2010 Second International Conference on Communication Software and Networks

Virtual password using Runge-Kutta method for internet banking

S. Mohammadi, Department of Industrial Engineering
University of KNTU
Tehran, Iran
Smohammadi40@yahoo.com

S. Z. Hosseini, Department of Industrial Engineering
University of KNTU
Tehran, Iran
Saeede626@yahoo.com

Abstract **-Today, user authentication is one of the fundamental procedures to ensure secure communication on online services. Among the authentication methods password –based is popular and widely used. So having a strong password authentication without any vulnerability is essential. In this paper, we propose a password-based remote user authentication with virtual password concept to secure users' passwords in on-line environments. We use Runge-Kutta method with linear functions as virtual function to hide user password from adversary in public channel. We analyzed the vulnerabilities and possible attacks in the proposed method and suggest proper seclusions. This paper, also considers how this method can defend against password cracker, man-in-the-middle, phishing and password file compromise attacks.**

I. Introduction

A. Related works

Remote User Authentication scheme allows the authenticated user to access the services offered by the remote system. There are many ways for remote user authentication while each one has their own advantages and disadvantages. In 1981 the first well-known hash-based password authentication introduced, but this scheme suffers from high hash computation overhead and password resetting problems [1]. Thereafter, many authentication schemes have been proposed based on hashed password [2–7] and on public key cryptography [4, 5, 8–12]. Some of these methods of authentication needs verification table. Verification table should be maintained on the remote server in order to validate the legitimacy of the registered users. But it vulnerable to some attacks such password file attack and server spoofing attack. This weakness may solve via solutions based on smart cards where a verification table is no longer required. In 2000, 2004, 2005 some methods introduced for user authentication based on smart card [13, 14, and 15]. Recently Da-Zhi Sun, Jing Xu proposed remote authentication schemes based on smart cards [16, 17] and so on.

B. Motivations

Password authentication is one of the most simple and effective approaches for authentication in a client/server environment. It has been widely deployed in banking and payment systems, computer networks.The password used in some protocols is often generated in one of the following two ways. Firstly, the password might be randomly selected from a known password set by a third party. In this case the need for users to be able to memorize the password will limit the size of the password set. As a result, the password will possess low entropy. Secondly, a user might be required to select his password from a known password set. In this case, the user is very likely to choose the password based on his personal preferences (such as name, birth date) again in order to memorize the password easily. This is preferred by user but is susceptible to many attacks and consequently, the price of to be easy to remember is that the password can be stolen by an adversary. As mentioned in [18], customer preferences were driven by their attitudes towards usability and convenience rather than their perceptions of security in authentication method. Most users prefer use of simple and desirable password (such as name, birth day) because of simplicity and rememberablity. Also most users have multiple accounts on the internet where each account needs a password. The goal of this research is that the user can use her/his simple password with high security, without any computation. In general we need the usability-security trade off, where convenience, quality, and usability are sacrificed when increasing layers of security are required. But we can show high security together with convenience and quality in authentication process. In this paper, we propose a protocol that allows a client to securely use a simple password for authentication, and also prevents phishing, man-in-the-middle, and password file and password cracker attacks. In the other words, we want to prevent users' passwords from being stolen by adversaries. Our protocol achieves client authentication without the client revealing his password to the server at any point. This method uses dynamic (one- time) password generation or the technique of one-time tickets together challenge/response technique. Most tickets in real life are one-time tickets in the sense that they can only be used once, such as movie tickets. Here user must use an external device/ token for computation of one-time password. Which this token would be able to run the function of generating one-time password. User must enter his/her simple password into device/token and use generated one-time password in authentication phase. Remote user authentication methods that work with password without smart card enforce to have password file/password verification information/Verification table in remote server for verification process. So password file exposed to some attacks such password file compromise attack. Our method can stand up to this attack. The rest of the paper is organized as follows. In Section 2, we propose the one-time ticket technique and requirements for this method. In Section 3, Runge-Kutta

978-0-7695-3961-4/10 $26.00 © 2010 IEEE
DOI 10.1109/ICCSN.2010.118

method with linear functions is explained step to step along with possible attacks. In Section 4, we describe Security analysis and comparison with newest similar virtual password. Finally, we conclude our paper and describe our future work in Section 5.

