A Distributed Online Certificate Status Protocol with a Single Public Key

Satoshi Koga[1] and Kouichi Sakurai[2]

[1] Graduate School of Information Science and Electrical Engineering,
Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan
satoshi@itslab.csce.kyushu-u.ac.jp

[2] Faculty of Information Science and Electrical Engineering,
Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka City, 812-8581, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract. The *Public Key Infrastructure* (PKI) technology is very important to support secure global electronic commerce and digital communications on networks. The *Online Certificate Status Protocol* (OCSP) is the standard protocol for retrieving certificate revocation information in PKI. To minimize the damages caused by OCSP responder's private key exposure, a distributed OCSP composed of multiple responders is needed. This paper presents a new distributed OCSP with a single public key by using *key-insulated signature scheme* [6]. In proposed distributed OCSP, each responder has the different private key, but corresponding public key remains fixed, so the client simply obtains and stores one certificate and can verify any responses by using a single public key.

Keywords: Public Key Infrastructure, Certificate Revocation, Online Certificate Status Protocol, Distributed OCSP, Key-Insulated Signature Scheme

1 Introduction

1.1 Background and Motivation

Recently, the Internet has been spread all over the world and it has used to be an infrastructure of electronic commerce. However a lot of threats exist on networks, for example wiretapping, alteration of data, and impersonation. It is important to support secure digital transactions and communications throughout existing networks. Confidentiality, integrity, authentication, and non-repudiation are all security requirements to prevent these threats. These requirements can be supported by a variety of different key management architectures. One of these architectures is a *Public Key Infrastructure* (PKI). A PKI is the basis of security infrastructure whose services are implemented and provided using public key techniques. Most of the protocols for secure e-mail, web service, virtual private networks, and authentication systems make use of PKIs.

In a PKI, a trusted third party called *Certification Authority* (CA) issues a certificate digitally signed by using its private signing key. A certificate is used to bind an entity's identity information with the corresponding public key. Nevertheless, certificates are revoked in case of breaking that binding before its expiration date. If user's private key is compromised or the user's personal information is changed, the user makes a request to the CA for revoking own certificate. The CA is the ultimate owner of certificate status and has the responsibility of publishing to the users that the certificate has been invalid. Thus, users do not simply check the expiration data on the certificate, but also check whether the certificate has been revoked or not. The validation of certificates status information is the current issues in PKI.

A certificate revocation can be implemented in several ways. The most well-known method is to periodically publish a *Certificate Revocation List* (CRL) [9, 7]. A CRL is a digitally signed list of revoked certificates and usually issued by a CA for the certificates it manages. In case of validating user's certificate, the verifier obtains the CRL stored in repository and should verify its validity and CA's digital signature. And the verifier should confirm whether user's certificate is contained in the CRL or not. The main advantage of the CRL systems is its simplicity, however, there are several problems pointed out [1, 23]. Especially, the main disadvantage of the CRL systems is its high communication costs between the user and the repository stored on CRLs. It is said that a certificate revocation rate around 10 percent per year is reasonable [20]. Therefore, the size of CRL will be quite long if the CA has many clients. That is, the validation performance is likely to be slow, since the verifier has to download the CRLs from each CA (or CA's repository) in a certification chain and verify each CRLs. This fact is critical problem if the client is the mobile terminal with restricted processing capacities, memory limitations, and network bandwidth. In order to reduce the size of CRLs, several modifications have been suggested. *Delta CRL* [9] is small CRL that includes information about the certificates that have been revoked since the issuance of a complete revocation list called *Base CRL*. And *CRL Distribution Points* was defined in [9]. CRL Distribution Points allow revocation information within a single domain to be divided into the multiple CRLs.

In order to reduce the communication costs, there are some alternative methods to CRL-based systems. The *Certificate Revocation Tree* (CRT) was proposed by Kocher [12]. CRTs are based on Merkle Hash Trees [14], in which the tree itself represents all certificate revocation information. Naor and Nissim proposed the *Authenticated Directory* [19], which improves the reduction in communication cost by balancing the hash tree. They introduced using a 2-3 tree, in which every node has two or three children. In [10, 11], the binary hash tree is extended to k-ary hash tree in which any node has at most k children. Micali proposed the revocation system using hash chains [15, 16], taking into account both user's and CA's efficiency.

