Computer Communications 31 (2008) 4367–4375

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Virtual password using random linear functions for on-line services, ATM machines, and pervasive computing

Ming Lei [a], Yang Xiao [a,*], Susan V. Vrbsky [a], Chung-Chih Li [b]

[a] Department of Computer Science, The University of Alabama, Box 870290, Tuscaloosa, AL 35487, USA
[b] School of Information Technology, Illinois State University, Normal, IL 61790, USA

ARTICLE INFO

Article history:
Available online 16 May 2008

Keywords:
Security
Password
On-line services
User ID
ATM machines

ABSTRACT

People enjoy the convenience of on-line services, Automated Teller Machines (ATMs), and pervasive computing, but online environments, ATMs, and pervasive computing may bring many risks. In this paper, we discuss how to prevent users' passwords from being stolen by adversaries. We propose a virtual password concept involving a small amount of human computing to secure users' passwords in on-line environments, ATMs, and pervasive computing. We adopt user-determined randomized linear generation functions to secure users' passwords based on the fact that a server has more information than any adversary does. We analyze how the proposed schemes defend against phishing, key logger, and shoulder-surfing attacks. To the best of our knowledge, our virtual password mechanism is the first one which is able to defend against all three attacks together.

1. Introduction

Users with important accounts on the Internet face many kinds of attacks, e.g., a user ID and password can be stolen and misused. There are many reports on thefts on ATMs as well. The secure protocol SSL/TLS [1] for transmitting private data over the web is well known in academic research, but most current commercial websites still rely on the relatively weak protection mechanism of user authentications via a plaintext password and user ID. Meanwhile, even though a password can be transferred via a secure channel, this authentication approach 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. Fig. 1 illustrates the total phishing reports received from September 2005 to September 2006 [2] according to the anti-phishing working group.
- *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. 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. Trojan Redirector was designed to redirect end-users network traffic to a location to where it was not intended [3]. This includes crime ware that changes host files and other Domain Name Service (DNS) specific information, crime ware browser-helper 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 front of terminals [10] or ATMs. This attack is most likely to occur in insecure and crowded public environments, such as an Internet Café, shopping mall, airport, etc. [14,18]. It is possible for an attacker to use a hidden camera to record all keyboard actions of a user for both a computer and an ATM machine. Video of the user's actions on a keyboard can be studied later to figure out a user's password and ID.

Many schemes, protocols, and software have been designed to prevent users from some specified attacks. However, to the best of our knowledge, so far, there is not a scheme which can defend against all three types of attacks listed above at the same time.

In this paper, we present a password protection scheme that involves a small amount of human computing in an Internet-based

* Corresponding author. Tel.: +1 205 348 4038; fax: +1 205 348 0219.
E-mail address: yangxiao@ieee.org (Y. Xiao).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.05.005

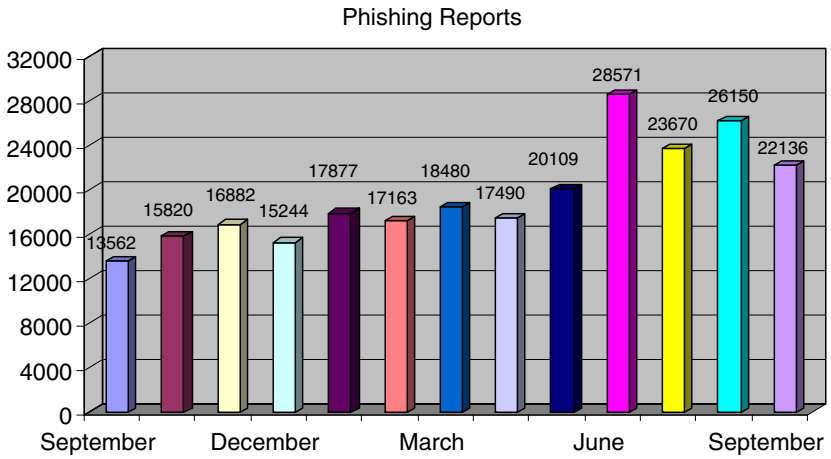

Fig. 1. Phishing report received September 2005–September 2006 [2].

environment, ATM, or pervasive computing, which will be resistant to a phishing scam, a Trojan horse, and shoulder-surfing attacks. We propose a virtual password concept that requires a small amount of human computing to secure users' passwords in on-line environments, ATM, and pervasive computing. We adopt user-determined randomized linear generation functions to secure users' passwords based on the fact that a server has more information than any adversary does. We analyze how the proposed scheme defends against phishing, Trojan horses, such as key loggers, and shoulder-surfing attacks. To the best of our knowledge, our virtual password mechanism is the first one which is able to defend against all three attacks together.

The idea of this paper is to add some complexity, through user computations performed by heart/hand or by computation devices, to prevent the three kinds of attacks. There is a tradeoff of how complex the computation by the users can be. One goal is to find an easy to compute but secure scheme for computing.

We believe that for some sensitive accounts such as on-line bank accounts, on-line credit card accounts, and ATMs, users are likely to choose a little additional complexity requiring some degree of human computing in order to make the account more secure.

The rest of the paper is organized as follows. In Section 2, we propose the virtual password scheme. We propose randomized linear generation functions in Section 3. In Section 4, we describe implementation issues of our scheme. We describe related work about password protection in Section 5. Finally, we conclude our paper and describe our future work in Section 6.

2. Virtual password

2.1. Virtual password concept

To authenticate a user, a system (S) needs to verify a user (U) via the user's password (P) which the user provides. In this procedure, S authenticates U by using U and P, which is denoted as: S → U: U, P. All of S, U, and P are fixed. It is very reasonable that a password should be constant for the purpose of easily remembering it. However, the price of easy to remember is that the password can be stolen by others and then used to access the victim's account. At the same time, we cannot put P in a randomly variant form, which will make it impossible for a user to remember the password. To confront such a challenge, we propose a scheme using a new concept of virtual password.

