International Journal of Information Processing and Management
Volume1, Number2, October 2010

A Mediated RSA-based End Entity Certificates Revocation Mechanism with Secure Concerned in Grid

Weifeng Sun, Juanyun Wang, Boxiang Dong, Mingchu Li, Zhenquan Qin[*]
School of Software of Dalian University of Technology Dalian Liaoning, China
wfsun@dlut.edu.cn, wjy48981581@gmail.com, bxdong7@gmail.com, mingchuli@dlut.edu.cn, qzq@dlut.edu.cn
doi: 10.4156/ijipm.vol1.issue2.13

Abstract

The End Entity Certificates (EECs) revocation mechanism in Grid Security Infrastructure (GSI) adopts Certificate Revocation List (CRL) currently. However, CRL is an inefficient mechanism with drawbacks of "time granularity problem" and unmanageable sizes. This paper presents a new EECs revocation mechanism MEECRM (Mediated RSA-based End Entity Certificates Revocation Mechanism) to eliminate "key escrow" problem. MEECRM combines with MyProxy - the online credential repository in Globus Tookit (GT). And some Schemes, such as HMAC, multi-SEM support and PVSS, have been introduced into MEECRM to increase the security and efficiency. MEECRM can ensure instantaneous revocation of invalid EECs in grid environments and can be used in many large-scale grid projects because of inheriting from MyProxy. Analyses also prove that MEECRM is secure.

Keywords: *Certificate revocation, Mediated RSA, Key escrow, Security mediator*

1. Introduction

The virtual trust relationship needs to be established and eliminated in a dynamic manner in grid environments, and breaks through the restrictions on location and traditional collaboration manner. Thus security becomes the biggest barrier against wide adoption of grids. And Grid Security Infrastructure (GSI) is an integrated solution to security issues in a grid.

The End Entity Certificate (EEC) [1] is a kind of long-term X.509 certificate issued by Certificate Authority (CA) in GSI. An EEC whose life-time is long expires in several weeks or one year after issued. EEC issues Proxy Certificates (PCs) and ensures the implementation of mutual authentication between grid entities in gird. PC [2] is a kind of conversation certificate based on X.509 certificate. A PC whose life-time is short expires in several hours. And it may be issued by an EEC or another PC. Grid users need to use PC to access grid services through portal.

Globus Tookit4 (GT4) [3] is the popular middleware to implement grid services, and it has been adopted by more and more large-scale grid projects. As the online credential repository in GT4, MyProxy [4] ensures users access the secure grid services through portal. Moreover, MyProxy provides a centralized and efficient management of users' long-term credentials.

MyProxy supports EEC revocation generally by using CRL [5] in GT4, but CRL is an inefficient and insecure mechanism which has the drawbacks of "time granularity problem" and unmanageable sizes of corresponding overhead [6]. Besides, MyProxy manages all the long-term credentials in the grid, which would cause "key escrow" [7] problem. In this paper, we provide an mRSA-based EEC revocation mechanism (MEECRM) combined with MyProxy. MEECRM could revoke invalid EECs in the grid, and it eliminates "key escrow" problem. Because MEECRM is combined with MyProxy, it can be suited for many grid projects. Further, some security and effective mechanisms is used in MEECRM to achieve higher performance.

The rest of the paper is organized as follow. In section 2, we give a brief introduction of some related works and some analysis of them. Section 3 describes how we introduce several methods, including NOVOMODO, SEM servers, HMAC PVSS and so on, to enhance our system in both

Supported by Nature Science Foundation of China under grant No.: 60673046, 90715037, University Doctor Subject Fund of Education ministry of China under grant No.: 200801410028, National 973 Plan of China under grant No.: 2007CB714205

[*]Corresponding author.

security and efficiency. In section 4, the use of MEECRM will be discussed in detail. In section 5, there will be our analysis about the security concerns of our system and it will be discussed in detail. Finally, in section 6, we will draw our conclusion of the MEECRM and discuss about our further work.

2. Related work

The oldest and most widely used revocation method is to publish a Certificate Revocation List (CRL) [5]. CRL is generated periodically. A CRL is composed of a CA-signed list with all the serial numbers of the revoked certificates that are revoked before their expiration. The relying party needs to fetch the whole CRL from a repository to check whether a certificate is in the latest CRL. However, CRLs can tend to grow into unmanageable sizes with time growing and bring severe bandwidth requirements and transmission costs inevitablely. Besides, the updates of CRLs are not real time scheme, and the long intervals between CRL distributions often result in stale revocation information ("time granularity problem"). All of these may bring serious security problem to the whole virtual organization.

