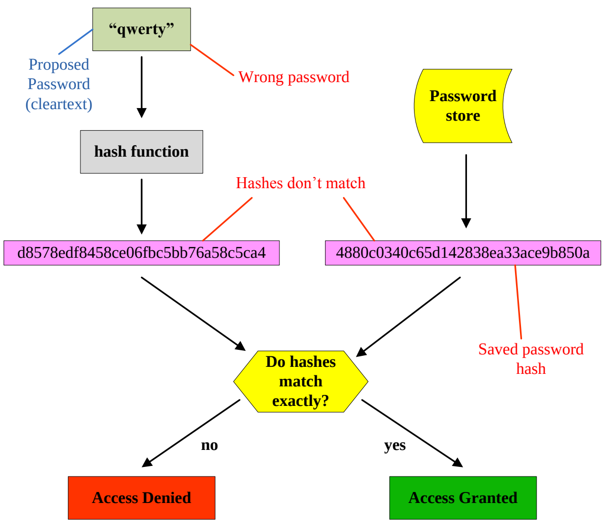


Figure 1: Hash Password Authentication Process [5]

Mobile Token authentication process on the server has a very critical time distinction. Users have to match the clock in the phone with the clock on the server, the difference in hours that allowed is less than 3 minutes and more than 3 minutes, more than that is considered wrong or the code has been used.

The chain process of the connection system between the server, client, user, Mobile Token, and the website has a strong connection and cannot be separated. If one system crash, the other series cannot be started or processed [11] (see Figure 2).

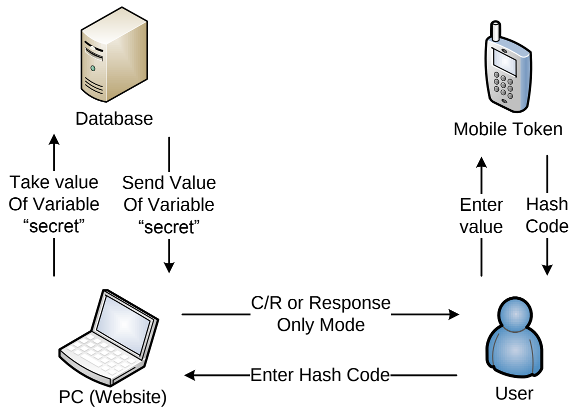

Database
Take value Of Variable "secret"
Send Value Of Variable "secret"
Mobile Token
Enter value
Hash Code
C/R or Response Only Mode
Enter Hash Code
PC (Website)
User

Figure 2: Connection Process in General

3.2 One Time Password Process

To avoid brute-forces [6] attacks to the hash value stored in the server, before the user's password is generating its hash value, firstly, add a random string called the salt. In the program, the value of salt is the value of the variable in the Mobile Token secret and also planted in the database. For example, if the user's password is "qwerty", before it generates its hash value, password is added salt in the form value of the secret variable for example "7fc04db" [10]. So the hash value will be calculated three values combined into "qwerty7fc04db" not only "qwerty".

When viewed from the MD5 value "qwerty7fc04db" is "77ed461ee664d2bb7ab75c16f338e943", while the value of MD5 original password without the combination "qwerty" is "d8578edf8458ce06fbc5bb76a58c5ca4". If the passwords do not use salt, then the attacker can easily decrypt the password using a brute force attack or rainbow table to get the value of the password in plain-text, since the site to store database decrypted already widely available. And it is impossible for attacker to build a database for mapping between the plaintext and hash completely.

In Self Generated code generation mode with Mobile Token, the server must be able to give toleration of time, because the time on the server and also Mobile Token is not always entirely correct. This time gap between the server's request for a password to the user generating the value of Mobile Token. Steps which have to be taken a note, those are:

1. Seconds when the server request for a password (OTP) from the user.
2. Seconds when Mobile Token is generating OTP.
3. Seconds when the server receives the OTP from the user

If you see above events then the important thing to note is a time gap between events 1, 2, and 3. For example, assume that time server has an exact time with a Mobile Token, if in the 0 second the server asks for a password from the user, because of the slow internet access, it is on 30th seconds the user be able to look at the browser that he must enter the OTP from the token. Later, on the 60th minute the token generates OTP. In the 65th second, the user submits the OTP value to the server and will be received by the server at the 90th second. To overcome the differences in generation time to the time of submit the OTP value, then, the way to do is examine all OTP possibilities in the time duration that is considered to be adequate, for example 180 seconds.