A *virtual password* is a password which cannot be applied directly but instead generates a *dynamic password* which is submitted to the server for authentication. A virtual password P is composed of two parts, a fixed alphanumeric F and a function B from the domain ψ to ψ, where the ψ is the letter space which can be used as passwords. We have $P = (F, B)$ and $B(F, R) = P_d$, where R is a random number provided by the server (called the random salt and prompted in the login screen by the server) and P_d is a dynamic password used for authentication. Since we call $P = (F, B)$ a virtual password, we call B a *virtual function*. The user input includes (ID, P_d), where ID is user ID. On the server side, the server can also calculate P_d in the same way to compare it with the submitted password.

It is easy for the server to verify the user, if B is a bijective function. If B is not a bijective function, it is also possible to allow the server to verify the user if it is at least injective as follows. The server can first find the user's record from the database based on the user' ID, and compute P_d, and compare 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 virtual password. The user should be free to pick the fixed part of the virtual password.

2.2. Virtual password usage

We have introduced the concept of the virtual password, and next, we detail how to apply it in an internet-based environment.

To use a virtual password, human computing is involved or a handhold device 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 run at the mobile device, such as a cellular phone, PDA, personal computer, or programmable calculator, to relieve the user from the complicated calculations and to overcome any short-term memory problem. If such a helper-application is involved, we should make sure that the helper-application itself should be unique to each user account and only work 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 virtual password scheme, there is a virtual function, which is a must, to 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 email. The user needs to remember this function together with the password they have chosen or save them in disks or emails.

We also note that a small amount of human-computing is involved in the authentication process. We have to choose B to make the calculation as simple as possible if the helper-application is not

used. A user has to remember both the fixed part and the function part, and as a result will require a little bit more effort to remember. However, the virtual password will be resistant to a dictionary attack, which is mostly caused by the fact that users like to create a password which is either related to their own name, date of birth, other simple words, etc.

In a traditional password scheme, users can change their passwords, and this fact is also true in our virtual password scheme. Different from the traditional scheme, users can change the fixed part of the virtual password or the virtual function, or even both.

2.3. Virtual function with a helper-application

If a helper-application is available for the user, the user needs to type the random salt into the helper-application, and subsequently, the dynamic password is generated by the helper-application. Then the user types in the generated dynamic password in the login screen. In this way, the extra time required is very small and the precision will be one hundred percent correct as long as the user types the correct random salt displayed on the login screen.

This works for the case when the user has a mobile device, such as a cellular phone or PDA. 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 virtual function is, the helper-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 get a new helper-application after the password change instead of remembering all the changed parts of the virtual password. Note that the server must make the corresponding changes too.

A one-way hash function and many of their functions (such as some known encryption algorithms) can serve as virtual functions.

2.4. Virtual function without a helper-application

If there is no helper-application for a user, the user needs to calculate the dynamic password from the virtual function with the inputs, random salt and the fixed part of the virtual password. The whole login process may take a little bit longer because it requires the user to perform some calculations. This must work for the user who has no mobile device, so in that case, the virtual function should not be too complicated for human computing.

For password changing, it is similar to traditional password changing. The user can choose a new password, which is the fixed part of the virtual password or a new virtual function, or both. After such changes, the user needs to remember the new virtual password.

The virtual function plays a critical role in the virtual password. There are an infinite number of virtual functions, so that designing an appropriate function is very critical to the success of our scheme.

In order to defend against phishing, key-loggers, and shoulder-surfing while the system is authenticating the user, this function should meet the following criteria:

- The function should have some random input provided by the server, which then allows the users to type in different inputs each time they log in the system. This ensures the key logger cannot steal the password because the real password is not typed and the typed inputs change each time.
- The function should be easy for the users. To make the system more secure, we could increase the complexity of the virtual function. However, this resulting function may be very difficult to remember or utilize. The objective is to design less complex but secure virtual functions.
- The function should be *unobservable*, i.e., the observed password the user types in for the login session does not disclose hidden secrets; therefore, adversaries cannot use the stolen information to login to the system.
- The function should be *insolvable*, i.e., the adversaries should not be able to solve the function with all the potential information they are able to obtain.

There are some functions which meet all the requirements which we listed above.

3. Randomized linear generation functions

In this section, we propose examples of virtual functions and analyze how they secure a user password. However, our proposed approach is not limited by these examples. We consider digits here as an example, but our scheme is not limited in the number of digits, nor is it even limited to using only digits.

3.1. Early attempts

In the beginning we pursued bijective functions as virtual functions since we thought that since bijective functions had reverse functions, it was easy to verify a user by the server. Later, we believed that it was not necessary for virtual functions to be bijective, but injective.

We began by using simple operators, such as plus or minus some secret value, e.g.,

$$B(x) = (x + a) \bmod Z, \tag{1}$$

which is easy to use, where x is the fixed part of the user's virtual password and a is the secret value. We immediately excluded this case of Eq. (1), since it is easy to attack [24] and violates the first requirement from the previous section because it has no random variable. We concluded that we could use a simple linear function that includes a random variable, such as

$$B(x) = (a * x + y) \bmod Z, \tag{2}$$

where x is the fixed part of the user's virtual password, y is a random number the system provides to the user in each login session, and a is kept as secret. This function (2) involves the random factor, but it violates the unobservable rule. This is because once the adversary obtains the function output k and random factor y (through shoulder-surfing or phishing), they can easily login to the system based on the stolen information. This is because with k and y, they can deduce $(a * x) \bmod Z = (k - y) \bmod Z$. Then the adversary can login to the system with $(a * x + y') \bmod Z$, where y' is a new random number.

Then, we considered the function

$$B(x_i) = ax_i + y_i \bmod Z, \tag{3}$$

where x_i is one digit of the password, y_i is a random number provided by the server, and a is kept as secret. However, we discovered that this function is subject to shoulder-surfing attacks. For example, if a hidden camera records a user's activity (actions of key boards), then it is easy to obtain ax_i if y_i is known.

