International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075, Volume-3 Issue-1, June 2013

Dynamic Password Schemes for Protecting Users from Password Theft for E-Banking

Shimna M. S., Sangeetha P. S.

Abstract-In this paper, we discuss how to prevent users passwords from being stolen by adversaries in online banking and automated teller machines. We propose dynamic password mechanisms in which a user has a mobile, in that mobile the dynamic password scheme is implemented using Android Operating System, so dynamic password requires a small amount of human computing to secure users passwords. Among the schemes, we have a default method (i.e., traditional password scheme), system recommended functions, user-specified functions, user-specified programs. A function/program is used to implement the dynamic password concept. For user-specified functions, we adopt secret little functions and a constant value, in which security is enhanced by hiding both. The computation of human can be reduce by using mobile applications with builtin dynamic password. Here the user only needs to input the system random digits which the system provides and then the dynamic password is automatically calculated for the user. Thus we can overcome the main attacks like phishing, key-logger, shoulder-surfing, mobile malwar attacks simultaneously.

Keywords: dynamic password, Net banking, secret little function, codebook, Phishing, key-loggers, shoulder-surfing, mobile malwar attack.

I. INTRODUCTION

Today, the Internet has entered into our daily lives, as many services have been moved online. Besides reading the news, searching for information, and other risk free activities we are using online, we have also become accustomed to other risk-related work, like paying using credit cards, checking/composing emails, online banking, and so on. Most current commercial online banking will ask their users to input their user identifications (IDs) and corresponding passwords for authentication.

Once a user's ID and the corresponding password are stolen by an adversary, the adversary can do anything with the victim's account, which can lead to a disaster for the victim. As a consequence of increasing concerns over such risks, protecting users' passwords on the web has become increasingly critical.

The secure protocol SSL/TLS [1] for transmitting private data over the web, but most current commercial online banking still rely on the relatively weak protection mechanism of user authentications via plaintext password and user ID. Now security token are provided for authentication. Tokens need secure channel. This is still vulnerable to attacks as follows,

Phishing: Phishers attempt to fraudulently acquire sensitive information, such as passwords and credit card details, by masquerading as a trustworthy person or business in an electronic communication [2]. For example, a phisher can set up a fake website and then send some emails to potential victims to persuade them to access the fake website. This way, the phisher can easily get a clear-text of the victim's password. Phishing attacks have been proven to be very effective.

Password Stealing Trojan: This is a program that contains or installs malicious code. There are many such Trojan codes that have been found online today, so here we just \briefly introduce two types of them[3].

1. Key loggers capture keystrokes and store them somewhere in the machine, or send them back to the adversary. Once a key logger program is activated, it provides the adversary with any strings of texts that a person might enter online, consequently placing personal data and online account information at risk.

2. Trojan Redirector was designed to redirect end-users network traffic to a location to where it was not intended. This includes crime ware that changes hosts files and other DNS specific information, crime ware browser-mobile objects that redirect users to fraudulent sites, and crime ware that may install a network level driver or filter to redirect users to fraudulent locations.

Shoulder Surfing: Shoulder surfing is a well-known method of stealing other's passwords and other sensitive personal information by looking over victims' shoulders while they are sitting in use a hidden camera to record all keyboard actions of a user. front of terminals[4][5][6]. This attack is most likely to occur in insecure and crowded public environments, such as an Internet Cafe, shopping mall, airport, etc

Mobile Malwar attack: The malwar attack stealprivate data from infected smartphones, including the address book and messaging history, and sends it to a command and control server[7]. Android malwar attack is new attack for mobile using android application

In this paper, we present a password protection scheme that involves a small amount human computing in an Internet-based environment, which will be resistant to a phishing scam, a Trojan horse, and shoulder-surfing, mobile malwar attack. We propose a dynamic password concept involving a small amount of human computing to secure users' passwords in on-line environments. We propose differentiated security mechanisms in which a user has the freedom to choose a dynamic password scheme ranging from weak security to strong security. Among the schemes, we have a default method (i.e., traditional password scheme), system recommended function, user-specified function, user specified program, etc. A function/program is used to implement the dynamic password concept which requiring a small amount of human computing. We further propose several functions serving as

Manuscript received June, 2013.

