A Method for Detecting the Exposure of a Secret Key in Key-Insulated Scheme

Younggyo Lee[†] **and Dongho Won**[††]

Department of Internet Information, Seoil College, Korea

Information Security Group, Department of Computer Engineering, Sungkyunkwan University, Korea

Summary
Dodis *et al* proposed a key-insulated signature scheme in 2003. In the scheme, total lifetime of a certificate is divided to time periods and different secret keys are used for each time period. The master secret key is stored in the physically secure device and is not used for signing directly. The different secret keys are used for signature in each time period and they are refreshed by a computation with the master key. Therefore, the scheme can minimize the damage caused by a secret key's exposure. However, it can not protect the user from the secret key's exposure perfectly. We propose a method which can detect even a single illegitimate signature due to the exposure of a secret key in the key-insulated scheme. The method uses the one-time hash chain based on NOVOMODO and the counter. And it requires small modification of traditional PKI. The method can prevent the users from compromising a secret key effectively in the key-insulated signature scheme.

Key words:
key-insulated signature, one-time hash chain, NOVOMODO, PSD

1. Introduction

PKI (Public Key Infrastructure) is a widespread and strong technology for providing the security (integrity, authentication, and non-repudiation) using public key techniques. The main idea of PKI is based on the digital certificate that is a digitally signed statement binding an entity (user or authority)'s identity information and his public key by CA (Certificate Authority)'s secret key. If the entity's secret key is compromised or the entity's identity information is changed, the entity makes a request to the CA for revoking its own certificate. The information whether the certificate is revoked or not is called CSI (Certificate Status Information) and CRL (Certificate Revocation Lists) is one of the most well-known methods for CSI [2, 7, 13].

1.1 CRL size and communication cost

The CRL system has an advantage of its simplicity, however, it has a disadvantage of high communication cost between the user and the CA's Repository (or Directory) storing the CRL. Therefore, in order to reduce the size of certificate revocation list (communication costs), computation cost and storage amounts, various methods have been suggested up to date. These are Delta-CRL, CRL DP (Distributed Points), Over-issued CRL, Indirect CRL, Dynamic CRL DP, Freshest CRL, CRT (Certificate Revocation Trees), NOVOMODO, Authenticated Directory *et al* [1, 2, 5, 6, 7, 9, 11, 12, 13, 14, 16].

12 OCSP and computation cost

If the client or user needs very timely CSI, an online certificate status service such as the OCSP (Online Certificate Status Protocol) is more convenient than the off-line method such as CRL *et al* [9]. In OCSP, since the client does not need to download a CRL from the CA's Repository, the high communication cost between the client and the CA's Repository and the storage spaces for storing the CRL are not required. However, if the CSI requests are centralized to an OCSP Responder, the OCSP Responder can have a risk of DoS (Denial of Service) attacks [15]. In order to reduce the risk of DoS attacks, the OCSP Responder can pre-produce a signed value for responses in a short time. However this may allow a possibility of the replay attacks [15, 16].

13 T-OCSP and D-OCSP

To reduce the overload of single OCSP Responder in "T-OCSP (Traditional-OCSP)", "D-OCSP (Distributed-OCSP)" is introduced [4, 15]. In D-OCSP, if distributed OCSP Responders have the same secret key, the possibility of OCSP Responder's secret key exposure is very high [16]. Therefore, in the general D-OCSP model, each OCSP Responder has a different secret key and clients must have all of the OCSP Responder's certificates for verifying CSI response of OCSP Responders. This gives an increase of storage amounts consumption or communication costs for acquiring the OCSP Responder's certificates. For solving this problem, the method of single public key was proposed in D-OCSP-KIS (Distributed

Manuscript received September 5, 2008.
Manuscript revised September 20, 2008.

OCSP based on Key-Insulated Signature) by Koga and Sakurai [15]. Also for solving the problem that the length of the single public key is in proportion to the number of OCSP Responder in D-OCSP-KIS, Daehyun Yum and Piljoong Lee proposed the D-OCSP-IBS(Distributed OCSP based on Identity-Based Signature) that the length of the single public key is constant and short [4]. In addition, a method for detecting the exposure of an OCSP Responder's session secret key in D-OCSP-KIS was proposed by Younggyo Lee *et al* [20, 21].