If the client needs very timely information of certificate status, an online certificate status service is required. The standard online revocation system is the *Online Certificate Status Protocol* (OCSP) defined in [18]. The OCSP provide the up-to-date response to certificate status queries and enable to reduce the communication costs in comparison with the CRL, because the client only request to return the status of certificate instead of obtaining the CRLs. The certificate status is returned by a trusted entity referred to as an OCSP responder. The response indicates the status of the certificate returning the value *good*, *revoked*, and *unknown*. Additionally, the OCSP responder signs each response it produces by using its private key. The CRL is published the data on all of revoked certificates, for example those data are issuer's name and its serial number. Since any client can obtain the CRL, this fact will be leading the privacy concerns. On the other hand, the OCSP responder simply returns the status of requested certificate and does not expose information about all revoked certificates. In mobile environment, the method of using the OCSP appears to be a good choice, because the client can retrieve timely certificate's status with a moderate resource usage. As the online protocol that are more extensive than OCSP, several mechanisms that build and validate certification path instead of end users are suggested [13, 22]. This paper only focuses on the certificate status checking mechanism.

In OCSP, the communication costs will be reduced, however, it substantially increases computation costs since a digital signature is a computationally complex operation. Consequently, it becomes highly vulnerable to *denial-of-service* (DoS) attacks, if the responder is centralized [16]. Another threat is the leakage of responder's private key. In case of compromising responder's private key, the attacker can generate the forged response that the revoked certificate is valid. As well as CA's private key, responder's private key exposures affect the serious impact for the client. So the countermeasure against those threats is important to provide the online certificate status service securely.

1.2 Related Work

To reduce the risk of DoS attacks, OCSP responders may pre-produce signed responses specifying the status of certificates at a specified time [18]. However, the use of pre-produced responses allows replay attacks in which an old response is replayed prior to its expiration date but after the certificate has been revoked. To avoid the replay attacks, the responder needs to generate pre-produced responses within a short period of time. But this consumes a lot of processing and this fact causes DoS attacks. In [17], the modification over OCSP using hash chain is suggested to reduce the computational load of the OCSP responder.

As well as CA's private keys, responder's private key must be stored very carefully. There are some approaches to protect the private key from attackers. A Hardware Security Module (HSM) may reduce the risk of key compromise. An attacker requires penetration or theft of the HSM to retrieve responder's private key. To evaluate the security of HSM objectively, the security requirements for cryptographic modules are specified in [21]. Another approach is to manage a share of responder's private key on different servers by using a threshold cryptography [4]. A proactive signature [3] is the enhanced threshold solution by periodic refreshment of shares. These approaches can be effective, but key exposures caused by operation mistakes appear to be unavoidable.

1.3 Our Contributions

As mentioned above, it is difficult to avoid all of threats completely. If the OCSP responder is centralized, the entire system is affected by DoS attacks and compromising responder's private key. That is, the entire service is not available in those cases. Therefore, minimizing damages caused by responder's private key exposure and DoS attacks is extremely important to employ the OCSP system.

A distributed OCSP (D-OCSP) model composed of multiple responders mitigates these damages. In case that each responder has the same private key, compromising any responder compromises the entire system [16]. On the other hand, if each responder has the different private key, compromising a responder cannot affect the others. Hence, this paper examines the D-OCSP that each responder has the different private key. In the general D-OCSP model, the CA issues each responder's certificate. However, the client's load becomes heavy in this model. Every time clients receive the response from the responder, they need to obtain responder's certificate. Moreover, when clients utilize the different responder, they need to get its certificate.

This paper presents a new D-OCSP with a single public key by using *key-insulated signature scheme* (KIS) based on the difficulty of the discrete logarithm problem [6]. The KIS is one of the methods for mitigating the damage caused by private key exposures. Using a KIS-enabled responder, compromise of responder's private key only affects at short time period. We focus on the property that all signatures can be verified by using fixed public key in KIS. This paper takes a different approach from KIS-enabled responder. The multiple private keys are generated using key update algorithm in KIS and assigned to the separate responders, respectively. Thus each responder has the different private key, but corresponding public key remains fixed and the client can verify any responses by using a single public key. Once the client obtained responder's certificate, she simply stores it and can utilize during its validity. Thereby, communication costs are more efficient in comparison with the general model. In our model, the client needs to check the validation of responder's private key as well as the traditional certificate. Our proposed D-OCSP applies the Micali's revocation system [16] and the client checks the validation of responder's private key efficiently than using like the CRL.