Mrs.Shimna M.S,Computer science, Vedavyasa Institute of technology, Malapuram, India.

Mrs.Sangeetha P.S, Working as a assistant professor. Department of computer science, Vedavyasa Institute of technology, Malapuram, India.

system recommended functions and provide a security analysis. We analyze how the proposed schemes defend against phishing, key logger, shoulder-surfing attacks, mobile malwar and multiple attacks. In user-specifie functions, we adopt secret little functions and constant value, in which security is enhanced by hiding both. To the best of our knowledge, our dynamic password mechanism is the first one which is able to defend against all above attacks together. We further propose a scheme to adopt ACT(Authentication Code Test) to be used for re-keying and to defend against Phishing attack.

The proposed functions include secret little functions, a constant value from a matrix and two other schemes called codebook and reference switching functions. Our objective is to produce a function achieving both: 1) ease of computation; and 2) security. However, since simplicity and security conflict, it is difficult to achieve both. The idea of this paper is to add some complexity, through user computations performed by devices, to prevent the above kinds of attacks. We believe that, for some sensitive accounts such as online bank accounts users are likely to choose additional complexity which requires some degree of human computing in order to make the account more secure.

The rest of this paper is organized as follows. We describe related work about password protection in Section II. In Section III, we propose the idea of the dynamic password, differentiated security mechanisms, and user-specified functions or programs, in which we propose the concept of secret little functions. One functions (the codebook approach) are proposed in Section IV. We provide some Security analysis in Section V. In Section VI, we describe implementation issues of our scheme. Finally, we conclude our paper and describe our future work in Section VII.

II. RELATED WORK

How to shield users' passwords from being stolen by adversaries is not a new topic, but it is always important because adversaries keep inventing more and more advanced attacks to break the current defense schemes. This results in more research on protecting users from such attacks. In this section, we briefly introduce the previous work on defending against user password-stealing attacks for the three major categories

Phishing attacks are relatively new but very effective. There are two typical types of phishing. First, to prevent phishing emails [8], [9], [10], a statistical machine learning technology is used to filter the likely phishing emails; however, such a content filter does not always work correctly. Blacklists of spamming/phishing mail servers are built in [11] and [12]; however, these servers are not useful when an attacker hijacks a virus-infected PC. In [13], key distribution architecture and a particular identity-based digital signature scheme were proposed to make email trustworthy. Second, to defend against phishing websites, the authors in [14] and [15] developed some web browser toolbars to inform a user of the reputation and origin of the websites which they are currently visiting.

In [16], the author presented a tricky method which can confuse a keylogger, which works as follows. Instead of typing your whole password into the login field, the user changes focus outside the login form and types some random characters between any two successive password characters. However, this trick does not shield the user from keylogger attacks. It only makes it slightly more difficult because it is very easy to record all the keys, mouse events, and applications of the focus.

The authors in [17] and [18] used a dynamic pad for the login system, which allows a user to click the dynamic keyboard on the screen instead of typing on the physical keyboard, but such a dynamic keyboard faces some of the same problems as above (i.e., an adversary can record all the mouse events with a combination of screen snapshots to figure out what the user clicks on the screen)

Alphanumeric password systems are easily attacked by shoulder-surfing, in which an adversary can record the user motions by a hidden camera when the user types in the password. In [19], the authors adopted a game-like graphical method of authentication to combat shoulder-surfing; it requires the user to pick out the passwords from hundreds of pictures, and then complete rounds of mouse-clicking in the Convex Hull. However, the whole process needs the help of a mouse and it takes a long time.

The author in [20] filed a patent to allow a user to make some calculations based on a system generated function and random number for the user to prevent password leaking. However, the scheme in [20] is not anti-phishing and the password can be stolen if an adversary uses a camera to record all the screens of the system and motions of the victim.

