# A Secure Mobile OTP Token

Fred Cheng

International Technological University and FPC Consultancy,
Los Altos Hills, California USA
fredtcheng@yahoo.com

**Abstract.** Implementing a mobile One-time Password (OTP) Token on a cellular phone is a hot topic since the past few years. The proposed solutions had made certain improvements on network security. But none of them can fully prevent the OTP seed (K) tracing from MIMT OTP code interception or Shoulder-surfing security attacks while also meet the following criteria – fully compliant with existing authentication systems, inter-operable with other token and easy to deploy or support. This paper presents a cipher called Rubbing Encryption Algorithm (REAL) and the implementation of a Mobile OTP Token using this algorithm. The newly designed REAL Mobile OTP Token addresses and improves the aforementioned issues successfully.

**Keywords:** One-time Password, OTP Token, Authentication, Encryption Algorithm, MITM Attack, Shoulder-surfing Attack, Security Attack.

## 1 Introduction

One-time Password (OTP) Token can automatically generate a series of dynamic password. It has gained a leading position in the Two-factor Authentication (2FA) system for better network security. As the cellular phone became popular in the past few years, many solutions were proposed to embed the OTP Token inside such mobile device [1][2][3]. But they encountered certain deficiencies such as the mobile token can not fully resist OTP seed (K) tracing by Man-in-the-middle (MITM) OTP code interception [23] and Shoulder-surfing [24] security attack. Other issues include poor interoperability and compliance with existing authentication systems, plus higher deployment and support cost. In particular, many proposals store the OTP generation seed and personal secrecies inside the cellular phone. It compromises network security when the phone is lost or stolen [4].

To address the aforementioned issues, we introduce a Rubbing Encryption Algorithm (REAL) in this paper. A user does not need to memorize and enter the key when decrypts a ciphertext by REAL. This special feature allows REAL to use a very long and complex key for encryption. So REAL can securely encrypt a short word length OTP code with very high security level. That is why the locally stored REAL OTP codes can retain its security even if the phone is lost or stolen. A user can lay the hardware token over the REAL ciphertext image on a cellular phone's screen to electronically "rub" (decrypt) the OTP code. This is why the cipher gets its name – "Rubbing Encryption ALgorithm" (REAL).

We also present the design of a REAL Mobile OTP Token. This token is compliant to the OTP token proposed by the Initiative for Open AuTHentication (OATH) [6]. A cellular phone is used as the OTP generating platform. Such token is fully interoperable with the existing authentication system. A low cost plastic card is used as the REAL key (code pointer) bearing hardware token. OTP codes are pre-generated using the OATH's event-based OTP code algorithm [5]. The codes are encrypted by a specific REAL key assigned to a user. The confidential REAL encrypted codes are stored in the Data File. The OTP generation program and each user's Data File are provisioned and downloaded through the Internet. After installation, the user activates the OTP program. User lays her hardware token over the REAL ciphertext image on the phone's screen to obtain the OTP code. The user then enters this OTP code into the login window to complete the 2FA process.

Each REAL hardware token carries one key on each side of the token. Three versions of the REAL Mobile OTP Token are implemented. The basic version works with codes from just one OTP generating seed. It provides the basic secure OTP code. An improved version works with OTP codes from two different OTP generating seeds. The token matches the two code generation seeds with the authentication server automatically. So a valid OTP code will not be mistakenly rejected. This token can resist attack on OTP seed (K) tracing by MITM data interception attack. The third token is similar to the second version but the REAL key is dynamically decoupled from the code pointer's physical locations on the hardware token. This version can resist both the Shoulder-surfing and the seed tracing from MITM interception attacks.

We organize this paper as follows. In section 2, we briefly provide the background and related work. The Rubbing Encryption Algorithm (REAL) is introduced in Section 3. The implementation of the REAL Mobile OTP Token is discussed in Section 4. We then review the design goals, analyze the security attacks and other security concerns in Section 5. We further conclude this paper and possible improvement work in Section 6.

## 2 Background and Related Work

Several implementations have emerged as the key methods to use a cellular phone in remote network authentication. One such solution focuses on using cellular phone as a standalone OTP token [1][7][8]. The mobile device is used as a computational platform to generate OTP code. These tokens usually do not have any capability to resist the OTP seed (K) tracing by MITM interception and Shoulder-surfing attacks. It stores the secret seed (K) and counter value in the phone. So the network security may be comprised if the phone is lost or stolen as the secrecies can be exposed.