The rest of this paper is organized as follows. In Section 2, we explain the traditional D-OCSP, in which the CA issues responder's certificate with a short life-time, and discuss the problems of traditional D-OCSP. In Section 3, we describe the proposed D-OCSP, including the validation of responder's private key and decentralizing processes of responders. Section 4 details the viewpoints of security and performance of our D-OCSP. Concluding remarks are made in Section 5.

2 Distributed OCSP

2.1 Model

In a distributed OCSP (D-OCSP), there are three entities, as shown in Fig1.

1. **Certification Authority (CA)**
   A Certification Authority (CA) is a trusted third party that has the responsibility of publishing the certificate revocation information. Compromise of CA's private key will have disastrous for the entire system, so the CA is isolated form the Internet in order to avoid unauthorized accesses.
2. **Responders**
   A responder is a trusted entity that sends the certificate status information to clients.
3. **Clients**
   Clients trust the CA's public key certificate and request the certificate status information to responders.

In this section, we explain the general D-OCSP model using responder's certificates. If each responder has the same private key, the compromising of any responder compromises the entire system [16]. Thus, we examine the D-OCSP that each responder has the different key-pair (PK_i, SK_i). The CA issues each responder's public key certificate digitally signed by its own private key. As well as the traditional public key certificate, the client needs to check revocation information of responder's certificates. There are some ways of checking those information [7]. The simplest method is to use the CRL issued by CA. Another way is to use the responder's certificate with a short lifetime. Using short-lived certificates, clients don't have to check the validation of responder's certificate. In this way, D-OCSP composed of n-responders is shown in Figure 1.

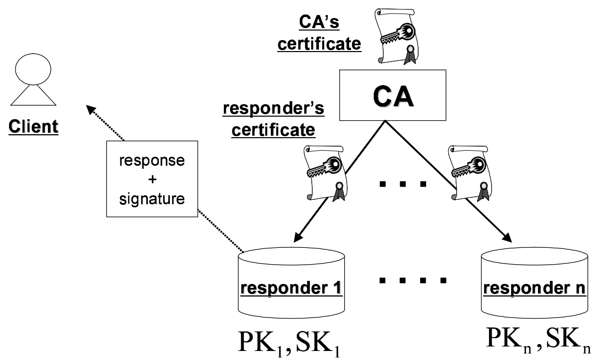

Fig. 1. A Distributed OCSP Model

2.2 Verification Processes

In case that the client receives the response from responder i, she should verify that response as follows.

1. The client obtains the certificate of responder's by online or offline.
2. The client verifies the digital signature contained responder's certificate by using CA's public key.
3. The client verifies the digital signature contained the response by using responder's public key.

(Problems)

1. *Client Efficiency*
   Every time the client receives the response, she should obtain the responder's certificates, since responder's certificate should be updated frequently. Therefore, the communication costs between the client and responders are not efficient. Even if the CA issues the long-lived responder's certificate, the client needs to download the different responder's certificate in case of receiving responses sent by the different responder. So the memory space of the client will be increasing.
2. *CA Efficiency*
   The CA needs to issue responder's certificates frequently. Thereby, the CA needs to produce a digital signature and the computational costs are increasing.

3 Proposed Method

This paper proposes a new D-OCSP with a single public key. In detail, we use a *key-insulated signature scheme* (KIS) [6] and responder's private keys are generated at once. And the client can verify any responses by using a single public key. Before suggesting the decentralization method, we explain the KIS in detail.

A lot of the digital signature schemes have been proposed, but they provide no security guarantees in case of private key exposures. To minimize the damage caused by the leakage of private keys, the notion of key-insulated security was introduced in [5] and a KIS is formalized in [6]. As in a standard signature scheme, the user begins by registering a single public key that remains fixed for the lifetime of the protocol, while the corresponding private key can be changed frequently. A master secret key is stored on physically secure device. The lifetime of the protocol is divided into distinct periods $1, ..., N$. At the beginning of period i, the user interacts with the secure device to derive a temporary private key SK_i. Even if SK_i is exposed, an attacker cannot forge signatures for any other time periods. Moreover, in a strong (t, N)-key-insulated scheme, an attacker cannot forge signature for any of remaining $N - t$ periods even if she obtains the private keys for up to t periods. Using a KIS-enabled responder, responses are signed using responder's private key SK_i at time period i. In that case, the attacker can forge the responses only during period i, if SK_i is compromised. That is, compromise of responder's private key only affects those responses at the point of compromise.

