October 19, 1998
RT0280
Computer Science 8 pages

Research Report

A Secure Key Registration System Based on Proactive Secret-Sharing Scheme

Masayuki Numao

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A Secure Key Registration System Based on Proactive Secret-Sharing Scheme

Masayuki Numao
Tokyo Research Laboratory, IBM Japan, Ltd.
numao@jp.ibm.com

Abstract

We designed a secure key registration system based on the proactive secret-sharing scheme. A user can register important data such as a session key to a distributed system in a (t, n)-threshold scheme, which means that the data can be recovered if t servers cooperate (in other words, that the data cannot be revealed unless t servers collude). The proactive scheme provides stronger security against an active adversary. We designed the protocol to generate an implicit secret, to distribute shares of it, and to reconstruct the secret for proactive secret-sharing without a dealer.

We also developed a prototype of a data archiving service framework on the Internet. To allow users to access the system via a Web browser, we implemented a system based on the PKI (public key infrastructure), where the client/server authentication is done by means of X.509 certification. We also used the publish/subscribe communication model to realize interaction between key management servers, because it is easy to implement the broadcasting channels used in the share update phase.

1. Introduction

As the number of Internet users explodes, many real-life activities are coming to be conducted over the Internet; for example, electronic commerce enable consumers to buy anything at any time from anywhere in the world. Since the Internet is an open network, maintaining confidentiality, privacy, authentication, and so on is becoming a big issue. One approach to resolving this issue is through the use of cryptographic technology, for example, SSL is a popular cryptographic protocol for authenticating the server (and the client) and creating a secure communication channel. These authentication and digital signature technologies rely on the key, which consists of a bit string. Key management is therefore a very important issue for the user, because if the user's private key is revealed, an adversary can impersonate him or her.

This paper presents a secure key registration system based on a distributed secret sharing mechanism called "proactive secret sharing" [OY91] [HJKY95]. Key registration is a basic technology for key recovery [Denning96] [Maher96] [Walker96], in which the session key is encrypted by using the public key of a key recovery agent and attached to the encrypted message as a key recovery field. When a message need to be recovered - for example, by the government for the purpose of law enforcement, or by an owner who has lost his master key - the key recovery field is sent to the key recovery agent, who then decrypts the session key by using his private key.

In this case, the key recovery agent becomes a single point of attack for an adversary intent on getting the private key; therefore secret sharing is used in order to distribute the attack points. As Maher mentioned [Maher96], it is securer if the private key is distributed and only constructed during the recovery phase. Such a key pair generation technique is realized by naïve application of the ElGamal scheme [Ped91] and by the sophisticated method developed by Boneh and Franklin [BF97] for RSA crypto.

The method of generating an implicit secret and distributing the share is naturally integrated into the proactive secret-sharing scheme. By applying Desmedt and Frankel's protocol [DF89], we also show that the secret is reusable in the sense that the user can send messages to multiple receivers, each of whom can decrypt only the messages addressed to him. We also extend the scheme to maintain multiple secrets for multiple users based on the (c, d; k, n)-multi-secret sharing scheme defined by Franklin and Yung [FY92]. This scheme is quite useful for the long-term key management, because it is possible to invalidate some of the keys without affecting other keys.

In summary, our distributed key registration system has the following features:

- A public key is constructed from periodically refreshed random seeds (shares) by the user as a key-encryption key.
- The private key (secret) is generated implicitly without a dealer; thus no one knows it until it is actually reconstructed.
- The secret is maintained in a (t, n)-threshold proactive scheme. At least t servers are necessary to

recover the secret, and no *t-1* servers can deduce anything.

- The message can be decrypted without revealing the secret. Thus the public key is used to encrypt messages for multiple receivers.

In Section 2, we will explain the share update protocol of proactive secret-sharing, which constitutes the basic technology of our application. Then in Section 3, we will explain the key registry framework as a target application and describe how existing secret-sharing schemes are used for this application. In Section 4, we will describe the protocol for the key generation, secret update, and message decryption based on proactive secret-sharing scheme. We also describe an Internet-based data archiving service framework as an application of our protocol in Section 5. Finally we summarize in Section 6.