An alternative proposal focuses on using the cellular network as a secure out of band channel to transmit or receive the OTP code to or from the authentication server [2][9][10]. The OTP code is transmitted as an image data or through the Short Message Service (SMS) in text form. This new channel effectively prevents the traditional MITM attack. But Shoulder-surfing attack is still an issue. Cellular QoS (quality of service) will affect the reliability of OTP generation when using cellular SMS. SMS is a best effort delivery service. Cellular service providers cannot guarantee a real-time or in-time delivery. Moreover, when a user is out of the cellular service coverage area, such as in a basement of the building or in the rural area, using SMS sometimes is not even possible. Besides, new software and hardware are needed to allow the authentication server to interface to SMS system. All these increase the total system cost and complicate the server management task.

The third and most recent approach involves using Subscriber Identity Module (SIM) on a cell phone and other newly proposed protocols such as the Liberty Alliance Federation Standard [9] or The Free Auth Project [10] to perform the authentication procedure [3] [13][14][15][16][17][18]. In this scheme, the mobile device also carries part of the authentication secret information. The authentication is carried out directly through the phone, cellular network to the remote server. Again, QoS of authentication are limited by cellular service coverage area. In particular, using different authentication protocols and OTP algorithms usually leads to a new authentication system. New software and hardware are required at server to implement such scheme as the proposed solutions are not fully compliant with the existing authentication servers. Though this approach may prevent the traditional MITM attack, the Shoulder-surfing attack may still be an issue.

Overall, a standalone cellular phone-based Mobile OTP Token has its merits. It can be easily implemented to have full compliance with the existing infrastructure. No additional cost and server work are required for such token. Its usage is also not limited by the cellular service's coverage. But we need to resolve the security issues associated with this approach. This is exactly what the REAL Mobile OTP Token sets out to do.

## 3 The Cipher System

The Rubbing Encryption Algorithm (REAL) operates through multiple steps to ensure the desired data security. REAL's general theory and its operation procedures are presented in the following subsections.

### 3.1 REAL General Theory

Using Shannon entropy theory, the uncertainty H(X) of a numeric image X, which consists of a series of T characters selected from Y different symbols, can be found from equation (1) [19].

$$H(X) = -\sum_{i=1}^{T} P_i (\mathrm{Log}_2 P_i) , \tag{1}$$

where $P_i$ is the occurrence possibility of a symbol $Y_i$.

When each symbol has an equal chance of occurrence and X selects equal number of symbol to form the T characters, we then have an equiprobable numeric image X. H(X) reaches its maximum value when above condition is met [20]. Since X selects equal number (n) of Y different symbols to form the T characters, T = nY, where n is a positive integer. Each symbol's occurrence possibility $P_i$ becomes

$$P_i = n/T = 1/Y . \tag{2}$$

Substituting $P_i$ into equation (1), H(X) becomes

$$H(X) = T(\log_2 Y)/Y. \quad (3)$$

Similarly, given a symbol S displayed in the same equiprobable numeric image X, its uncertainty H(S) can be found as follows.

$$H(S) = T(\log_2 Y)/ Y^2. \quad (4)$$

The relationship on image and symbol's uncertainties to the image size (T) and symbol's variety (Y) can be found from equation (3) and (4). When using more variety of symbols (Y), both the image and symbol's uncertainties decrease.. Symbol's uncertain reduces even at a faster rate when its variety increases. Both the numeric image and symbol's uncertainties increase when image size (T) increases.

We can obtain a better security to a given plaintext by forming a higher uncertainty numeric image as the ciphertext. Maintaining higher uncertainty for the symbols used in the original plaintext is also an important step. Optimizing the image size (T) and symbol variety (Y) according to equation (3) and (4) is the key task to maintain the desirable high security for both the image X and symbol S.

In encrypting, REAL places the original plaintext symbols S as part of the characters of image X. This image X will be displayed on a screen for viewing. Such image X is called a REAL Image. A proper REAL encryption ensures that S' uncertainty always stay not less than other symbols that are not shown in the plaintext. REAL Image size T will be chosen to have a high uncertainty level and fit the display screen size at the same time. The higher Shannon uncertainty values both the REAL Image and its symbol have, the securer such REAL Image is.

The REAL Image can be easily expanded into a multiple dimensional spatial form factor. By maintaining an equiprobable image in each dimension, the multiple dimensional REAL Image will retain its peak uncertainty. It will be a very secure spatial REAL Image. The multiple dimensional REAL Image feature enables a new spatial encryption method to protect the desired security.

## 3.2 REAL Encryption Procedures

Fig. 1 shows the general REAL encryption procedure. For ease of discussion, a REAL Image X of size T and the set of symbol with numerals of 0 to 9 are chosen to illustrate the proposed algorithm.