II. One- time ticket technique

A. Virtual password

In general we need a function/method to convert a simple/fixed password to dynamic/one-time password. So user can have a new one in each login. But User must calculate the password via an application and then login on server. We use virtual password concept [19] to achieve one-time password/ticket for authentication. 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) = Pd, where R is a random number provided by the server (called the random salt and prompted in the login screen by the server) and Pd 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, Pd), where ID is a user ID. On the server side, the server can also calculate Pd 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 b bijective function. If *B* is not a bijective function, it is also possible to allow the server to verify the user as follows. The server can first find the user's record from the database based on the user' ID, and compute *Pd*, 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 [19]. Our objective is to produce a function with high security to overcome the possible attacks in password–based remote user authentication. However, since "simplicity" and "security" conflict with each other, it is a challenging task to achieve both, if possible. The idea of this paper is to add some complexity, through user computations performed by computation devices, to prevent the existence attacks. We believe if function is easy to compute, user yet doesn't desire calculate any digit. It's better user has a computation devices /token to get one-time ticket free of any calculation and only remember her/his simple password. For some sensitive accounts such as on-line bank accounts and on-line credit card accounts, users are likely to accept a little additional complexities, in order to making the account more secure.

B. Virtual function with a Token-based implementation

In the proposed token-based implementation, the following entities involve:

• A user U.

• An impersonal compliant authentication token T with a small display.

• A client (i.e. browser) C that is used by U to access an SSL/TLS-based application.

• An SSL/TLS-enabled server S that hosts the application.

The token T set the virtual function and able to calculate one-time password. Similar to PKCS #11 cryptography token, this token T must capable of communicating with client browser and make a connection with client browser, when token connects to the client pc. In the other words, this token should access to the browser's SSL/TLS cache after SSL session established (which holds the server certificate for the SSL/TLS session).when an SSL session established between user and server, authentication of server is happened. If it was valid, the browser gives the server's identity to token for staying in the authentication process. Then the sever can give the random number to browser C. Afterward the user requires to put random number into token, then the token pass generated one-time password to browser with marking as "*". As a result in the case of shoulder surfing attacker cannot see one-time password. In the next section, the relation between token and client browser is shown, in detail. All the capabilities that added to token are for increasing security and stand up to man-in-the-middle, phisher and password file attacks. 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 the proposed scheme. In order to defend against existence attacks while the system is authenticating the user with password, this function should have the following properties. Having such properties enables suitable defending technology developed against the password crackers, phishing, man-in-the-middle, password file attacks.

- The function should be capable to convert n digits of simple password to m random digits; m>n. It's very difficult to password cracker including dictionary attack and brute force to break the one-time password. If it is successful, the amount of work have to do for getting one-time password exceed the time that one-time password can be correct.
- The number of variables and equations in the function must be noticed. Because with the proper number, phisher cannot find the simple password from function, even with multiple attacks.
- Existence of a random number in the function is necessary to generate dynamic password each time.
- Our purpose is to build complexity with relations and dependencies among equations in the function. Since the user doesn't desire to remember other value (constant) except simple password. Increasing the number of constant value in the function is not a good idea.

III. Runge-Kutta method with linear functions

In this section, we explained proposed virtual function step by step. In every step, vulnerabilities and strength of this virtual function are presented versus password attacks. Final version removes the possibility of single and multiple password attack.

A. Version I

The first version is the ordinary linear function with simple operators like plus. This bijective function had reverse characteristic, so verification process in server side was very easy.

$$\begin{cases} w_0 = H(pass) mod\ Z \\ f(t_i, w_i) = t_i + w_i; i = 0, \dots n \\ w_{i+1} = f(t_i, w_i)\ mod\ Z;\ Z = 10 \end{cases} \quad (1)$$

Where t_i (for i=0,…, n) is one digit of simple password and w_0 is hash of simple password. w_0, w_1,…,w_{n+1} are one-time password. Server stores hash of password for verification of password, for two reasons. The one, sever needs H (pass) either to verify the user with reverse function or with original function. The second, the server store hash of passwords in password file for security. The application that calculates function must be able to hash the simple password inputted by user.

