Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Multiple hashes of single key with passcode for multiple accounts

LEE Kok-wah[‡1], EWE Hong-tat[2]

([1]*Faculty of Engineering & Technology, Multimedia University, Jalan Ayer Keroh Lama, 75450 Bukit Beruang, Melaka, Malaysia*)

([2]*Faculty of Information Technology, Multimedia University, Jalan Multimedia, 63100 Cyberjaya, Selangor, Malaysia*)

E-mail: kwlee@mmu.edu.my; htewe@mmu.edu.my

Received Mar. 11, 2007; revision accepted May 9, 2007

Abstract: A human's e-life needs multiple offline and online accounts. It is a balance between usability and security to set keys or passwords for these multiple accounts. Password reuse has to be avoided due to the domino effect of malicious administrators and crackers. However, human memorability constrains the number of keys. Single sign-on server, key hashing, key strengthening and petname system are used in the prior arts to use only one key for multiple online accounts. The unique site keys are derived from the common master secret and specific domain name. These methods cannot be applied to offline accounts such as file encryption. We invent a new method and system applicable to offline and online accounts. It does not depend on HTTP server and domain name, but numeric 4-digit passcode, key hashing, key strengthening and hash truncation. Domain name is only needed to resist spoofing and phishing attacks of online accounts.

Key words: Key management, Memorability, Key hashing, Key strengthening, Multiple accounts, Information security
doi:10.1631/jzus.2007.A1183 **Document code:** A **CLC number:** TN918; TP309

INTRODUCTION

For friendly environment, cost effectiveness and efficiency, human civilizations are heading towards a paperless and electronic society. Every human is getting numerous offline and online accounts. These accounts require authentication to gain system access. There are three types of authentication approaches: secret, token and biometrics.

Secret is about something you know like password or key. Token is about something you have like smart card. Biometrics is about something you are like fingerprint. For the sake of cost and compatibility, secret in the form of key is the most popular authentication approach.

According to Forrester Research (Kanaley, 2001), an active Internet user manages an average of 15 keys on a daily basis. Most people, who are majority-wise, not using the password management tools, either maintain the same key for all the accounts, write down different keys for different accounts, or keep closely related keys for various accounts. These are all poor password management practices.

The HTTP basic authentication protocol (even over SSL) (Franks *et al.*, 1999) allows a server to know the key of each account. This causes possible malicious server attacks from the administrators and crackers. The server may be untrustworthy or compromised.

For another HTTP specification, i.e. HTTP digest authentication protocol, challenge-response protocol is used (Franks *et al*., 1999). The server can still see the clients' keys. Since the response from a client to a server is not specific to the server, HTTP digest authentication protocol is vulnerable not only to malicious server attacks, but to password file compromise attacks, spoofing attacks and phishing attacks.

If a key is reused, the success of an attack on an account in a weak system may cause a strong system to be compromised. This password reuse can trigger a domino effect from the weakest system to the strongest system (Ives *et al*., 2004).

‡ Corresponding author

Therefore, every key has to be uniquely set for each account, regardless of weak or strong system, to get rid of the risk when one system is compromised. However, according to (Adams and Sasse, 1999), users can only be expected to cope with a maximum of four or five keys that are unrelated and regularly used. This reflects the need to balance the usability and security.

To address this problem, some key management tools are invented. These tools allow users to remember only one master secret as master key and assign unique keys to multiple accounts. They allow users either to choose their own master key and then store the site keys somewhere safe, or to assign fixed keys to each website that can be computed whenever they are needed.

The examples of the first approach are Password Safe and Windows Live ID. The examples of the second approach are LPWA (Lucent Personal Web Assistant), HP Site Password, Password Multiplier, SPP (Single Password Protocol), PwdHash and Passpet. A special example using the hybrid approach is CPG (Compass Password Generator).

Password Safe is a password vault that can be used for offline and online accounts. However, its mobility is low due to the requirement to have a safe storage for multiple keys encrypted by a common master key. Windows Live ID is mobile and requires a single sign-on server for online accounts. It has the risk of single point of failure.

Other key management tools require a domain name to create site keys for multiple online accounts. These tools cannot be used for offline account without a domain name. These prior arts use single sign-on server, key hashing, key strengthening and petname system techniques to allow users to use a single key for accessing multiple online accounts.