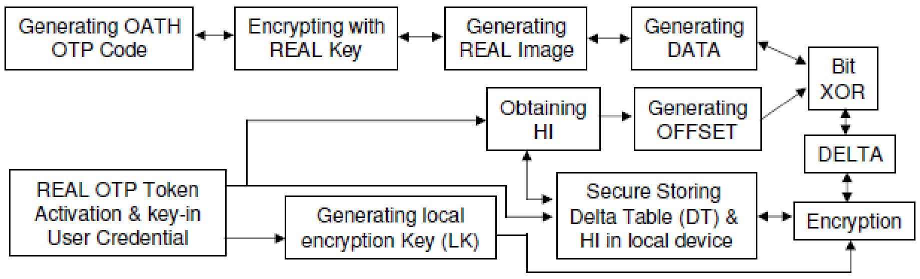

Generating OATH OTP Code
Encrypting with REAL Key
Generating REAL Image
Generating DATA
Bit XOR
Obtaining HI
Generating OFFSET
DELTA
REAL OTP Token Activation & key-in User Credential
Generating local encryption Key (LK)
Secure Storing Delta Table (DT) & HI in local device
Encryption

**Fig. 1.** REAL cipher and Mobile OTP Token operation procedure

**Key Generation.** REAL derives its encryption key from the specific spatial data on REAL Image. Since REAL can encrypt and display ciphertext data in multiple dimensional form factor, its encryption key can be of the same multiple dimensions as well. The REAL key is embedded on its hardware token.

To generate a key, we first choose a desired REAL Image form factor that fits well with the given display screen size and an easy reading symbol font size (as shown in Fig. 3.a.). The encryption key is the character locations that display the plaintext symbols in a REAL Image. Given a two dimensional REAL Image with a plastic card (REAL hardware token), we can randomly place pointers around the card's periphery as REAL key. For D characters of plaintext, we use "D+1" number of pointers as REAL key. The extra one pointer is used as the indicator to choose either front side or back side of the token during the REAL operation. Locations of the pointer can be denoted as $W_i$ where i = 0 to D. Fig. 2. shows an example of such a REAL hardware token with seven code pointers for a 6-digit OTP code. In this example, we have two different sets of key on each side of the token.

We can find the total possible number (N) of key or total number of different hardware token from the equation

$$N = C(T, (D+1)) \,. \qquad (5)$$

Given a REAL Image size of 40 (T) and OTP code size of 6 (D), we can have 18.6 million different tokens or keys (on each side of token). Each token can only decrypt its own encrypted REAL Image (REAL ciphertext).

**Generating REAL Image.** REAL places the plaintext's symbols into the corresponding $W_i$ locations in REAL Image according to each symbol's occurring sequence. We use the following example to illustrate REAL Image's generation.

Assuming D = 6 and plaintext code = 807235, we then have $D_5 = 8$, $D_4 = 0$, $D_3 = 7$, $D_2 = 2$, $D_1 = 3$ and $D_0 = 5$ as the plaintext symbols. Row I and Row II of Table 1 show how the plaintext symbols can be randomly assigned to each $W_i$ location. The pseudo code to generate such REAL Image is shown as follows.

```
REAL_Image_GEN(K, i, T)
0  OTP(i) = Truncate(HMAC-SHA-1(K, i));
1  OTP(i) = Concatenate (D_5|D_4| D_3|D_2| D_1|D_0);
2  Sequentially placing D_5 through D_0 into the
   corresponding $W_i$ locations of the REAL Image;
3  Fill in an odd random number (3 in our example) in $W_6$
   to indicate using key on the front side of the token;
4  Fill the rest of REAL Image elements with randomly
   chosen symbols so that the Image reaches
   equiprobable state,      //This is REAL_Image(i);
5  DATA(i) = Concatenate(elements of REAL_Image(i));
6  End of program.
```

In this pseudo code, K is a 160-bit randomly chosen OTP seed, i is the event counter value, T is the REAL Image size and REAL_Image(i) is the encrypted OTP code generated from OATH OTP formula [5] as shown in step 0. DATA(i) is the concatenation of all the elements of the REAL_Image(i).

