

Enhancing Security of Security-Mediated PKI by One-time ID

Satoshi Koga¹, Kenji Imamoto¹, and Kouichi Sakurai²

¹ Graduate School of Information Science and Electrical Engineering,
Kyushu University

6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan
{satoshi,imamoto}@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 Security Mediator (SEM) approach was proposed by Boneh et al. at USENIX Security Symposium in 2001. However, their Security-Mediated PKI has a drawback that it is vulnerable to Denial-of-Service (DoS) attacks, since an attacker can send a lot of requests to the SEM server. To prevent DoS attacks, some solutions are proposed. In this paper, we show that the use of Message Authentication Code (MAC), which is one of the solution proposed, cannot avoid DoS attacks because an attacker can reuse the request for replay attacks. In order to prevent DoS attacks efficiently, this paper proposes Security-Mediated PKI using one-time ID. Our proposed system can avoid DoS attacks and protect the privacy of signers, since one-time ID can be used only once.

Keywords: Public Key Infrastructure, Certificate Revocation, Security Mediator, One-time ID

1 Introduction

1.1 Background

A Public Key Infrastructure (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 the PKI. In the PKI, a certificate is used to bind an entity's identity information with the corresponding public key. When a certificate is issued, its validity is limited by a pre-defined expiration time. Nevertheless, certificates are revoked in case of breaking that binding before its expiration date. Thus, the certificate verifier must check not only the expiration date on the certificate but also the revocation information of it.

A certificate revocation system can be implemented in several ways. The most well-known method is to periodically publish a Certificate Revocation List (CRL)

[6, 8], which is a digitally signed list of revoked certificates and usually issued by the Certification Authority (CA). One of the shortcomings of CRL systems is that the time granularity of revocation is constrained by CRL issuance periods. It is necessary to obtain timely information regarding the revocation status of a certificate. The most popular mechanism that provides real-time status of a certificate is the Online Certificate Status Protocol (OCSP) [13]. The OCSP provides the up-to-date response to certificate status queries. Since the user just requests to return the status of certificate, the communication costs can be reduced in comparison to the CRL.

Recently, Boneh et al. observed that existing revocation techniques, including OCSP, don't provide immediate revocation [3]. Supporting immediate revocation with existing revocation techniques would result in heavy performance cost and very poor scalability. To provide immediate revocation, SEcurity Mediator (SEM) approach was proposed [2, 3]. The basic idea of SEM is as follows. They introduce a new entity, referred to as a SEM: an online semi-trusted server. To sign or decrypt a message, a client must first obtain a message-specific token from its SEM. Without this token, the user cannot accomplish the intended task. To revoke the user's ability to sign or decrypt, the security administrator instructs the SEM to stop issuing tokens for that user's future request. The SEM approach provides several advantages. This approach can eliminate the need for CRLs since private-key operations cannot occur after revocation. That is, this approach enables immediate revocation. Recently, Vanrenen et al. proposed a distributed SEM architecture [14], and Bicakci et al. proposed Privilege Management Infrastructure based on SEM approach [1].

1.2 Motivation

As Boneh et al. pointed out in [3], one of the drawbacks of SEM approach is that it is vulnerable to Denial-of-Service (DoS) attacks. The SEM server must generate the token for every user's request. That is, for every simple request sent by an attacker, the SEM server must generate the token, which is a computationally intensive operation. Consequently, it becomes highly vulnerable to DoS attacks. The goal of this paper is to prevent DoS attacks.

1.3 Related Work

To prevent DoS attacks, there are some solutions as follows [3].

1. Digital signatures

The simple solution is to authenticate the user by verifying digital signature [3]. For example, a user can generate a partial signature on each request message as follows. (See Section 2.3 for notations.)

$$\langle EC(m), EC(m)^{d_u} \pmod n \rangle$$

The SEM computes a digital signature $S(m) = (PS_{sem} * PS_u) \pmod n$ and verifies it by using public key. Although this method does not prevent DoS

attacks, since a SEM would need to perform two modular exponentiations to authenticate each request [3].

2. Secure Sockets Layer (SSL)

Another solution is to rely on more general encapsulation techniques, such as SSL, to provide a secure channel for communications between SEMs and users [3]. However, the user must validate the SEM's certificate in order to establish secure channel. Therefore, the burden of a user becomes heavy since the certificate validation processing is intensive operations.