The present invention can be applied to offline and online accounts with good mobility. Domain name is not necessary but optionally needed to resist phishing attacks and spoofing attacks. A single sign-on server is also not needed. The required components are numeric 4-digit passcode, key hashing, key strengthening and hash truncation.

To allow diversity of site keys from a single master key, there are two optional entries: username ID and domain name (or website) URL. Domain name that is also used to resist phishing attacks can be replaced by adopting an anti-phishing tool. In other words, the proposed new method and system can be used together with an anti-phishing tool.

These anti-phishing tools are SpoofStick, Netcraft Toolbar, Earthlink Toolbar, SiteKey, DSS with SRP (Dynamic Security Skins with Secure Remote Password Protocol), Petname Tool, TrustBar and Passpet (Yee and Sitaker, 2006).

RELATED WORKS

Here, we discuss the prior arts of key management tools, where a single key can be used for multiple accounts, in a deeper context. Anti-phishing tools will not be discussed. Accounts are divided into two types: offline and online. Offline accounts have no domain name while online accounts have domain name. Example of offline accounts is file encryption; whereas example of online accounts is email.

Password Safe is an application software originally developed by Bruce Schneier (http://www.schneier.com/passsafe.html). It uses the Twofish encryption algorithm to protect the stored passwords by a master password. Users need only to remember one master password to access multiple passwords. Its mobility depends on the available password database. It can be used for both offline and online accounts, but cannot resist spoofing attacks and phishing attacks.

Windows Live ID is also known as "Microsoft Passport Network" (http://www.passport.net). Users need a master password to sign on a central server. This central server will authenticate users for multiple servers which have joint network. Besides single point of failure, it has high cost of integration. Some security loopholes are reported (Kormann and Rubin, 2000). It can be used for online accounts only, but can resist phishing and spoofing attacks.

LPWA (Gabber *et al*., 1997; Matias *et al*., 1997) uses key hashing of master password and domain name to generate a specific site password via a server. It has single point of failure but not the high cost of integration. However, the malfunction of central authority will mean the breakdown of all services. It can be used for online accounts only and can resist phishing and spoofing attacks. As info, it has stopped providing the services.

HP Site Password (Karp, 2003; Karp and Poe,

2004) is also called "System-Specific Passwords" or "Site-Specific Passwords". A master password and a system name are concatenated, hashed using MD5 (Rivest, 1992) and converted into Base64 encoding (Borenstein and Freed, 1992) to get a site password. It is not centralized using a server but operates as stand-alone application in the terminal computers. It can be used for online accounts only and cannot resist phishing and spoofing attacks.

It is important to note here there were few successful collision attacks over the MD5 in the years 2004~2006 (Wikipedia, 2007a). The successor of MD5, which is SHA-1, is also discovered to be subject to collision attacks on its reduced version in the years 2004~2006 (Wikipedia, 2007b). Consequently, NIST announced that SHA-1 would be phased out by the year 2010 in favour of SHA-2 variants: SHA-224, SHA-256, SHA-384 and SHA-512 (NIST, 2002; Lilly, 2004).

CPG (Luo and Henry, 2003) is also known as "Common Password Method". It assigns unique random numbers to different website accounts. The random number is hashed using MD5 and converted using a binary-to-text transform to generate a specific password for multiple accounts. The random number is encrypted and stored in an account server or proxy server. When a user needs to access a specific account, the encrypted random number is retrieved from the server, decrypted, hashed and converted into a specific password to authenticate the access. Therefore, it has the weakness of single point of failure, but does not involve the high cost of integration as in LWPA. It can be used for online accounts only and can resist phishing and spoofing attacks.

Password Multiplier (Halderman *et al*., 2005) uses key hashing and key strengthening. There are two levels of hash iterations using the inputs of username, master password and site name. Both the numbers of hash iterations are fixed for 100 s and 1/10 s respectively. It is a stand-alone application without using a server and implemented using browser extension to Mozilla Firefox. It can be used for online accounts only and can resist phishing and spoofing attacks.

SPP (Gouda *et al*., 2005) is also a stand-alone application. It applies the techniques of challenge-response protocol, one-time server-specific ticket and key hashing using MD5 or SHA-1. The site password is hashed from the one-time ticket, server name and master password. The one-time ticket and site password will be updated after every login access. It can be used for online accounts only and can resist phishing and spoofing attacks.