14 Our Contributions

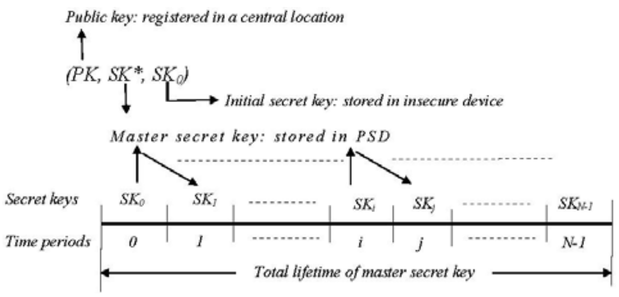
However, the study for preventing the exposure of a user's secret key is hardly accomplished up to date. Dodis *et al* proposed a key-insulated signature scheme at 2003 in [17, 18] In the scheme, the master secret key is stored in the physically secure device and not used for signing directly. Total lifetime of the master secret key is divided into time periods and the different secret keys refreshed by the master key are used for each time period. Therefore the scheme can minimize the damage caused by the secret key's exposure but can not protect the user from the secret key's exposure in a time period perfectly. Just a single illegitimate signature by the exposure of a secret key can give extensive damage to the user in E-business or E-commerce. In this paper, we propose a method that can detect even a single illegitimate signature caused by the exposure of a secret key in the key-insulated scheme. The method uses the one-time hash chain based on NOVOMODO and the counter. The method can prevent users from compromising the secret key effectively in the key-insulated signature scheme. The rest of this paper is organized as follows. In section 2, we present the key-insulated signature scheme and NOVOMODO. In section 3, we propose a method for detecting the exposure of a secret key in the key-insulated scheme. In section 4, we analyze the proposal in detail. In section 5, we show the characteristics of proposal and compare the proposed method to other methods. Finally, in section 6, we conclude our paper.

2. Key-insulated signature scheme and NOVOMODO

In this section, we present the key-insulated signature scheme proposed by Dodis *et al* first and show Micali's NOVOMODO that our proposal based on.

2.1 Key-insulated signature scheme

In the key-insulated signature scheme, the master secret key (SK^*) is stored in the physically secure device (PSD) and not used for signing directly. Total lifetime of the master secret key is divided into time periods and the different secret keys refreshed by the master key are used for each time period as shown in Fig. 1. The secret key (SK_i) stored in the insecure device is refreshed at discrete time periods via interaction with the physically secure device which stores the master secret key. Therefore the scheme can minimize the damage caused by the secret key exposure until the secret key is changed. The key-insulated signature scheme needs a 5-tuple of poly-time algorithms *(Gen, Upd*, Upd, Sign, Vrfy)* such that:



Public key: registered in a central location
(PK, SK^*, SK_0)
Initial secret key: stored in insecure device
Master secret key: stored in PSD
Secret keys
SK_0
SK_1
SK_i
SK_j
SK_{N-1}
Time periods
0
1
i
j
N-1
Total lifetime of master secret key

Fig. 1 The concept of key-insulated signature scheme

- *Gen* : the key generation algorithm
- U_{pd}^*: the physically secure device key-update algorithm
- U_{pd} : the user key-update algorithm
- *Sign* : the signing algorithm
- *Vrfy* : the signing verification algorithm

At first, a user generates (PK, SK^*, SK_0) using $Gen(1^k, N)$ and publishes PK in the central location. The user stores SK^* in the physically secure device and stores SK_0 in the general device. SK_0 is used for signing during the time period 0. When a time period (if the secret key is SK_i) is finished, the user gets $SK'_{i,j}$ from the secure device using $U_{pd}^*(i,j,SK^*)$. The user computes the next time period's secret key $SK_j = U_{pd}(i, j, SK_i, SK'_{i,j})$ using SK_i and $SK'_{i,j}$. *Sign* algorithm is used for signing a message $M(Sign_{ski}(i,M) \rightarrow i,s)$ and *Vrfy* algorithm is used for verifying the signature s of $M(Vrfy_{PK}(M,<i,s>) \rightarrow b)$ [17, 18].
After computing SK_j, the user erases SK_i and $SK'_{i,j}$. SK_j is used for signing a message during the time period j without further access to the physically secure device. Therefore the scheme can minimize the damage caused by the secret key's exposure. However the scheme cannot protect the user from the exposure of user's secret key in a time period perfectly. If the secret key (SK_i in Fig. 1) of a time period is exposed incidentally, it can be used for signing by an attacker acquiring it until it is changed (SK_j in Fig. 1). Therefore the key-insulated signature scheme can minimize the damage caused by the secret key's exposure but can not protect the user from the secret key's exposure perfectly.