Online Certificate Status Protocol (OCSP) [8] is another approach that a CA answers a query about a certificate C by returning its own digital signature of C's validity status at the current time. However OCSP is problematic in bandwidth and computation because each validity proof must be combined with CA's digital signature.

Micali [9] has proposed NOVOMODO to use one-way hash function to transmit the certificate status in a short value. The CA, at the time of issuing a certificate to the user, generates two random 20-byte values X and Y. CA uses a one-way hash function to operate on these values. The result of several-times hash operation on X is called validation target, and the result of one-time hash operation on Y is called revocation target. CA transmits relevant target to the relying party to release the current status of a certificate. Sushan [10] has implemented NOVOMODO into MyProxy to revoke the invalid PC in grid.

Boneh [6] has proposed Semi-trusted Mediator (SEM) with mediated RSA (mRSA) cryptosystem to realize fine-grained control over security capabilities. MRSA is a simple threshold variant of RSA public key cryptosystem. The CA, at the time of issuing a certificate for the user, splits the relevant private key d into two parts d_{SEM} and d_U , and makes $d = d_{SEM} + d_U \bmod \Phi(n)$. Then CA distributes d_{SEM} to SEM and d_U to the user. SEM and user must cooperate with each other to accomplish decryption or signature operation. Our contribution is to provide a new EEC revocation mechanism in grid: mRSA-based End Entity Certificates Revocation Mechanism (MEECRM). MEECRM can be combined with MyProxy to revoke the invalid EECs on the base of Sushan's work [10].

In HMAC[11], the hardware tokens are made with unique serial number, capacity of on-board HMAC computations and capacity to keep some hidden parameters (HMAC secret keys) inside the token which must not be known to outside world, except known the party who need to authenticate a message or an user. Usage of HMAC capable token can be made in user authentication. The only way to be authenticated is to own the legitimate token as well as the client's password. Thus, attackers only with the legitimate token or the client's password can't undermine the system.

Berry Schoenmakers[12], developed a publicly verifiable secret sharing (PVSS) scheme which is a verifiable secret sharing scheme with the property that the validity of the shares distributed by the dealer can be verified by any part. Hence verification is not limited to the respective participants receiving the shares. After introducing this development into our gird system, our system can achieve securer schemes when tying to verify users.

There should be an effective mechanism easily to be used and solve "key escrow" problem. Furthermore, after a careful analysis of these mechanisms, we find they are ways to increase the security or efficiency of some specific parts during the Certificates Revocation interactions. In other words, some of these works could be integrated together and form a new system as a whole, improving both security and efficiency, in grid.

3. MEECRM

In MEECRM, we join a layer of SEM function module under MyProxy. SEM is a semi-trusted third party. Its main function is helping the valid grid users to accomplish decryption and signature operations, and these operations would be used in the process of mutual authentication and issuing PCs in grid; checking whether users' EECs have been revoked. When a user U wants to access grid services, SEM checks current status of the user's EEC. Only if the user is valid, SEM would correspond with U to accomplish relevant operations. Without the help of SEM, U cannot complete decryption or signature operation independently, in other words, U cannot visit grid services successfully. Suppose U discovers his own private key part d_U is compromised, which makes him feel that he must revoke his EEC at once. In this situation, MyProxy informs SEM (or SEMs in multi-SEMs support scenario) to stop helping the invalid user who uses U's EEC to visit grid services immediately. Thereby the invalid user can not visit grid services with U's compromised EEC. In other words, U's compromised EEC has been revoked already. From the process of revocation of U's EEC, we see that MEECRM could revoke invalid EEC efficiently without adding too much corresponding overhead compared with CRL in grid. And it is secure without the drawback of "time granularity problem". In addition, MyProxy only holds a part of the private key d_{SEM} in MEECRM. Even attackers can breach MyProxy Server (MPS) successfully, they cannot visit grid services without the cooperation of users. Naturally they cannot bring any significant threats to the whole virtual organization. So MEECRM also solves the "key escrow" problem.