3. Message Authentication Code (MAC)

More cost-effective approach is to use MAC or keyed hash [3]. This situation requires a shared secret key between a SEM and each user. This paper shows that this approach cannot prevent DoS attacks. An attacker can reuse requests generated by legal users, and send a lot of requests to a SEM. Even if the SEM authenticates requests by checking MAC, it is vulnerable to DoS attacks based on replay attacks, as discussed in Section 3.

4. The DoS resistant protocol

To avoid DoS attacks from any malicious stranger, it is important to make him have a large burden in sending a lot of requests. One approach to solve the DoS problem is to make the client compute some form of proof of computational effort [5, 12]. This approach is effective to DoS attacks, however computational costs of the legal user are also increasing as well as DoS attackers. In our proposed method, only attackers require exhaustive computations.

1.4 Our Contributions

This paper shows that traditional solutions cannot prevent DoS attacks. An attacker can reuse requests generated by legal users, and send a lot of requests to a SEM. Even if the SEM authenticates requests by checking MAC, it is vulnerable to DoS attacks based on replay attacks.

It is necessary to avoid DoS attacks based on replay attacks efficiently. This paper proposes the SEM approach using One-time ID. One-time ID is a user's extraordinary identity, which has two properties as follows: (1) an attacker cannot specify who is communicating even when she eavesdrops on one-time ID, and (2) One-time ID can be used only once. Our proposed method requires a shared secret key between a SEM and each user. A user sends a request containing a message and one-time ID. One-time ID can be derived by shared secret key and be used only once. By checking one-time ID, a SEM server can confirm that requesting user is legal user. In our proposed method, the user sends a request containing Message Authentication Code (MAC). Therefore, an attacker cannot create the legal request. Moreover, an attacker cannot reuse requests for replay attacks because the One-time ID is used only once. Our proposed system has the following benefits.

1. DoS-resilient

A SEM can efficiently detect illegal requests, since an attacker cannot generate one-time ID unless she obtains a shared secret key. Moreover, our

proposed system can prevent replay attacks. An attacker cannot reuse the request generated by legal user because one-time ID can be used only once.

2. Efficiency

One-time ID can be generated by hash computations. Therefore, computational costs of a SEM and a user are more efficient, compared with signature-based solutions. Additionally, communications between a SEM and a user are the same as the existing Security-Mediated PKI.

3. Privacy

In existing approaches, a user must send a request contained a public key certificate. Thus, an attacker can trace the user's identity by tracking user's request. It is easy to know the user's private information (e.g. user's name, affiliation, address and so on.) from the serial number of his certificate. In our proposed system, an attacker cannot trace user's identity even if she eavesdrops on one-time ID, because one-time ID dynamically changes. That is, our proposed system can protect the privacy of signers, compared with existing approaches.

The rest of this paper is organized as follows. In Section 2, we explain the SEM approach. In Section 3, we show that the traditional approach cannot prevent DoS attacks. In Section 4, we describe our proposed system and Section 5 discusses as to security of our proposed system. Concluding remarks are made in Section 6.

2 Security-Mediated PKI

2.1 Model

In a Security-Mediated PKI, there are three entities, as shown in Fig.1.

1. Certification Authority (CA)

A Certification Authority (CA) is a trusted third party that issues certificates. Compromise of CA's private key will affect the entire system, so the CA is isolated from the Internet to prevent unauthorized accesses.

2. SEM (SEcurity Mediator)

A SEM is an online semi-trusted server. A single SEM serves many users.

3. User

Users trust the CA and SEM. To sign or decrypt a message, they must first obtain a message-specific token from its SEM.

2.2 An overview of a SEM

The basic idea of SEM approach is as follows [2]. We introduce a new entity, referred to as a SEM. A SEM is an online trusted server. To sign or decrypt a message, Alice must first obtain a message-specific token from the SEM. Without this token Alice cannot use her private key. To revoke Alice's ability to sign or decrypt, the security administrator instructs the SEM server to stop issuing