We focus on the property that all signatures can be verified by using fixed public key. This paper takes a different approach from KIS-enabled responder. Suppose the total number of responders is n in our D-OCSP, n private keys are generated using key update algorithm in KIS and assigned to the separate n responders, respectively. Thus each responder has the different private key, but corresponding public key remains fixed. Thus, verifiers can verify responses sent by any responders using a single public key. The details of these processes are described in Section 3.2.

Besides a key-insulated model, alternate approaches have been proposed. The first such example is a *forward-secure signature scheme* (FSS) [2]. This scheme can prevent compromise of private keys at the previous time periods, even if an attacker exposes the current private key. However, once the attacker exposes the current private key, she can easily derive the private keys of the future periods. Like a proposed model, a D-OCSP model using FSS has the advantage that the client can verify any responses using a single public key, but this model cannot minimize the impact caused by compromising responder's private keys. Another approach is a *intrusion-resilient signature scheme* (IRS) proposed in [8]. This scheme adds key-insulation to a proactive refresh capability which may be performed more frequently than key updates. IRS can be tolerant multiple corruptions of both the user and the physically secure device. Any signatures are secure if both of devices are compromised, as long as the compromises are not simultaneous. Compared to FSS and KIS, IRS has a high security. In our method, however, a master secret key stored on physically secure device is only used during private key generations. Thus master key is deleted after that generations are finished. Taking into account the computation costs, this paper examines the decentralizing method of CA using KIS.

3.1 Validation of responder's private key

In this section, we examine the validation method of responder's private key. The client needs to check that a responder's certificate has not been revoked. There are some ways of checking those information [7]. The simplest method is that the client checks the offline verification using like a CRL issued by the CA. While, the CA may choose not to specify any method of revocation checking for responder's certificate. In that case, responder's certificate with a very short lifetime should be issued. In the traditional D-OCSP mentioned in Section 2, the client doesn't have to check the validation of responder's certificate. However, the D-OCSP using responder's certificate with a short lifetime has disadvantages. The first problem is that communication costs are inefficient, since the client should obtain the responder's certificate in case of receiving the response. Moreover, CA's computational costs become high because of updating responder's certificate frequently.

In our model, each responder has the different private key, but corresponding public key remains fixed. As well as the traditional model, if this private key is compromised, this private key needs to be revoked and the CA publishes all user that this private key is invalid. We utilize Micali's revocation system proposed in [16]. Micali's revocation system uses the hash-chain and is efficient as to computational costs. Our model uses a one-way hash function H satisfying the following properties, as well as Micali's system.

(One-way hash function)

1. H is at least 10,000 faster to compute than a digital signature scheme.
2. H produces 20-byte outputs.
3. H is hard to invert, given Y, finding X such that $H(X) = Y$ is practically impossible.

(Issuance of responder's certificate)

1. Let T be the total number of time-periods. For example, T is 365 if each responder's certificate expires 365 days after issuance. The CA produces T hash value using H as follows.

$$X_T \xrightarrow{h} X_{T-1} \xrightarrow{h} X_{T-2} \xrightarrow{h} \cdots \xrightarrow{h} X_1$$

   Let n be the total number of responders. The CA repeatedly produces n hash-chain as different input value $X_{T,i}$. $X_{t,i}$ denotes the hash value at time priod t for validation of responder j. These hash values are stored on the CA.

$$\begin{array}{c} X_{T,1} \xrightarrow{h} X_{T-1,1} \xrightarrow{h} X_{T-2,1} \xrightarrow{h} \cdots \xrightarrow{h} X_{1,1} \\ X_{T,2} \xrightarrow{h} X_{T-1,2} \xrightarrow{h} X_{T-2,2} \xrightarrow{h} \cdots \xrightarrow{h} X_{1,2} \\ \vdots \\ X_{T,n} \xrightarrow{h} X_{T-1,n} \xrightarrow{h} X_{T-2,n} \xrightarrow{h} \cdots \xrightarrow{h} X_{1,n} \end{array}$$

2. The CA issues responder's certificate C_{res} by using own private key. SN is the serial number of certificate and V represents the validity period. I and S denote issuer and subject of certificate, respectively.