2. Proactive Secret Sharing

Proactive secret-sharing [HJKY95] provides strong security for a secret-sharing scheme against the active attacker. The technology is especially useful in applications such as certificate authority (CA) and key recovery (KR), since private keys in these applications need to be very securely maintained for long periods. It combines the secret-sharing technology [Shamir79] with a periodical share update process to ensure the overall security of a system. Through the use of this refreshment mechanism, old shares become useless; thus, to steal a secret, an attacker needs to intrude on at least *t* servers in the same time frame if security is maintained in a *t*-threshold secret-sharing scheme.

Proactive schemes have been applied to various security technologies such as secret sharing, signature sharing, and secure communication by using crypto scheme such as ElGamal, RSA, and DSA, which were surveyed by Canetti et al. [CGHN97]. Here we will explain proactive secret-sharing, which was introduced by Herzberg et al. [HJKY95]. The technology is based on Verifiable Secret Sharing (*VSS*) [CGMA85]. Suppose that a secret s is defined over a prime field Z_q^*, where *q* is a prime number, that $p = mq+1$ is also prime with a small integer *m*, and that a total of *n* servers maintain the secret.

[Initialize]

The dealer generates a polynomial of degree *t-1* using random numbers $d_1, .. , d_{t-1}$:

$$f(x) = s + d_1 x + ... + d_{t-1} x^{t-1}$$

Then, the dealer sends the share $s_i = f(i) \bmod q$ to the server *i*. Since this is Shamir's secret-sharing scheme [Shamir79], any *t* servers can reconstruct the secret by using Lagrange interpolation, while *t-1* cannot get any information.

[Share Update]

Each server *i* generates a polynomial of degree *t-1* by using random numbers $d_1, .., d_{t-1}$:

$$f_i(x) = d_1 x + ... + d_{t-1} x^{t-1}$$

This satisfies $f_i(0)=0$. Server *i* then sends the value $s_{ij} = f_i(j) \bmod q$ to server *j*, which updates its new share $s_j^{(new)}$ as follows:

$s_j^{(new)} = s_j^{(old)} + s_{1j} + \ldots + s_{nj} \bmod q$.

Since the new shares lie on the polynomial $f^{(new)}(x) = f^{(old)}(x) + f_1(x) + \ldots + f_n(x)$, the new polynomial still maintains the secret value *s* at *x=0* , because $f^{(new)}(0) = s + 0 + \ldots + 0 = s$.

In both the **[Initialize]** and **[Share Update]** phases, the dealer and the servers generate polynomials satisfying constraints $f(0)=s$ and $f_i(0)=0$, respectively. To prevent the dealer and servers from distributing wrong shares, Feldman's verifiable secret sharing (*VSS)* [Feldman87] is applied: When the dealer or servers generate a polynomial whose coefficients are $d_0, .., d_{t-1}$, they first broadcast $g^{d_0}, g^{d_1}, ..., g^{d_{t-1}}$ where *g* is an element of Z_p^* of order *q*.

When the server *i* receives his share $s_i = f(i)$, he can verify the value by checking:

$$g^{s_i} \stackrel{?}{=} (g^{d_0})(g^{d_1})^i (g^{d_2})^{i^2} ... (g^{d_{t-1}})^{i^{t-1}} \pmod p$$

If this holds, server *j* announces *i*-correct, otherwise, it announces *i*-wrong. If more than *t* *i*-corrects are announced, server *i* is included in the good set of servers. Subsequently, proactive secret sharing performs a share recovery protocol for servers in which VSS fails.

[Share Recovery]

Suppose that *r* is a server in which VSS failed. Each VSS-verified server *i* generates a polynomial of order *t-1* by using random numbers $d_0, d_1, .., d_{t-1}$:

$$f_i(x) = d_0 + d_1 x + ... + d_{t-1} x^{t-1}$$

which satisfies $f_i(r)=0$. Server *i* then sends the value $s_{ij} = f_i(j) \bmod q$ to the other VSS-verified server *j*, which generates a share for *r* as follows:

$s_j^{(for\ r)} = s_j + s_{1j} + \ldots + s_{nj} \bmod q$.

Server *j* then sends $s_j^{(for\ r)}$ to the server *r*.

Server *r* interpolates the polynomial *f'(x)* to recover its share at $f'(r)=f(r)$. Note that since the $f'(0)$ is randomized, server *r* does not get the secret *s*.

