Secure Password Pocket for Distributed Web Services*

Jae Hyung Koo,[1] and Dong Hoon Lee[2]

[1] Center for Information Security Technologies (CIST),
Korea University, Seoul, Korea
ideao@cist.korea.ac.kr[1]
[2] Graduate School for Information Security (GSIS),
Korea University, Seoul, Korea
donghlee@korea.ac.kr[2]

Abstract. Password authentication (PA) is a general and well-known technique to authenticate a user who is trying to establish a connection in distributed web services. The main idea of PA is to remove complex information from users so that they can log on servers only with a human-memorable password at anywhere. So far, many papers have been proposed to set up security requirements and improve the efficiency of PA. Most papers consider practical attacks such as password guessing, impersonation and server compromise which occur frequently in the real world. However, they missed an important and critical risk. A revealed password of a user from a server may affect other servers because most people tend to use a same password on different servers. This enables anyone who obtains a password to easily log onto other servers. In this paper, we first introduce a new notion, called "password pocket" which randomizes user's password even if he/she types a same password on different servers. When our password pocket is used, an exposed password does not affect other servers any more. The cost of a password pocket is extremely low since it needs to store only one random number securely.

1 Introduction

In a client-server environment, there exist several mechanisms to authenticate a client trying to log on a server such as cryptographic secure module, biometric data, information which only legitimate user knows and so on. Cryptographic module adopts mathematically secure algorithms such as message authentication code [7, 15] and digital signature in public key infrastructure (PKI) [10, 11]. In spite of providing very strong user authentication, cryptographic module is not widely installed in many servers because of its high cost and difficulty in key management. Biometric authentication [6, 12] is also hard to take in since installation cost is too high. Furthermore, users generally do not like to give their biometric information because of privacy. Password based authentication is a

* This work was supported (in part) by the Ministry of Information&Communications, Korea, under the Information Technology Research Center (ITRC) Support Program

typical technique using information which a person knows and does not require high cost for additional devices such as biometric information scanner. The only requirement for users is to memorize passwords which they registered in server. Thus, password authentication is the most attractive one.

Although password based authentication (shortly PA in this paper) is a nice solution, PA still has a drawback. If a user wants to log on multiple servers, he/she should memorize a number of passwords. Obviously, it is hard for user to memorize several passwords. One approach to remedy the drawback is single-sign-on (SSO in short) [14]. As the name denotes, a user can get services from different servers without logging on separately. When a user logs on SSO gateway, he/she automatically logs on other servers which he/she has permission. SSO is a very convenient tool from the user's point of view but it has important problem. An exposed password may harm entire SSO system since anyone with the password can use all applications from servers in which the password is registered. We note that even if a user does not use SSO, the problem happens very often. In other word, many users set a same password on different servers. A simple but bad solution is to let users memorize multiple passwords. As we mentioned, memorizing several passwords is not so easy for user. So, it is desirable to design a method to achieve two goals simultaneously: i)*set user free from memorizing multiple passwords* and ii) *guarding user's passwords in other servers from a revealed password.*

Smartcard [5] is one of hardware solutions to protect secret information from out side attack. Also, it provides portability so that the owner can use the secret information for authentication at anywhere. It is possible for a user to store a number of passwords in a smartcard. However, to use a smartcard, every computer has to be equipped with a smartcard reader. Another adoptable device may be a universal serial bus (USB) [16]. Recently, most computers have USB and lots of users use the device. Therefore, saving passwords into a USB seems to be the best way for password based authentication for distributed services without additional equipments. But, the device should guarantee the confidentiality of the passwords and linkability to match each password with its corresponding server. Confidentiality of passwords is necessary because USB can be stolen. A very simple method to achieve confidentiality is to encrypt all passwords with a symmetric key which is securely managed in USB. The key also could be encrypted with a password [9] for user authentication. Obviously, it requires memory spaces. Furthermore, although a user does not need to memorize multiple passwords, he/she has to generate different passwords. Making several passwords is not easy for human-being.

Our goal is to build a mechanism to randomize a user's password even though he/she enters a same password on different web sites. We adopt an *one-to-one one-way random mapping function* $h(\cdot)$ in which same input results in same output but it is hard to guess pre-image from function value, e.g. m from $h(m)$. The word 'random mapping' means, even only one bit changes, it is computationally infeasible to guess the function value. We use a USB device but it just stores

a random number in a secure area. So our idea is also memory efficient. More detailed description is shown in Sect. 3.