$$C_{res} = Sig_{SK_{CA}}(PK_{res}, SN, I, S, V, X_{1,1}, ..., X_{1,n})$$

(Validation of responder's private key)

1. The CA delivers the hash value $X_{t,i}$ to responder i, if responder i's private key SK_i is valid at t period.

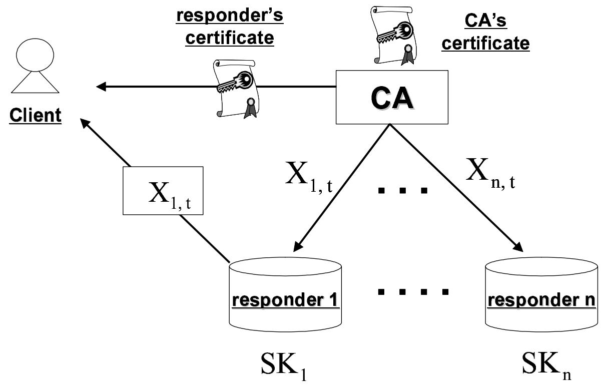

Fig. 2. Proposed D-OCSP

2. When responder i returns the response to the client at period t, she also delivers the hash value $X_{t,i}$ to the client.

$$\langle i, t, X_{t,i}, R, Sig_{SK_i}(R) \rangle$$

3. When the client receives the response by responder i, she verifies the digital signature by using responder's public key PK_{res}. Then the client can check the validation of responder's private key using hash value $X_{t,i}$ and $X_{1,i}$ contained responder's certificate. In detail, the client checks the following equation. If that equation is satisfied, the client can certify that SK_i is valid.

$$X_{1,i} = H^{t-1}(X_{t,i})$$

In this way, the client can verify the validation of the responder's private key. The responder's certificate is not revoked during its validity unless all of responder's private keys are revoked.

3.2 Decentralizing method of responder

We describe the decentralizing process using KIS based on the difficulty of discrete logarithm problem [6]. Let $R_1, ..., R_n$ be responders in our model. Using the following processes, a D-OCSP composed of $R_1, ..., R_n$ is constructed (Figure2).

Step1: generation of responder's private keys

1. *Key pair generation*
   Let p and q be prime numbers such that $p = 2q+1$ and let g, h be a element of order q in the group $\mathbb{Z}_p$. A responder's public key PK_{res} is generated by choosing $x, y \in_R \mathbb{Z}_q$ and setting $v = g^x h^y$. SK^* denotes the master key to be used generating of responder's private keys. During the generation processes, SK^* is stored on the CA.

$$x_0^*, y_0^*, ..., x_t^*, y_t^* \leftarrow \mathbb{Z}_q$$
$$v_i^* = g^{x_i^*} h^{y_i^*}$$
$$SK^* = (x_1^*, y_1^*, ..., x_t^*, y_t^*)$$
$$PK_{res} = (g, h, v_0^*, ..., v_t^*)$$

2. *Responder's private key generation*
A partial key SK_i' is generated as follows. SK_i' is used to derive R_i's private key.

$$x_i' = \Sigma_{k=1}^t x_k^* (i^k - (i-1)^k)$$
$$y_i' = \Sigma_{k=1}^t y_k^* (i^k - (i-1)^k)$$
$$SK_i' = (x_i', y_i')$$

By using partial keys derived above, n private keys are generated. Once all private keys is derived, SK_i' and SK^* are deleted.

$$x_i = x_{i-1} + x_i'$$
$$y_i = y_{i-1} + y_i'$$
$$SK_i = (x_i, y_i)$$

The CA delivers the private key SK_i to R_i securely. Thus, each responder has the different private key.

3. *Issuance of responder's certificate*
As mentioned section 3.1, the CA issues the responder's certificate C_{res} as follows.

$$C_{res} = Sig_{SK_{CA}}(PK_{res}, SN, I, S, V, X_0^1, ..., X_0^n)$$

Step2: Signature and verification algorithm

1. *Signature algorithm*
When R_i returns the response M to the client, she generates a digital signature $\langle i, (w, a, b) \rangle$ by using SK_i as follows.

$$r_1, r_2 \leftarrow \mathbb{Z}_q$$
$$w = g^{r_1} h^{r_2}$$
$$\tau = H(i, M, w)$$
$$a = r_1 - \tau x_i$$
$$b = r_2 - \tau y_i$$