2.2 NOVOMODO

In NOVOMODO, a user's certificate $Cert_{user}$ includes two 20-byte (160-bit) hash values in addition as shown in Fig. 2. The one *(X_{365})* is used as "validity target" and the other *(Y_1)* is used as "revocation target."

$$Cert_{user} = Sig_{SK_{CA}}(PK_{user}, SN, I, S, V, \overset{\text{Validity target}}{\overset{\downarrow}{X_{365}}}, \underset{\text{Revocation}}{\underset{\uparrow}{Y_0}})$$

Fig. 2 The structure of certificate in MOVOMODO

These values are produced by applying one-way hash function to two different 20-byte values randomly selected in CA. When the time interval is one day, The value X_{365} is computed by 365 hashing operations from X_0: $X_1=h(X_0)$, $X_2=h(X_1)$, .., $X_{365}=h(X_{364})$; and Y_1 by 1 hashing operation from Y_0: $Y_1=h(Y_0)$. The CA keeps secretly the initial values X_0, Y_0 and all the intermediate values X_i. The CA releases the corresponding intermediate hash value to each user as a certificate's "validity proof" *(X_i)* or "revocation proof" *(Y_0)* at initial time of each interval. In E-business or E-commerce based on PKI, the verifier getting the user's certificate and corresponding hash value compares two values using the hash function. If the result of comparison is the same, the verifier gets the message authentication and integrity concerning the released hash value *(X_i or Y_0)* periodically and confirms the user's certificate status by the hash value [16]

Table 1 The comparison of storage amounts for hash chain stored in each time interval in CA

time interval		1 day	1 hour	1 minute	15 seconds	1 second
X_i	1 certificate	7.3 K	175.2 K	10.3 M	41 M	615 M
	1 million certificates	7.3 G	175.2 G	10512 G	42048 G	630720 G
10 levels (A_i~J_i)	1 certificate	73 K	1752 K	103 M	410 M	6150 M
	1 million certificates	730 G	1752 G	105120 G	420480 G	6307200 G

Unit : bytes.

The well-known CRL holds the list of revoked certificate in the list. If a certificate does not appear in the CRL, then the certificate is deduced to be valid. Thus CRL is a double negative scheme but NOVOMODO is a positive/negative scheme that it releases the validity or revocation status. A big advantage of NOVOMODO is that it uses the hash function of small computation cost and storage amount (always produces 20-byte outputs) as given in Table 1. CA may provide the hash value to a client as OCSP response or release it periodically or post it on the WWW (World Wide Web). The hash value is small. However, since the inversing of the hash function is practically impossible, anybody (e.g., attacker) cannot forge it and it offers the message authentication and integrity If the certificate includes 10 validity targets and 1 revocation target and CA releases a different validity proof, a user can use the certificate as 10 certificates with different levels [16].

3. A method detecting the exposure of a secret key in the key-insulated signature scheme

As we mentioned earlier, the key-insulated signature scheme can minimize the damage caused by the exposure of a secret key but can not protect the user from the exposure of a secret key perfectly. Therefore, in this section, we propose a method that detects even a single illegitimate signature by the exposure of a secret key of user. This proposal uses the one-time hash chain based on NOVOMODO and the counter. And its framework (includes the certificate format) is similar to general PKI.

3.1 Requirements

Our proposal has some requirements as follow.

1) Both the user's signature and the verifier's certificate status validation (using OCSP) are established by 1 : 1 in real time.