If the system uses granularity 10 seconds then the server must calculate the OTP value starts from the 0, 10, 20, 30, 40, to 180 second in increments of 10 seconds (see Table 1). In this system, the generated OTP is in the form of 32 characters, and is only used 6 first characters of combined MD5. In doing authentication, the server should be comparing all OTP values from the 0 second to 180 seconds or up to a maximum tolerance. In this example the value of the epoch [7] 0 seconds is EPOCH/10 = 134589691.

Table 1. Authentication Result of OTP Self Generated Mode

Second	EPOCH/10	Combination	OTP
0	134589691	7fc04db134589691	7e8970
10	134589692	7fc04db134589692	501e90
20	134589693	7fc04db134589693	672eb5
30	134589694	7fc04db134589694	aae1e7
40	134589695	7fc04db134589695	54358e
50	134589696	7fc04db134589696	1d605d
60	**134589697**	**7fc04db134589697**	**8f3e13**
70	134589698	7fc04db134589698	cb4756
80	134589699	7fc04db134589699	6114ac
90	134589700	7fc04db134589700	39e6f0
100	134589701	7fc04db134589701	ea3b7f
110	134589702	7fc04db134589702	36705b
120	134589703	7fc04db134589703	913a48
130	134589704	7fc04db134589704	6e5673
140	134589705	7fc04db134589705	a3c438
150	134589706	7fc04db134589706	182f7b
160	134589707	7fc04db134589707	298420
170	134589708	7fc04db134589708	1a5e1a
180	134589709	7fc04db134589709	177e2b

In Table 1, the user sends OTP "8f3e13" then the authentication will be successful when the server calculates the OTP value on the 60 second since the server request OTP from the user, but in fact there is the possibility of time between server and Mobile Token is not exactly 100% so that the server had to give a time tolerance not only forward, but also backwards. Because it could be the time at the server is faster than the time on the token. For example, when the time on the server shows EPOCH/10 = 134589709, it could be time token shows EPOCH/10 = 134589699 (Mobile Token's time is late 100 seconds).

If a time tolerance is 2 minutes (120 seconds), the server must provide a tolerance of next 2 minutes and 2 minutes to the before, relatively to the time when server receives the OTP from the user and doing authentication. So, if a server authenticates the EPOCH/10 = 1000, then the server must calculate the entire value of OTP, starts from EPOCH/10 = 880 until EPOCH/10 = 1120.

Generation and authentication in the Challenge/Response mode (C/R) is actually similar to the mode Self Generated. When the self-generated mode has additional salt from the epoch value, on the C/R mode, it is more salt. Combination is not only added with the secret variable but also added with the challenge. In Table 2 servers perform OTP calculations for the 0 second to 180 seconds with a 10 second granularity.

It is assumed that the resulting Challenge value is 7777 with a secret variable "7fc04db".

Table 2. Calculation Result of OTP C/R Mode

Second	EPOCH/10	Combination	OTP
0	134589691	7fc04db1345896917777	b1db26
10	134589692	7fc04db1345896927777	360bac
20	134589693	7fc04db1345896937777	f0c2fd
30	134589694	7fc04db1345896947777	488e38
40	134589695	7fc04db1345896957777	09b65d
50	134589696	7fc04db1345896967777	f706dd
60	134589697	7fc04db1345896977777	3451c3
70	134589698	7fc04db1345896987777	d9c456
80	134589699	7fc04db1345896997777	d9873c
90	134589700	7fc04db1345897007777	86970c
100	134589701	7fc04db1345897017777	fe427b
110	134589702	7fc04db1345897027777	321768
120	134589703	7fc04db1345897037777	fd2e8e
130	134589704	7fc04db1345897047777	1a0346
140	134589705	7fc04db1345897057777	6f6d99
150	134589706	7fc04db1345897067777	8b2beb
160	134589707	7fc04db1345897077777	a6500e
170	134589708	7fc04db1345897087777	f94ee4
180	134589709	7fc04db1345897097777	1da125