For example, suppose that you have a virtual password 123 and the function $(3 * x + y) \bmod 10$. In the login session, the system provides the random number, 456. You calculate $(3 * 1 + 4) \bmod 10 = 7$, $(3 * 2 + 5) \bmod 10 = 1$, and $(3 * 3 + 6) \bmod 10 = 5$, so that it is 715. Once the adversary steals the random number and the dynamic password the user input, the adversary can determine $3 * x_1 = 3$, since $(7 - 4) \bmod 10 = 3$, $3 * x_2 = 6$ since $(1 - 5) \bmod 10 = 6$, and $3 * x_3 = 9$ since $(5 - 6) \bmod 10 = 9$. Now, the adversary is able to access the system. If the system generated random number is 789, the adversary can enter the system by providing the

password 048, where $0 = (7 + (7 - 4)) \bmod 10$, $4 = (1 + (8 - 5)) \bmod 10$, and $8 = (5 + (9 - 6)) \bmod 10$. This is the correct real password.

3.2. Further attempts

Next, we added a constant factor to the linear function:

$$B(x_i) = [a(x_i + y_i) + c] \bmod Z, \tag{4}$$

where a and Z are relatively prime, x_i is one digit from the fixed part of the user's virtual password, y_i is one random digit provided by the system, and a and c are the constant factors of the linear function, which the user has to remember. The $B(x_i)$ is a bijective function if and only if $\gcd(a,Z) = 1$ [26], where gcd denotes the Greatest Common Divider function.

In fact, we can prove that this function can withstand phishing, key logger, and shoulder-surfing attacks by the follow security analysis. Let $x_1,x_2,\ldots,x_n$ and $[a(x_i + y_i) + c] \bmod Z$ denote the user's fixed password, and the virtual password, respectively, where $y = y_1,y_2,\ldots,y_n$ is random number provided by the system/server.

- *Defense against phishing:* For phishing, once a user is lured to type in the dynamic password $(k_1,k_2,\ldots,k_n)$ in the faked page, then the dynamic password will be recorded by the adversaries. However, we claim that this does not help the adversaries to figure out the virtual password of the user, because it is impossible to get the solution of $x_1,x_2,\ldots,x_n$, a, and c based on the information they have stolen. We can present the digits the phisher caught in the form of equations and there are n equations the phisher can build as: $[a(x_i + y_i) + c] \bmod Z = k_i$, (for $i = 1,\ldots,n$), where y_1 *to* y_n and k_1 *to* k_n are known by the phisher. However, since there are $n + 2$ variables (a, c, and x_1 to x_n), but only n equations, it is impossible to solve the equations to get the solution. Therefore, we claim that our scheme is phishing-proof.
- *Defense against the key logger:* The key logger code logs all the key strokes at the operating system level so that such logs are delivered to some adversaries who analyze what the victim has keyed in their system, and then try to extract the user password. Such a key logger will be very effective if the user typed their password in an unsafe machine on which the key logger is installed. In our scheme, the key logger can still catch the entire user's key strokes, but they still need to solve the above mentioned n equations, where the only variables the logger can be aware of are k_1 to k_n. We also assume that the user does not add some noise into the logger as [15]. However, if noises are created, we claim that the adversaries are not able to have the equations at all. Even if they build the equations, they cannot solve the function because the available knowledge for the adversaries is not enough.
- *Defense against shoulder-surfing:* To protect the user from shoulder-surfing, we assume that the watchers have a good memory or they use some other devices, such as a camera, to record all the information that the user will use, including the random digits that the system provides in the screen and the keys the user types into the password field. With the help of the virtual password, at the prospective of the watchers, the information available to them are the dynamic password $k_1,k_2,\ldots,k_n$ and the random digits $y_1,y_2,\ldots,y_n$. The watcher will face the same challenge that the phisher encounters, and will have difficulty in solving the above mentioned n equations.

We have claimed that function (4) will protect the user from a phishing attack, key loggers, and shoulder-surfing. However, we later found out that function (4) has at least one drawback, e.g., it cannot stand for multiple attacks as follows.

- *Challenge for multiple attacks:* If the user is lured to try to login to any phishing website more than twice, it will leak his/her password. Suppose that Bob has a set up a phishing website abc.com and Alice is a user of the website with password $k_1,k_2,\ldots,k_n$ with Eq. (4). Now if Alice tries to login twice to the phishing website abc.com, Bob can compare the dynamic passwords which Alice input, and then Bob can easily figure out how to login to the real website with Alice's account. The reasons are explained as follows. For any given ith digit of the fixed part of Alice's virtual password, if Alice has tried more than twice to login to the fake website, then Bob could obtain the two equations below: $[a(x_i + y_i) + c] \bmod Z = k_i$, and $[a(x_i + y'_i) + c] \bmod Z = k'_i$. Now Bob can know that $[a(y'_i - y_i)] \bmod Z = (k'_i - k_i) \bmod Z$, and as a result, it can calculate a. After a is identified by the adversary, the system is broken. Then Bob can use Alice's account to login to the real website in the following way. For the ith digit, Bob can just type in $(k_i + a(y''_i - y_i)) \bmod Z$, where k_i is the first time Alice typed in the ith digit in the fake website, y_i is the ith random digit provided by the fake website, and y''_i is the ith digit the system will display on the screen, which Bob needs to login to Alice account.

Such leaking can also occur in shoulder-surfing if the watcher can record all the information for the same victim twice.

We continued in our attempts to design a virtual function. The result was to make Eq. (4) more complex. We revised Eq. (4) to become Eq. (5) so that is uses the value of a digit in the dynamic password to calculate a subsequent digit in the dynamic password. We called this new function a *randomized linear generation* function because c is random number as follows.

$$B(x_i) = \begin{cases} k_1 = (ax_1 + y_1 + c) \bmod Z \\ k_i = (ak_i - 1 + y_i + cx_i) \bmod Z', \end{cases} \tag{5}$$

where a is a constant which the user needs to remember but c is not. The most interesting part of the function is that c will be a random number which the user randomly chooses each time the user tries to login to the system. Since $\gcd(a,Z) = 1$, the above function is also a bijective function regardless of the c value. How the server authenticates the user's validity is different from the previous scheme based on Eq. (5). Because c is also unknown to server, the server knows that $c \in \{0,1,\ldots,Z-1\}$. The authentication is the same as that for Eq. (4) and can be done as follows.