PwdHash (Ross *et al*., 2005) is implemented using browser extensions to Mozilla Firefox, Internet Explorer and Opera. Its key hashing inputs the domain name of remote site into a pseudo-random function controlled by user's master password. The domain name acts as a hash salt. It can be used for online accounts only and resist phishing and spoofing attacks.

Passpet (Yee and Sitaker, 2006) is also implemented using browser extension to Mozilla Firefox. It applies the techniques of petname system, key hashing, key strengthening and UI customization. Petname system is a naming system possessing the properties of globality, security and memorability (Wikipedia, 2007c). It is used for anti-phishing attacks. Key hashing and key strengthening in Passpet are alike the Password Multiplier using the SHA-256, except that its first level of hash iterations is flexible in amount allowing updates according to the computer technology advancement without changes of software. It uses local storage for login access via a fixed machine, and remote storage in a server for login access with mobility feature. The remote server stores the first level of hash iterations and site label file that is encrypted from the site label list. Due to the dependency of server for newly used machines, Passpet has some risks of single point of failure. However, there is no high cost of integration. It can be used for online accounts only and can resist phishing and spoofing attacks.

METHODS

Our proposal here requires users to remember an at least 128-bit master key and a numeric 4-digit passcode. We name this method and system "multi-hash key".

The passcode is used together with key hashing, key strengthening (Manber, 1996; Abadi *et al*., 1997; Kelsey *et al*., 1997) and hash truncation to generate 20 unique hashes at 20 security levels for 20 accounts. Each security level has one account. These hashes are

site keys. All the security levels are ranked from the highest security (#1) to the lowest security (#20). This is because knowing the multihash key at the higher level can reveal the multihash key at the lower level, but not the reverse.

Twenty accounts are set according to the survey that an active Internet user manages an average of 15 keys on a daily basis (Kanaley, 2001). Five accounts are added by assuming that there are five offline accounts. The number of accounts can be increased by changing the settings or remembering another pair of (master key, passcode).

There are three pseudo-codes for multihash key to show how the method and system are working. These are determination of hash iterations of multiple security levels, generation of multihash keys as site keys, and changes of key pair (master key, passcode).

Fig.1 shows the determination of 20 security levels via the experiments to locate the lower bound b_L and upper bound b_H for 1-second hash iterations for an old computer that is slow but still popular. Each security level is partitioned by 2^8.

1 Settings:
  b_L=lower bound for 1-second hash iterations;
  b_H=upper bound for 1-second hash iterations;
  s_i=security level (i=1, 2, 3, …, 20);
  s_1=the highest security level;
  i=20.
2 For s_i, the bound b_i is
  $b_i \leftarrow 0.2b_L+2^8\times(i-1)$, $b_i \le 2.0b_H$;
  $i \leftarrow i-1$.
3 If i=0, exit; else go to Step 2.

Fig.1 Pseudo-code of determination of hash iterations of multiple security levels

Fig.2 presents the generation of multihash keys as the site keys. A user needs to remember the selected security level for a specific account. In case of forgetfulness, all the 20 security levels have to be tried one by one.

Necessary entries are master key d and numeric 4-digit passcode d_n. Optional entries are username ID and website (or domain name) URL. The username and website are used to create diversity of multihash key from a key pair (master key, passcode). Domain name can also help to resist phishing and spoofing attacks.

1 User selects security level s_i;
  Twenty bounds of hash iterations: b_1, b_2, b_3, …, b_{20};
  Optional entries: username ID, website URL;
    or else NULL;
  Necessary entries: numeric 4-digit passcode d_n, key d.
2 $H_b \leftarrow$SHA-512 ($d \| d_n$, 1) for one round of hash iteration.
3 $H_b(n_1, n_2)$=bit truncation of H_b from bit n_1 to bit n_2.
4 Settings of b_i: Choose either Fixed or Random
  if Fixed
    if i=1
      $b_i \leftarrow (b_1-2^8+1)+H_b(1, 8)$, $b_i \le b_2$;
    else if 1<i<20
      $b_i \leftarrow (b_{i-1}+1)+H_b(8i-7, 8i)$, $b_i \le b_{i+1}$;
    else if i=20
      $b_i \leftarrow (b_{19}+1)+H_b(153, 160)$, $b_i \le b_{20}+2^8$.
  else if Random
    $b_i \leftarrow random[b_1-2^8+1, b_{20}+2^8]$.
5 For b_i rounds of hash iterations:
  if ID=URL=NULL
    $H_i \leftarrow$SHA-512(d, b_i);
  else if ID=NULL
    $H_i \leftarrow$SHA-512(d || URL, b_i);
  else if URL=NULL
    $H_i \leftarrow$SHA-512(d || ID, b_i);
  else if ID and URL are not NULL
    $H_i \leftarrow$SHA-512(d || ID || URL, b_i).
6 H=hash used as login key for offline and online accounts
  $H \leftarrow H_i(1, 256)$, 256-bit truncation of H_i from MSB bit.
7 *Bin2Txt*(H)=Binary-to-text encoding of H.
8 Copy and paste key hash into the key field.
9 If login is fine, exit; else go to Step 1.