On the C/R mode, there is an additional field which has to be joined before calculating its hash value, the challenge. Challenge value is known by the server and also by Mobile Token (when users type challenge to Mobile Token), so that both token and the server will be able to calculate the same OTP and proceed the authentication process.

3.3 Program Structure

To the application of Dynamic Mobile Token, it has uniqueness on its secret variable. These variables are planted in the main class, to create a permanent secret value, the function fopen() in PHP is used to create a java file. Then the fwrite() function is used to write a series of program code. The making of secret variables such as the following examples on one line of PHP.

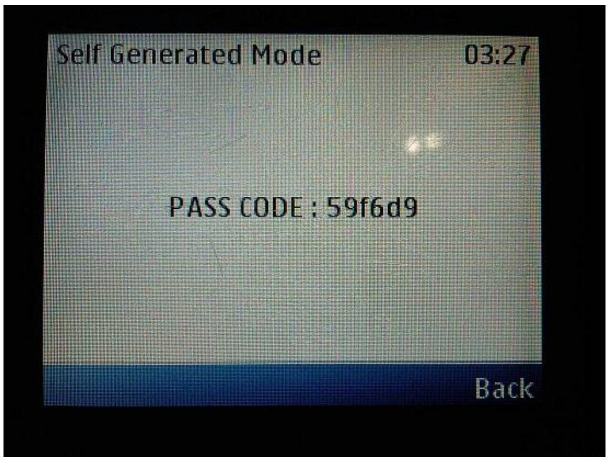
```
$myFile = "MobileToken.java";
$fh = fopen($myFile, "w") or die("can't open file");

fwrite($fh, "int secret = $_SESSION[secret_id];\r\n");
```

The code program above works to create the variable "secret" has a different value from any Mobile Token. If the file has been created, the next file you process is md5.java, this file is created using the same function fopen() to create a java file and fwrite() to write a series of md5 algorithm on it.

4. IMPLEMENTATION AND RESULT

In this implementation, the example used is a type of Nokia C3 mobile phone with the time difference between the server and the Mobile Token is about 10 seconds.



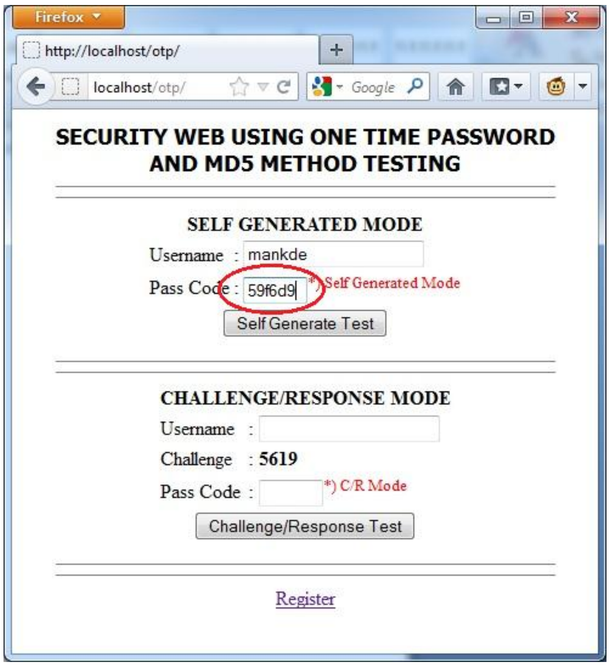
Self Generated Mode 03:27

PASS CODE : 59f6d9

Back

Figure 3: Pass Code in Self Generated Mode on the Mobile Token

Figure 3 shows the Mobile Token generate random numbers with Self Generated Mode. Pass Code then entered into the website in order to avoid the possibility of life span of OTP have been expired (see Figure 4).