3.1. Interaction of MEECRM

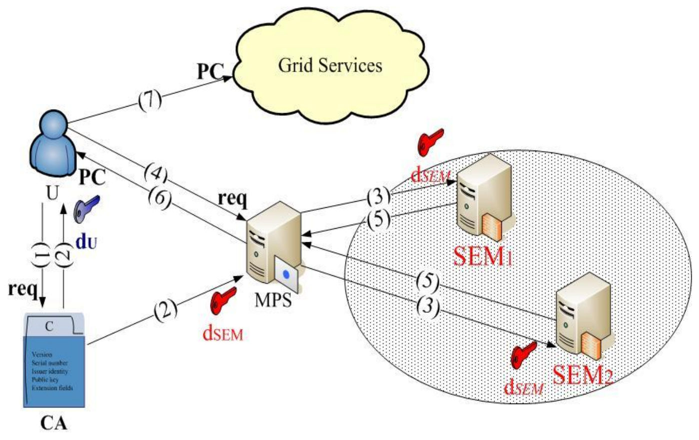

Figure 1. Interaction of MEECRM

Figure 1 shows the interaction of MEECRM. Grid users U could visit grid services through the following steps:

(1) U initiates an application of joining in grid organization to CA.

(2) CA generates two random 20-byte values X_0 and Y_0. Value Y_1 is computed by hashing Y_0 once; and X_N by hashing X_0 N times: $X_1 = H(X_0)$, ..., $X_N = H(X_{N-1})$. In addition to traditional quantities such as a serial number S_N, a public key PK, a user name U, an issue date D_1 and an expiration date $D_2 = D_1 + N$,CA issues a long-term certificate $C = SIG_{CA}(S_N, PK, U, D_1, D_2, ..., Y_1, X_N)$ for U. At the same time, CA splits relevant private key into d_{SEM} land d_U . After that, CA corresponds d_{SEM} to MPS via secure channel, and corresponds d_U to U.

(3) MPS transmits $C's$ id information and relevant d_{SEM} to SEM for later use.

(4) U initiates applications via relevant commands to visit grid services. If the application is asking for SEM to help him accomplish decryption or signature operation, U needs to transmit the results of relevant operation using d_U to MPS through portal.

(5) SEM answers the request with d_{SEM}, and transmits the results to MPS.

(6) MPS transmits the results of response to U. In a typical scenario, U requests MPS to sign a new PC to visit grid services, and then MPS would transmit a new PC to U.

(7) U visits secure grid services using PC or does relevant work using MPS's response.

Now U wants to store his long-term credentials into MPS's credential repository via "myproxy-store" command. In current grid environments: U's EEC C and the whole private key d will be stored; but in MEECRM: only C is stored into credential repository, and private key part d_{SEM} has been stored when C is issued. Thus the credentials stored in the repository are composed of C and private part d_{SEM}.

3.2. Deployment of SEM

SEM is introduced into MEECRM to help users accomplish relevant operation. The number of SEM depends on the number of users the MPS manages and the quantum of traffic in the grid. In a typical scenario, one MPS manages all the long-term credentials in the grid, and a couple of SEMs can carry out all the requests from grid users in such grid environments. SEM1 answers the requests of decryption operation while SEM2 answers the requests of signature operation. Both of the SEMs maintain the same Certificate Revocation Form (CRF) by themselves. CRF stores the information of the revoked EECs.

Both of the SEMs should update their own CRF when a user executes "myproxy-destroy" command. If grid security administrator discovers disaccord between the two CRFs, which imply that one of the SEM may have been attacked. Attackers may have tampered the CRF. Grid security administrator can recover the form via logs.

The identity of a SEM can be certified by the respective CA. Thus a SEM can authenticate itself to a grid entity using traditional certificate-based handshake approach.

3.3. Extension deployment of multi-SEM support

In this section, we first describe a Multi-SEM support which offers a way to avoid single point failure. Then a deployment of publicly verifiable secret sharing scheme will be described.

The section above has discussed a simple deployment of muti-SEM support and it reduces the work load of SEM compared to single SEM deployment. However, Boneh [6] has proposed a Multi-SEM Support which offers a way of obtaining service from any of a set of SEMs and avoid a single point of failure of SEM and which would also be adopted to MEECRM.

Before going to introduce the interaction with multi-SEM support, there would be a slight change, when CA generates d_{SEM}. The mRSA key generation algorithm is shown in Figure 2.

```
Algorithm: mRSA.multi-key (executed by CA)
Choose a collision-resistant hash function H : $\{0,1\}^* \rightarrow [1,\ldots,L]$ where
$L \geq 1024$. Let $k$ (even) be the security parameter. Assume client $U_i$
is authorized to obtain service from $\{SEM_0, SEM_1, \ldots, SEM_m\}$.
(1) Generate random $k/2$-bit primes: $p, q$
(2) $n_i \leftarrow p_i q_i$
(3) $e_i \overset{r}{\leftarrow} Z^*_{\phi(n_i)}$
(4) $d_i \leftarrow 1/e \bmod \phi(n_i)$
(5) $x \overset{r}{\leftarrow} Z_{n_i} - \{0\}$
(6) For each $j \in [0,\ldots,m]$, construct a server bundle for $SEM_j$:
        $d_i^{sem} \leftarrow d_i - \mathrm{H}(x, SEM_j) \bmod \phi(n)$
(7) $SK_i \leftarrow (n_i, x)$
(8) $PK_i \leftarrow (n_i, e_i)$
```