Fig.2 Pseudo-code of generation of multihashes as site keys

The 512-bit hash of the concatenated master key and passcode is truncated into 20 partitions with 8-bit each from the MSB bit. This increases the randomness of specific keys for different accounts. If an attacker does not know the exact security, then 5 120 hashes have to be checked for any key pair (master key, passcode). If the attacker knows about the security level, then 2^8 hashes have to be validated for any key pair (master key, passcode).

For the settings of bound b_i, it can be either fixed or random. If the fixed option is chosen, the number of hash iterations will use the standard settings. A user is mobile and can use this method without remembering the number of hash iterations while accessing offline and online login account from different computing systems. If the random option is chosen, the number of hash iterations will be randomly selected by a user within a given range. User's mobility is

weakened unless one can remember the random values of hash iterations while accessing offline and online logins. However, if a user can remember the hash iterations, this option offers stronger resistance to dictionary attack. The best option is a hybrid scheme. Choose fixed option for lower security levels and random option for higher security levels.

Depending on the value existence of username ID and domain URL, the master key undergoes different key hashing and key strengthening using SHA-512 to generate hash H_i. H_i is then encoded from binary to text to fulfill the demands of password requirements such as alphanumeric, mixed lowercase and uppercase, and with punctuation marks.

Here we propose a binary-to-text encoding of *Bin2Txt*(*H*) as in Table 1. Base64 encoding is not used as there are only two punctuation marks included (Borenstein and Freed, 1992). *Bin2Txt*(*H*) converts 6 binary bits into one 8-bit ASCII character. It has a bit expansion of 33%. All types of ASCII characters are included: lowercase, uppercase, digit and punctuation marks. The last group of 4 binary bits of *H* from 253rd to 256th is padded with 2 binary bits of 0 at the right or LSB side. The output of *Bin2Txt*(*H*) is a string of 43 ASCII characters and is used as key hash.

Lastly, copy the hash as site key into the clipboard and paste it on the prompt key field for authentication access. **Remember to clear the clipboard before leaving the computer**.

Table 1 Binary-to-text encoding of *Bin2Txt*(*H*)

Bin	Txt	Bin	Txt	Bin	Txt	Bin	Txt
00	a	16	A	32	0	48	(
01	b	17	B	33	1	49	)
02	c	18	C	34	2	50	*
03	d	19	D	35	3	51	+
04	e	20	E	36	4	52	,
05	f	21	F	37	5	53	-
06	g	22	G	38	6	54	.
07	h	23	H	39	7	55	/
08	i	24	I	40	8	56	:
09	j	25	J	41	9	57	;
10	k	26	K	42	!	58	<
11	l	27	L	43	“	59	=
12	m	28	M	44	#	60	>
13	n	29	N	45	$	61	?
14	o	30	O	46	%	62	@
15	p	31	P	47	&	63	~

Fig.3 illustrates how to change from an old key into a new one. A user can change either the master key, passcode, security level, username or the domain name. There are also proposed usages of 20 security levels as shown in Fig.4.

1 Access the account that needs key change and start from the exit point in Step 9 as in Fig.2.
2 Open another application interface, create a new key pair (master key, passcode) as in Fig.2.
3 Copy and paste the new key hash into the field of “change key” or “change password”.
4 If change is valid, exit; else go to Step 1.