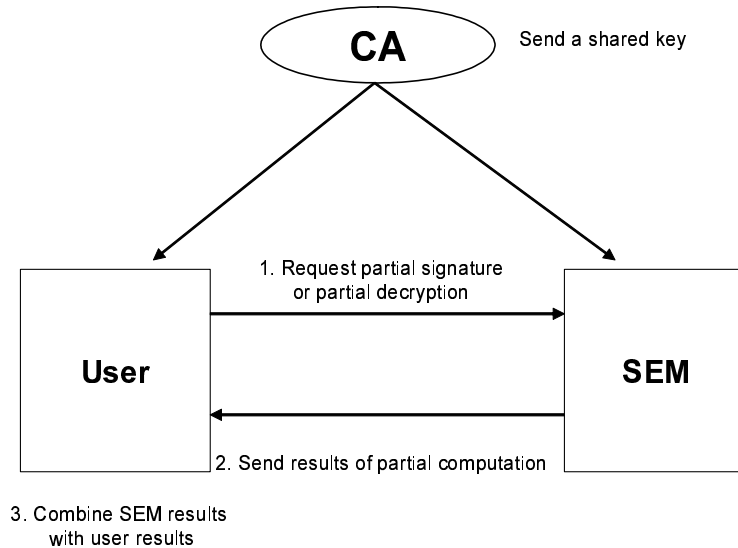


Fig. 1. A general SEM algorithm

tokens for Alice’s public key. At that instant, Alice’s signature capabilities are revoked. Fig. 1 shows the general SEM algorithm.

A SEM approach is to enable immediate revocation of user’s key. This method provides two additional benefits over standard revocation techniques: (1) simplified signature validation, and (2) enabling revocation in legacy systems. The SEM approach naturally provides the following semantics for digital signatures:

Binding Signature Semantics: a digital signature is considered valid if the public key certificate associated with the corresponding private key used to generate the signature was valid at the time the signature was issued.

2.3 Mediated RSA

This section describes in detail how a SEM interacts with users to generate tokens. The SEM architecture is based on a variant of RSA called Mediated RSA (mRSA). The main idea is to split each RSA private key into two parts using simple 2-out-of-2 threshold RSA [4].

Each user U has a unique public key and private key. The public key PK includes n and e , where the former is a product of two large distinct primes (p, q) and e is an integer relatively prime to $\phi(n) = (p - 1)(q - 1)$. There is also a corresponding RSA private key $SK = (n, d)$ where $d * e = 1 \pmod{\phi(n)}$. However, as mentioned above, no one party has possession of d . Instead, d is effectively split into two parts: d_u and d_{sem} which are secretly held by the user

and the SEM, respectively. The relationship among them is:

$$d = d_{sem} + d_u \pmod{\phi(n)}$$

The CA generates a distinct set: $\{p, q, e, d, d_{sem}, d_u\}$ for the user. The first four values are generated as in standard RSA. The fifth value, d_{sem} , is a random integer in the interval $[1, n]$, where $n = pq$. The last value is set as: $d_u = d - d_{sem} \pmod{\phi(n)}$. The protocol of key generation algorithm is as follows. Let k be a security parameter. After CA computes, d_{sem} is securely communicated to the SEM and SK is communicated to the user.

(Algorithm: key generation)

- (1) Generate random $k/2$ -bit primes: p, q
- (2) $n \leftarrow pq$
- (3) $e \xleftarrow{r} Z_{\phi(n)}^*$
- (4) $d \leftarrow 1/e \pmod{\phi(n)}$
- (5) $d_{sem} \xleftarrow{r} 1, \dots, n - 1$
- (6) $d_u \leftarrow (d - d_{sem}) \pmod{\phi(n)}$
- (7) $SK \leftarrow (n, d_u)$
- (8) $PK \leftarrow (n, e)$

2.4 mRSA signatures

According to PKCS #1v2.1 [11], RSA signature generation is composed of two steps: message encoding and cryptographic primitive computation. We denote by $EC()$ the encoding function. This encoding includes hashing the input message m using a collision resistant hash function. $EC()$ is the EMSA-PKCS1-v1.5 encoding function, recommended in [11].

1. The user sends the message m to the SEM.
2. The SEM checks the user's certificate. Only if this certificate is valid, the SEM computes a partial signature $PS_{sem} = EC(m)^{d_{sem}} \pmod{n}$, and replies with it to the user.
3. The user computes $PS_u = EC(m)^{d_u} \pmod{n}$. Then, the user receives PS_{sem} and computes $S(m) = (PS_{sem} * PS_u) \pmod{n}$. It then verifies $S(m)$ as a standard RSA signature. If the signature is valid, the user outputs it.