2) The user's certificate includes a hash value (of 20 bytes) of "detection mark" for detecting the own secret key's exposure.

32 Proposal

[Initial procedure : the user's certificate issuance by CA]

1) In this proposal, the user's secret key is restricted by the number of signatures. Let *K* be the total number of signature usages for a user. For an example, *K* is 10,000 if each user's certificate is expired after 10,000 signing operations. Thus, the certificate of the user is expired after 10,000 certificate status validations. The user computes Z_0 by 10,000 hashing operations from random input value Z_K using *h* as follows.

$$Z_K \xrightarrow{h} Z_{K-1} \xrightarrow{h} \cdots Z_i \cdots \xrightarrow{h} Z_1 \xrightarrow{h} Z_0$$

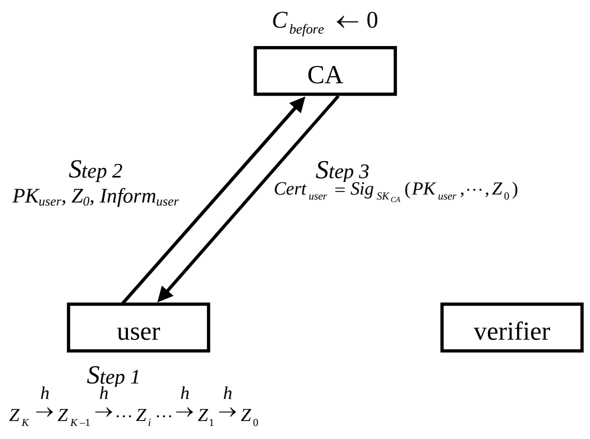

$C_{before} \leftarrow 0$
CA
Step 2
PK_{user}, Z_0, $Inform_{user}$
Step 3
$Cert_{user} = Sig_{SK_{CA}}(PK_{user}, \cdots, Z_0)$
user
verifier
Step 1
$Z_K \xrightarrow{h} Z_{K-1} \xrightarrow{h} \cdots Z_i \cdots \xrightarrow{h} Z_1 \xrightarrow{h} Z_0$

Fig. 3 The procedure of certificate issuance by CA

2) The user sends the own public key, the own identification information, and the final hash value, Z_0 safely to CA for the request of own certificate issuance.

3) The CA issues the user's certificate $Cert_{user}$ by using own secret key. In $Cert_{user}$, the hash value Z_0 is also included as follows. SN is the serial number of certificate and V represents the validity period. I and S denote issuer and subject of certificate, respectively. And the CA sets the counter C_{before} for user to 0. The counter C_{before} is stored and managed in CA.

$$Cert_{user} = Sig_{SK_{CA}}(PK_{user}, SN, I, S, V, Z_0)$$
$$C_{before} \leftarrow 0$$

4) The user stores his own master secret key, the input value and all intermediate values of step 1 in PSD securely.

[Service procedure : signature and certificate status validation]

1) When the user signs, he sends his own certificate and the hash value Z_i with the signed message M to the verifier. The Z_i is taken out from PSD and delivered in the order of $Z_1, Z_2, Z_3, ..., Z_K$

$$M, Sig_{SKuser}(M), Cert_{user}, Z_i$$

2) After receiving the signed message from user, the verifier requests the user's certificate status information to CA. Then he also delivers the user's certificate and the hash value Z_i with the OCSP request.

$$OCSP\ request, Cert_{user}, Z_i$$

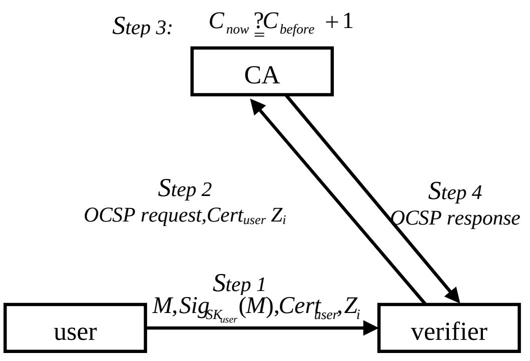

Step 3: $C_{now} \overset{?}{=} C_{before} + 1$
CA
Step 2
OCSP request,$Cert_{user}$ Z_i
Step 4
OCSP response
Step 1
$M, Sig_{SK_{user}}(M), Cert_{user}, Z_i$
user
verifier