3. Key Registration and Secret Sharing

Key registration is a service for providing public keys with which users encrypt their data (usually session keys). This service is first required by key recovery [Denning96] [Maher96] [Walker96]: whenever a message is encrypted by using a session key, the session key itself is encrypted

under the public key of the key recovery agent and attached to the encrypted message as the key recovery field. The message format is $\{E_k(M), PK_{Rec}(k), PK_{KRA}(k)\}$, where $E_k()$ is symmetric encryption by the key k, and $PK_r()$ is public key encryption by means of the key r. The first term is a message encrypted by a session key k, the second term is the session key encrypted by the receiver's public key, and the third term is a key recovery field, which is the session key encrypted by means of the public key of the key recovery agent. Compared with the key escrow scheme, in which the user's master key must be actually sent to the escrow agent, key recovery is safer from the viewpoint of the user's privacy, because only the session key is escrowed and the escrow is performed on the user's side.

To distribute the attack points, multiple agents are introduced and the secret is divided into pieces and distributed to the agents. Verheul designed an ElGamal key-sharing technique [VT97]: The server i has a private share s_i and a public share $y_i = g^{s_i} \bmod p$, and the user can construct a new public key $Y = \prod_i y_i$ by selecting any subset of servers. The corresponding private key is $\sum_i s_i$, but it does not exist until the selected servers' secrets are summed. This scheme is very secure, because of the following points:

(1) The user has control over the public key construction.
(2) The private key never needs to be constructed until the recovery phase.
(3) The servers have no information on the public key or private key

In this scheme, there is no interaction among the servers and between the servers and user in the key generation phase, and therefore the servers are autonomous. But one big disadvantage is that it is a fragile (t, t)-threshold scheme, and thus if one of the selected servers breaks down, then the secret is lost.

Thus, we need some protocols for migrating from (t, t)-threshold into (t, n)-threshold proactive scheme. We will apply the some of the protocols that aim at generating an implicit secret without the assistance of a mutually trusted party (MTP): Ingermarson and Simmons first pointed out the importance of such implicit secret generation [IS90]. Pedersen then applied the idea to verifiable secret sharing [Pedersen91]. Jackson et al. later generalized the idea to support any monotone access structure in the perfect secret-sharing scheme [JMOK95]. The basic idea is to have t servers first generate shares of a unanimous (t, t)-threshold scheme, then each server acts as a dealer as if the secret were its own private share, computes the shares for the other servers, and distributes the shares to other n servers.

It needs an interaction between the user and the servers; that is, the user tells the servers about which servers are selected, and interaction among the servers to perform share distribution. Thus, from the security point of view, (1) and (2) are satisfied, and (3) is slightly modified, but robustness (4) can be realized as follows:

(3)' t-1 servers have no information on the private key (secret).
(4) Any k servers can recover the secret.

From the practical point of view, the system is desired to serve to multiple users, which means that multiple secrets can be registered and retrieved independently. We will apply a multi-secret sharing scheme defined by Franklin and Yung [FY92] to extend our scheme to handle multiple secrets. They defined (c, d; k, n)-multi-secret scheme, which is a protocol between a dealer and n servers with the distribution phase in which the dealer distributes the shares of k secrets to the servers, and the recovery phase where any subset of at least d servers can reconstruct all k secrets and no subset of at most c servers can deduce anything. And they provided an implementation of a (t-k+1, t+1, k, n)-multi-secret sharing scheme based on Shamir's secret-sharing (where $k \le t$) : Let s_1,.., s_k be the secrets. Assume that a_1,.., a_n and e_1,…, e_k are pre-selected element of Zp that are known to the dealer and all servers. The dealer first generates a polynomial f of the degree t which satisfies the equations $f(e_i)=s_I$ for $1 \le i \le k$, then distribute the value $f(a_j)$ to the server j ($1 \le j \le n$) as his share. Any t+1 servers can interpolate their shares to recover $f(x)$, and hence get all k secrets. We modified the protocol to enables k users to request servers to generate the secrets and to recover them independently.

4. Key Management Protocol Based on Proactive Secret Sharing

4.1. Model and Definition

We assume a system of n servers $P=\{P_1, .., P_n\}$ that will share k secrets $X=\{x_1, .., x_k\}$ through a (t-k+1, t+1; k, n)-multi-secret-sharing scheme. The goal of the scheme is for k users $U=\{u_1, .., u_k\}$ to register the secrets (keys) in or retrieve them from the system in a (t+1, n)-threshold scheme without relying on a trusted dealer, which means that if the t+1 servers are honest, then all users can retrieve their registered secrets confidentially, and no subsets of at most t-k+1 dishonest servers can destroy or steal any secrets.