3 Disadvantages of a SEM approach

One of the drawbacks of SEM approach is that it is vulnerable to Denial-of-Service (DoS) attacks, as pointed in [3]. There are three types of DoS attack:

against SEM’s bandwidth, memory, and CPU. The purpose of the first attack is that a SEM cannot receive any more messages. The second one is performed to make a SEM store large quantities of waste states. The last one is the attack which makes a SEM compute a lot of quite inefficient processings. In SEM approach, we focus on DoS attacks against SEM’s CPU.

The SEM server must generate the partial signature for every user’s request. If an attacker can send a lot of requests, the SEM must compute a lot of partial signatures. Computations of a partial signature require a modular exponentiation. Consequently, it becomes highly vulnerable to DoS attacks.

To prevent DoS attacks, the SEM authenticates incoming requests. Only if a legal user sends a request, the SEM computes a partial signature. The SEM does not respond any request, sent by a party whom the responder cannot specify. Additionally, the SEM should confirm that the request is fresh. If an attacker can eavesdrop on communications between a legal user and a SEM, she can reuse a request created by legal users for replay attacks. That is, an attacker can send a lot of legitimate requests to the SEM. To prevent DoS attacks based on replay attacks, the SEM only responds to new requests.

In [3], several solutions are proposed. However, these solutions cannot prevent DoS attacks efficiently. Most cost-effective approach is to use Message Authentication Code (MAC) or keyed hash [3]. This situation requires a shared secret key k between a SEM and each user. A SEM can authenticate the request by checking MAC. However, it is vulnerable to replay attacks. An attacker can reuse requests generated by legal users, and send a lot of requests to a SEM. Suppose that the legal user sends requests $\langle MAC_k(m_1), m_1 \rangle, \langle MAC_k(m_2), m_2 \rangle, \dots, \langle m_n, MAC_k(m_n) \rangle$, as shown in Fig. 2. $MAC_k(m_i)$ denotes the MAC using shared key k as the input message m_i . An attacker can eavesdrop these requests and send them to the SEM. This attack is called as replay attack. The SEM misunderstands that these request sent by an attacker are legal requests, and computes partial signatures, since the SEM cannot detect replay attacks.

4 Our proposed method

In MAC approach proposed in [3], the SEM cannot detect replay attacks. Suppose that an attacker, who doesn’t know secret value, can eavesdrop and modify the data between the SEM server and the legal user. Under this environment, the goal of this paper is to prevent DoS attacks efficiently, and to protect the privacy of signers. This paper proposes Security-Mediated PKI using one-time ID.

4.1 One-time ID

To prevent leakage of user’s identity and DoS attacks, Krawczyk proposed “One-time ID” [9]. One-time ID is a user’s extraordinary identity, which has two properties as follows: (1) an attacker cannot specify who is communicating even when she eavesdrops on one-time ID, and (2) One-time ID can be used only

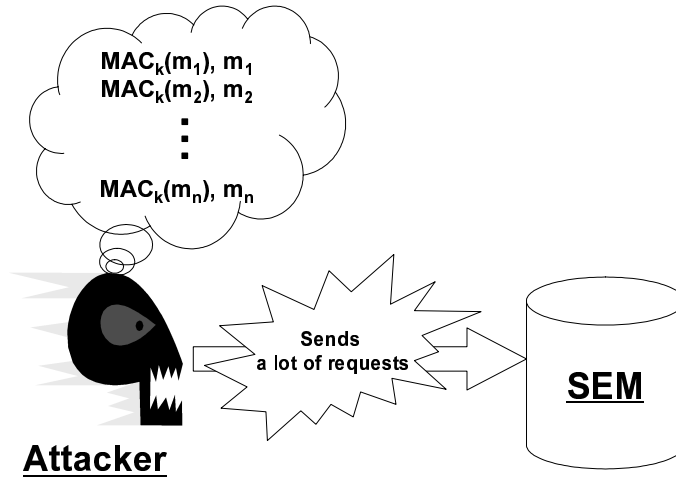


Fig. 2. DoS attack based on replay attack

once. To realize perfect forward secrecy, Imamoto et al. proposed new method of One-time ID calculation [7].

Taking into account the computational cost of users and a SEM, we utilize one-time ID protocol with small computational complexity, like a P-SIGMA. When one-time ID is generated, this method does not require modular exponentiations. The user and a SEM can compute one-time ID using one-way hash function. The SEM can authenticate incoming request by checking one-time ID. Additionally, the SEM can confirm that sending request is fresh because one-time ID is used only once.