Let $B^{-1}(x)$ be the reverse function of $B(x)$. After the server gets the user's keyed dynamic password $k_1,k_2,\ldots,k_n$, and the fixed part of the virtual password of the user, $x_1,x_2,\ldots,x_n$, the server can perform the following verification in Table 1.

The algorithm above guarantees that if the user has input the correct password, the system will grant him/her entrance whatever the random number he/she picked. However, it is also true that for each user, there will be multiple (exactly Z) acceptable dynamic passwords existing for each specific login session. This may increase the probability that the adversary's random input happens to be the correct password. However, if length of the password is long enough, the probability is very small, i.e., $Z/2^n$, where n is the length of the password.

Table 1
Verification

```
Verify()
{For each digit u ∈ {0,1,...,Z − 1}{
For each digit in the dynamic password the user typed
{w_i = B^{-1}(k_i, u)}
if (w_1,w_2,...,w_n = x_1,x_2,...,x_n) return true}
Return false
}
```

The scheme with Eq. (5) can defend against phishing, key logger, shoulder-surfing, and multiple attacks as follows:

- *Defense against phishing, key loggers, and shoulder-surfing:* It protects the user's from password stealing based on the same theory that the adversary cannot solve the function because the adversary does not have enough information. We only use phishing as an example here. We now list the following equations: $k_1 = (ax_1 + y_1 + c) \bmod Z$ and $k_i = (ak_{i-1} + y_i + cx_i) \bmod Z$ $(i = 2,\ldots,n)$. For the phisher, the $c,a,x_1,\ldots,x_n$ are unknown, and they only know the $k_1,k_2,\ldots,k_n$ and $y_1,y_2,\ldots,y_n$, so that they cannot solve the function to get the solution. These similar unsolvable equations exist against key loggers and shoulder-surfing.
- *Defense against multiple attacks:* We claim that multiple dynamic password leaking in the virtual password scheme with a linear generation function can be secured using a random number. If an adversary has tricked a user into logging into his fake website twice, the adversary obtains $k_1 = (ax_1 + y_1 + c) \bmod Z$ and $k_1' = (ax_1 + y_1' + c\prime) \bmod Z$, where a, x_1, c, and c' are unknown to the adversary, and then what information the adversary can figure out is the $(c\prime - c) \bmod Z = (y_1' - y_1) + (k_1' - k_1)$. Since the c and c' were randomly chosen by the user, $(c' - c)$ does not provide any information. If the adversary cannot work out some clue about the first digit of the dynamic password k_1, he/she cannot find about k_2 and later digits in the dynamic password. Therefore, using the linear function with a random number can remove the possibility of multiple dynamic password attacks.

However, at this moment, we discovered that both the scheme with Eq. (4) and the scheme with Eq. (5) suffer a drawback: a small key space.

For Eq. (4), i.e., $B(x) = [a(x + y) + c] \bmod Z$, if $Z = 10$, the choice of a is 4 since $\gcd(a,Z) = 1$ and the choice of c is 10. Therefore, the total number of choices for pair (a,c) is 40, i.e., the key space is 40. With each choice of pairs (a,c), and y, an attacker can guess 40 choices of x values.

The key space of size 40 is not acceptable for on-line services, but it may be OK for ATMs. It is different from 40 phishing attacks, since a human is not likely to tolerate more than 2 or 3 phishing attacks without being able to get into the system. But the 40 keys is for the adversary to login the system, not to phish a human. On average, 20 tries can let him get into the system. Even if the system is designed to lock the account after 3 wrong passwords are entered, the adversary can collect 10 successful phishing results from 10 different users. This can be easily done. Then, for each account, he can try 3 passwords, and the chance he could get onto at least one account is higher than 1/2, (precisely, $1-(37/40)^{10}$).

Eq. (5) also suffers the same problem of a small key space.

Hence, we considered a little more complex function as follows.

$$B(x_i) = [a(x_i + y_i) + x_{[(i+1) \bmod n]+1}] \bmod Z, \tag{6}$$

where n is the length of the fixed part of the virtual password.

We can easily demonstrate that the above function protects the system from phishing, key logger, and shoulder-surfing attacks with similar methods from those for Eq. (4).

However, the key space using Eq. (6) is still 40 after one successful phishing. The adversary can guess a and x_1, and then he can solve x_2, then x_3, and so on. In other words, each pair of (a, x_1) determines the whole x.

Furthermore, similar to the scheme in Eq. (4), we can prove that the above function is subject to multiple attacks. Such leaking can also occur in shoulder-surfing if the watcher can record all the information for the same victim twice.

We then considered the following function

$$B(x_i) = \begin{cases} k_1 = (ax_1 + y_1 + x_2 + c) \bmod Z, i = 1 \\ k_i = (aK_{i-1} + y_i + x_i + cx_{i+1}) \bmod Z, i = 2,\ldots,n \end{cases} \tag{7}$$

where c is a user-chosen random number.

The random constant c is irrelevant when $i > 1$ since the adversary can pick up any one just like the user can. So, the adversary can choose 0 to cancel the effect.

Finally, instead of cx_{i+1}, we chose $c + x_{i+1}$ (since they can basically function the same, i.e., adding some randomness) to result in Eq. (8).

3.3. Random Linear Function

We then propose a function that uses the value of a digit in the dynamic password to calculate a subsequent digit in the dynamic password. We call this new function also a *randomized linear generation* function:

$$B(x_i) = \begin{cases} k_1 = (ax_1 + y_1 + x_2 + c) \bmod Z, i = 1 \\ k_i = (ak_{i-1} + y_i + x_i + c + x_{i+1}) \bmod Z, i = 2,\ldots,n \end{cases} \tag{8}$$

where a is a constant which the user needs to remember but c is not. The most interesting part of the function is that c will be a random number which the user randomly picks each time when the user tries to login to the system. Since $\gcd(a, Z) = 1$, the above function is also a bijective function regardless of the c value. Because c is also unknown to server, the server knows that $c \in \{0,1,\ldots,Z-1\}$. The authentication could be done as follows.