1.1 Related Works

A password is widely used in a client-server setting for user authentication. Basic idea is to allow permission to a person who exactly knows password registered in a server by comparing it with the entered password. To enhance security, i.e. to protect communication between a client and a server, a secure channel is usually constructed through an encryption scheme. The key used in the encryption scheme is derived from a password. By checking the validity of the encryption and additional message, the server decides to open connection. The key is continuously used after authentication until current connection is closed. This mechanism is very popular especially in internet banking to prevent user's information from being disclosed. There have been lots of papers to build password based authentication with secure channel. Most of the papers focus on finding security breaches in password based schemes such as guessing password and remedying them [1–4]. Unfortunately, all of them do not consider the situation of password exposure. Although their schemes are secure, whenever a user stores a same password on different servers and the password is revealed, anyone with the password can easily log onto the servers as though he/she is the owner of the password.

Single sign on (SSO) [14] is a cost-effective password management which concentrates on enhancing user's convenience. It is very attractive in enterprise environment with various servers or distributed web services. In a distributed web services without SSO, a user has to log on as many servers as the number of services he/she wants to get. For example, if a user tries to transfer money to an account registered in an auction server, buy a goods from a merchant and receive a receipt through email, then he/she must log on three servers, on-line banking server, auction server and email server. SSO removes the tiresome login phases based on the assumption that all of the servers are registered in the SSO gateway. I.e. whenever a user logs on the gateway, he/she can automatically logs on other servers registered in the gateway. However, as we mentioned, instead of setting a user free from memorizing multiple passwords it has critical security breach when a password is stolen. In our scheme, we adopt the basic idea of SSO but use a portable USB in place of SSO gateway.

Our contributions are two folds.

- We pointed out potential risk of a disclosed password in distributed web services which is more critical in SSO. As we noted, even though a password based authentication scheme is secure, we can easily logs on servers with a obtained password if the owner set a same password on different servers
- We propose a new and practical concept, called 'password pocket' in which even a user types a same password, the actual passwords used to authenticate user to servers are different. The actual password depends on the server's

internet address (i.e. URL : uniform resource locator). The password pocket does not store all passwords. It keeps only a random number in its secure area. Hence, the required size of memory is very small.

2 Preliminaries

We briefly introduce several building blocks which are adopted in our scheme.

One-to-One One-Way Random Mapping Function. The notion of *one-wayness* is that if we know input value it is very easy to compute the result but it is infeasible to compute the value of an inverse function. *One-to-one random mapping* guarantees two properties: i) same input always derives same output, ii) even if only one bit of input changes the output is not predictable. There exist several practical functions with the above properties such as SHA-1 [13].

Password Authenticated Key Agreement. Password authenticated key agreement is widely adopted to construct a secure channel only with a password. Basically, the secure channel consists of encryption algorithms and the key used in the algorithms is derived from the password and some related information. Following is a simple method to build a secure channel between a user U and a server S. pw denotes user's password and g^a, g^b are the result of cryptographic operation with random inputs a, b.

- $\mathbf{U} \rightarrow \mathbf{S} : E_{h(pw)}(g^a)$ where $E(\cdot)$ is a symmetric algorithm and $h(\cdot)$ is a hash function.
- $\mathbf{U} \leftarrow \mathbf{S} : E_{h(pw)}(g^b)$, $MAC_K(g^a)$ where $K = h(g^{ab})$.
- $\mathbf{U} \rightarrow \mathbf{S} : MAC_K(g^b)$.

MAC is a cryptographically secure message authentication code [7, 15] in which only a person knows actual key (K) and input message can generate a correct value.

3 Password Pocket for Distributed Servers

In this section, we introduce a password pocket. As we mentioned in Sect. 1, we use a USB as a portable device. The structure of a password pocket is simple. There are one small secure memory (128 bits is sufficient) for a random number and relatively large general memory for one-to-one one-way random mapping function and temporal values. General memory does not need to be secure but it should guarantee read-only property for the area in which the function is stored. In fact, we only consider randomizing passwords. So we do not deal with the way to send a password to a server securely. There are many schemes providing methods to transmit data to a server in a secure manner and we can adopt one of them to enhance the security.