In the previous sections, we have briefly introduced some schemes without including methods which need hardware support. None of the schemes above can prevent phishing, Trojan horse, and shoulder-surfing at the same time. A one-time password (OTP) does not use a static password, and therefore can prevent replay attacks. There are several approaches to generate and distribute OPTs: 1) a time-synchronization method; 2) a mathematical algorithm to generate a password based on old ones; and 3) a mathematical algorithm to generate a challenge. In the time synchronization method, a password is generated based on the current time, while the server and the client programs need time-synchronization via a physical electronic token. In the second method, OPTs must be used by a predefined order such as using a one-way hush function. In the third method, challenge-response can be done by a physical token. Therefore, an OPT method needs a electronic token, mobile phone, or out-of-band channel such as SMS messaging. The idea of our proposed dynamic password is similar to OPT. However, the approaches are different.

The author in [21] in the secret little function approach, the secret little function is built between the server and the user (a human being), whereas in a OPT method, communications are done between the server and a physical device (either a token, a computer, or a cellular phone). This proposed a virtual password concept involving a small amount of human computing to secure users' passwords in online environments. User needs to calculate the virtual password from the VPF with the inputs, the random salt, and the hidden password. The whole login process may take a little bit longer because it requires the user to perform some calculations.

III. DIFFERENTIATED DYNAMIC PASSWORDS AND SECRET LITTLE FUNCTIONS

In Section III-A, we propose the idea of the dynamic password. In Section III-B, we propose differentiated security mechanisms in which a user has the freedom to choose a dynamic password scheme ranging from weak security to strong security. In Section III-C, we propose user-specified functions or programs, in which we propose the concept of secret little functions. We discuss dynamic password function with a mobile-application in Section III-D. We present *ACT* authentication in Section III-E

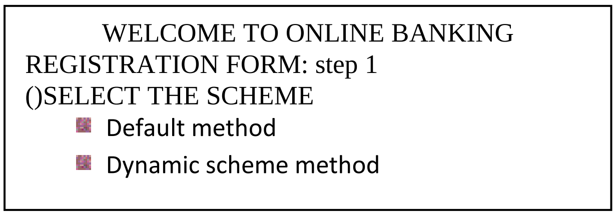
A. Dynamic Password:

To authenticate a user, a system (S) needs to verify a user (U) using the user's password (X) and ID (also denoted as U) which the user provides. In this procedure, S authenticates U by using U and X, which is denoted as: $S \rightarrow U$: U, X. Both U and X are fixed. It is reasonable that a password should be constant so that it can be easily remembered. However, the price of being easily remembered is that the password can be stolen by others and then used to access the victim's account. At the same time, we can not put X in a randomly variant form because it would be impossible for a user to remember the password. To confront such a challenge, we propose a scheme using the new concept of dynamic password.

A *dynamic password* is a dynamic password that is generated differently each time from a *dynamic password scheme* and then submitted to the server for authentication. A dynamic password scheme P is composed of three parts, a fixed alphanumeric X (i.e., the real password, also called the hidden password) , a constant value C which is selected from a matrix (matrix contain alphabet(case sensitive), numbers and some symbols) and a function F from the domain ψ to ψ, where the ψ is the letter space which can be used for passwords. Since we call $P = (X,C,F)$ a dynamic password scheme, we call F a dynamic password function (DPF). The result (denoted as D) of the DPF is called a dynamic password, and F may have some hidden parameters, H, which are the secrets between the server and the user. If this is the case, we denote F with $FH(...)$. Note that the DPF can be a secret between the server and the user. Let $X = x1x2...xn$ denote a vector standing for the hidden password, where xi ($i = 1...n$) is a digit, and n is the length of the vector. Let $R = r1r2...rn$ denote the random number provided by the server, also called the random salt, a constant value c_i standing for hidden parameter(where i=1) and prompted in the login screen by the server. $D = d_1,...,d_n$ is the dynamic password used for authentication. The user input includes (U, D), where U is the user ID. On the server side, the server can also calculate D in the same way to compare it with the submitted password. We use $D= FH(X,C,R)$ or $FH(xi,c_i,ri) = d_i$ interchangeably in the rest of this paper. It is easy for the server to verify the user if F is a bijective functions. If F is not a bijective function, it is also possible to allow the server to verify the user as follows. The server first finds the user's record from the database based on the user's ID (i.e., U), then it computes D and compares it with the one provided by the user. A bijective function makes it easier for the system to use the reverse function to deduce F's dynamic password. Therefore, we do not assume that F is a bijective function. The user should be free to pick the hidden password. We propose a differentiated security mechanism in the next subsection to allow the user to choose a DPF.