Possible attacks:

- Crack passwords attacks are successful for this function, because n digits simple password covert n random digits. Brute force and dictionary attacks with more attempts can discover one-time password.
- Malicious server attack is the type of attacks that an attacker first set up a malicious server and allured people to register with him using a password. Assume user beguiled and revealed his one-time password, then the attacker can easily calculate fixed password.
- A shoulder surfing can easily log generated password and uses repeatedly. We should use the challenge/response technique in the next version of the function to generate dynamic password (response). So the function must have a random input provided by the server (the challenge), which then allows the users to type in different input each time they login to the system.
- This method is vulnerable to "password file compromise" attack. The attacker first steals the password file of the server, for example by breaking. Then attacker can easily detect simple password by dictionary attack (hash of all password in dictionary). Because of the goal, that is to free user of complex password, probability of success of the dictionary attacks is increased.

B. Version II

$$\begin{cases} w_0 = H(pass) mod\ Z \\ f(t_i, w_i) = t_i + w_i;\ i = 0 \dots n \\ w_{i+1} = hf(t_i, w_i)\ modZ;\ Z = 10 \end{cases} \quad (2)$$

Where t_i (for i=0,…,n) is one digit of fixed password , $w_0 \dots w_{n+1}$ are one-time password and h is random number. The server stores hash of passwords in file password for verification that vulnerable to some threats. However this function generate dynamic password in each login, but doesn't withstand toward all password.

Possible attacks:

- When phisher tricks a user to enter one-time password in the fake page, the one-time password ($w_0 \dots w_{n+1}$) will be recorded by the attacker. If attacker can get random number (with sniffing), the fixed password $t_0 \dots t_n$ disclose to adversary by the below equation: $w_{i+1} = h\ (t_i + wi\ )\ mod\ Z$
- Unfortunately, brute force attack exists yet for example n digit fixed password need 10^n permutation (one-time password), which is not difficult for password crackers.
- This version is vulnerable to sever spoofing attacks (man-in-the-middle). To launch server spoofing attack, a malicious server S first pretends to be client C and tries to login on another benign server S', who has C as a client. Responding to the login request from S, S' sends the random number h to the malicious server S. After malicious server S obtains h from sever S'. S stops communicating with S'. Later on, when C tries to login on S', S fools C and sends the stolen h, then receive one-time password from C. After that, S logs in to benign server S', again. Then S success to crack the server and log in to the server.
- This password authentication protocol is unprotected to Password file compromise attack the same as the pervious method.

C. Version III

The function (2) is extended to Euler method to enhance the complexity of method.

$$\begin{cases} w0 = H(pass)\ mod\ Z \\ f(t_i, w_i) = t_i + w_i + a\ ;\ i = 0 \dots n \\ K_1 = f(t_i, w_i) \\ w_{i+1} = (w_i + hK_1)\ modZ;\ Z = 10 \end{cases} \quad (3)$$

where H(pass) is hash of simple **password** , t_i (i=0,…,n) is one digit of simple password, $w_0 \dots w_{n+1}$ is one-time password; $w_i \in \{0 \dots z\text{-}1\}$ and a is identity of server that SSL protocol provided. The $f(t_i, w_i)$ is a bijective function if and only if gcd (h,Z)=1 [20]. Digit calculation depends on previous calculated digit, that

causes complexity of the function. Also, in this function, w_{i+1} depends on K_1 that makes Eq(2) more complex.

- When attacker tricks a user to enter one-time password in the fake page, the one-time password will be recorded by the attacker. $w_0 \ldots w_{n+1}$ are known and $t_0 \ldots t_n$, h and *a* are unknown by the phisher. But it does not help attacker. The reason is, there are n+3 variables ($t_0 \ldots t_n$, h , *a*) and n+2 equations. These equations are :
  $w_0 = H(pass) \bmod Z$
  $w_{i+1} = [w_i + h(t_i + w_i + a)] \bmod Z$ (for i=0,…,n).
- As mentioned, in the beginning of communication between user and the server, SSL protocol is run. The server identity is provided by user's browser to application that calculates the function. That causes to prevent server spoofing attack. The *a* factor stop the server spoofing attack, because the malicious server S initials a SSL session with user C, so the password calculation is based on malicious server's identity. When malicious server S establish SSL session with benign server S', this one-time password is not worked.