In the initial phase, a user $\mathcal{U}$ registers a randomized password on a server $\mathcal{S}$ as followings:

1. $\mathcal{U}$ chooses a password pw from password dictionary $\mathcal{D}$. In the real world, pw could be a meaningless word generated by $\mathcal{U}$.
2. $\mathcal{U}$ enters pw into a password pocket with his/her ID and $\mathcal{S}$'s address (it can be URL or IP address) Svr_Addr.
3. The password pocket fetches a random number r stored in the secure area and computes $h(ID||pw||r||Svr\_Addr)$.
4. $\mathcal{U}$ registers $h(ID||pw||r||Svr\_Addr)$ on $\mathcal{S}$.

We assume that there is no attack while r is used to generate a randomized password. $h(\cdot)$ is the function we explained in Sect. 2. We note that even if same pw is used for multiple servers, each server gets a different password because of the different addresses (Svr_Addr).

Whenever $\mathcal{U}$ wants to log onto a server, he/she just needs to type ID and pw into the password pocket. Then, the password pocket sets ID, pw, r, Svr_Addr as input values and returns $h(ID||pw||r||Svr\_Addr)$. An essential assumption is that a secure module guards r and pw from attacks such as memory dump and data capture.

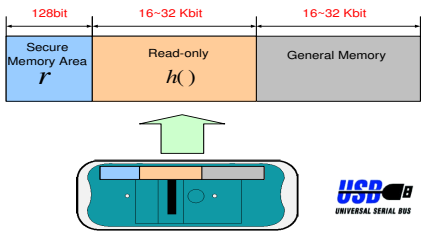

Fig. 1. Structure of Password Pocket

Remark 1. [*Random number escrow and update*]: Even though the probability is low, a password pocket could be broken or stolen. We provide two mechanisms: i) random number escrow for 'broken' and ii) random number update for 'stolen'. For the former one, we can use a key escrow technique [8] in which a user escrows his/her secret key to prepare the case of losing the key. So, after reconstructing the random number, the user installs it into a new device and uses a password pocket as before. For the latter one, a user may want to change the random number. In fact, updating the random number is very complex because all passwords registered in servers should be also updated. We note that the user may not need to update the random number since the probability of guessing correct password is very low. More detailed explanation for the security against the stolen password pocket is showed in Sect. 4.

4 Security Analysis of Password Pocket

Basically, we assume that there exists a cheap USB device with secure area and insecure area. The secure area may be very small but it is tamper-proof. A random number r is stored in the secure area and it is infeasible to find or get r. For the security of password pocket, we consider *password exposure from a server* and *password guessing with a stolen password pocket*. Because of the lack of pages, we omit the security proof but it will be shown in final paper.

5 Efficiency of Password Pocket

Password pocket is cost and memory efficient idea. It requires only a quite small secure memory for guarding a random number r and a little larger general memory for storing $h(\cdot)$ and computing values. So it can be installed in a small size USB device and a user can carry the device by inserting it into a key-holder. In this paper, we set the length of r as 128 bit and it is sufficient because the probability of guessing correct r is $\frac{1}{2^{128}}$. Only tens of kilo-bits of general memory is required to manage and execute a function $h(\cdot)$. Usually, a mapping does not require much time ($1.25 \times \frac{1}{10^6}$ second to execute a hash function SHA-1[13] on 3.21GHz Pentium 4 processor with 1 Gbyte RAM). In fact, we also consider a method to adopt our password pocket, called *virtual password pocket* (VPP). We will provide the description of VPP in the final paper. Table 3 shows the property comparison among the mechanisms. PP denotes a password pocket.

Table 1. Property comparisons

	Password-only	*Smartcard*	*PP*
secure for stolen password	X	△	O
secure memory	X	O (encryption key)	O(random number)
additional memory	X	O (encrypted passwords)	O ($h(\cdot)$)
additional device	X	O(reader)	O(USB port)
password guessing attack	feasible	infeasible	infeasible
memorable	easy	irrelevant	easy
cost	very low	high	low

In general, a smartcard authenticates a user with the owner's password pre-set in it. Therefore, if the stolen password from a server is same as or similar to the owner's password, all passwords could be revealed. As we mentioned, secure memory is tamper-proof and an encryption key (smartcard) or a random number (PP) should be managed in this area. In a smartcard, because all encrypted passwords should be stored, a sufficient size of additional memory is required. Our password pocket needs a USB as a portable device but almost all of the

computers adopt USB port. Hence, using USB does not require additional cost. All mechanisms in Table 1. excluding 'password-only' guarantee security against password guessing attack.

We simulated our password pocket with a hash function, SHA-1 [13]. Table 1. shows the results.

Table 2. Simulation results. The values of random numbers and P_Passwords are hexadecimal.