SECURITY WEB USING ONE TIME PASSWORD AND MD5 METHOD TESTING

SELF GENERATED MODE

Username : mankde

Pass Code : 59f6d9 *) Self Generated Mode

Self Generate Test

CHALLENGE/RESPONSE MODE

Username :

Challenge : **5619**

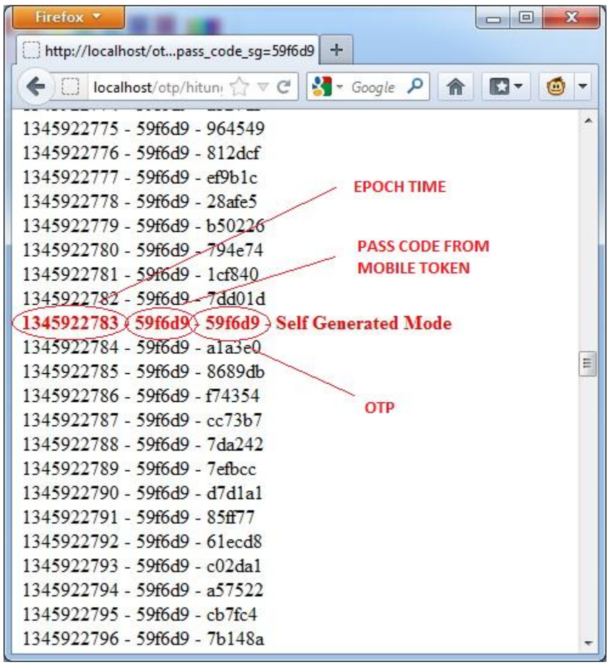
Pass Code : *) C/R Mode

Challenge/Response Test

Register

Figure 4: Entering the Pass Code on the Self Generated Text Box

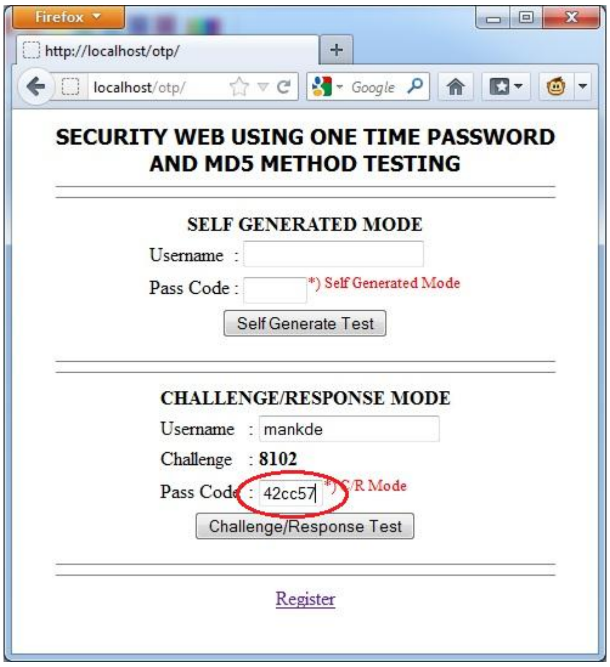
The results of Pass Code generation after checking the time in the period up to EPOCH-180 to EPOCH+180 to avoid any differences of Mobile Token and server, more or less than 180 seconds (see Figure 5).