Fig. 4 The procedure of signature and certificate status validation

3) When the CA receives the OCSP request from the verifier, it repeatedly computes the hashing operation until the hashing operation result of Z_i is equal to the hash values Z_0 contained in the certificate.

$$h(Z_i)^i \overset{?}{=} Z_0$$

If this holds, the CA can confirm the message authentication and integrity about the hash value Z_i. And the counter C_{now} (=i; the number of hashing operation in the present request) is compared with the stored counter C_{before} (the number of hashing operation in the previous request) by the following condition.

$$C_{now} \overset{?}{=} C_{before} + 1$$

If the condition is satisfied, then the CA confirms that the user's secret key was not used in an illegitimate signature. Otherwise, CA concludes that the user's secret key was exposed and used illegitimately by an attacker.

4) In step 3, if the condition about the counter is satisfied, the CA increases the counter C_{before} by 1 and sets the counter C_{now} to "*0*." And the CA delivers the corresponding OCSP response including the user's certificate status information ("*good*") to the verifier. Otherwise, he returns the response ("*revoked*") to the verifier. And he revokes the user's certificate and informs user that user's secret key is exposed and used by an attacker illegitimately. The CA goes to step 1 to process the next OCSP request.

$$OCS\,\mathrm{Pr}esponse$$
$$C_{before} \leftarrow C_{before} + 1$$
$$C_{now} \leftarrow 0$$

4. Analysis

In this section, we analyze the proposed method in detail for each step.

1) The hash value Z_0 included in the certificate
The hash value Z_0 included in the user's certificate is used as "detection mark" for detecting the exposure of the user's secret key. The hash value is a final value of hash chain computed by the user. When Z_i is received with the request of the certificate status information from the verifier, the CA can have the message authentication and integrity about Z_i by comparing Z_i with Z_0through the hashing computation.

2) The hash values of $Z_K,...,Z_i,...,Z_1$ stored in PSD
Since the hash values of $Z_K,...,Z_i,...,Z_1$ are computed using the one-way hash function, the inverse computation is impossible. These values are computed in a user's PC and stored in PSD securely. When user signs a message, these values are taken out from PSD and delivered in order of $Z_1,...,Z_i,...,Z_K$ to the verifier with signed message step by step. Therefore an attacker cannot compute the hash values. In other words, these values are uniquely delivered values and the attacker cannot forge and reuse the hash values.

3) The hash value Z_i delivered to the CA via the verifier
The hash value Z_i with signed message is delivered to the verifier and it with the OCSP request is sent to the CA. The value cannot be forged by an attacker and she cannot re-compute the previous hash value Z_{i+1}. And the CA can have the integrity about Z_i received by hashing operation and comparing Z_i with Z_0 included in certificate. Therefore Z_i does not need the signature when it is delivered to the CA via the verifier.

4) The counter C_{before} and C_{now} in the CA
The counter C_{before} and C_{now} are used for judging the illegitimate signature by the user's secret key. The C_{before} is increased one by one in normal case and its value represents the number of the previous hashing operations. The C_{now} is the number of present hashing operations when comparing the Z_i and Z_0 in CA. Therefore if the user's signature is not used illegitimately, the difference of these counters is ($C_{now} = C_{before} + 1$). Otherwise, we acknowledge that a secret key of user is compromised and the illegitimate signature use is performed by the attacker. In case that the difference of these counters is 0 ($C_{now} = C_{before}$), the illegitimate signature use is performed using the hash value of latest signature. In case that the C_{now} is less then the C_{before} ($C_{now} < C_{before}$), the illegitimate signature is performed using the old hash value. In cay case, even if the attacker forges illegitimate signature only once, the proposed method can detect it immediately.