B. Differentiated Security via a DPF

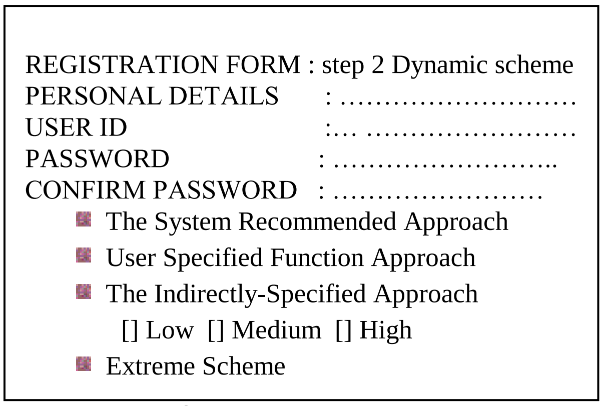
We have introduced the concept of the dynamic password; next, we detail how to apply it in an Internet-based environment. We propose a differentiated security mechanism for system registration in which the system allows users to choose a registration scheme ranging from the simplest one (default) to a relatively complex one, where a registration scheme includes a way to choose a dynamic password function. The more complex the registration, the more secure the system is, and the more user involvement is required. A screenshot of the first step of the proposed registration is shown in Fig. 1.



WELCOME TO ONLINE BANKING
REGISTRATION FORM: step 1
()SELECT THE SCHEME

- Default method
- Dynamic scheme method

Fig 1: Screenshot for user registration step1



REGISTRATION FORM : step 2 Dynamic scheme

PERSONAL DETAILS	: ……………………….
USER ID	:… ……………………
PASSWORD	: ……………………..
CONFIRM PASSWORD	: ……………………

- The System Recommended Approach
- User Specified Function Approach
- The Indirectly-Specified Approach
  [] Low [] Medium [] High
- Extreme Scheme

Fig 2: Screenshot for user registration step 2: Dynamic Scheme

Whether a dynamic password scheme is used or not, the user is required to input the read password and ID in Step 2 of the registration. In Fig 1, a user has the freedom to choose a default approach in the traditional way or a more complex scheme as proposed in this paper. A user can choose a recommended dynamic password function, define his/her own dynamic password function, or even define a common program to share between the user and the server to calculate the password as in Fig 2.

1) The system recommended approach is that, after the system receives a registration request, it automatically generates a function. The users do not have to provide extra information about the function to the server except for some necessary parameters, called hidden parameters (H).

2) The user specified function approach is the one in which users themselves can choose any function they like. However, such freedom is based on the assumption that the user has some basic knowledge about DPFs, which can be introduced by an online introduction.

3) The indirectly-specified approach, instead of letting either the user or the server make the full decision, allows a user to specify the desired security degree. Then the server will assign a function according to that degree.

4) An extreme scheme is that the user can even provide a program in C or Java instead of a function. This requires a very advanced user.

Note that, except for the default approach, either human computing is involved or mobile (or a computer) which can be programmed to compute the dynamic password is needed. We could develop a smart application to make the complex calculation for the user which can be run on the mobile device, such as a cellular phone, PDA, smart phone, iphone, personal computer, or programmable calculator, to relieve the user from complicated calculations and to overcome any short-term memory problem. If such a mobile-application is involved, we should make sure that the mobile-application itself is unique to each user account and only works for the corresponding user account. Regardless of the approach chosen, a user's registration in the system is similar (i.e., the user submits a user ID and a fixed password). The one difference from a traditional approach is that in the dynamic password scheme, a constant value and a DPF must be set during the registration phase. The server then delivers this function information to the user via some channels, such as displaying it on the screen or in an email. The user needs to either remember this function together with the password they have chosen, and the constant value or to save them in disks or emails. The user-specified password and the system generated function are combined to form a dynamic password scheme.