Let $B^{-1}(x)$ be the reverse function of $B(x)$. After the server gets the user's keyed dynamic password $k_1,k_2,\ldots,k_n$, and the fixed part of the virtual password of the user, $x_1,x_2,\ldots,x_n$, the server can perform the verification of Table 1. The algorithm above guarantees that if the user has input the correct password, the system will grant him/her entrance whatever the random number he/she picked. However, it is also true that for each user, there will be multiple (exactly Z) acceptable dynamic passwords existing for each specific login session. This may increase the probability that the adversary's random input happens to be the correct password. However, if the length of the password is long enough, the probability is very small, i.e., $Z/2^n$, where n is the length of the password.

A scheme with Eq. (8) can defend against phishing, key logger, shoulder-surfing, and multiple attacks as follows.

- *Defense against phishing, key logger, and shoulder-surfing:* It protects the user's from password stealing based on the same theory that the adversary cannot solve the function because the adversary does not have enough information. We only use phishing as an example here. We now list the following equations: $k_1 = (ax_1 + y_1 + x_2 + c) \bmod Z$ and $k_i = (ak_{i-1} + y_i + x_ic + x_{i+1}) \bmod Z$ $(i = 2,\ldots,n)$. For the phisher, the c, a, $x_1,\ldots,x_n$ are unknown, and they only know the $k_1,k_2,\ldots,k_n$ and $y_1,y_2,\ldots,y_n$, so that they cannot solve the function to obtain the solution. These similar unsolvable equations exist against key loggers and shoulder-surfing.
- *Defense against multiple attacks*: We claim that multiple dynamic password leaking in the virtual password scheme with a linear generation function can be secured using a random number. If an adversary has tricked a user into logging into his fake website twice, the adversary obtains $k_1 = (ax_1 + y_1 + x_2 + c) \bmod Z$ and $k_1' = (ax_1 + y_1' + x_2 + c\prime) \bmod Z$, where a, x_1, c, and c' are unknown to the adversary, and then what information the adversary can figure out is the $(c\prime - c) \bmod Z = (y_1' - y_1) + (k_1' - k_1)$. Since the c and c' were randomly chosen by the user, $(c' - c)$ does not provide any information. If the adversary cannot work out some clue about the first digit of

the dynamic password k_1, he/she cannot find about k_2 and the later digits in the dynamic password. Therefore, using the linear function with a random number can remove the possibility of multiple dynamic password attacks.

3.4. Comments on functions (1)–(8)

Although some functions among (1)–(8) are not perfect as discussed, they are much more secure than using no functions, i.e., $f(x) = x$. There is always a tradeoff between complexity and security. We believe that some functions of (1)–(8) or their variants will be very useful for on-line services, ATMs, and pervasive computing, regardless of their drawbacks.

4. Implementation and evaluation

In order to implement the virtual password scheme to safeguard users when they are surfing online, we implemented the scheme, and demonstrate that a little human computing can defeat phishing, key logger, and shoulder-surfing attacks. In this section, we will evaluate our practical implementation of the virtual password scheme.

4.1. Screenshots of system implementation

We just implemented a simple scheme for illustration purposes, e.g., the randomized linear function. With the consideration of user usability, we set $Z = 10$, so that the available values for a are $\{1,3,7,9\}$. This does not decrease the scheme's security due to the limited value of a, since in the linear generation function, the value of the dynamic password will rely heavily on the random c. The function $K_n = (aK_n - 1 + yn + xn + c + x_{n+1}) \bmod 10$ may be too difficult for users to calculate in their mind, especially since K_{n-1} is hard to remember. In our implementation, we will allow the system to display the password file without marking the content as "*", which will make it easier for the user to know the previous digit he/she has entered. In Fig. 2, we demonstrate a testing website in which a virtual password scheme was implemented.

Even though such a calculation is a little complicated for some people, our helper-applications could relieve the users of this required human computing. In Figs. 3 and 4, we implement two versions of such helper-applications for a personal computer and a mobile device, respectively.

4.2. Password-security survey for users' responses

Currently, most of the websites allow a user to have only one fixed unique password. In our scheme, however, the password is dynamic and a user needs to make some computations for each login if the user has not installed the helper-application, which is significantly different from the traditional way that the user just inputs a password. The traditional way may seem more comfortable to the user, but the price of such comfortableness is that the password could be stolen by adversaries. If considering the fact that users tend to pick passwords that are usually used in cross-systems for easy recall, or those related to the users' privacy, such as DOB, nick name, and so on, the traditional password is more vulnerable. Although it is tedious for users to make some calculations each time to login to the system, the well-trained user can finish the entire login process in a short time.

We distributed a survey (Figs. 5–11) to collect users' responses for our system implementation with a total of 86 responses. We found that the respondents have an average of 10 or more online accounts, as shown in Fig. 5, but the majority of them are unaware of how to defend against phishing, key loggers, and shoulder-surfing. Meanwhile, many of the respondents have no idea about what the three attacks are as illustrated in Figs. 6–8. This severe fact indicates that it is urgent to take some action to protect innocent users from those types of attacks. As shown in Fig. 9, we also found