**Table 1.** Generating a REAL Image

| | $W_6$ | $W_5$ | $W_4$ | $W_3$ | $W_2$ | $W_1$ | $W_0$ |
|---|---|---|---|---|---|---|---|
| I | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
| II | 3 | 8 | 0 | 7 | 2 | 3 | 5 |

**Offset Generation.** To further protect the plaintext data, REAL does not store the encrypted DATA(i) directly. A random value (Offset) is used to generate a logic operation difference (Delta) between DATA(i) and Offset (i). Delta then indirectly represents its corresponding REAL ciphertext. By safely guarding the random Offset, Delta is very secure and so is the REAL Image. Offset (i) is generated by a one-way-hashing operation from the (i-1)th index value. This hashed index value is called HI. Offset(i) generation procedure is shown in the pseudo code below.

```
Offset_GEN
0  i = 0,       //initialize program loop counter;
1  HI(0)= TRUNCATE(HMAC-SHA-1(K_1, 1)), //K_1 is a 160
   bit random number;
2  Bit_159 to Bit_0 = HMAC-SHA-1(HI(i), 1);
3  Bit_319 to Bit_160 = HMAC-SHA-1(HI(i), 19);
4  Offset(i) = Concatenate(Bit_319 to Bit_0);
5  i = i + 1;
6  HI(i) = TRUNCATE(HMAC-SHA-1(HI(i-1), 1));
7  If i > Max_Count, go to Step 8,    //Max_Count =
   total counts of DATA(i);
8  End of program.
```

**Delta Table and Secure Storage.** Delta(i) is the Bit-Exclusive-OR (B-XOR) difference between the related DATA(i) and Offset(i). Their relationship can be described as

$$\text{Delta(i)} = \text{B-XOR(DATA(i), Offset(i))} . \tag{6}$$

Delta Table (DT) is the compilation of the entire Delta(i) in a special relationship to the corresponding HI(i). That is, Delta(i) is not stored according to the original sequence of DATA(i). Delta(i) is rearranged to follow the value significance of its related HI(i). The new tabulated Delta(i) form the DT. DT and the last HI data will then be further encrypted and securely stored in the designated device.

## 3.3 REAL Decryption Procedures

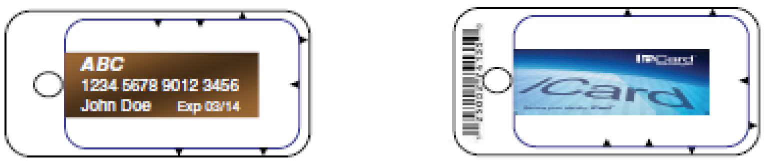
To decrypt a REAL ciphertext, we mainly follow the procedures of section 3.2 but in a reversed order (Fig. 1). Obtaining the last HI(i-1) and Delta(i) value is the first step. A new HI(i) value is generated using the procedure shown in step 8 of Offset_GEN listed in Section 3.2. Once HI(i) is available, Delta(i) can be found by sorting through Delta Table (DT). Following the same step 2 through 4 procedures shown in Offset_GEN, Offset(i) can be generated from HI(i). DATA(i) can then be obtained through the Bit-Exclusive-OR operation of Delta(i) and Offset(i). Subsequently, REAL Image(i) can be reconfigured from DATA(i). By overlaying the unique REAL hardware token on top of the REAL Image(i), the plaintext data will be (rubbed out and) indicated by the pointers on the token (as shown in Fig. 3b).

## 4 Design the Secure REAL Mobile OTP Token

Our design goals and considerations are as follows. The REAL Mobile OTP token will work as a standalone token. It will be fully compliant to existing OATH server, inter-operable with other OATH tokens and easy to deploy and low support cost. The network security will not be compromised even if the phone is lost or stolen. Moreover, the token will have the capability to resist security attacks such as OTP seed tracing by MITM code interception or Shoulder-surfing.

### 4.1 REAL Mobile OTP Hardware Token

We choose a plastic card with the size of 1” x 2” as hardware token (as shown in Fig. 2). The material of our token conforms to credit card’s ISO/IEC FDIS 7810 standard [21]. The token can be easily put on key chain or kept in a wallet. The REAL encryption key is embedded on the token by the code pointer’s locations. The code pointer is the solid black triangle printed on the token periphery. The token making process is very similar to a regular credit card. So the cost of this hardware token is comparable to a credit card.



a. REAL Mobile OTP Token (Front) b. REAL Mobile OTP Token (Back)

**Fig. 2.** a. Token’s periphery is transparent. Overlaid symbols can be clearly read through the token. b. Barcode serial number is printed on the back side to correlate the token with the specific REAL encryption keys. One token carries two set of keys with one each on the front and back side. REAL Image will indicate which set of key to be used.

### 4.2 REAL Mobile OTP Client Software

REAL Mobile OTP Token has two major parts of software. They are Program File and Data File. The Program File contains a set of executable programs to generate the REAL Image and other housekeeping tasks.