Fig.3 Pseudo-code of changes of key pair (master key, passcode)

Security levels: usages
1 Password file and key management tool like password vault.
2 Finance=>Very important Internet banking.
3 Finance=>Important Internet banking.
4 Finance=>Stock trading.
5 Finance=>Insurance, income tax, ...
6 Very important personal encrypted files, email accounts, instant messengers, …
7 Important personal encrypted files, email accounts, instant messengers, …
8 Very important accounts in working/studying place like email.
9 Important accounts in working/studying place like database.
10 Other accounts in working/studying place like library.
11~20 Other not frequently used offline and online accounts.

Fig.4 Proposed usages of 20 security levels

RESULTS AND DISCUSSION

Table 2 compares various key management tools with multihash key from the aspects of usability, security and possible implementation. A lot of comparisons are attributed by Yee and Sitaker (2006) on Passpet. New features used for comparisons are applicability to offline and online accounts, integrated usages together with other key management tools and possible implementations. It is important to note here that multihash key can be used together with Passpet to earn “Yes” for items 7~9 under the security features in Table 2.

Table 2 Comparisons of key management tools

Features	Key management tools											
	Plain browser	Password autofill	Password safe	Windows live ID	LPWA	HP site password	CPG	Password multiplier	SPP	Pwd Hash	Passpet	Multi-hash key
Usability												
1. Make logging in more convenient	No	Yes	No	Yes	Yes	No	?	Yes	?	No	Yes	?
2. Work with existing websites	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
3. Allow site-by-site migration to tool	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
4. Change individual site keys	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	–	Yes	Yes
5. Log in from other computers	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6. Only need to memorize one secret	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	–	Yes	Yes
7. Enable changing the master secret	–	–	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8. Applicability to offline accounts	–	–	Yes	No	No	No	No	No	No	No	No	Yes
9. Applicability to online accounts	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10. Integrate usages together with other tools	–	–	Yes	No	No	Yes	No	No	No	No	No	Yes
Security												
1. Unique key for each account	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2. Resist offline dictionary attacks	No	No	No	No	No	No	No	Yes	No	No	Yes	Yes
3. Adapt to increasing CPU power	No	No	No	No	No	No	No	Yes	No	No	Yes	Yes
4. Avoid storing keys	Yes	No	No	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
5. Avoid a single central authority	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
6. Resist phishing by fake login forms	No	No	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
7. Resist mimicry of browser UI	No	No	No	No	No	No	No	No	No	No	Yes	No
8. Help the user identify websites	No	No	No	No	No	No	No	No	No	No	Yes	No
9. Stop entering secrets in webpages	No	No	No	No	No	Yes	?	Yes	?	No	Yes	?
Possible implementation												
1. Stand-alone application	No	No	Yes	No	No	Yes	No	Yes	Yes	Yes	No	Yes
2. Single sign-on server	No	No	No	Yes	Yes	No	Yes	No	No	No	No	No
3. Browser extension	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

?: Unknown situation depending on implementation

Multihash key can be used for both offline and online accounts. Possible implementations are stand-alone application and browser extension. These are simple interfaces to input a password or key with unique key images for multiple accounts. Memorability is improved since there is only one secret for various login accounts.

Server is not used and hence there is no central authority. There are no single point of failure and high cost of integration. It is mobile and there is no encrypted storage of site keys. Since there is no integration, multihash key can be used for any existing computer systems.

The passcode is optional to be remembered by a user because it can be converted to be an 8-bit password supplement in one of the two methods of key strengthening (Manber, 1996; Abadi *et al*., 1997). Master key is the password, and when it is combined with the password supplement, they form the full password. Another key strengthening method is also called key stretching, which uses a large amount of hash iterations (Kelsey *et al*., 1997).

The variant of SHA-2, which is SHA-512, is used in the key hashing and key strengthening. This is because there are possible collision attacks to MD5 and SHA-1. The hash truncation creates a 256-bit hash as site key. The unused truncated bit creates a 128-bit security strength (256/2=128) preventing the compromised site keys at the higher security level from revealing the site keys at the lower security level. The passcode also has this feature but is very much less powerful.

For the experimental data of lower bound b_L and upper bound b_H of an old, slow but still popular computer, a tested computer system is a desktop PC with Pentium II 266 MHz 192 MB RAM running on

Windows XP Professional Edition. The b_L is 7600 (correct to the digit of hundred) and b_H is 8200.