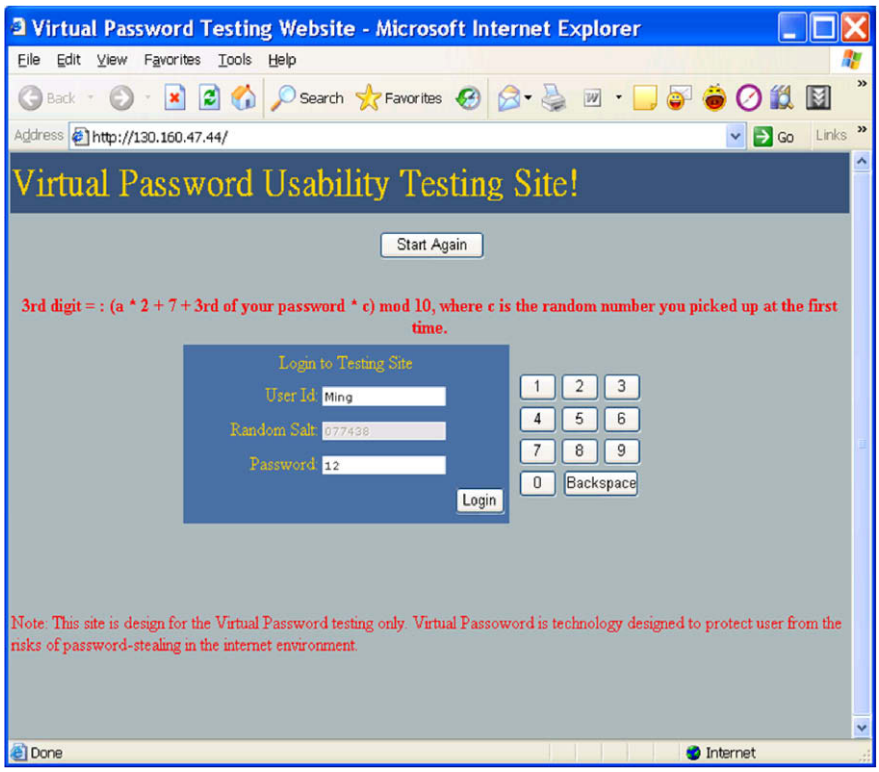

Fig. 2. Testing website login page.

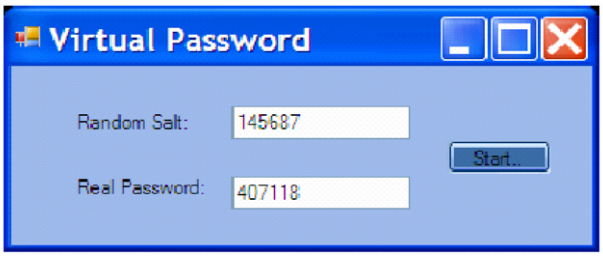

Fig. 3. Helper-application for personal computer.

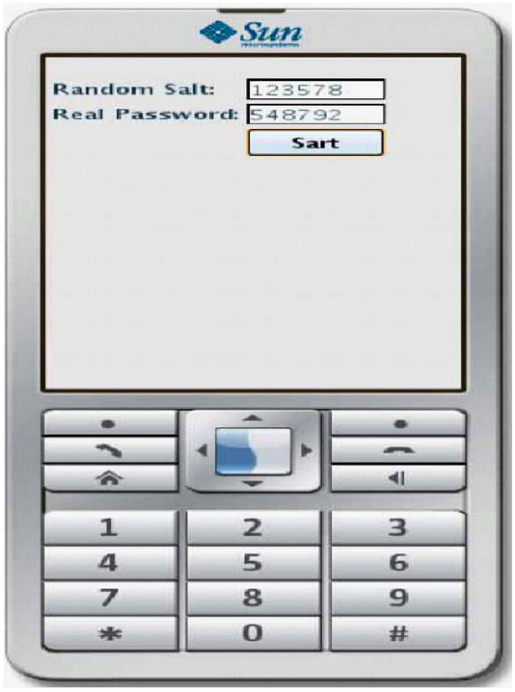

Fig. 4. Helper-application for cell phone.

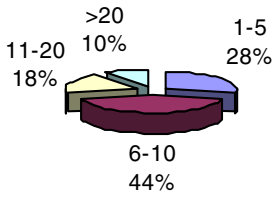

Fig. 5. How many online account you have?

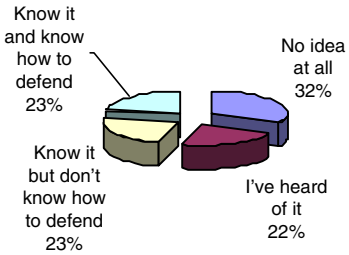

Fig. 6. Knowledge about phishing attack.

that most of the people could complete the single digit calculation easily, without help from the calculator. This makes our virtual

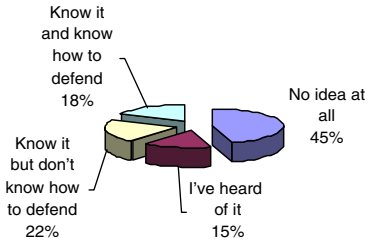

Fig. 7. Knowledge about key logger attack.

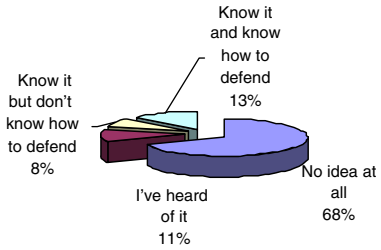

Fig. 8. Knowledge about shoulder-surfing attack.

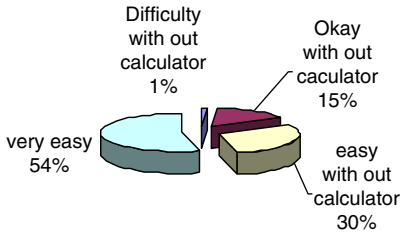

Fig. 9. How comfortable to do the single digit calculation?

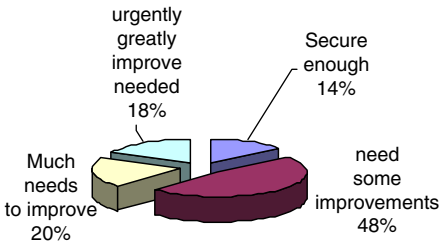

Fig. 10. Current internet secure enough to protect your password?