Here, we will distinguish the security level of users and dealers as follows: the dealer knows the secrets before they are distributed to the servers, whereas a user can use only the public part of the secrets to encrypt his own secret (key) without knowing the actual secrets. Therefore, in our scheme, no one knows the secrets until t+1 servers reconstruct them.

In the following subsections, p and q denote large primes such that $p=mq+1$, where m is a small integer, and $g \in Z_q^*$ is a generator of order q. Assume that $a_1, .., a_n$ and $e_1, \ldots, e_k$ are pre-selected elements of Zp that represent the positions of the servers and secrets, respectively. We can assume that $n \geq 2t-k+2$, which means that the system allows at most $t-k+1$ cheating servers but that at least $t+1$ are still honest.

4.2. Outline of the Protocols

We illustrate how our protocol is used to the key registry in figure 1. Every server i ($1 \leq i \leq n$) maintains a secret share s_i and public share $y_i = g^{s_i} \bmod p$, and broadcasts the public share. At **[Initialization]** described in section 4.3, all the implicit secrets maintained in the system are set to 0. By means of periodical **[Share Update]** described in section 4.4, all the servers' shares are updated, while all the secrets are unchanged.

In some time frame, when user j ($1 \leq j \leq k$) chooses l servers out of n servers (where $n \geq l \geq 2t-k+2$) and constructs the public key by multiplying the public shares of selected servers, he informs all the servers about the selection $SELj=\{i_{j1}, .. i_{jl}\}$. Then, each selected server Pi $i \in SEL_j$ distributes his share to other servers by **[Share Distribution]** described in section 4.5, to register the sum of the shares as the j-th secret. The corresponding public key is also constructed and published. Thus the user can encrypt his message by the public key as described in section 4.6 **[Message Encryption]**.

For all the other time frame, all the servers update their shares by **[Share Update]**.

Now, when user j wants to decrypt his message, he requests reconstruction of the j-th secret to at least $t+1$ servers. Then the servers authenticate the request and if it is validated, the servers send the shares for the j-th secret to the user by **[Message Decryption]** described in section 4.7. After getting the shares, the user can decrypt the message.

4.3. Initialization

1. Each server P_i generates a polynomial f_i of degree t, which satisfies the condition $f_i(e_l) = 0$ for $1 \leq l \leq k$.
$$f_i(x) = d_{i,0} + d_{i,1}x + \ldots + d_{i,t}x^t$$
2. For all other servers P_j, P_i sends $s_{ij} = f_i(a_j) \pmod q$ to P_j via a secure channel. At the same time, P_i broadcasts $g^{d_{i,0}}, g^{d_{i,1}}, \ldots, g^{d_{i,t}}$.
3. After receiving s_{ij}, P_j first checks the correctness of its share as
$$g^{s_{ij}} \stackrel{?}{=} (g^{d_{i,0}})(g^{d_{i,1}})^{a_j}(g^{d_{i,2}})^{a_j^2}\ldots(g^{d_{i,t}})^{a_j^t} \pmod p$$
And the validity of the polynomial as
$$1 \stackrel{?}{=} (g^{d_{i,0}})(g^{d_{i,1}})^{e_l}(g^{d_{i,2}})^{e_l^2}\ldots(g^{d_{i,t}})^{e_l^t} \pmod p$$
for $1 \leq l \leq k$.
4. If above VSS is verified, P_j computes its initial share as $s_j = s_{1j} + \ldots + s_{nj} \pmod q$.

After this initialization protocol is completed, the shares interpolate a polynomial which satisfies $f(e_l) = 0$ for $1 \leq l \leq k$.