All the pre-generated OTP codes are encrypted by REAL and stored as Delta Table (DT) inside the confidential Data File. Data File consists of DT and last HI value. The entire Data File are generated right after a user securely logins and activates the REAL Mobile OTP Token provisioning work. The confidential Data File is encrypted by server using a key (LK) generated from user’s credential information (UC) as shown in the following pseudo code.

$$LK = \text{HMAC-SHA-1}(\text{HMAC-SHA-1}(UC, UC), UC)\,, \tag{7}$$

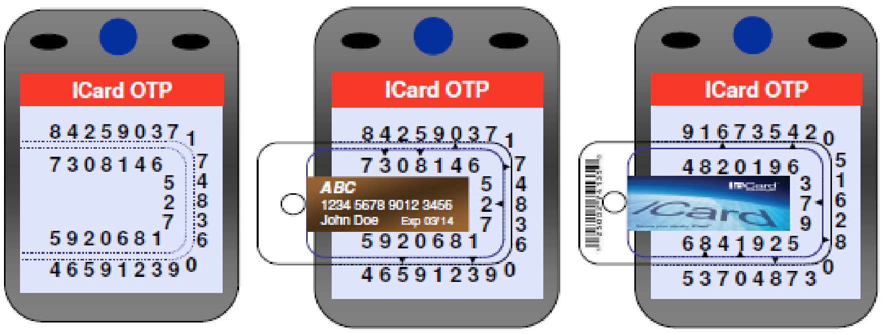
where UC is the user credential information such as the userID, password or PIN.

The encrypted Data File and Program File are zipped together after provisioning and can be downloaded into a user's cellular phone through a secured Internet connection. After the auto-installation, both the Program File and the encrypted confidential Data File are securely ported to the cellular phone.

## 4.3 Using REAL Mobile OTP Token

Once the Program File and Data File are properly installed, a user can activate the token through cell phone's screen or menu. After keying in the user credential information (UC), the phone will display a REAL Image as shown in Fig. 3.a. The user overlays and properly aligns her hardware token (always using front side first) on the screen to decrypt (rub) the OTP code as shown in Fig. 3.b. The first pointer points to a symbol $N_{1f}$ with a value of 3. Since $N_{1f}$ is an odd number, it means that we should use key on the front side to rub OTP code. If $N_{1f}$ is an even number, we should use key on the back side.

OTP code rubbing sequence starts from the most top left symbol on outer ring and begins from the second pointer (pointing to a symbol $N_2$) location. $N_i$ is $D_{(7\text{-}i)}$ of the OTP code. Following this method, we can find the OTP code as 807235. In Fig. 3.c, $N_{1f}$ is 8. Using the back side code pointers, we find the OTP code to be 478818.



a. REAL Image b. Rubbing (Decrypting) OTP Code (using front key) c. Rubbing OTP Code (using back side key)

**Fig. 3.** a. A REAL Image contains an REAL encrypted OTP code. Phone's screen size, token's physical dimension, optimal symbol font size for ease of reading (rubbing) and the desirable card open space for server's Logo lead us to have a 40 symbols REAL Image. b. $N_{1f}$ read from REAL hardware token's front side is an odd number 3. Front side of token is used to rub (decrypt) the OTP Code. The OTP code is 807235. c. $N_{1f}$ read from front side of REAL hardware token is an even number 8. Back side of the token should be used to rub (decrypt) the OTP Code. The OTP code is 478818.

## 4.4 Token That Resists Certain Security Attacks

Man-in-the-middle (MITM) can be in the network to intercept a user's static password for next round use. This is the so called "Replay" attack. The OTP's dynamic password has successfully thwarted such attack [22]. REAL Mobile OTP Token (MR1 Token) also has the same capability to prevent such MITM replay attack.

**Man-in-the-middle data intercept attack [23].** MITM can obtain series of OTP codes by continuingly intercepting a user's login password. Even with a dynamic password, a hacker can try to trace the seed (K) through the pseudo random sequence the captured passwords show. This is a direct attack on an OTP algorithm. If the malicious person has enough pseudo random number data base and computation power, a typical OTP token can not resist such attack effectively.

An improved REAL Mobile OTP Token can provide extra protection when such event happens. We name this version MR2 Token. Each user will be assigned with two sets of OTP generating seed ($K_A$ and $K_B$). The user will use one of the two OTP tokens each time but in a mixed random order. The OTP codes generated by token A will be encrypted by the REAL key on front side of token. Token B will use the REAL key on the back side to encrypt and decrypt its OTP codes. So even though the user carries only one REAL Mobile OTP hardware token, he actually has two tokens with him all the time.