5. Characteristics and comparisons

In this section, we explain the characteristics of the proposed method and compare it to traditional PKI and key-insulated signature scheme. The detailed characteristics are as follows:

[Immediate exposure detection of a secret key of user]

The user's secret key is kept securely and is used at the signature in PKI. However in the key-insulated scheme, the possibility of each user's secret keys exposure is higher than that of server's because the user's PC has the weak points concerning the system security in general (e.g., firewall) and each secret key is stored in un-physically secure device. In the proposal, the certificate status request for each user's signature is checked one by one using the one-time hash chain. Even if the attacker that acquires a secret key of the user uses only one-time illegitimate signature, the CA can detect it immediately at the procedure of the certificate status validation.

[The computation costs and storage amounts in the user]

Table 2: The computation costs and storage amounts in the user by the number of signature

The number of signatures	100	1,000	5,000	10,000
The number of hashing operations	100	1,000	5,000	10,000
The number of stored hash values (Max.)	100	1,000	5,000	10,000
Storage amount for the hash values (Max.)	1.95 Kbytes	19.53 Kbytes	97.66 Kbytes	195.31 Kbytes

In the proposal, the user uses the hash chain to CA for delivering the unique value that an attacker cannot forge it. Each user computes the number of 10,000 hashing operations and stores the input and all the intermediate values safely. Table 2 shows the computation costs and storage amounts of the user by the number of signature. In case the number of 10,000 signatures, 10,000 hashing operations are required and this is equaled to only the number of 1 signature operation because the hash operation is at least 10,000 times faster in computation than a signature operation [15, 16]. 195.31 K bytes are needed for storing the hash chain and the hash chain is reduced according to delivery of hash value at the signature. Therefore the computation cost and the storage amounts are not the overload to user.

Table 3: The comparison of the proposal with other methods

		proposal	traditional PKI	key-insulated signature scheme
certificate	structure	modified (+ 20bytes)	-	-
	validity	10,000 times (+- is possible)	365 days	365 days
OCSP request form		adds certificate, hash value	-	-
OCSP response form		-	-	-
additional computation costs for CA		average 5,000 hashing operations / OCSP request (during service)		-
additional storage amount		2 bytes /user (1. 907 M bytes in total)		-
additional communication costs		certificate, hash value / OCSP request (acceptor $\rightarrow$ CA)	-	-
additional computation costs for user	initially	10,000 hashing operations / user	-	$Gen(1^k,N)$ for PK,SK^*,SK_0
	each time interval	-	-	Upd^*, Upd
additional storage amount for user		maximum 195.31 K bytes	-	secret key at each period
additional loads for acceptor		receive hash value, send certificate, hash value	-	-
possibility of wrong response		×	O	O
security of user's secret key		high	no	medium
detection of user's secret key exposure		O	×	×
possible number of illegitimate signature		only 1	from hundreds to thousands of times	from several to hundreds of times
detector of illegal signature		CA	×	×
detection time of illegitimate signature		at once	× (later)	× (later)
signer : verifier		1 : 1	n : n	n : n
implementation		Internet banking, E-commerce, E-business	All kind	All kind

Notes. O : Supported, × : Not supported for the number of certificates is 1,000,000 and the number of usage times (K) is 10,000

[The number of use times of user's secret key]

In the proposed method, the secret key (the certificate) of a user is used for the limited number of times because one-time hash value is used in the validity proof of user's secret key. As above, it is supposed that the number of times K is 10,000. In this case, the CA computes 1 hash operation at the first OCSP request and the 10,000 hash operations at 10,000-th OCSP request for detecting the exposure of the user's secret key. Thus, the CA computes average 5,000 hashing operations. Of course, K can be set bigger (e.g., 20000, 50000, 100000) or smaller (e.g., 8000, 5000, 2000). However setting K bigger than 10,000 is inefficient because the required computation time is larger than 1 digital signature time. If different 3 hash values are added to the user's certificate, the number of usage times of user's secret key is increased and the number of hash operation is decreased in the CA. Suppose that these 3 hash values are computed by 5,000 hashing operations from different initial values. The total number of usage times K is 15,000, but the CA only computes average 2,500 hashing operations for 1 OCSP request. When the user spends all of the number of usage times, it computes a new hash-chain and transfers the only final hash value to the CA unless its secret key is compromised or public key and other information is changed [20-22].