We also note that a small amount of human-computing is involved in the authentication process. We have to choose a DPF to make the calculation as simple as possible if the mobile-application is not used. A user has to remember the hidden password, a constant value and the function (i.e., DPF), and as a result more effort is required to remember them. However, the dynamic password will be resistant to a dictionary attack, mostly because users like to create a password which is either related to their own name, date of birth, other simple words, and so on. In a traditional password scheme, users can change their password. This is also true in our dynamic password scheme. Unlike the traditional scheme, users can change the hidden password, constant value and the DPF all or any one or two etc ....

C. User-Specified Functions/Programs

The strongest security approaches let the user define a user specified function and constant value . Here secret encryption algorithms (i.e., user-specified DPFs) is used because algorithms are very personal to a particular user and should not be known by others except the server. On the other hand, for example, a wireless local area network (WiFi) needs open encryption algorithms to allow products from different companies to communicate with each other. Otherwise, one company's WiFi card could not communicate with that of another company. However, in our application, communication is only between a user and a server so that it is good to use secret encryption algorithms and constant value, since this improve security by hiding the both. So even a very simple function will be secure because the attackers do not know what kind of functions the user chose and the value .Examples of simple functions can be as follows:

1) Flip one bit in the password and add constant;
2) flip one digit in the password and add constant ;
3) Add one to each odd digit and minus one in each even digit and add constant ;
4) The first digit of the password is tripled; 100x+birthdate*c, where x is the real password in an integer form transferred from ASCII codes, c is constant value ;
5) Reversing even bits of the real password and constant value in a binary form etc

User specified functions can be infinite. Since attackers do not know the function forms (i.e., secret encryption algorithms and constant value), these simple functions are very secure. Note that user specified functions do not need to be bijective. We call these simple and secure functions *secret little functions*. They are useful in our context. One problem is that extra effort is required in programming the function into the server upon the creation of an account, so human intervention is needed. Another constraint is that secret little functions must use the random number provided by the server; otherwise, it would still be subject to Key-logger attacks since the attackers do not need to know the function but can simply input the same capture inputs again to gain access. Advanced users can also define a program to be used. Note that the secret little function is that the function F defined in $P = (X,C,F)$.

D. DPF With a Mobile-Application

If a mobile-application is available for the user, the user needs to type the random salt into the mobile-application; subsequently, the dynamic password is generated by the mobile application. The user then types the generated dynamic password in the login screen. In this way, the extra time required is very small and the precision will be 100% correct as long as the user types the correct random salt displayed on the login screen. However, such mobile devices are not able themselves to communicate with the server to which the user wants to login. No matter how complex the DPF is, the mobile-application can always generate the correct dynamic password for the user. This case is the most sophisticated one, and it is also the most convenient approach for the user.

For password changing, the user only needs to change the password in mobile-application instead of remembering all the changed parts of the dynamic password. Note that the server must make the corresponding changes too. Here that the mobile-application can communicate with the server, the user only needs to type the random salt in the mobile-application, and then the rest of the work is done by the mobile-application. The mobile-application can generate the dynamic password and provide the password, which is built into the mobile-application for the corresponding user. For password or constant value or functions changing, the mobile-application communicating with the server is a better way to change them and make them more secure (i.e., the mobile-application can periodically make the change request to server and update the corresponding change by the user into the mobile application). The whole process can be completely transparent to the user.

For mobile-application we are using the android application which is the latest one. Here we are going to implement our dynamic password scheme in android operating system in mobile

E. ACT Authentication

In this subsection, we propose a scheme to adopt *ACT* to be used for re-keying and defending against phishing. The purpose of this scheme is that it can be used for authentication of the server and the client also before re-keying of the previous scheme. This *ACT* scheme can be very useful when the web browser or other client side applications, such as the mobile-application can have an authentication function implemented. This scheme can also be used to protect from phishing via emails.

ACT [22] is an authentication scheme which was originally designed for sensors to authenticate a broadcast message sender in a sensor network based on a public one-way hash function *F*.

We could use the methodology to defend against phishing attacks. Here the server side and client side will choose the same public hash function; we discuss how it works in the registration phase, sign-on phase, and password change phase.