The first symbol ($N_{1f}$) pointed by the first pointer on the front side is still used as the indicator to select front and back side's key. During the REAL Image generation procedure, an odd number of $N_{1f}$ is used when an OTP code is from token A and an even $N_{1f}$ is used with an OTP code from token B. The authentication server will know the sequence of which token is used as the provisioning is provided by server itself. A user can operate as if she has only one token. The two pseudo random OTP codes are used in a mixed random order. So the intercepted OTP codes can no longer provide a meaningful pseudo random sequence. It makes random number seed tracing very difficult. This attack is then prevented.

**Shoulder-surfing attack [24].** Shoulder-surfing attack happens when a malicious person secretly observes the action and screen while a user is using her OTP token. The malicious person may then have the REAL Image and code pointer information to retrace the secrecy or OTP code. He may not have the OTP code as the code usually displays in other non-numeric symbols during the login process.

To fend off such attack, we employ a random offset to decouple the direct relationship among pointer locations and OTP code symbols in REAL Image. Token of such feature is called MR3 Token. The first symbol ($N_{1f}$) pointed by the first pointer on the front side is still used as the indicator to select front and back side's key. But both $N_{1f}$ and $N_{1b}$ will be used as an adder to each symbol's numeric value pointed by the rest of the six code pointers. The ten's digit will be dropped if the added value is greater than or equal to 10. The general equation is as follows.

$$D_{if} = (\text{Value of } N_{1f} + \text{Value of } N_{(7-i)\,f}) \bmod 10\ , \tag{8}$$

$$D_{ib} = (\text{Value of } N_{1b} + \text{Value of } N_{(7-i)\,b}) \bmod 10\ , \tag{9}$$

where $D_{if}$ is the ith digit of OTP code and $N_{(7-i)\ f}$ is the symbol pointed by its corresponding (7-i)th pointer when using REAL key on front side. When making the REAL Image, each $N_{(7-i)\ f}$ value should be adjusted according to equation (8).

In Fig. 3.b, the $N_{1f}$ is equal to 3, an odd number. We use the key on front side of hardware token to rub the OTP code. The second pointer indicates $N_{2f} = 8$. From equation (8), we find $D_{5f} = 1$. Following this new decryption procedure, we have the full OTP code as 130568.

The REAL Image in Fig. 3.c shows $N_{1f} = 8$, an even number. We use REAL key on the back side to decrypt the OTP code. This code is actually generated from Token B. The first pointer on the back side shows a number $N_{1b} = 6$. We find $N_{2b} = 4$. So $D_{5b} = 0$. Following the procedure, we can obtain the full OTP Code as 034474.

Both $N_{1f}$ and $N_{1b}$ are randomly chosen and the modular operation also adds additional randomness to the codes. The codes obtained from Fig. 3.b and Fig 3.c show no physical direct relationship with the original code pointer locations. It then retains the security and resists the attack from Shoulder-surfing.

## 5 Analysis

### 5.1 Design Goals Review

For a 5,000 pre-generated OTP codes stored in a REAL Delta Table, the entire code size is about 200 KBytes. It is sufficient for a consecutive 2.7 years use with an average daily usage of 5 OTP codes. The entire software code size of REAL Mobile OTP Token is less than 3 Mbytes. Both software program deployment and technical support can be easily done through Internet. There is no need to change or add any hardware or software on the existing OATH authentication server. The hardware token can be made, delivery and activated following the same logistic like credit card. It helps to achieve easy deployment and low cost operation.

The token uses the same OATH OTP algorithm to generate the OTP codes. It ensures that the OTP codes are 100% compliant to any OATH authentication server. As long as both tokens use the same key (K) and counter value (C), REAL Mobile OTP Token can maintain full interoperability with other tokens. The token can directly replace any existing or expired OATH OTP token.

### 5.2 OTP Code Security and Integrity

The plaintext OTP code is generated by the same OATH algorithm. The REAL decrypted OTP code is as secure as the one generated by an OATH compatible token. A detailed OATH OTP code security analysis can be found in Appendix A of RFC4226 [5] for further reference.

REAL encrypted OTP codes are encrypted again by Hashed Index (HI) value when they are placed into Delta Table (DT). DT and HI value are further encrypted using a key (LK) generated by user credential data. It prevents these data from being tampered. They provide the desired level of integrity and security protection.