For a better tested computer system, it is a laptop PC with Centrino Duo 1.66 GHz 1.5 GB RAM running on Windows XP Home Edition. Its b_L and b_H are 81700 and 93700 respectively. A newer and faster computer system tends to have a larger range to support more security levels and accounts. The size of partition can also be increased from 2^8 to a higher value like 2^{12}, while keeping or changing the number of digits of passcode.

Using the proposed settings, the key strengthening has an access time from 0.2 s to 2 s. This is an efficient range of acceptable login processing time. It can be calibrated to be parallel with the advances in computer technologies for new releases of multihash key. Moore's Law is a good rule to judge the calibration, which is about one bit faster for every two years (Wikipedia, 2007d).

Key hashing and key strengthening are also good techniques to resist offline and online dictionary attacks as well as pre-computation attacks. To prevent phishing attacks and spoofing attacks, multihash key can either be used together with other anti-phising tools like petname system and Passpet, or include domain name URL in its key hashing process. Malicious server attack is also prevented as different accounts have unique passwords. For homograph attack due to visually similar Unicode graphic symbols, the implementation of multihash key has to be able to support Unicode characters.

LIMITATIONS

Multihash key can be implemented as a stand-alone application with no change of setting at the server side. However, it is vulnerable to password file compromise attacks and message log file attacks. Nevertheless, domino effect of password reuse can be avoided. To get rid of password file compromise attacks and message log file attacks, some countermeasures (Gouda *et al*., 2005) can be adopted by changing the settings of authentication approach at client and server sides.

Acting as a stand-alone application, multihash key requires a user to perform extra steps. These steps are creating a key, copying and pasting it to a login prompted textbox. The user also needs to remember the security level of an account, an at least 128-bit master key and a numeric 4-digit passcode. These cause the solution to be not user-friendly.

To facilitate the application, multihash key has to be integrated into the user interface of each authentication application. Therefore, the item 1 of usability in Table 2 about convenient logging in depends on implementation.

For security level, it can be jotted into a notebook in plaintext form because it is not an essential secret. Alternatively, for online account, the user can be reminded about the security level whenever the user sends the username to the server. This allows an attacker to reduce the number of hash testing by 20 times, or 4.32 bits ($=\log_2 20$).

For numeric 4-digit passcode, it gives an extra security of 13.29 bits ($=\log_2 10000$) and is not an essential secret. This passcode can be made constant for user with poor memory. For user who can remember 128-bit master key, 4-digit passcode and security level, the effective security strength is 145.61 bits. For user who can remember only the 128-bit master key, its security strength is 128 bits. Hence, security can be compensated for better usability.

Using multihash key, limited number of multiple offline and online accounts can be supported as compared to the almost infinite number of online accounts for LPWA, HP Site Password, CPG, Password Multiplier, SPP, PwdHash and Passpet. For more accounts, faster computer system is needed to have larger bound range. Or else, the partition between any two security levels has to be reduced.

CONCLUSION

The proposed invention of multihash key requires users to remember a master key and passcode to generate unique key hashes or site keys for multiple accounts. For security level, username and domain name of a specific account, users can choose to write them down somewhere as there are not critical secrets. This is a balance between the usability and security.

Multihash key can be used for offline and online accounts, where existing similar key management tools without encrypted site key storage can only be applied to online accounts. It is hoped that this pro-

posal can release the human memory burden on required passwords or keys for various types of increasing accounts. To have better resistance to phishing and spoofing attacks, try to use multihash key together with an anti-phishing tool like petname system and Passpet.

ACKNOWLEDGEMENT

The authors would like to thank Mr. Ching-Weng Hong with Multimedia University, Malaysia, for conducting an experiment to look for the bounds of key strengthening. Thanks are also given to the anonymous reviewers for their comments.

References

Abadi, M., Lomas, T.M.A., Needham, R., 1997. Strengthening Passwords. Technical Reports of SRC (Systems Research Center) SRC-1997-033. Palo Alto, CA, USA, p.1-11.

Adams, A., Sasse, M.A., 1999. Users are not the enemy. *Commun. ACM*, **42**(12):40-46. [doi:10.1145/322796.322806]

Borenstein, N., Freed, N., 1992. Base64 Content-Transfer-Encoding. MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies. RFC 1341. IETF, Sterling, Virginia, USA, p.17-19.

Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., Stewart, L., 1999. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617. IETF, Sterling, Virginia, USA, p.1-34.