[The computation costs and storage amounts in CA]

In this proposal, the CA should manage the counter C_{before} for each user. The integer variable of 2 bytes is needed for storing the maximum value 10,000 in it. Suppose that the CA is a big CA with 1,000,000 issued certificates. The storage amounts of 1.907 M bytes are needed for 1,000,000 certificates. The storing and managing of the quantity does not give an overload to CA. As we mentioned earlier, the CA computes averagely 5,000 hashing operations when K is 10,000. The computation costs also do not give an overload to the CA.

[The additional communication costs]

In this proposal, the hash value of 20 bytes is delivered from the user to the verifier additionally and the communication cost is small. And the user's certificate and the hash value are sent to the CA with OCSP request. The CA can acquire the user's certificate in Directory directly. In any case, the CA should acquire 1 certificate and 20-byte hash value additionally in this proposal.

[The PKI structure and procedure having small modifications]

There are slight differences between our proposal and the traditional PKI in the structure (including the certificate format) and procedure. The differences in our PKI structure and procedure can be summarized as follows: (1) each user has a hash chain and sends the hash value Z_i with the signed document, (2) verifier sends the hash value Z_i with the OCSP request, and (3) CA maintains the counter C_{before}, computes the hash operations and comparisons. The difference in our certificate format is that each user's certificate has a hash value of 20 bytes for the detection mark in addition.

[The frequency of the communicating with the PSD]

In this proposal, the user needs to communicate with the PSD to get the hash values of $Z_K,...,Z_i,...,Z_1$ whenever he signs a message. Thus the frequency of the communicating with the PSD is increased, and the frequency of the PSD's connection to insecure environments is also increased. After all, it puts the PSD in a higher risk of exposure. As the master secret key is stored in PSD, the key-insulated signature system will be broken by the only one exposure of PSD. Therefore, for reducing the risk of the master secret key, two PSDs are required as like PKIPE(Parallel Key-Insulated Public Encryption) [6]. The master secret key is stored in PSD_1 and the hash values are stored in PSD_2 In PSD_2, the user-defined password can be used to protect the hash values against abuse [6]

We compare the proposed method to traditional PKI and key-insulated signature scheme such as in Table 3. In Table 3, we see that the proposal has some disadvantages. These disadvantages include the facts that the user's secret key has a limited usage times, the small computation costs and storage amounts is increased in CA and user and the small communication costs is increased between verifier and CA. However the proposal has some advantages. One big advantage of this proposal is the detection the exposure of user's secret keys. And it has an advantage of delivering correct response to verifier. In addition, it has an advantage of accounting the user's certificate rate by the number of times.

6. Conclusions

In the key-insulated signature scheme, the damage of the secret key's exposure can be minimized. But, even just one-time illegitimate signature by the exposure of a secret key can give extensive damage to the user in E-business or E-commerce. Therefore we propose a method that can detect immediately even just one-time illegitimate signature by the exposure of a secret key of user. The method uses the one-time hash chain based on NOVOMODO and its structure and procedure are similar to the traditional PKI.

We analyze the proposed method in detail and compare it to the traditional PKI and the key-insulated signature scheme. Our proposal uses the one-time hash chain requiring small resource and can prevent the users from compromising the secret key effectively and perfectly in the key-insulated signature scheme. The method can increase the security of the user's secret key in PKI.

References

[1] A. Malpani, R. Housley, T. Freeman, *Simple Certificate Validation Protocol(SCVP)*, IETF Internet Draft, June, 2002.

[2] C. Adams, P. Sylvestor, M. Zolotarev, R. Zuccherato, *Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols*, IETF RFC 3029, February, 2001.

[3] Dae Hyun Yum, Jae Eun Kang, and Pil Joong Lee, "Advanced Certificate Status Protocol," in *MMM-ACNS 2003*, LNCS 2776, pp. 229-240, 2003.

[4] Dae Hyun Yum, Pil Joong Lee, "A Distributed Online Certificate Status Protocol Based on GQ Signature Scheme," in *ICCSA 2004*, LNCS 3043, pp.471-480, 2004.

[5] Jianying Zhou, Feng Bao, and Robert Deng, "An Efficient Public-Key Framework," in *ICICS 2003*, LNCS 2836, pp.88-99, 2003.