## 5.3 Real Image Security Level

REAL Image (RI) contains the OTP code. Its security level affects how secure the OTP code is protected. Given an image size of 40 (T) and a 6-digit (D) OTP code, equation (5) shows a total number of 37.2 million different code pointer patterns (REAL key) can be embedded on both sides of the hardware token. Without the aid of the hardware token and a known OTP code, to guess the correct key from a REAL Image, the possibility ($P_1$) is 1 out of 37.2 million. Even with a known REAL Image alone, equation (5) also shows that to directly guess the correct 6-digit OTP code the possibility ($P_1$) is 1 out of 3.8 million.

On the other hand, a traditional OATH token display the full OTP code on the screen without any encryption. The chance ($P_1$) to obtain the correct code from the image on the LCD screen will be 1 out of 1. If just considering the displayed image alone, REAL OTP Token's security level is much stronger than a traditional OATH token. In other word, REAL securely encrypts the 6-digit OATH OTP code inside the REAL Image. So REAL does not degrade the OATH OTP code security level.

## 5.4 Security Attacks

Man-in-the-middle (MITM) Replay Attack is a decades old problem. A dynamic password from an OTP token such as our MR1 token can successfully thwart such attack [22]. The MR2 is a two OTP tokens in one physical token form. It can send a stream of OTP codes generated from two different OTP tokens in a mixed random order. The intercepted OTP codes by MITM are just a randomly mixed number series. The original pseudo random sequence of an OTP generating algorithm is broken and difficult to trace. So MR2 token can resist the OTP seed tracing by MITM interception attack. The MR3 token randomly decouples the REAL key from its hardware token code pointer's physical locations. It effectively breaks the key and pointers' physical linking information that a malicious person tries to get. The MR3 token thus successfully resist the attack from Shoulder-surfing or Seed tracing by MITM code interception.

## 5.5 Other Security Concerns

**Cellular phone is lost or stolen.** Cellular phone is small and prone to get lost or stolen. A malicious person has to crack the user's credential first before activating the REAL OTP function on a cellular phone. Even if the cellular phone is activated, trying to correctly guess the OTP code with a known REAL Image but without the specific hardware token, the possibility ($P_1$) is 1 out of 3.8 million. It is much tougher than a brute force guess the 6-digit OTP code (1 out of 1 million).

**Hardware token is lost, stolen or secretly copied.** A regular software or hardware token is a standalone OTP code generator. The loss of such token means losing all the future OTP codes. REAL hardware token does not contain any electronic to generate OTP code. The token will only work with a specific cell phone that has the user Data File. So losing a REAL hardware token or the pointer pattern being photo copied, the security will not be compromised. Having hardware token alone, the possibility ($P_2$) to correctly guess the OTP code is like to guess a correct REAL Image. The possibility ($P_2$) is 1 out of $10^{40}$. It is $10^{34}$ times tougher than a brute force guess of the 6-digit OTP code.

**Confidential File is secretly copied or stolen.** A malicious person may copy the user specific confidential Data File without user's knowledge. The person can then set up a cell phone that imitates the user's REAL Mobile OTP token environment. It can generate correct REAL Image independently. But the intruder will face the difficulty of guessing the correct OTP code without the hardware token even if user credential information is correctly guessed. Intruder will then meet the difficulty of probability $P_1$ (1 out of 3.8 million) that we discussed previously.

**Limitations.** Security attack technique advances daily. Trojan and MITM attack though not found in the cellular phone today, they have successfully infiltrated the PCs world and created huge damages. The existing 2FA system alone can not solve this issue [25]. Layered security solutions and discipline are needed. REAL Mobile OTP Token will become part of the security solutions for this new challenge.

## 6 Conclusion and Future Work

Using cellular phone as a One-time Password (OTP) token to generate dynamic session password is becoming popular recently. But it has met certain difficulties. Some of the solutions can not fully prevent the OTP seed tracing by MITM code interception or Shoulder-surfing security attacks. Other have issues regarding poor compliance and interoperability with existing authentication infrastructure, geographical limitation due to poor or no cellular service, plus high deployment and support cost. In particular, many of the implementations may comprise the security when phone is lost, stolen or data file is secretly copied.

A Rubbing Encryption Algorithm (REAL) is used as the base cipher for a new Mobile OTP Token. REAL decryption does not require entering encryption key on local computing device. It allows REAL to use a long and complex key to encrypt a short word length plaintext such as the OTP codes. This feature ensures high level of security on REAL ciphertext (REAL Image). The ciphertext can be securely stored in a cellular phone even if the device gets lost or stolen. A user can use an inexpensive non-electronic plastic hardware token to electronically "rub" (decrypt) the OTP code from the phone screen's REAL Image. We implement and analyze an OATH Compliant Mobile OTP Token using REAL. This token is in compliant and interoperable with existing authentication infrastructure. Token's deployment and support is easy and with low cost. The token also has capability to resist the MITM data interception to trace OTP generating seed. Furthermore, it can resist the Shoulder-surfing attack as well.