Possible attacks:

- If a user login at least twice in the fake page, the phisher discovers $W_0 \ldots W_{n+1}$, $W'_0 \ldots W'_{n+1}$. There are 2n+4 equations:

  $W_0 = H(pass) \bmod Z$ and $W_{i+1} = [W_i + h(t_i + W_i + a)] \bmod Z$ (for i=0,…,n),
  $W'_0 = H(pass) \bmod Z$ and $W'_{i+1} = [W'_i + h'(t_i + W_i + a)] \bmod Z$ (for i=0,…,n)
  And n+4 variables: ($t_0, \ldots, t_n$, h, h' , *a*).
  So, the phisher can detect above variables by means of those equations, even without shoulder surfing for h and h'.

- This method does not prevent Brute force attack, yet. Due to the fact that password cracker can try all combination of n (simple password length) digits.
- Remote user authentication is vulnerable to Password file compromise attack.

D. Version IV

The function (3) extended to Heun method to difficult breaking of the function.

$$\begin{cases} w0 = H(pass, h) \bmod Z \\ f(t_i, w_i) = t_i + wi + a \; ; i = 0 \ldots n \\ K_1 = hf(t_i, w_i) \\ K_2 = hf\left(t_i + \frac{2}{3}h, w_i + \frac{2}{3}K_1\right) \\ w_{i+1} = \left[w_i + \frac{1}{4}(K_1 + 3K_2)\right] mod Z; \; Z = 10 \end{cases} \quad (4)$$

Where, H (pass, h) is hash of password and random number h, that server send to user (challenge) while t_i (for i=0,…,n) is one digit of fixed password, $w_0 \ldots w_{n+1}$ is one-time password; $W_i \in \{0 \ldots z\text{-}1\}$ and *a* is identity of server.

- The calculation of the each digit (one-time password) depends on the previous digits; this causes complexity of the function. In addition to, in this version W_{i+1} depends on K_1 and K_2 linear functions. This method is improved pervious method because if the attacker can't find k_1 then can't calculate k_2. Moreover, it's hard the attacker detected the Heun method.
- If attacker can't find random number h and be able to trick the user at least one that the user login to fake page, the attacker can't solve the equation(4).
- The server store hash of simple password and random number h for every user. Here, random number h provided by the user' application in each login. To stand up password file compromise attack, user application must provide h'(new random number) and H(pass, h'). Then, together generated one-time password send to the server and server sort them for further login. As a result after each login, password verification information is updated in sever side.

Possible attacks:

- If the attacker can find random number h by means of for example sniffing. And the user is tricked to login to any phishing website at least once; it will disclose his/her one-time password. Then the phisher can compare the one-time password which user inputs ($w_0 \ldots w_{n+1}$). This is because of the information that the attacker could collect from the one lured user login. The attacker can find $t_0, \ldots, t_n$ and *a* (n+2 variables) from solving n+2 equations and h, as explained in pervious section.
- However, random number changes in each login and update password verification information in sever, but attacker to be able to detect simple password. The attacker can steal password file from the server by means of breaking. It's probable that attacker can detect simple password via dictionary attack.

E. Version V

The function (4) improved to Modified Euler method with increased dependency within equation.

$$\begin{cases} w_0 = H^2\,(pass, h) mod Z \\ f(t_i, w_i) = t_i + w_i + a; i = 0 \dots n \\ K_1 = hf(t_i, w_i) \\ K_2 = hf(t_{i+1}, w_i + K_1) \\ w_{i+1} = \left[w_i + \frac{1}{2}(K_1 + K_2) \right] mod Z;\ Z = 10 \end{cases} \quad (5)$$

Where, H^2(pass, h) is double hash password and random number , that server send to user(challenge).while t_i (for i=0,…,n) is one digit of fixed password, $w_0 \dots w_{n+1}$ is one-time password; $W_i \in \{0 \dots z-1\}$ and a is identity of server that user browser provided by established SSL session. The application that calculates Eq.(5) must be able to hash (double) the simple password inputted by user and random number received from server.