ID	*Password*	*Random number (128 bit)*	*Server's Address*	*P_Password*
Robert	Password	00112233445566778899aabbccddeeff	*www.security.com*	f001e795ac95f23a
Robert	Password	00112233445566778899aabbccddeeff	*www.npc*05*.org*	073e160a41738cd2
bobert	Password	00112233445566778899aabbccddeeff	www.security.com	9ba86eb376b36756
Robert	*Passpord*	00112233445566778899aabbccddeeff	www.security.com	128d6116fb2faa8c
Robert	Password	ff112233445566778899$aabbccddeeff$	www.security.com	1060904b14a42155

5.1 Other Application of Password Pocket

Password pocket is applicable even in home network environment. If a password pocket is embedded in a mobile equipment such as a cellular phone, a user can control the devices at home from remote outside with randomized passwords. In this case, two of the input values should be changed. $Time_{Current}$ denoting current time and Cmd representing user's command are inserted instead of ID and Svr_Addr. And we set a control server at home which has the same random number and the password pocket which the user takes. Simply Cmd consists of device name (DN) and type of command (TC). The role of the server is to interpret user's command, activate a device connected to home network and report the result to the user. The messages from a user to the server are $h(Time_{Current}||pw||r||Cmd)$, $Time_{Current}, Cmd$. If the value of $h(\cdot)$ is correct, then the server extracts DN and TC to operate command. Clearly, since only who knows the password pw and the random number r can compute the function value, no one without permission can access to the devices at user's home.

6 Conclusion

We pointed out a potential but critical security breach in password based authentication which frequently occurs in distributed web services where a user only uses a same password on different servers. To remedy the security breach, we proposed a practical method, called 'password pocket' which randomizes a password so that a user can register different passwords on multiple servers only with a memorable password. Furthermore, the only requirement for the password pocket is very small secure memory to keep a random number in a secure manner. Hence, password pocket is a practical idea for distributed web services.

7 Acknowledgements

We really thank the reviewers for their helpful advices. Our scheme is general and can be used in all password based key agreement schemes without modification. So it is hard to compare our scheme with password based key agreement schemes. The entire security of our scheme is similar to that of smartcard. However, usually the computation power and storage ability of USB are much higher than those of smartcard.

We also want to give thanks to Bum Han Kim and Sang Pil Yun for their advice and assistance for the system-design.

References

1. M. Bellare, D. Pointcheval and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. Eurocrypt '2000, 2000.
2. V. Bokyo, P. Mackenzie, and S. Patel. Provably Secure Password-Authenticated Key Exchange using Diffie-Hellman. Eurocrypt 2000
3. E. Bresson, O. Chevassut and D. Pointcheval. Security Proofs for an Efficient Password-based Key Exchange. ACM CCS 03, 2003.
4. E. Bresson, O. Chevassu and D. Pointcheval. New Security Results on Encrypted Key Exchange. PKC 2004, vol. 2947 of LNCS, 2004.
5. The ISO 7816 Smart Card Standard. Available at "http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx".
6. A. Jain and S. Prabhakar. Biometrics Authentication. INSIGHT: A Publication of the Institute for the Advancement of Emerging Technologies in Education, Vol. 2, pp. Vision 29-52, EdPress, Jan 2003.
7. B. Kaliski Jr. and M. Robshaw. Message Authentication with MD5, CryptoBytes (1) 1, Spring 1995.
8. Key Escrow. Available at "http://www.epic.org/crypto/key_escrow/".
9. RFC 2898PKCS #5: Password-Based Cryptography. Available at "http://ww-w.f-aqs.org/ rfcs/rfc2898.html".
10. P1363 Standard Specifications for Public-Key Cryptography. Available at "http://grouper.ieee.org/groups/1363/".
11. Public-Key Infrastructure (X.509) (pkix). Available at "http://www.ietf.org/html .charters/pkix-charter.html".
12. S. Qidwai, K. Venkataramani and B. Kumar. Face Authentication from Cell Phone Camera Images with Illumination and Temporal Variations. Proc. First International Conference Biometric Authentication (ICBA), Springer Verlag, LNCS 3072, 2004.
13. Secure Hash Standard (SHA1). Available at "http://www.itl.nist.gov/fipspubs/fip180-1.htm".
14. Single Sign on. Available at "http://www.opengroup.org/security/sso/".
15. D. Stinson. Cryptography - Theory and Practice, CRC Press, Boca Raton, 1995.
16. Universal Serial Bus. Available at "http://www.usb.org/".