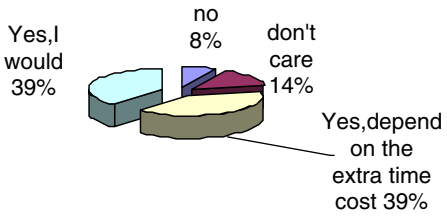

Fig. 11. Would you like to improve your password security with a little bit extra time?

password scheme applicable even for people who do not have any mobile devices with them. As we described in the previous section, we could design some simple bijective function as a randomized generation function to allow for easy human computing. In Figs. 10 and 11, respondents express their demand for a more

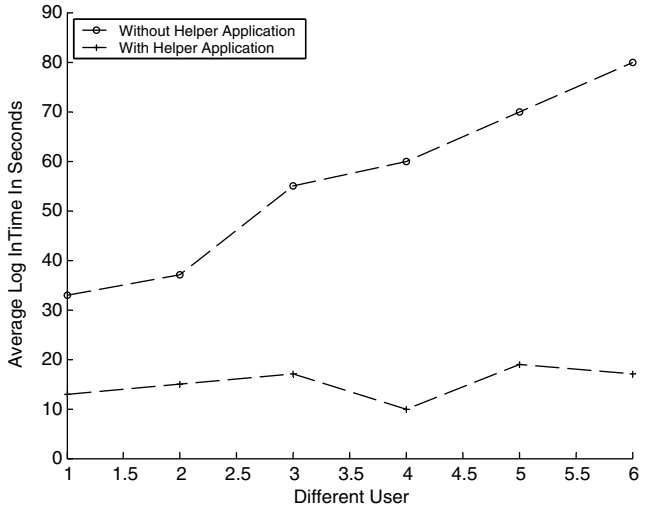

Fig. 12. Average log in time.

secure internet and most of them would accept the cost of spending a little more extra time to sign onto the system for an improvement in password security. We argue that the extra time will be acceptable to most of the people based on Fig. 9. Furthermore, if the helper-application is available for users, the extra time will be very small, and there is no extra time at all if a user's mobile device can communicate with the server.

4.3. Usability test

We test whether the approach is easily used by users or not. In order to test the usability of our scheme, we conducted a usability test with six student volunteers. In our testing, each volunteer was asked to try to login to our test website for two rounds. For the first round, they needed to calculate the password by themselves, and for the second round, they used a helper-application to calculate the password for them. They completed each round 10 times and recorded the time it took them to complete their login. Figs. 12 and 13 demonstrate the usability testing results.

From Figs. 12 and 13, we can see that the user time to login to the system if they do not utilize a helper-application can vary depending on their ability to perform simple calculations.. The

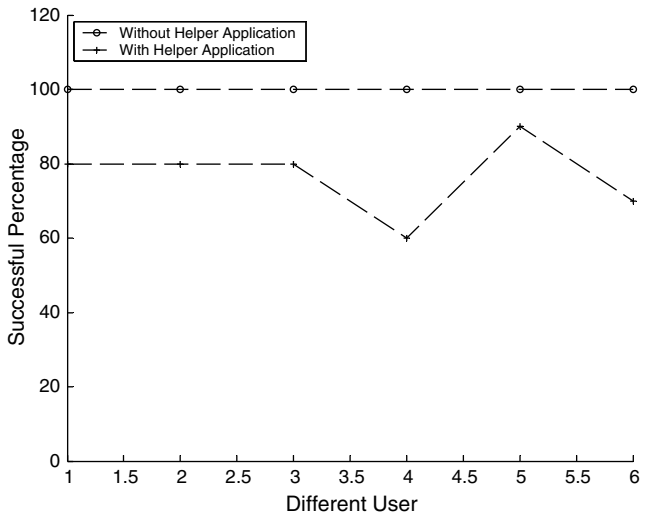

Fig. 13. Average successful rate (in percentage).

users will take an average of 47 s to login to the system, but with help from the helper-application which runs at the mobile device or lap top, users will only take around 15 s to login to the system. In another aspect, without a helper-application, the login success rate is around 75%, but with a helper-application, the login success rate will be always 100%. We argue that it is worth it to take a little bit longer (an extra 10 s with a helper-application) to login to the system, if such a login will be guaranteed secure. This is especially important when a user logs into some important system via the internet, such as an online banking account or credit card account.

5. Related work

Phishing attacks are relatively new but very effective. There are two typical types of phishing. First, to prevent phishing emails [25,27,28], a statistical machine learning technology is used to filter the likely phishing emails; however, such a content filter does not work correctly all the time. Blacklists of spamming/phishing mail servers are built in [29,30]; however, these servers are not useful when an attacker hijacks a virus-infected PC. In [9,22,23], a path-based verification has been introduced. In [12], a key distribution architecture and a particular identity-based digital signature scheme have been proposed to make email trustworthy. Second, to defend against phishing websites, the authors in [19,31] have developed some web browser toolbars to inform a user of the reputation and origin of the websites which they are currently visiting. In [4–8], the authors implement password hashing with a salt as an extension of the web browser [4,7,8], a web proxy[11], or a stand alone Java Applet [13]. Regardless of the potential challenges considered in an implementation, such password hashing technology has a roaming problem and other problems.

Unlike phishing, malicious Trojan horses, such as a key logger, are not attacks, and sophisticated users can avoid them. Such programs are also easy to develop [15] and there is a great deal of freeware that you can download from the internet to prevent them.

Alphanumeric password systems are easily attacked by shoulder-surfing, in which an adversary can watch over the user's shoulder or record the user motions by a hidden camera when the user types in the password. In [20], the authors adopt 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. In [21], the authors propose a scheme to ask a user to answer multiple questions for each digit. In this way, it is resistant to shoulder-surfing only to a limited degree, because if an adversary catches all the questions, then they will know what the password is. In [21], a game-based method is designed to use cognitive trapdoor games to achieve a shield for shoulder-surfing. The author in [24] has 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 [24] is not anti-phishing and the password can possibly be stolen if an adversary uses a camera to record all the screens of the system and motions of the victim. Any of the schemes above cannot prevent against phishing, Trojan horse, and shoulder-surfing at the same time. There are other related research [16,17,32]. Some of the linear functions adopted in this paper are in fact linear congruential generator functions, which were also been used as ciphers in [33] and [34].

6. Conclusion

In this paper, we discussed how to prevent users' passwords from being stolen by adversaries. We proposed a virtual password

concept involving a small amount of human computing to secure users' passwords in on-line environments, ATMs, and pervasive computing. We adopted user-determined randomized linear generation functions to secure users' passwords based on the fact that a server has more information than any adversary does. We analyzed how the proposed scheme defends against phishing, Trojan horses, such as key loggers, and shoulder-surfing attacks. We also implemented the system to do some tests and survey feedback indicates the feasibility of such a system.