2. *Verification algorithm*
The client can verify R_i's signature by using PK_{res} as follows.

$$v_i = \prod_{k=0}^t (v_i^*)^{i^k}$$
$$\tau = H(i, M, w)$$
$$w = g^a h^b v_i^\tau$$

4 Evaluations

1. *Security*
   Suppose that an attacker steals R_i's private key SK_i and hash value X_t^i at time period t. In this case, she cannot derive any other responder's private keys unless she obtain SK^* (SK^* is deleted after generating responder's private keys). And if an attacker can get the hash value X_t^i, she cannot derive the hash value $X_{t+1,i}(H(X_{t+1,i}) = X_{t,i})$ because H is a one-way function. Therefore, an attacker cannot cheat that SK_i is valid after period $t+1$ and our model can minimize the damage caused by responder's private key exposures.
2. *Communication costs*
   In the traditional D-OCSP, the client should get the responder's certificate in case of receiving the response from the responder. On the other hand, our model can mitigate the communication costs, because the responder's certificate is only one. The client stores responder's certificate and need not to obtain it by online or offline during the certificate's validity.
3. *Validation of responders*
   In our model, validation of responder's private key is performed by using hash-chain, without using CRL. As mentioned above, hash computation is much faster than digital signature computations. In case of checking the status of responder's certificate, the client just computes t-times hash computations.
4. *CA Efficiency*
   The CA should store the hash value securely. The total size of those value amounts $20nT$-bytes. However, the CA does not have to store all hash values and only store $X_{1,i}$ ($20n$-bytes), since hash computations is very fast. At period t, $X_{t,i}$ is derived by $T-t$ times hash computations. In the traditional D-OCSP, the CA should issue the responder's certificate with a short lifetime. In our model, the CA can issue long-lived responder's certificate, because the client can validate the responder's private key. Thus our model is more efficient than traditional model.

Table 1 shows the comparison between our model using KIS and the traditional D-OCSP using DSA. As the comparison items, we consider the total size of responses, the verification cost of the client (validation of responder's certificate and verification cost), and signing cost of responder. Let $size(C_{res})$ be the size of responder's certificate. (For example, the size of traditional public key certificate is about 800-byte.) Let q, t be the parameter of digital signature scheme. We consider that $q = 160$ and $t \approx n$. The computational cost is represented as the number of multiplications over $\mathbb{Z}_p$ or $\mathbb{Z}_q$. Let $\mathrm{EX}_{\mathbb{Z}_p}$ be the number of multiplications required to compute an exponentiation. In our method, computational cost is less efficient than traditional D-OCSP, but the client may verify any responses by using a single public key. Additionally, the client just obtains the responder's certificate at a time.

Table 1. Comparison between traditional D-OCSP and our D-OCSP

	Traditional (DSA)	Proposal (KIS)
size of resposes	$2q + size(C_{res})$	$3q + 160$
validation of certificate	nothing	t-hash computaions
signing verification cost	3+2EX$_{Z_p}|q|$	$t + 2$+3EX$_{Z_p}|q|$
signing cost	2+EX$_{Z_p}|q|$	2+2EX$_{Z_p}|q|$

5 Conclusions

In order to minimize the damage caused by responder's private key exposure and DoS attacks, the distributed OCSP model composed of the multiple responders is required in real world. This paper suggests the new distributed OCSP model using key-insulated signature scheme. In our model, the client needs to check the validation of responder's private key as well as the traditional certificate. Our proposed D-OCSP applies Micali's revocation system and the client check the validation of responder's private key efficiently than using like the CRL. In mobile environment, the client has the restricted processing capacity as well as the bandwidth. So our future work is to reduce the computation costs of clients.