Firefox
http://localhost/ot...pass_code_sg=59f6d9
localhost/otp/hitun
Google
EPOCH TIME
PASS CODE FROM MOBILE TOKEN
OTP

```
1345922775 - 59f6d9 - 964549
1345922776 - 59f6d9 - 812dcf
1345922777 - 59f6d9 - ef9b1c
1345922778 - 59f6d9 - 28afe5
1345922779 - 59f6d9 - b50226
1345922780 - 59f6d9 - 794e74
1345922781 - 59f6d9 - 1cf840
1345922782 - 59f6d9 - 7dd01d
1345922783 - 59f6d9 - 59f6d9 Self Generated Mode
1345922784 - 59f6d9 - a1a3e0
1345922785 - 59f6d9 - 8689db
1345922786 - 59f6d9 - f74354
1345922787 - 59f6d9 - cc73b7
1345922788 - 59f6d9 - 7da242
1345922789 - 59f6d9 - 7efbcc
1345922790 - 59f6d9 - d7d1a1
1345922791 - 59f6d9 - 85ff77
1345922792 - 59f6d9 - 61ecd8
1345922793 - 59f6d9 - c02da1
1345922794 - 59f6d9 - a57522
1345922795 - 59f6d9 - cb7fc4
1345922796 - 59f6d9 - 7b148a
```

Figure 5: Password Calculation Result on Self Generated Mode

Challenge/Response mode (C/R) is not much different from its application to the Self Generated Mode. Mobile Token can generate Pass Code if it gets challenge from the server. Figure 6 shows the challenge generated code is “8102”, then the challenge code is entered into the Mobile Token (see Figure 7) and the result is a Pass Code (see Figure 8) is entered into the text box pass code.


SECURITY WEB USING ONE TIME PASSWORD AND MD5 METHOD TESTING

SELF GENERATED MODE

Username :

Pass Code : *) Self Generated Mode

Self Generate Test

CHALLENGE/RESPONSE MODE

Username : mankde

Challenge : 8102

Pass Code : 42cc57 *) C/R Mode

Challenge/Response Test

Register

Figure 6: Challenge Value and Pass Code Input on Challenge / Response Mode

Challenge Value randomly generated between 1000 to 9999. When the value of Pass Code on Mobile Token is entered into the website and the submit button is pressed, then the counting starts with matching the Pass Code with a value of OTP

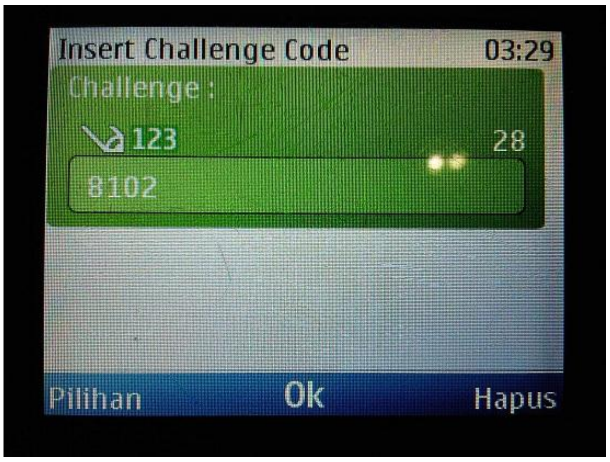


Figure 7: Challenge code is inputted on the Mobile Token

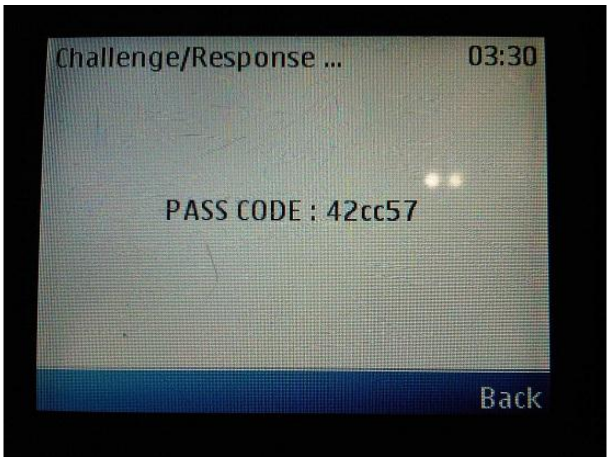


Figure 8 : Pass Code in Challenge/Response on Mobile Token

The results of the code generation Challenge/Response mode is then calculated, the Pass Code compatibility with the OTP. C/R Mode also calculates OTP of EPOCH-180 to EPOCH+180 (see Figure 9).