In the registration phase, upon a registration request, in addition to preparing the general password, user id, and other information, the server needs to generate a chain key *Km*, ...,*K*0 by randomly choosing the *Km* and then producing the *Kn*−1 = *F*(*Kn*) where *n* = 0, 1, ...,*m*. The server will then pass the *K*0 to the client.

In the sign-on phase, once the sign-on request arrives at the server side, the server presents the sign-on screen with authentication code to the clients, which is encrypted by the latest key. For example, the *4*th time the user signs in to the system, the server produces the authentication code as EK_3 (K_4) and passes this to the client. When the client receives the page, they first need to decrypt the authentication code with the current key (i.e., *K3,which is saved*) then to get the K_4 and then save this K_4 thus verified this sign-on screen is from the right server in the following way: if *F*(*K3*) = *Key*, it is verified and the currently held key is updated to be *K4*; and a message is forward from the user(mobile scheme) automatically to the server when it is decrypted , without any user intervention otherwise, it is denied. In this way, the client and the server can verify them each other and be protected from phishing attacks because the phisher has no knowledge\ of the *K*0, ...,*Km* and therefore is not able to fake the authentication code. The server should use the latest used key to encrypt the current key. The server then presents the login screen along with the new authentication code *Kn*, which makes it difficult for the phisher to fake a login screen with the correct authentication code to lure the client. Because the number of keys from *K*0 to *Km* is finite, the server will eventually use up the entire authentication codes, even if we choose a very large value of *m*. The server and client should build a scheme to regenerate their authentication code, which we refer to here as re-keying of the authentication key refreshes. This is an easy job and can be conducted once both the client and the server verify each other. The server and client needs to generate a new chain of keys, as it did in the user registration phase, and to deliver the first of the keys to the client and the client save that as key for decrypting. This *ACT* scheme will work effectively to shield the clients from phishing attacks, and it could be used together with our dynamic password scheme to protect the user's password.

IV. CODEBOOK

Here, we propose an approach to the dynamic password function. They are not perfect, but they are acceptable in the hostile password phishing environment. For this approach, some small codebooks will be needed. A codebook should be small enough to be printed on a pocket-sized card, stored in a saved e-mail, or stored in a PDA or cell phone for the user to carry. Our ultimate goal is to design a zero-knowledge interactive proving protocol. Thus, our next ideal function is not to give away enough information to significantly compromise the user's account.

A. Codebook

A **codebook** is a type of document used for gathering and storing codes. In the setup session, the user decides the length of the password, *n*. The server then generates *n* 64-digit random numbers. Suppose that we are doing this to protect a 4-digit PIN (i.e., *n* = 4). The sever outputs four random numbers, *R*0, *R*1, *R*2, and *R*3, with each having 64-digits. Let *r*(*i*,0), *r*(*i*,1), *r*(*i*,2), ..., *r*(*i*,*63*) denote the 64- digits of *Ri*. ACT generates chain-key which is setup and stored in codebook. The user's codebook contains the hidden password, constant value and the user specified function, current authentication decrypting key and ACT keys. So the user need not to remember these thing, in case a user loses their codebook, the system will not be compromised and the user can easily ask for a new codebook without changing the parameters .

V.SECURITY ANALYSIS

A. Security Analysis for Secret Little Functions

User specified functions can be infinite. Since attackers do not know the function forms and constant value (i.e., secret encryption algorithms and value), so these simple functions are very secure. Therefore, secret little functions and constant value can easily prevent phishing, shouldersurfing, key-logger, and even multiple attacks. How many successful phishing attacks can be tolerated by secret little functions? The answer is infinite and here a constant value is also added so it becomes more difficult for phishing attack . For key-logger/shouldersuffer, secret little functions and constant value can remain secure after the adversary obtains many dynamic passwords. Next, we prove that the proposed scheme can prevent the following attacks.

1) ***Phishing***. Since each time, the user inputs a dynamic password; the phishing attacker could get the dynamic password, but cannot obtain the real password. The dynamic password is different each time. Also ACT is performed for server and client authentication.