- Because of the double hash of simple password and random number store in the server, password file compromise attack can't success. It's very hard that attacker can detect simple password via dictionary attack.

F. Version VI

The function (5) extended to four steps Runge-Kutta method with increased the levels of the equation.

$$\begin{cases} w_0 = H\text{^}2(pass, h)\ mod Z \\ f(t_i, w_i) = t_i + w_i + a\ ; i = 0 \dots n \\ K_1 = hf(t_i, w_i) \\ K_2 = hf\left(t_i + \frac{h}{2}, w_i + \frac{1}{2}K_1\right) \\ K_3 = hf\left(t_i + \frac{h}{2}, w_i + \frac{1}{2}K_2\right) \\ K_4 = hf(t_i + 1, w_i + K_3) \\ w_{i+1} = \left[w_i + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \right] mod Z \end{cases} \quad (6)$$

Z is modified to a long prime number. The reason ıs the length of one-time password. Before versions suffer from brute force attack because n digits simple password make change to random n+1 digit. But this method makes change to m digits random password, as m>>n and changeable, so Brute force can't be able to crack dynamic password. In result he/she can't solve the equations to crack the simple password (t_i; i=0..n). The final version can defend against server spoofing attack, phishing, password file compromise, brute force, dictionary attacks.

IV. Security analysis

A. 4.1 suppositions

We have some reasonable suppositions, for the proposed virtual function to be secure. First, this remote user authentication is used secure socket layer (SSL). The current industry standard for securing communication over the Internet is SSL, which was developed by Netscape in 1994 [19]. Part of the protocol is a handshake protocol that is responsible for (mutual) authentication and key establishment. SSL has been built into all major web browsers, web servers, email clients, email servers, etc. The SSL protocol runs mainly in three steps:

A. Server authentication: The server sends its certificate to the client. The client authenticates the server using the certificate. Note that the certificate contains the public key of the server.

B. Key establishment: The client generates a session key, encrypts it with the public key of the server, and sends the result to the server. The server then decrypts the message using its private key and obtains the session key. Henceforth the session key between the client and the server is established.

C. Secure communication: All the subsequent communication is encrypted with the session key.

A higher level protocol can easily layer on top of the SSL protocol transparently because SSL is application protocol independent. So we can assume the proposed method is set on top of SSL protocol. In other words, our protocol runs in the third step (secure communication) of SSL. So all message in our protocol can encrypted with session key and increasing security. Notice the client authenticates the server using the first step of SSL protocol and server authenticates the client using our remote authentication after the SSL session established. In general web browser checks the server's certificate. A web browser, such as Internet Explorer and NetScape, is usually preloaded with a list of trusted certificate authorities. When a client tries to access a web site whose certificate has expired or is not signed by a trusted certificate authority, the web browser usually prompts a warning message to the client saying that the certificate should not be trusted.

B. 4.2 Comparison

In this section, we show that our new method has advantages over the M. Lei and Y. Xiao method [19] that is newest method of remote user authentication with virtual password recently.

- Lei's method go n digits fixed password to m digits which n=m, because Z= {1..9}. This causes brute force with high amount works can detect dynamic password.
- In the Lei's method neither consider password file in remote server, nor have any mechanism for protection password file. So that is vulnerable to password file compromise attack. An attacker can break the server and access the fixed password.
- Lei used two values (constant and variable) more than the proposed function. This is hard for user to remember constant or secret value more than fixed

password. But we use only fixed password (constant) in our function and build complexity with dependencies among equations without more variable. Our goal is to free user from any calculation and remembering value. User only type her/his password into token, afterward the token is responsible for the rest of works.