Figure 2. mRSA Key Generation for multiple SEMs

After CA generated the d_{SEM} keys for a user, it distributed the d_{SEM} keys to each SEM servers. When a user wants to cooperate with SEMj to decrypt or encrypt the signatures the user simply computes H(x; SEM j) as the corresponding d_{SEM} half-key.

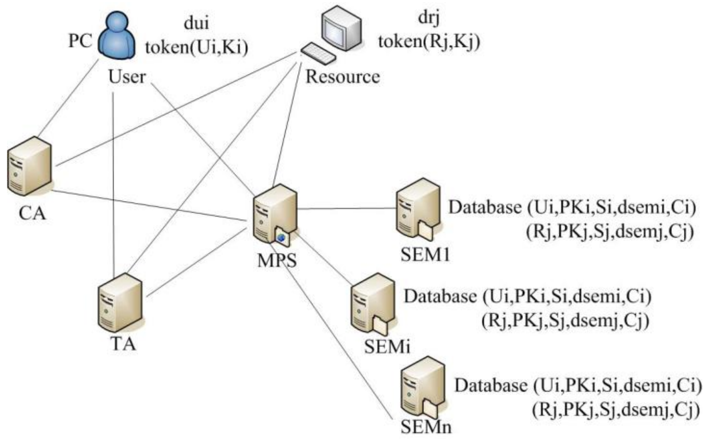

Figure 3. Multi-SEM support

Figure 3 shows the interaction of MEECRM with multi-SEM support. Grid users U could visit grid services through the following steps:

The (1)，(6)，(7) steps are the same with 3.1 And the differences are discussed below.

(2) CA generates two random 20-byte values X_0 and Y_0. Value Y_1 is computed by hashing Y_0 once; and X_N by hashing X_0 N times: $X_1 = H(X_0)$, ..., $X_N = H(X_{N-1})$. In addition to traditional quantities such as a serial number S_N, a public key PK, a user name U, an issue date D_1 and an expiration date $D_2 = D_1 + N$,CA issues a long-term certificate $C = SIG_{CA}(S_N, PK, U, D_1, D_2, ..., Y_1, X_N)$ for U. At the same time, according to the mRSA key generation algorithm in Figure 2, CA generates and transports the d_{SEM} keys which serve a grid user U to MPS via secure channel, and corresponds d_U to U.

(3) MPS distributes the d_{SEM} keys to relevant SEMs.

(4) When U initiates applications via relevant commands to visit grid services. U can choose to cooperate with SEMj to decrypt or encrypt the signatures and the user get the mRSA half key though hash function described at Figure 2.

(5) SEM j answers the request with d_{SEM} , and transmits the results to MPS.

3.4. Deployment with PVSS

Next, the Multi-SEM support scenario make it possible that we introduce a publicly verifiable secret sharing (PVSS) scheme, which enable the system with a reconstruction protocol in which the secret key is recovered by pooling the shares of a qualified subset of the participants and resist malicious SEM sever to recover the d_{SEM} key alone, into the system. After the CA generated the d_{SEM} keys for a user, we adopt this scheme to protect the d_{SEM} key which could be also called secret value in PVSS scheme.

Both 3.1 and 3.3, we described that CA generates and transports the d_{SEM} keys which serve a grid user U to MPS via secure channel. However In PVSS, there is no need for that secure channel between CA and those SEMs. All communication is done over public channels using public key encryption. Consequently, the secret will only be hidden computationally.

In this scheme, CA wishes to distribute shares of a secret values d_{SEM} among n participants SEM1, . . . , SEMn. A monotone access structure describes which subsets of participants are qualified to recover the secret. For example, the access structure may be a (t, n)-threshold schemes, $1 < t < n$, which means that any subset of t or more participants will be able to recover the secret; any smaller subset will be unable to gain any information about the secret, unless a computational assumption is broken.

As a common structure for PVSS schemes we consider the following protocols. Note that initialization is done without any interaction between the CA and the SEMs. In fact, SEMs may join or leave the system dynamically; the only requirement is that a SEM holds a registered public key.