2) ***Shoulder-surfing***. Since each time, the user inputs a dynamic password; the shoulder-surfing attacker could get the dynamic password, but cannot obtain the real password. The dynamic password is different each time.

3) ***Key-logger***. Since each time, the user inputs a dynamic password, the Key-logger attacker could get the dynamic password, but cannot obtain the real password. The dynamic password is different each time.

4) ***Replay attack.*** The dynamic password does not suffer the replay attack since each time, the server generated a different random number. We can prove mathematically that secret little function and constant value is much more secret than any other symmetric cipher (such as DES, AES, and so on) under the brute-force attack with the following theorem, while the brute-force attack is to try every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.

5)***Mobile malwar attack*****:** The dynamic password scheme is implemented in mobile in encrypted form, i.e the database of this scheme is codebook in which all confidential information's are save in encrypted form and again the whole scheme is encrypted

Theorem 1: Under the brute-force attack, secret little function and constant value is much more secret than any other symmetric cipher.
Proof: Let *P*, *C*, and *K* denote the plaintext, ciphertext, and key, respectively, for any given symmetric cipher. Let *LP* , *LC*, and *LK* denote the lengthes of *P*, *C*, and *K*, respectively. For an attacker to try to use the brute-force attack to the symmetric cipher, the number of alternative keys is 2*LK*. In other words, the attacker at worst needs to try 2*LK* to get the correct key. On the other hand, for a secret little function and constant value approach, the number of alternative keys is the number of secret little functions and the value. Since the number of secret little functions and constant value is not only infinite, but also uncountable infinite, the number of alternative keys for the secret little function and constant value approach is infinite, denoted as ∞. Since we have 2*LK* _ ∞, we then prove the theorem.

B. Security Analysis for Codebook:

The codebook can prevent shouldersurfing, mobile malwar attack and key-logger since a human is not likely to tolerate more than two or three phishing attacks without being able to get into the system.

VI. IMPLEMENTATION AND EVALUATION OF SCHEME IN E-BANKING

We implement secret little functions, constant value and demonstrate that they defeat phishing, key-logger, and shoulder-surfing attacks in a PC machine. Our mobile-applications help to relieve the users from calculation. The dynamic scheme is implemented in mobile-application in which ACT is performed and dynamic password is calculated after entering the random salt number which is entered into the sign-on screen by the user.After entering the dynamic password the bank sent an access code to the user's mobile number which is given during registrarion which is generated in online. Here during registration human computation is needed. A user response test is to test the user's feeling on the time spent to register the dynamic password. Our password scheme is dynamic and requires a user to make some computations .

Q/A	How comfortable to do equation calculation?
Very easy with mobile-application	80%
Difficult without a calculator	2%
Ok without a calculator	9%
Easy without a calculator	9%
Q/A	Would you like to improve your password security with a little bit extra time?
Yes	61%
No	6%
Dno't care	16%
Yes, but depends on extra time	17%

Table 1.1 Responses of Users

A survey in Table 1.1 is used to collect 50 users' responses for our system implementation. It was found that most of the respondents could complete the equation calculation easily with mobile-application. This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination The most of the surveyed people showed their need for more secure internet with the cost of spending a little extra time

VII. CONCLUSION

We discussed the challenges of protecting users' passwords on the online banking and how to prevent users' passwords from being stolen by adversaries. We propose dynamic password mechanisms in which a user has a mobile, in that mobile the dynamic password scheme is implemented using Android Operating System, so dynamic password requires a small amount of human computing to secure users passwords. Among the schemes, we have a default method (i.e., traditional password scheme), system recommended functions, user-specified functions, user-specified programs. A function/program is used to implement the dynamic password concept. For user-specified functions, we adopt secret little functions and a constant value, in which security is enhanced by hiding both. The computation of human can be reduce by using mobile applications with builtin dynamic password. Here the user only needs to input the system random digits which the system provides and then the dynamic password is automatically calculated for the user. The user then enter the dynamic password into the sign-on phase and then bank sent access code to mobile number which is provide during registration. This scheme is encrypted and implemented into mobile. Thus we can overcome the main attacks like phishing, key-logger, shoulder-surfing, mobile malwar attacks simultaneously. We further proposed several functions serving as system recommended functions andprovided a security analysis. We analyzed how the proposed schemes defend against phishing, key-logger, shoulder-surfing attacks, mobile malwar attack and multiple attacks. In conclusion, user-defined functions (secret little functions) are better. We believe that for some important accounts such as bank accounts, some users needs more secure, especially when usinga computer in an insure environment such as the Internet cafe.