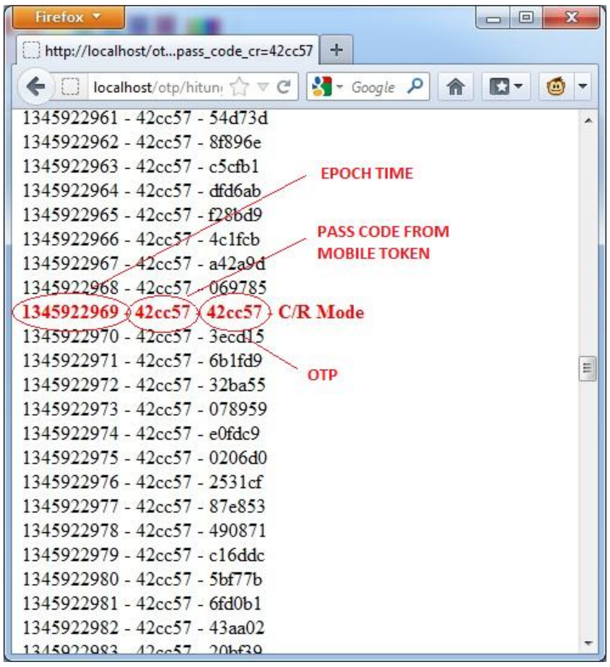
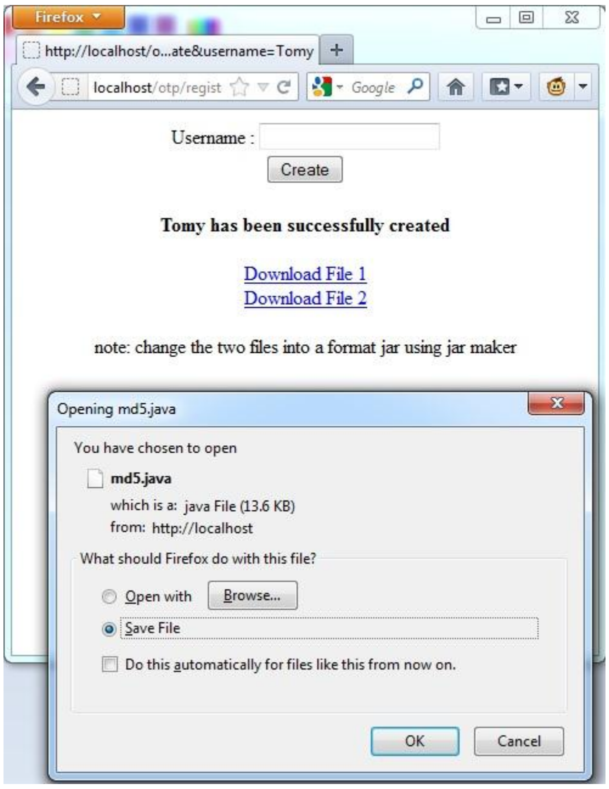

```
1345922961 - 42cc57 - 54d73d
1345922962 - 42cc57 - 8f896e
1345922963 - 42cc57 - c5cfb1
1345922964 - 42cc57 - dfd6ab
1345922965 - 42cc57 - f28bd9
1345922966 - 42cc57 - 4c1fcb
1345922967 - 42cc57 - a42a9d
1345922968 - 42cc57 - 069785
1345922969 - 42cc57 - 42cc57   C/R Mode
1345922970 - 42cc57 - 3ecd15
1345922971 - 42cc57 - 6b1fd9
1345922972 - 42cc57 - 32ba55
1345922973 - 42cc57 - 078959
1345922974 - 42cc57 - e0fdc9
1345922975 - 42cc57 - 0206d0
1345922976 - 42cc57 - 2531cf
1345922977 - 42cc57 - 87e853
1345922978 - 42cc57 - 490871
1345922979 - 42cc57 - c16ddc
1345922980 - 42cc57 - 5bf77b
1345922981 - 42cc57 - 6fd0b1
1345922982 - 42cc57 - 43aa02
1345922983 - 42cc57 - 20bf39
```
Firefox
http://localhost/ot...pass_code_cr=42cc57
localhost/otp/hitun
Google
EPOCH TIME
PASS CODE FROM MOBILE TOKEN
OTP