V. Conclusion

However there are numerous methods for authentication in electronic world such smartcard, biometric and etc, but password is the common way that widely used because they are easily implemented. In this paper, we have proposed a new remote user authentication via virtual password with a new virtual function. This function uses four steps Runge-Kutta method with linear function that generate one-time password securely. With this scheme user have any concern about using simple password. The equations of the function is enough correlated without too variables and constants and has low computation time. It is very difficult that the attacker can discover the function. This method is fit for online environment such e-banking and ATM and etc. we analyzed how the proposed method can secure against password cracker, phishing, man-in-the-middle, password file attacks. Our future work is implementation this method in internet banking.

References

[1] L. Lamport, Password authentication with insecure communication, Communications of the ACM 24 (11) (1981) 770–772.

[2] A. Shimizu, T. Horioka, H. Inagaki, A password authentication method for contents communications on the Internet, IEICE Transactions on Communications E81-B (8) (1998) 1666–1673.

[3] T.C. Yeh, H.Y. Shen, J.J. Hwang, A secure one-time password authentication scheme using smart cards, IEICE Transactions on Communications
E85-B (11) (2002) 2515–2518.

[4] C.C. Lee, M.S. Hwang, W.P. Yang, A flexible remote user authentication scheme using smart cards, ACM Operating Systems Review 36 (3) (2002) 46–52.

[5] C.C. Lee, L.H. Li, M.S. Hwang,Aremote user authentication scheme using hash functions, ACM Operating Systems Review 36 (4) (2002) 23–29.

[6] M.L. Das, A. Saxena, V.P. Gulati, A dynamic ID-based remote user authentication scheme, IEEE Transactions on Consumer Electronics 50 (2) (2004) 629–631.

[7] W.C. Ku, A hash-based strong-password authentication scheme without using smart cards, ACM Operating Systems Review 38 (1) (2004) 29–34.

[8] C.C. Chang, W.Y. Liao, A remote password authentication scheme based upon ElGamal's signature scheme, Computers & Security 13 (2) (1994) 137–144.

[9]D. PJablon, Strong password-only authenticated key exchange, ACM Computer Communications Review 26 (5) (1996) 5–20.

[10] M.S. Hwang, L.H. Li, A new remote user authentication scheme using smart cards, IEEE Transactions on Consumer Electronics 46 (1) (2000)
28–30.

[11] J.J. Shen, C.W. Lin, M.S. Hwang, A Modified remote user authentication scheme using smart cards, IEEE Transactions on Consumer Electronics 49 (2) (2003) 414–416.

[12] A.K. Awasthi, S. Lal, A remote user authentication scheme using smart cards with forward secrecy, IEEE Transactions on Consumer Electronics 49 (4) (2003) 1246–1248.

[13] H.M. Sun, An efficient remote user authentication scheme using smart cards, IEEE Trans. Consum. Electron. 46 (4) (2000) 958–961.

[14] Amit K. Awashti, Sunder Lal, An enhanced remote user authentication scheme using smart cards, IEEE Trans. Consum. Electron. 50 (2) (2004) 583–586.

[15] W.C. Ku, S.T. Chang, Impersonation attack on a dynamic ID-based remote user authentication scheme using smart cards, IEICE Trans. Commun. (5) (2005) 2165 –2167.

[16] Jing Xu , Wen-Tao Zhu , Deng-Guo Feng , An improved smart card based password authentication scheme with provable security, Computer Standards & Interfaces xxx (2008) xxx–xxx

[17] Da-Zhi Sun , Jin-Peng Huai , Ji-Zhou Sun , Jian-Xin Li , Cryptanalysis of a mutual authentication scheme based on nonce and smart cards, Computer Communications xxx (2009) xxx–xxx

[18] A.O. Freier, P. Karlton, P.C. Kocher, The ssl protocol version 3.0 internet draft, March 1996. http://wp.netscape.
com/eng/ssl3/draft302.txt.

[19] M. Lei , Y. Xiao , S. V. Vrbsky, C.-Chih Li ,Virtual password using random linear functions for on-line services, ATM machines, and pervasive computing, Computer Communications xxx (2008) xxx–xxx

[20] D. Stinson, Cryptography Theory and Practice, second ed., 2005.