In the future, we plan to study how to design smarter functions to implement this scheme in all mobile-application. We would also like to develop dynamic passwords scheme, which will be able to run at a customer's wireless device, such as even in simple cellular phone or a PDA. With such an application, the user protect the passwords from the password theft.

REFERENCES

1. T. Dierks and C. Allen, *The TLS Protocol—Version 1.0*, IETF RFC 2246, Jan. 1999.
2. [Online].Available:http://en.wikipedia.org/wiki/Phishing
3. [Online].Available: http://www.eweek.com/article2/0,1895,1940 23,00.asp
4. V. A. Brennen. (2004). *Cryptography Dictionary*, vol. 2005, 1.0.0 ed.: http://cryptnet.net/fdp/crypto/crypto-dict/en/cryptodict. Html
5. M. Kuhn. (1997). *Probability Theory for Pickpockets—ec-PIN Guessing* [Online]. Available:

http://www.cl.cam.ac.uk/?mgk25
6. B. Moller. (1997, Feb.). *Schw¨achen des ec-PIN-Verfahrens* http://www.informatik.tu-darmstadt.de/TI/ Mitarbeiter/moeller
7. *en.wikipedia.org/wiki/**Malware***
8. J. Mason, "Filtering spam with SpamAssassin," in *Proc.HEANetAnnu. Conf.*, 2002.
9. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering june e-mail. In learning for text categorization," in *Proc. Workshop*, May 1998
10. T. A. Meyer and B. Whateley, "SpamBayes: Effective open-source, Bayesian based, e-mail classification system, in *Proc. CEAS*, 2004.
11. MAPS. (1996). *RBL—Realtime Blackhole List* [Online]. Available *and Phishing Attacks*, Cryptology ePrint Archive, Rep. 2004/155 [Online]. Available: http://eprint.iacr.org/2004/155
12. *The Spamhaus Project. The Spamhaus Block List* [Online]. Available http://www.spamhaus.org/sbl
13. E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, A. Tironi, and L. Zaniboni,
14. A. Herzberg and A. Gbara. (2004). *Trustbar: Protecting (Even Naive) Web Users From Spoofing*
15. Netcraft.*Anti PhishingToolba r*[Online].Available: http://www.mail-abuse.com/services/mds−rbl.html
16. C. Herley and D. Florencio, "How to login from an Internet cafe without worrying about keyloggers," in *Proc. SOUPS*, 2006.
17. [Online] http://www.citibank.co.jp/en/service/cap/virtualpad
18. [Online]. Available: http://obr.typepad.com/financial
19. S. Wiedenbeck, J. Waters, L. Sobrado, and J. Birget, "Design and evaluation of a shoulder- surfing resistant graphical password scheme," in *Proc. Working Conf. Adv. Vis. Interfaces*.
20. G. T. Wilfong, "Method and apparatus for secure PIN entry," U.S. #5 940 511, United States Patent and Trademark Office, Assignee: Lucent Technologies, Inc., Murray Hill, NJ, May 1997.
21. Differentiated Virtual Passwords, Secret Little Functions,and Codebooks for Protecting User From Password Theft Yang Xiao, *Senior Member, IEEE,* Chung-Chih Li, Ming Lei, and Susan V. Vrbsky
22. A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security protocols for sensor networks," *Wirel. Netw.* vol. 8, no. 5, pp. 521–534, 2002

AUTHORS PROFILE

Shimna M. S., B.Tech, doing M Tech in computer science and engineering at Vedavyasa Institute of Technology Malappuram, Kerala, India.

Sangeetha P. S., B.Tech, MS(IT) Working as a assistant professor . department of computer science , Vedavyasa Institute of Technology, Malappuram, Kerala, India.