Proposed system requires a shared secret key between a SEM and each user. The key-pair of a user is generated by CA, as well as the traditional SEM approach. And the CA generates a shared secret key between a SEM and a user, and sends it to both a SEM and user securely.

We describe the protocol of our system as follows. A user sends a request containing a message and one-time ID. One-time ID can be derived by shared secret key and be used only once. By checking one-time ID, a SEM server can confirm that requesting user is legal user. Only if requesting user is legal, a SEM generates a partial signature.

Our method has the following benefits.

1. DoS-resilient

A SEM can efficiently detect the attacker's request, since an attacker cannot generate one-time ID unless she obtains a shared secret key. Moreover, proposed system can prevent replay attacks. An attacker cannot reuse the request generated by legal user because one-time ID can be used only once.

2. Efficiency

One-time ID can be generated by hash computations. Therefore, computa-

tion costs of a SEM and a user are more efficient, compared with traditional methods. Additionally, communications between a SEM and a user are the same as the existing Security-Mediated PKI.

3. Privacy

In existing approaches, a user must send a request contained a public key certificate. Thus, an attacker can trace the user's identity (e.g. public key certificate) by tracking user's request. In our proposed system, an attacker cannot trace user's identity even if she eavesdrops on one-time ID, because one-time ID is used only once. That is, our proposed system can protect the privacy of signers, compared with existing approaches.

4.2 Preliminaries

(Notations)

- U : user.
- PK : a public key of U .
- SK : a partial private key of U .
- d_{sem} : a partial private key of a SEM.
- k : a shared secret key between U and a SEM.
- $MAC_k()$: Message Authentication Codes using shared key k , such as HMAC [10].
- $H()$: collision-resistant hash function.
- OID_i : U 's one-time ID with i -th session.
- m : message.

(Assumptions)

1. There is a shared secret key between the user and the SEM.
2. The secure channels are established between the CA and users, the CA and the SEM. This is the same assumption as the existing SEM approach [2, 3].
3. Communication between the SEM and users does not have to be protected. An attacker can eavesdrop and modify the contents of request message and response message.

4.3 SEM using one-time ID

(Setup)

1. As shown in Section 2.3, the CA generates PK, SK, d_{sem} , respectively. Then the CA generates k and issues the public key certificate of U . $\langle k, d_{sem} \rangle$ are securely communicated to the SEM and $\langle k, SK \rangle$ is communicated to U .

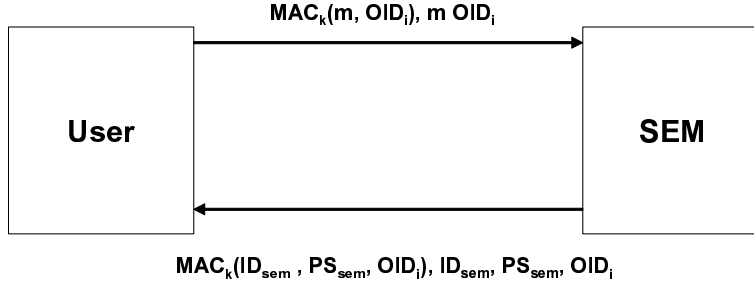


Fig. 3. Our proposed method

2. At first session, a SEM generates the list of OID_1 and U 's certificate. OID_1 is derived as follow.

$$OID_1 = H(1, k)$$

(Signature)

1. U makes a request using one-time ID. First, U generates OID_1 and sends the following request.

$$\langle MAC_k(m, OID_1), m, OID_1 \rangle$$

2. First, the SEM confirms that the contents of a request is fresh by using k . That is, the SEM check the MAC value $MAC_k(m, OID_1)$. Then, the SEM checks who U is by using OID_1 . If U is a legal user, the SEM validates U 's certificate.
3. If U 's certificate is valid, the SEM generates a partial signature $PS_{sem} = EC(m)^{d_{sem}} \bmod n$ and sends the following response. Let ID_{sem} denote SEM's ID.

$$\langle MAC_k(ID_{sem}, PS_{SEM}, OID_1), ID_{sem}, PS_{sem}, OID_1 \rangle$$

4. U computes $PS_u = EC(m)^{d_u} \bmod n$. Then U receives above response and confirms that contents of a response is fresh by checking $MAC_k(PS_{SEM}, OID_1)$. And U computes $S(m) = PS_{sem}^m * PS_u \bmod n$. It then verifies $S(m)$ as a standard RSA signature. If the signature is valid, outputs it.