Initialization All system parameters are generated as part of the initialization. Furthermore, each participant SEMi registers a public key to be used with a public key encryption method Ei. The actual set of participants taking part in a run of the PVSS scheme must be a subset of the registered participants.

Distribution The protocol consists of two steps:

(1) Distribution of the shares. The distribution of a secret value d_{SEM} is performed by the dealer CA. The CA first generates the respective shares d_{SEM} i for SEMi for i = 1, . . . , n. For each SEMi the CA publishes the encrypted share Ei (d_{SEM_i}). The CA also publishes a string PROOFD to show that each Ei encrypts a share d_{SEM_i} . Furthermore, the string PROOFD commits the dealer to the value of secret d_{SEM} , and it guarantees that the reconstruction protocol will result in the same value of d_{SEM} .

(2) Verification of the shares. Any party knowing the public keys for the encryption methods Ei may verify the shares d_{SEM_i} . For each participant SEMi a non-interactive verification algorithm can be run on PROOFD to verify that Ei (d_{SEM_i}) is a correct encryption of a share for SEMi. Since anyone may verify a share, it may be ruled out that a participant complains while it does not received a correct share. In case one or more verifications fail, we therefore say that the dealer fails, and the protocol is aborted.

Reconstruction The protocol consists of two steps:

(1) Decryption of the shares. The participants decrypt their shares d_{SEM_i} from Ei(d_{SEM_i}). It is not required that all participants succeed in doing so, as long as a qualified set of participants is successful. These participants release d_{SEM_i} plus a string $PROOFSEM_i$ that shows that the released share is correct.

(2) Pooling the shares. The strings $PROOFSEM_i$ are used to exclude the participants which are dishonest or fail to reproduce their share d_{SEM_i} correctly. Reconstruction of the secret d_{SEM} can be done from the shares of any qualified set of participants.

As discussed above, with the PVSS scheme, the secret key d_{SEM} are divided into n parts. The secret would not be known to outside world unless pooling t parts or more together.

We described two different extensions, in this section, to protect the SEM servers, because the sever side is more likely to suffer from malicious attacks. One extension is part 3.3 in which provide an mRSA key generation algorithm for multiple SEMs, and the other is part 3.4 deployment with PVSS. In part 3.3, it is better for a CA to transmit the d_{SEM} key in security channel, however, in PVSS, all communication is done though public channels using public key encryption method. In this way, extension with PVSS would more likely to be adopted widely. However, noted that when CA splits relevant private key into d_{SEM} and d_U , the method itself can ensure the safety of the system without the help of the two extensions discussed above and it is hard to tell whether the workload of computing mRSA key generation algorithm for multiple SEMs in part 3.3 or the PVSS algorithm is greater, so these two alternative methods could be chosen according to the physical environment if additional security concerns are needed.

3.5. Deployment of HMAC capable token in user authentication

Here we describe MEECRM using a HMAC capable token in user authentication or public key infrastructure (PKI) to derive user private key or produce message digest for digital signature scheme. The unique hardware token will be linked together with the user password cryptographically to provide a more secure/stronger solution, especially to provide protection against denial-of-service (DoS) attacks on a SEM.

First of all, TA must first select a random generated master identity secret (for identification purpose), denoted as $S_{identity}$, which must also be known by the SEM server which need to authenticate a user or incoming message.

TA generated the token the hardware tokens with unique serial number i, capacity of on-board HMAC computations and capacity to keep some hidden parameters (HMAC secret keys) inside. And the token can be in any hardware which can plug in or connect to PC, with USB port, for example.

The token can be owned by authenticated users or hardware resources in grid infrastructure such as PCs. Here we regard HMAC(a , b) as applying HMAC using “a” as the key and “b” is the data we want to compute the digest. So, TA need to compute $K_{identification}$ = HMAC($S_{identity}$, i) and then insert this key in to the token. This is the secret key keep inside the token which should not be known to outside. Since only TA and authentication SEM server know about $S_{identity}$ and always be able to compute HMAC($S_{identity}$, i), which means server can authenticate any message (M) received from client consist of digest HMAC($K_{identification}$, M).

When registering a user, we must let user choose an authentication password (P). Then the SEM server need to record user’s ID (or name), corresponding token serial number and his password in database, and the password must be saved in encrypted form.