REAL can be used in multiple dimension form factor also. It has potential to be used in other applications with new implementations [26]. REAL can also be further enhanced on its security strength. These are the future work for us.

**Acknowledgments.** We will like to thank Innovative Card Technologies, Inc. for their diligent efforts on marketing the OATH Compliant REAL Mobile OTP Token. Their feedback provided many important inputs on algorithm and token design fine tuning, enhancement and direction for future improvement.

## References

1. RSA.: RSA SecureID, Software Authenticator, `http://www.rsa.com/node.aspx?id=1313`
2. Mizuno, S., Yamada, K., Takahashi, K.: Authentication Using Multiple Communication Channels. In: DIM 2005, November 11 (2005)
3. Kostiainen, K., Ekberg, J.E., Asokan, N.: On-board Credentials with Open Provisioning. In: ASIACCS 2009 (March 2009)
4. Wikipedia.: Two-factor Authentication – Challenges, `http://en.wikipidia.org/wiki/two-factor_authentication`
5. M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D. Ranen, O.: HOTP: An HMAC-Based One-time Password Algorithm, The Internet Society, Network Working Group. RFC4226 (December 2005)
6. Initiative for Open AuTHentication.: Oath Vision, `http://www.openauthentication.org/about`
7. Verisign.: Authentication for Business Partners and the Mobile Workforce, `http://www.verisign.com/authentication/enterprise-authentication/enterprise-otp/`
8. Deepnet Security: MobileID - A Mbile, To-way and To-factor Athentication, `http://www.deepnetsecurity.com/products2/MobileID.asp`
9. Aloul, F., Zahidi, S., El-Hajj, W.: Two Factor Authentication Using Mobile Phones. In: 2009 IEEE/ACS International Conference on Computer Systems and Applications (2009)
10. Liao, K., Sung, M., Lee, W., Lin, T.: A One-Time Password Scheme with QR-Code Based on Mobile Phone, doi: 10.1109/NCM.2009.324
11. Liberty Aliance Project: Liberty Alliance, `http://www.projectliberty.org/`
12. FreeAuthProject.: The FreeAuth Project, `http://www.freeauth.org/site`
13. Abe, T., Itosh, H., Takahashi, K.: Implementing Identity Provider on Mobile Phone. In: DIM 2007, November 2 (2007)
14. Haverinen, H., Asokan, N., Maattanen, T.: Authentication and Key Generation for Mobile IP Using GSM Authentication and Roaming. In: ICC 2001 (2001)
15. Hallsteinsen, S., Jorstad, I., Thanh, D.: Using the Mobile Phone as s Security Token for Unified Authentication. In: ICSNC 2007. IEEE Computer Society, Los Alamitos (2007)
16. Thanh, D., Jonvik, T., Feng, B., Thuan, D., Jorstad, I.: Simple Strong Authentication for Internet Applications Using Mobile Phones. IEEE GLOBECOM (2008)
17. Wangensteen, A., Lunde, L., Jorstad, I., Thanh, D.: A Generic Authentication System Based on SIM. In: The International Conference on Internet Surveillance and Protection, ICISP 2006 (2006)
18. Thanh, D., Jonvik, T., Thuan, D., Jorstad, I.: Enhancing Internet Service Security Using GSM SIM Authentication. In: IEEE GLOBECOM (2006)
19. Stinson, D.: Cryptography – Theory and Practice, pp. 44–67. CRC Press, Boca Raton (1995)
20. Shastri, A., Govil, R.: Optimal Discrete Entropy. Applied Mathematics E-Notes 1, 73–76 (2001)
21. International Organization for Standardization.: ISO/IEC 7810:2003. November 17 (2009)
22. Lamport, L.: Password Authentication with Insecure Communication. Communications of the ACM 24(11), 770–772 (1981)

23. Wikipedia.: Man-in-the-middle Attack (April 15, 2010), `http://en.wikipedia.org/wiki/Man_in_the_middle_attack`
24. Wikipedia.: Shoulder Surfing (Computer Security) (April 15, 2010), `http://en.wikipedia.org/wiki/Shoulder_surfing_computer_security`
25. Schneier, B.: The Failure of Two-factor Authentication. Communications of the ACM (April 2005)
26. Cheng, F.: A Novel Rubbing Encryption Algorithm and The Implementation of the Web-based One-time Password Token. In: COMPSAC 2010, July 19 (2010)