(Update of one-time ID)

1. SEM's update
In session i , the SEM stores two one-time IDs $\langle OID_{i-1}, OID_i \rangle$ for U . If

one-time ID sent by U is correct, the SEM computes the partial signature and updates one-time ID as follows.

$$OID_{i+1} = H(k, OID_i)$$

And the SEM OID_{i-1} is deleted. If connection error is occurred, U does not receive the response. In that case, U does not update one-time ID. So, the SEM stores OID_i for authenticating U .

2. U 's update

If $S(m)$ is valid signature, U updates one-time ID as follows.

$$OID_{i+1} = H(k, OID_i)$$

If some error is occurred, U sends the same request once again. The SEM sends the same response. It is notice that the SEM doesn't have to compute a partial signature again. The SEM only stores the same response.

5 Discussions

1. Signature forgery

As mentioned in [2, 3], the user cannot generate signatures after being revoked. And the SEM cannot generate signatures on behalf of the user.

However, an attacker can collude with the malicious user. If an attacker compromises a SEM and learns d_{sem} , he can create a signature by colluding with user. The existing SEM approach has the same problem.

2. DoS attacks

After authenticating requests, a SEM validates a user's certificate and generates a partial signature. Suppose that an attacker can send a lot of requests to a SEM. An attacker cannot generate a legal request unless he can obtain a k . Our proposed system can prevent DoS attacks, since the SEM can detect illegal requests.

3. Replay attacks

An attacker can reuse requests generated by legal users. Since one-time ID is changed for each session, an attacker cannot reuse a request for replay attacks.

4. Man-in-the-middle attack

An attacker can modify the contents of request message. Suppose that U sends $\langle OID_i, m \rangle$. An attacker modifies $\langle OID_i, m' \rangle (m \neq m')$. To prevent this attack, U sends $\langle MAC_k(m, OID_i), m, OID_i \rangle$. Unless an attacker obtains k , an attacker cannot generate the legal request. Thus, our proposed system can prevent man-in-the-middle attacks.

5. Privacy

In existing approaches, a user must send a request contained a public key certificate. Thus, an attacker can trace the user's identity (e.g. public key certificate) by tracking user's request. Even if the contents of the request are encrypted with k , an attacker can trace the user's identity. This fact will be

leading the privacy concerns. The privacy of signer can be protected by using SSL, but the burden of both users and the SEM may be heavy to establish the secure channel.

In our proposed system, the user doesn't have to send a request containing user's certificate, since the SEM can specify the user by checking one-time ID. Additionally, an attacker cannot trace user's identity even if she eavesdrops on one-time ID, because one-time ID is used only once. That is, our proposed system can protect the privacy of signers from any eavesdropper, compared with existing approaches.

5.1 Analysis of SEM's computational costs

In this section, we analyze the computational costs of the SEM. N_{DoS} denotes the number of illegal requests sent by an attacker. N_u denotes the number of legal requests sent by legal users. P and H mean the modular exponentiation and hash computation, respectively. In the traditional SEM approach [2], the computational cost of the SEM is $P(N_{DoS} + N_u)$. On the other hand, in our proposed system, the computational cost of the SEM is $H(N_{DoS} + N_u) + PN_u$.

We evaluate the number of hash computations. It is assumed that $P = 1000H$, since H is at least 1,000 times faster to compute than P . In the traditional SEM, the number of hash computations is $1000N_{DoS} + 1000N_u$. In our proposed SEM, the number of hash computations is $N_{DoS} + 1001N_u$. In our proposed system, the computational costs of the SEM are more efficient than those of traditional SEM.

5.2 Other solutions

There are some solutions to avoid DoS attacks based on replay attacks. This paper describes these methods in detail. However, these approaches cannot protect the privacy of signer.

(Challenge and response)

First, U sends a request message req and identifier of the user ID_u to the SEM. Then the SEM responds a random number referred to as a *Nonce*. After receiving a *Nonce*, U generates a MAC using shared key k and the *Nonce* sent by the SEM.