We believe that proposed virtual functions or their variants will be very useful for on-line services, ATMs, and pervasive computing, regardless of their drawbacks. We do not prefer very-secure but un-practical approaches in our research.

Our current and future work is to design more intelligent virtual functions.

Acknowledgements

The authors thank Li Liu's involvement in the preliminary work. Dr. Yang Xiao's work is partially supported by US National Science Foundation (NSF) under the Grant CNS-0716211.

References

[1] T. Dierks, C. Allen. The TLS Protocol – Version 1.0, IETF RFC 2246, January, 1999.
[2] Anti-phishing working group. Available from: <http://www.antiphishing.org/>.
[3] Available from: <http://www.eweek.com/article2/0,1895,1940623,00.asp/>.
[4] B. Ross, C. Jackson, N. Miyake, D. Boneh, J. Mitchell, Stronger password authentication using browser extensions, in: Proceedings of 14th USENIX Security Symposium.
[5] E. Gaber, P. Gobbons, Y. Mattias, A. Mayer, How to make personalized web browsing simple, secure, and anonymous, in: Proceedings of Financial Crypto'97, LNCS, vol. 1318, Springer-Verlag, 1997.
[6] E. Gabber, P. Gibbons, D. Kristol, Y. Matias, A. Mayer, On secure and pseudonymous user-relationships with multiple servers, ACM Transactions on Information and System Security 2 (4) (1999) 390–415.
[7] E. Jung, Passwordmaker. Available from: <http://passwordmaker.mozdev.org/>.
[8] J. la Poutr'e, Password composer. Available from: <http://www.xs4all.nl/?jlpoutre/BoT/Javascript/PasswordComposer/>.
[9] J.R. Levine, A flexible method to validate SMTP senders in DNS, April 2004. Available from: <http://www1.ietf.org/proceedings_new/04nov/IDs/draft-levine-fsv-01.txt/>.
[10] V.A. Brennen, Cryptography Dictionary, vol. 2005, 1.0.0 ed., 2004.
[11] Available from: <http://www.bell-labs.com/project/lpwa/>.
[12] E. Damiani et al., Spam attacks: P2P to the rescue, in: Proceedings of Thirteenth International World Wide Web Conference, 2004, pp. 358–359.
[13] M. Abadi, L. Bharat, A. Marais, System and method for generating unique passwords, US Patent 6 (141) (1997) 760.
[14] M. Kuhn, Probability theory for pickpockets – ec-PIN guessing, Available from: <http://www.cl.cam.ac.uk/?mgk25/>, 1997.
[15] C. Herley, D. Florencio, How To login from an Internet Café without worrying about Keyloggers, in: Proceedings of Symposium on Usable Privacy and Security (SOUPS)'06.
[16] Available from: <http://www.citibank.co.jp/en/service/cap/virtualpad/>.
[17] Available from: <http://obr.typepad.com/financial_innovations/2005/11/ing_direct_adds.html/>.
[18] B. MÖLLER, Schwächen des ec-PIN-Verfahrens. Available from: <http://www.informatik.tu-darmstadt.de/TI/Mitarbeiter/moeller/>, February, 1997, Manuscript.
[19] A. Herzberg, A. Gbara, Trustbar: protecting (even naive) web users from spoofing and phishing attacks, Cryptology ePrint Archive, Report 2004/155, 2004. Available from: <http://eprint.iacr.org/2004/155/>.
[20] S. Wiedenbeck, J. Waters, L. Sobrado, J. Birget, Design and evaluation of a shoulder-surfing resistant graphical password scheme, in: Proceedings of the working conference on Advanced visual interfaces,Venezia, Italy.
[21] V. Roth, K. Richter, R. Freidinger, A PIN-entry method resilient against shoulder-surfing, in: Proceedings of the 11th ACM Conference on Computer and Communications Security, 2004, pp. 236–245.
[22] IETF, MTA Authorization Records in DNS (MARID), June, 2004. Available form: <http://www.ietf.org/html.charters/OLD/marid-charter.html/>.
[23] G. Ateniese, K. Fu, M. Green, S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage, in: Proceedings of the 12th Annual Network and Distributed System Security Symposium, 2005.
[24] G.T. Wilfong, Method and apparatus for secure PIN entry, US Patent #5,940,511, United States Patent and Trademark Office, Assignee, Lucent Technologies, Inc., Murray Hill, NJ, May, 1997.
[25] J. Mason, Filtering spam with SpamAssassin, in: Proceedings of HEANet Annual Conference, 2002.
[26] D. Stinson, Cryptography Theory and Practice, second ed., 2005.
[27] M. Sahami, S. Dumais, D. Heckerman, E.Horvitz, A Bayesian approach to filtering junk E-Mail. in: Learning for Text Categorization, The 1998 Workshop, May, 1998.
[28] T.A. Meyer, B. Whateley, SpamBayes: effective open-source, Bayesian based, email classification system, in: Proceedings of the CEAS, 2004.
[29] MAPS, RBL – Realtime Blackhole List, 1996. Available from: <http://www.mail-abuse.com/services/mds_rbl.html/>.
[30] The Spamhaus Project, The Spamhaus Block List. Available from: <http://www.spamhaus.org/sbl/>.
[31] Netcraft, Anti-Phishing Toolbar. Available from: <http://news.netcraft.com/archives/2004/12/28/netcraft_antiphishing_tool%bar_available_for_download.html/>.
[32] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, D.E. Culler, SPINS: security protocols for sensor networks, Wireless Networking 8 (5) (2002) 521–534.
[33] B. Sun, C.-C. Li, K. Wu, Y. Xiao, A lightweight secure protocol for wireless sensor networks, Computer Communications 29 (13–14) (2006) 2556–2568.
[34] B. Sun, Y. Xiao, C.-C. Li, H.-H. Chen, T.A. Yang, Security co-existence of wireless sensor networks and RFID for pervasive computing, Computer Communications, Special Issue on Secure Multi-Mode Systems and their Applications for Pervasive Computing.