The SEM server can send a random generated challenge value (RC) to the client, then the client will have to response by compute u = HMAC($K_{identification}$, RC) using the token and then compute R_{client} =HMAC(P, u). R_{client} is the response value which the client has to send back to server for verification. At the server side, it must know which user is trying to login, then it will retrieve the required parameters from the database (i.e. the user specific token serial number and password), and compute the correct response value R_{server} locally (since server know about the master identity secret). Then, server can compare R_{client} with R_{server} to determine if the user is a legitimate user and own the corresponding legitimate token. By imposing the HMAC capable token in an innovative way, an attacker with only either a stolen token or just knowing the user password is not enough to compromise the system. He cannot compute a correct response value without both of the components.

4. Use of MEECRM

Application of MEECRM mainly consists of mutual authentication, mutual message exchange method, revocation of EECs and issuing PCs.

4.1. Mutual authentication protocol

$$U \rightarrow R \qquad EC_{PK_R}(ID_U \| m \| T_1) \tag{1}$$
$$SEM \rightarrow R \qquad PC_{1sem} \tag{2}$$
$$R \rightarrow U \qquad EC_{PK_U}(ID_R \| m \| K_s \| T_2) \tag{3}$$
$$SEM \rightarrow U \qquad PC_{2sem} \tag{4}$$
$$U \rightarrow R \qquad EC_{K_5}(m \| T_3) \tag{5}$$

Figure 4. Mutual authentication protocol

Suppose mutual authentication is needed between the grid user U and a resource provider R. U fetches R's public key PK_R firstly through online requests. Then U and R use the following protocol in figure4 to accomplish mutual authentication:

(1) U encrypts random message m, his own id and current timestamp T_1 with PK_R to generate C_1. U transmits C_1 to R.

(2) SEM decrypts C_1 using R's d_{SEM} , and transmits the result to R.

(3) R decrypts C_1 with the help of SEM, and then R fetches m now. R encrypts m, his own ID, random conversation key K_s and timestamp using PK_U to generate C_2. R transmits C_2 to U. SEM will not help R decrypt C_1 only if R is valid, so U is convinced that both of R's identity and current status T_2 of R's EEC are valid.

(4) SEM decrypts C_2 using U's d_{SEM} , and transmits the result to U.

(5) U decrypts C_2 with the help of SEM, and then U fetches m and K_s now. U encrypts m using K_s to generate C_3. U transmits C_3 to R. R is convinced that both of U's identity and current T_3 status of U's EEC are valid if R could decrypt C_3 using K_s successfully.

4.2. Mutual message exchange method

Similar to mutual authentication protocol, we now suppose that user U wants to exchange a private message with a resource provider R.

(1) U fetches R's public key PK_R through online requests.

(2) When U considering encrypting his message using the fetched public key PK_R, he separates the whole message into two parts: (i) a short preamble containing a per-message key encrypted with R's public key, and (ii) the body containing the actual private message encrypted using the per-message key.

(3) Then the whole message is sent to R.

(4) To decrypt the message from U, R sends preamble to it SEM. SEM decrypts the preamble using R's d_{SEM} and transmits the result to R.

(5) R decrypts the preamble with the help of SEM, and then R fetches the per-message key now.

(6) R completes the decryption of the body part using the per-message key and, ultimately, to get the actual private message.

4.3. Revocation of EECs

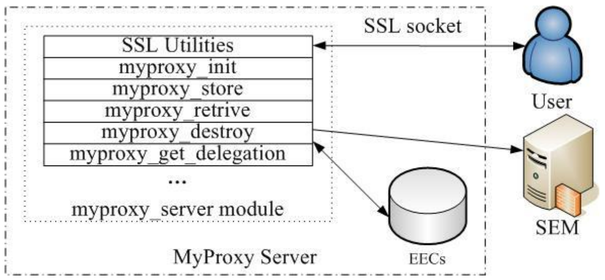

Figure 5. Revocation of EECs

Figure 5 shows how to revoke invalid EECs in MEECRM. Suppose grid user U discovers his EEC C is compromised. U revokes C via “myproxy-destroy” command and the MPS changes $C's$ status into invalid. At the same time, MPS informs SEM that C has been revoked. SEM puts the information of C into his own CRF. From now on, SEM refuses to cooperate with the user who wants to visit grid services with C. In other words, malicious users can not visit grid services although they may have invalid C and relevant d_U . Now C has been revoked without too much corresponding overhead.