1. $U \rightarrow SEM$: req, ID_u
2. $SEM \rightarrow U$: $Nonce$
3. $U \rightarrow SEM$: $MAC_k(m, Nonce), m, Nonce$
4. $SEM \rightarrow U$: $MAC_k(PS_{sem}, Nonce), PS_{sem}, Nonce$

- Security
The SEM can confirm that the request is fresh by checking *Nonce*. The SEM must store the *Nonce*. It is possible to DoS attack against SEM's memory.
- Efficiency
The number of rounds is four. It is inefficient of communications, compared with those of traditional method.

(Timestamp)

U generates a MAC using timestamp. TS_1 denotes the time of generating a request. The SEM check that TS_1 is fresh. Instead of timestamp, the user can send a request containing sequence number.

1. $U \rightarrow SEM : MAC_k(m, TS_1), m, TS_1, ID_u$
2. $SEM \rightarrow U : MAC_k(PS_{sem}, TS_2), PS_{sem}, TS_2$

- Security
The SEM sets a time α . Only if $T \leq TS_1 + \alpha$, where T is the time of receiving request, the SEM can accept this request.
- Efficiency
The computational and communicational costs are the same as those of traditional method.

6 Conclusions

The traditional SEM approach has the disadvantage that it is vulnerable to DoS attacks, since an attacker can send a lot of requests to the SEM server. To prevent DoS attacks, some solutions are proposed. However, these approaches cannot prevent DoS attacks. This paper proposes Security-Mediated PKI using one-time ID. Our proposed system can avoid DoS attacks with small computational complexity. Additionally, our proposed system can protect the privacy of signers, since one-time ID can be used only once.

Suppose that the SEM's partial private key d_{sem} is compromised. If a revoked user colludes with an attacker who obtains d_{sem} , a revoked user can generate a signature. This is the security issue in traditional SEM. Our future work is to improve this issue.

References

1. K. Bacakci, and N. Baykal, "A New Design of Privilege Management Infrastructure with Binding Signature Semantics," 1st European PKI Workshop (EuroPKI 2004), LNCS 3093, pp. 306–313, Springer-Verlag, 2004.
2. D. Boneh, X. Ding, G. Tsudik, and C. M. Wong, "A Method for Fast Revocation of Public Key Certificates and Security Capabilities," In proceedings of the 10th USENIX Security Symposium, pp. 297–308, 2001.
3. D. Boneh, X. Ding, and G. Tsudik, "Fine-grained Control of Security Capabilities," ACM Transactions on Internet Technology (TOIT), Volume 4, pp.60–82 , 2004

4. C. Boyd, "Digital multisignatures," Cryptography and Coding, Oxford University Press, pp. 241–246, 1989.
5. E. Bresson, O. Chevassut, and D. Pointcheval, "New Security Results on Encrypted Key Exchange," International Workshop on Practice and Theory in Public Key Cryptography (PKC 2004), LNCS 2947, pp.145-158, 2004.
6. 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>
7. K. Imamoto, and K. Sakurai, "A Design of Diffie-Hellman Based Key Exchange Using One-time ID in Pre-shared Key Model," The 18th International Conference on Advanced Information Networking and Applications (AINA 2004), pp. 327–332, 2004.
8. ITU/ISO Recommendation. X.509 Information Technology Open Systems Interconnection - The Directory: Authentication Frameworks, 1997.
9. H. Krawczyk, "The IKE-SIGMA Protocol," Internet Draft, 2001.
<http://www.ee.technion.ac.il/hugo/draft-krawczyk-ipsec-ike-sigma-00.txt>
10. H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," IETF RFC2104, 1997. <http://www.faqs.org/rfcs/rfc2104.html>
11. R. Labs, "PKCS #1v2.1: RSA cryptography standard. Tech. rep.," RSA Laboratories, 2002.
12. K. Matsuura, and H. Imai, "Modified Aggressive Mode of Internet Key Exchange Resistant against Denial-of-Service Attacks," IEICE TRANS. INF.&SYST., Vol.E83-D, No.5, pp.972-979, 2000.
13. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol-OCSP," IETF RFC2560, 1999. <http://www.ietf.org/rfc/rfc2560.txt>
14. G. Vanrenen, and S. Smith, "Distributing Security-Mediated PKI," 1st European PKI Workshop (EuroPKI 2004), LNCS 3093, pp. 218–231, Springer-Verlag, 2004.