[6] Jong-Phil Yang, Chul Sur, Hwa-Sik Jang, and Kyung-Hyune Rhee, "Practial Modification of An Efficient Public-Key Framework," *2004 IEEE International Conference on e-Technology*, e-Commerce and e-Service, March 2004.

[7] Jose L. Munoz, Jordi Forne, Oscar Esparza, and Miguel Soriano, "A Certificate Status Checking Protocol for the Authenticated Dictionary," in *MMM-ACNS 2003*, LNCS 2776, pp.255-266, 2003.

[8] Leo Reyzin, *General Time/Storage Tradeoffs for Hash-Chain Re-comoutation*, unpublished manuscript.

[9] M. Myers, R. Ankney, A. Mappani, S. Galperin, C. Adams, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, IETF RFC 2560, June, 1999.

[10] NIST FIPS (Federal Information Processing Standards Publication) 186-1, *Digital Signature Standard*, December, 1998.

[11] P.C.Kocher, "On Certificate Revocation and Validation," in *Financial Cryptography (FC'98)*, LNCS 1465, pp.172-177, Springer-Verlag, 1998.

[12] Paul. Kocher, *A Quick Introduction to Certificate Revocation Tree(CRTs)*, Technical Report, Valicert, 1999.

[13] R. Housley, W. Ford, W. Polk, D. Solo, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, IETF RFC 2458, January, 1999.

[14] R. Housley, W. Ford, W. Polk and D. Solo, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, IETF RFC 3280, April, 2002.

[15] Satoshi Koga, Kouichi Sakurai, "A Distributed Online Certificate Status Protocol with a Single Public Key," in *Public Key Cryptography 2004*, LNCS 2947, pp.389-401, 2004.

[16] Silvio Micali, "NOVOMODO; Scable Certificate Validation And Simplified PKI Management," in *1st Annual PKI Research Workshop Preproceedings*, pp.15-25, 2002.

[17] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung, "Key-Insulated Public Key Crytosystems," in *EUROCRYPT 2002*, LNCS 2332, pp. 65-82, 2002.

[18] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung, "Strong Key-Insulated Signature Schemes," in *PKC 2003*, LNCS 2567, pp. 130-1442, 2003.

[19] Goichiro, Yumiko Hanaoka, and Hideki Imai, "Pararell Key-Insulated Public Key Encryption," in *PKC 2006*, LNCS 3958, pp. 105-122, 2006.

[20] Younggyo Lee, Injung Kim, Seungjoo Kim, and Dongho Won, "A Method for Detecting the Exposure of an OCSP Responder's Session Private Key in D-OCSP-KIS," in *Euro PKI 2005*, LNCS 3545, pp.215-226, 2005.

[21] Younggyo Lee, Jeonghee Ahn, Seungjoo Kim, and Dongho Won, A Method for Detecting the Exposure of an OCSP Responder's Private Key using One-Time Hash Value," in *IJCSNS International Journal of Computer Science and Network Security*, VOL. 5 No.8, pp. 179-186, August 2005.

[22] Younggyo Lee, Jeonghee Ahn, Seungjoo Kim, and Dongho Won, "A PKI System for Detecting the Exposure of a User's Secret Key," in *EuroPKI 2006*, LNCS 4043, pp.248-250, 2006.

[23] http://www.eskimo.com/~weidai/benchmarks.html.

Younggyo Lee received the B.E. and M.E. degrees in Electronic Engineering from Hanyang University in 1986 and 1991. He received the Ph.D. degree in Computer Engineering in Sungkyunkwan University. He is currently a professor of Department of Internet Information of Seoil College in Korea. His current research interest is in the area of cryptography, particularly regarding the design of secure PKI protocols.

Dongho Won received his B.E., M.E., and Ph.D. degrees from Sungkyunkwan University in 1976, 1978, and 1988, respectively. After working at ETRI (Electronics & Telecommunications Research Institute) from 1978 to 1980, he joined Sungkyunkwan University in 1982, where he is currently professor of School of Information and Communication Engineering. His interests are on cryptology and information security. Especially, in the year 2002, he was occupied the president of KIISC (Korea Institute of Information Security & Cryptology).