4.4. Update of EECs

CA updates C's proof of status by computing and transmitting an up-to-time proof to SEM. M time intervals after C's issuance, CA hashes X_0 m($m \leq N$) times: $X_1 = H(X_0)$, ..., $X_m = H(X_{m-1})$. Then CA gets X_m and transmits it with U's serial number to SEM. SEM receives them and check whether if U's certificate is valid or revoked. MPS selects U's certificate $C = SIG_{CA}(S_N, PK, U, D_1, D_2, ..., Y_1, X_N)$ from SEM and hashes X_m N-m times. So MPS gets the result of X_N and finds out whether if the result equals the X_N in U's certificate. If so, SEM is convinced that U's certificate is valid during the m*th* time interval after C's issuance and the update process is completed.

4.5. Issuing PC

Grid user U could submit an application of issuing a new PC via “myproxy-get-delegation” command. U encrypts message m with d_U to generate a part of mRSA digital signature PS_U . U transmits PS_U to MPS through portal. At the same time, SEM encrypts message m with d_{SEM} to generate another part of mRSA signature PS_{sem}. SEM transmits PS_{sem} to MPS. MPS generates an integrated mRSA signature S with PS_U and PS_{sem}. MPS uses S to sign a new PC for U, and then U can visit grid services using that PC.

5. Security analysis

We focus on the security problems in the process of issuing a new PC and mutual authentication. There are several security concerns:

(1) CA is free from attacks which intend to compromise the key pair generated after user initiates an application of joining in grid organization.

(2) The user can not generate signatures or decrypt any cryptograph after being revoked in MEECRM.

(3) Introducing timestamp in the mutual authentication protocol can avoid replay attack.

(4) MEECRM with SEM can protect private Messages.

(5) MEECRM with Multi-SEM Support can avoid failure when a single SEM serves do not work.

(6) An alternative deployment with PVSS and its security concerns.

(7) MEECRM makes use of HMAC in user authentication or public key infrastructure (PKI).

There will be discussions about the security concerns in more details, just as follows.

5.1. Security concerns on CA

CA, serving as a trusted entity and issuing key-pair, is an idea target for any attacker who wants to compromise the system. However, it is useless for malicious attackers to try to get the key-pair generated after users’ initiation. Note that, in traditional CRL mechanism, CA has to keep the revocation list and leave the potential opportunity for attacker to forge the list behind. In our system, a CA only generates client’s keys and does not need to keep them. In fact, a CA distributes the key-pair to the user and the relevant SEM immediately after the pair being generated. CA must erase them to assure that any successful future attack on the CA does not result in client's keys being compromised.

5.2. Security concerns by adopting mRSA

Each corresponding message has been encrypted by the other's public key during the process. Boneh [6] has proved that the user cannot do relevant operation after being revoked in mRSA cryptosystem because of lacking the half key d_{SEM} on SEM server. And a malicious user who is trying to forge a conversation message on the behavior of U or SEM cannot succeed due to the security of mRSA cryptosystem in which no one can forge d_U or d_{SEM} on himself. Besides SEM or U would not be able to accomplish decryption or signature operation independently which has also been proved impossible in theory by Boneh [6]. So the conversation between the grid entities is confidential.

5.3. Security concerns by introducing timestamp to provide timeliness service

During the process of authentication, each message is combined with the current timestamp which has been introduced in the mutual authentication protocol in 4.1. Message recipients only accept the message which is timely they think. So the protocol can fight against replay attacks.

5.4. Mutual authentication protocol provides protection against attacking to private Messages

We recall that public key-encrypted message is usually composed of two parts: (1) a short preamble containing a per-message key encrypted with public key, and (2) the body containing the actual private message encrypted using the per-message key.

Soon after someone sends a message to the Resource Provider (RP) in this system, for example, A user called Wood sends a message m to the RP, RP just need to sends the preamble part to its SEM and SEM responses with a token which enables RP to complete the decryption of the per-message key and, ultimately, to read Wood's message. However, this token contains no information useful to anyone other than Wood. Hence, communication with the SEM does not need to be secret or authenticated.

5.5. Multi-SEM supporting provides protection against single point failure

As we know, each SEM serves many clients, a SEM failure -whether due to hostile attacks or accidental causes - stops all of its clients from decrypting data and generating signatures. To avoid such failure, mRSA can be modified to allow a single client to use multiple SEMs. We could either simply replicate a SEM to avoid accidental (non-malicious) failures, though this method cannot prevent from hostile attacks, or we could allow a client to be served by multiple SEMs, each with a different mRSA setting. The different mRSA method would require the CA to run the mRSA key generation algorithm t times (if t is the number of SEMs) for each client, obviously it is complicated. Our approach allows a SEM client to have a single public key and a single certificate while offering the flexibility of obtaining service from any of a set of SEMs. At the same time, each SEM maintains a different mRSA half-key for a given client. Thus, if any number of SEMs(who support a given client) collude, they are unable to impersonate that client, i.e., unable to compute the client's half-key.