References

1. A. Arnes, M. Just, S. J. Knapskog, S. Lloyd, and H. Meijer, *Selecting Revocation Solutions for PKI*, 5th Nordic Workshop on Secure IT Systems (NORDSEC 2000), 2000.
   `http://www.pvv.ntnu.no/ andrearn/certrev/`
2. M. Bellare, and S. K. Miner, *A Forward-Secure Digital Signature Scheme*, Advances in Cryptology - CRYPTO '99, LNCS 1666, pp.431-448, Springer-Verlag, 1999.
3. R. Canetti, R. Gennaro, A. Herzberg, and D. Naor, *Proactive Security: Long-term protection against break-ins*, RSA CryptoBytes, Volume 3, No. 1, 1997.
   `http://www.rsasecurity.com/rsalabs/cryptobytes/`
4. Y. Desmedt and Y. Frankel, *Threshold Cryptosystems*, Advances in Cryptology - CRYPTO '89, LNCS 435, pp.307-315, Springer-Verlag, 1990.
5. Y. Dodis, J. Katz, S. Xu and M. Yung, *Key-Insulated Public Key Cryptosystems*, Advances in Cryptology - EUROCRYPT 2002, LNCS 2332, pp.65-82, Springer-Verlag, 2002.
6. Y. Dodis, J. Katz, S. Xu, and M. Yung, *Strong Key-Insulated Signature Schemes*, Public Key Cryptography - PKC 2003, LNCS 2567, pp.130-144, Springer-Verlag, 2003.
7. R. Housley, W. Polk, W. Ford, and D. Solo, *Certificate and Certificate Revocation List (CRL) Profile*, IETF RFC3280, 2002.
   `http://www.ietf.org/rfc/rfc3280.txt`
8. G. Itkis, and L. Reyzin, *SiBIR: Signer-Base Intrusion-Resilient Signatures*CAdvances in Cryptology - CRYPTO 2002, LNCS 2442, pp.499-514, Springer-Verlag, 2002.
9. ITU/ISO Recommendation. X.509 Information Technology Open Sysytems Interconnection - The Directory: Authentication Frameworks, 2000.

10. H. Kikuchi, K. Abe, and S. Nakanishi, *Performance Evaluation of Certificate Revocation Using k-Valued Hash Tree*, 2nd International Workshop on Information Security (ISW '99), LNCS 1729, pp.103-117, Springer-Verlag, 1999.
11. H. Kikuchi, K. Abe, and S. Nakanishi, *Certificate Revocation Protocol Using k-Ary Hash Tree*, IEICE TRANS. COMMUN., Vol. E84-B, No.8, 2001.
12. P. C. Kocher, *On Certificate Revocation and Validation*, Financial Cryptography (FC '98), LNCS 1465, pp.172-177, Springer-Verlag, 1998.
13. A. Malpani, R. Housley, and T. Freeman, *Simple Certificate Validation Protocol*CIETF Internet-Draft, 2003.
`http://www.ietf.org/internet-drafts/draft-ietf-pkix-scvp-13.txt`
14. R. C. Merkle, *A Certified Digital Signature*, Advances in Cryptology - CRYPTO '89, LNCS 435, pp.218-238, Springer-Verlag, 1990.
15. S. Micali, *Efficient Certificate Revocation*, Technical Memo MIT/LCS/TM-542b, Massachusetts Institute of Technology, 1996.
16. S. Micali, *NOVOMODO; Scalable Certificate Validation And Simplified PKI Management*, 1st Annual PKI Research Workshop, pp.15-25, 2002.
`http://www.cs.dartmouth.edu/ pki02/`
17. J. L. Munoz, J. Forne, O. Esparza, I. Bernable, and M. Soriano, *Using OCSP to Secure Certificate-Using Transactions in M-commerce*, Applied Cryptography and Network Security (ACNS 2003), LNCS 2846, pp.280-292, Springer-Verlag, 2003.
18. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, *X.509 Internet Publik Key Infrastructure Online Certificate Status Protocol-OCSP*, IETF RFC2560, 1999.
`http://www.ietf.org/rfc/rfc2560.txt`
19. M. Naor, and K. Nissim, *Certificate Revocation and Certificate Update*, 7th USENIX Security Symposium, pp.217-228, 1998.
`http://www.usenix.org/publications/library/proceedings/usenix98/`
20. A. Nash, W. Duane, C. Joseph, and D. Brink, *PKI - Implementing and Managing E-Security*, Osborne Media Group, 2001.
21. National Institute of Standards and Technology (NIST), *Security Requirements for Cryptographic Modules*, FIPS 140-2, 2001.
`http://csrc.nist.gov/publications/fips/`
22. D. Pinkas, and R. Housley, *Delegated Path Validation and Delegated Path Discovery Protocol Requirements*, RFC3379, 2002.
`http://www.ietf.org/rfc/rfc3379.txt`
23. R. L. Rivest, *Can We Eliminate Certificate Revocation Lists ?*, Financial Cryptography (FC '98), LNCS 1465, pp.178-183, Springer-Verlag, 1998.