Figure 9: Password Calculation Result on Challenge/Response Mode

In addition, there is a register link at the bottom of the website (see Figure 4 or Figure 6) to create a new username. When the newly username created, secret variable is automatically made randomly by checking first to the database so that there is no same secret value (see Figure 10).



Username :

Tomy has been successfully created

note: change the two files into a format jar using jar maker

Opening md5.java

You have chosen to open

md5.java

which is a: java File (13.6 KB)
from: http://localhost

What should Firefox do with this file?

Open with
[✓ Save File]
Do this automatically for files like this from now on.

Figure 10: Register Menu

In Figure 10, when the username "Tomy" is made. The system will automatically create a java file where there are two files, md5.java and Token.java. Variables secret is planted in the Token.java file, the user must change the files using the jar maker because PHP does not support in compiling java.

5. CONCLUSION

Dynamic Mobile Token based on the previous explanation and application is helping much in the security level of a website. In addition, in its application, it can be applied to any website because the program is made as simple as possible.

Security created by the Mobile Token and OTP are able to deal with key logger attack, brute force attack, and other attacks, because the Pass Code is generated in the form of dynamic passwords with salt, such as epoch time and "secret" variables [10].

This application emphasizes how a Mobile Token can be run without the initial authentication process first, so the process can be shortened to avoid the difficult process of authentication [12].

6. ACKNOWLEDGMENTS

Our thank goes to Department of Information Technology Udayana University, Bali, Indonesia who has helped organize this research.

7. REFERENCES

[1] Kurniawan, Yusuf Ir. MT. 2004. KRIPTOGRAFI Keamanan Internet dan Jaringan Komunikasi. Informatika.

[2] Kiddo. 2010. Hacking Website: Menemukan Celah Keamanan & Melindungi Website dari Serangan Hacker. Mediakita

[3] Rivest, Ronald L. 1992. The MD5 Message Digest Algorithm.

[4] N. Haller, Bellcore, and C. Metz. 1996. A One-Time Password System. Kaman Sciences Corporation.

[5] Friedl, Sthepen J. 2005. An Illustrated Guide to Cryptographic Hashes. http://www.unixwiz.net/techtips/iguide-crypto-hashes.html.

[6] J. Black, M. Cochran, T. Highland. 2006. A Study of the MD5 Attacks: Insights and Improvements.

[7] Unix Time. http://en.wikipedia.org/wiki/Unix_time

[8] Cheng Fred. July 2010. A Secure Mobile OTP Token. International Technological University

[9] Arya Sapoetra Y. June 2010. Rancang Bangun Arsitektur Library Sistem Autentikasi One Time Password Menggunakan Prosedur Challenge-Response. Informatics Engineering, Pembangunan Nasional "Veteran" University, East Java.

[10] Sharma Nidhi, Sharma Alok, and Sharma Monika. 2011. A More Secure Hashing Scheme for Information Authentication. Proceedings published in International Journal of Computer Applications (IJCA).

[11] Mohan R, Partheeban N. April 2012. Secure Multimodal Mobile Authentication Using One Time Password. International Journal of Recent Technology and Engineering (IJRTE), Volume-1, Issue-1.

[12] Fadi Aloul, Syed Zahidi, Wassim El-Hajj. 2009. Two Factor Authentication Using Mobile Phones. Digital Library Telkom Institute of Technology (IEEE).

[13] Dr. Mark D. Bedworth PhD BSc FSS. February 2008. A Theory of Probabilistic One-Time Password. Computer Science Computer Engineering and Applied Computing, Security and Management.

[14] MD5 Algorithm. http://en.wikipedia/wiki/MD5