5.6. Advantage of deployment with PVSS

There are two main advantages adopting PVSS. Firstly, it is unnecessary to build a security channel between CA and the relevant SEM servers, which obviously reduces the cost of the system. Secondly, directly breaking the encryptions used in our PVSS scheme implies breaking the Diffie-Hellman assumption which has been proved infeasible in [12].

The special PVSS scheme is secure in the random oracle model. That is, (1) the reconstruction protocol results in the secret distributed by the dealer for any qualified set of participants, (2) any non-qualified set of participants is not able to recover the secret.

5.7. HMAC provide protection against DoS attacks

We adopt HMAC in user authentication or public key infrastructure. In our approach, we adopted HMAC to authenticate client requests. In this way to provide protection against denial-of-service (DoS) attacks on a SEM. As discussed in section 3, this would require a shared secret key between a SEM and each client. A CA could help in the generation and distribution of such shared secrets at the time of mRSA key generation. Yet another alternative is to rely on more general encapsulation techniques, such as SSL, to provide a secure channel for communication between SEMs and clients. An attacker with only either a stolen token or just knowing the user password is not enough to compromise the system. He cannot compute a correct response value without both of the hardware token and the user’s password.

From the discussion above, it shows that our security concerns cover every parts of the system among them are the trust entity CA, the user in grid, the MPS sever and the SEM servers. Firstly, CA is carefree because nobody can compromise the key-part generated by it. Secondly, the confidential information transmitted between users and SEMs is protected by mRSA cryptosystem and the preamble part which encrypted with public key. Thirdly, there are alternative choices to protect the SEM side such as Multi-SEM support and deployment with PVSS. At last, HMAC has been introduced to authenticate client requests and protect against denial-of-service (DoS) attacks on a SEM.

6. Conclusions

We form a new approach from retrieving credential to visit grid services to revocation of such credential by mainly combining MyProxy, NOVOMODO, SEM function module, PVSS and HMAC. We now achieve simplified validation of digital signatures, efficient credential revocation for grid systems and fast revocation of signature and decryption capabilities. At the same time, we reduce the complex certificate management and the workload when computing signature. We also provide alternative ways to extend the system, depending on the security level. So MEECRM is an efficient and secure mechanism to revoke the invalid EECs in grid environment.

Up to now, the research on MEECRM remains in theory. In future work, we will consummate MEECRM and do relevant experiment testing to improve the performance of MEECRM.

7. References

[1] V. Welch, I. Foster, C. Kesselman, etc, X.509 Proxy Certificate for Dynamic Delegation, In 3rd Annual PKI P&D Workshop, 2004.
[2] S. Tuecke, V. Welch, D. Engert, L. Perlman, and M. Thompson, RFC3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, 2004.
[3] I. Foster, C. Kesselman, Globus: A Metacomputing Infrastructure Tookit[J], International Journal of Supercomputer Applications, 1997, 11(2), pp.115-128.
[4] J.Novotny, S. Tuecke, V. Welch, An Online Credential Repository for the Grid: MyProxy, In 10th IEEE International Symposium on High Performance Distributed Computing, 2001.
[5] R Housley, W Ford, W Polk, D Solo, X.509 Internet Public Key Infrastructure Certificate and CRL Profile, IETF RFC2459, 1999.
[6] D. Boneh, X. Ding, G. Tsudik, Fine-Grained Control of Security Capabilities, ACM Transactions on Internet Technology, Vol.4, No. 1, 2004, pp.60-82.

[7] J. Basney, M. Humphrey, and V. Welch, The MyProxy online credential repository, Software: Practice and Experience, 35(9), 2005.
[8] X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP), IETF RFC2560, http://www.ietf.org/rfc/rfc2560.txt.
[9] S. Micali, NOVOMODO: Scable Certificate Validation And Simplified PKI Management, In 1st Annual PKI Research Workshop, 2002.
[10] S.Zhao, A.Aggarwal, R.D.Kent, A Framework for Revocation of Proxy Certificates in a Grid, In SNPD, 2007, 8th ACIS International on, pp.532-537.
[11] Shanhui Tan, Integrate HMAC Capable Token into User Authentication Mechanism and Public Key Infrastructure, SANS Institute InfoSec Reading Room, 2003.
[12] Berry Schoenmakers, A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting, In Advances in Cryptology—CRYPTO '99, Vol. 1666 of Lecture Notes in Computer Science,Springer-Verlag, pp.148-164,1999.