Gabber, E., Gibbons, P., Matias, Y., Mayer, A., 1997. How to make personalized web browsing simple, secure, and anonymous. *LNCS*, **1318**:17-31. [doi:10.1007/3-540-63594-7]

Gouda, M.G., Liu, A.X., Leung, L.M., Alam, M.A., 2005. Single Password, Multiple Accounts. Proc. 3rd Int. Conf. on Applied Cryptography and Network Security. New York City, NY, USA, p.1-12.

Halderman, J.A., Waters, B., Felten, E.W., 2005. A Convenient Method for Securely Managing Passwords. Proc. 14th Int. Conf. on World Wide Web 2005. Chiba, Japan, p.471-479. [doi:10.1145/1060745.1060815]

Ives, B., Walsh, K.R., Schneider, H., 2004. The domino effect of password reuse. *Commun. ACM*, **47**(4):75-78. [doi:10.1145/975817.975820]

Kanaley, R., 2001. Login Error Trouble Keeping Track of All Your Sign-ons? Here's a Place to Keep Your Electronic Keys, but You'd Better Remember the Password. San Jose Mercury News, Feb. 4, 2001.

Karp, A.H., 2003. Site-Specific Passwords. Technical Report of HP Laboratories Palo Alto HPL-2002-39 (R.1). Palo Alto, CA, USA, p.1-9.

Karp, A.H., Poe, D.T., 2004. System-Specific Passwords. USPTO Published Application for Patent US2004/0025026. Alexandria, VA, USA, p.1-6.

Kelsey, J., Schneier, B., Hall, C., Wagner, D., 1997. Secure applications of low-entropy keys. *LNCS*, **1396**:121-134. [doi:10.1007/BFb0030404]

Kormann, D.P., Rubin, A.D., 2000. Risks of the passport single signon protocol. *Computer Networks*, **33**:51-58. [doi:10.1016/S1389-1286(00)00048-7]

Lilly, G.M., 2004. Device for and Method of One-Way Cryptographic Hashing. USPTO Patent US6829355. Alexandria, VA, USA, p.1-8.

Luo, H., Henry, P., 2003. A Common Password Method for Protection of Multiple Accounts. Proc. 14th IEEE 2003 Int. Symp. on Personal, Indoor and Mobile Radio Communication (PIMRC 2003). Beijing, China, **3**:2749-2754. [doi:10.1109/PIMRC.2003.1259242]

Manber, U., 1996. A simple scheme to make passwords based on one-way functions much harder to crack. *Computers and Security*, **15**(2):171-176. [doi:10.1016/0167-4048(96)00003-X]

Matias, Y., Mayer, A., Silberschatz, A., 1997. Lightweight Security Primitives for E-commerce. Proc. USENIX Symposium on Internet Technologies and Systems. Monterey, California, USA, p.95-102.

NIST, 2002. FIPS PUB 180-2: Secure Hash Standard. CSRC, NIST. Gaithersburg, MD, USA, p.1-79.

Rivest, R., 1992. The MD5 Message-Digest Algorithm. RFC 1321. IETF, Sterling, Virginia, USA, p.1-21.

Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J.C., 2005. Stronger Password Authentication Using Browser Extensions. Proc. 14th USENIX Security Symposium (SEC'05). Baltimore, MD, USA, p.17-32.

Wikipedia, 2007a. MD5. Wikipedia the Free Encyclopedia. Accessed on Feb. 1, 2007, <http://en.wikipedia.org/w/index.php?title=MD5&oldid=142373953>

Wikipedia, 2007b. SHA Hash Functions. Wikipedia the Free Encyclopedia. Accessed on Feb. 1, 2007, <http://en.wikipedia.org/w/index.php?title=SHA_hash_functions&oldid=141311777>

Wikipedia, 2007c. Petname. Wikipedia the Free Encyclopedia. Accessed on Feb. 1, 2007, <http://en.wikipedia.org/w/index.php?title=Petname&oldid=93050718>

Wikipedia, 2007d. Moore's Law. Wikipedia the Free Encyclopedia. Accessed on Feb. 1, 2007, <http://en.wikipedia.org/w/index.php?title=Moore%27s_Law&oldid=142016849>

Yee, K.P., Sitaker, K., 2006. Passpet: Convenient Password Management and Phishing Protection. Proc. Symposium on Usable, Privacy and Security. Pittsburgh, PA, USA, p.32-43. [doi:10.1145/1143120.1143126]