4.4. Share Update

This protocol is applied to all the servers in the normal

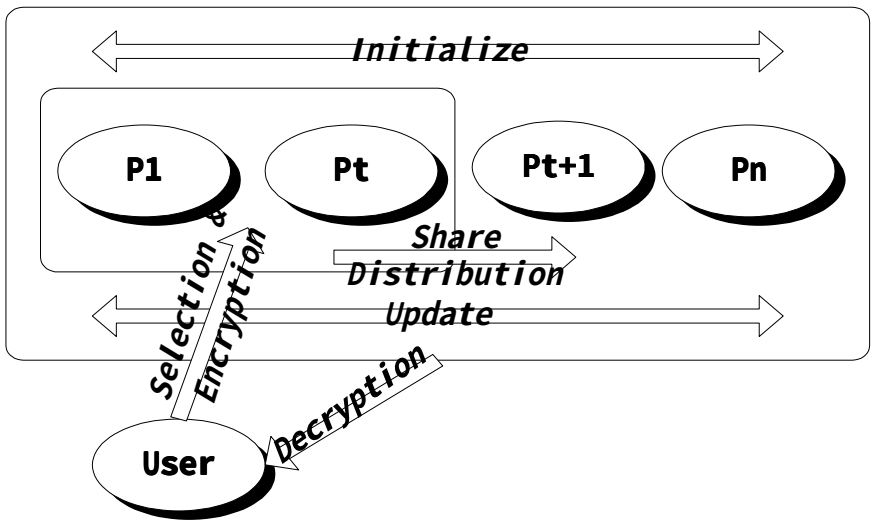

Figure 1. Outline of the Protocols

update phase. The protocol is the same as the initialization protocol up to 3.

4. If the above VSS is verified, P_j computes its new share as $s_{j(,new)} = s_{j(old)} + s_{1j} + \ldots + s_{nj} \pmod q$.

4.5. Share Distribution

This protocol is applied to the servers when user *j* interacts with the servers to generate an implicit secret and to register it as the *j*-th secret. We assume that all the servers are informed of the selection SEL_j via a public channel.

1. Each selected server P_i ($i \in SEL_j$) generates a polynomial f_i of degree *t*:

   $$f_i(x) = d_{i,0} + d_{i,1}x + \ldots + d_{i,t}x^t$$

   that satisfies the condition $f_i(e_j) = s_i$, and $f_i(e_l) = 0$ for $l=1,..,j-1,j+1,..,k$.
   At the same time, other unselected server $P_{i'}$, ($i' \notin SEL_j$) generates a polynomial $f_{i'}$ of the degree *t* which satisfies the condition $f_i(e_l) = 0$ for $1 \le l \le k$.
2. For all other servers P_j, P_i sends $s_{ij} = f_i(a_j) \pmod q$ to P_j via a secure channel. At the same time, *Pi* broadcasts $g^{d_{i,0}}, g^{d_{i,1}}, \ldots, g^{d_{i,t}}$.
3. After receiving s_{ij}, P_j first checks the correctness of his share by:

   $$g^{s_{ij}} \stackrel{?}{=} (g^{d_{i,0}})(g^{d_{i,1}})^{a_j}(g^{d_{i,2}})^{a_j^2}\ldots(g^{d_{i,t}})^{a_j^t} \pmod p$$

   P_j then checks that the local secret was correctly transferred from the selected server $i \in SEL_j$:

   $$g^{s_i} \stackrel{?}{=} (g^{d_{i,0}})(g^{d_{i,1}})^{e_j}(g^{d_{i,2}})^{e_j^2}\ldots(g^{d_{i,t}})^{e_j^t} \pmod p$$

   (Remember g^{s_i} was already published as a public share of the server.)
   P_j also checks that the local secret of zero was correctly transferred from the unselected server $i' \notin SEL_j$:

   $$1 \stackrel{?}{=} (g^{d_{i',0}})(g^{d_{i',1}})^{e_l}(g^{d_{i',2}})^{e_l^2}\ldots(g^{d_{i',t}})^{e_l^t} \pmod p$$

4. If above VSS is verified, P_j computes its new share as $s_{j(,new)} = s_{j(old)} + s_{1j} + \ldots + s_{nj} \pmod q$.

After this protocol is complete, the new shares interpolate a polynomial that satisfies $f(e_j) = \sum_{i \in SEL_j} s_i$ and $f_i(e_l) = 0$ for $l=1,..,j-1,j+1,..,k$.

4.6. Message Encryption

In section 4.5, the *j*-th secret key is constructed as $x_j = \sum_{i \in SEL_j} s_i$ and set as the value of $f(e_j)$. The corresponding public key is publicly calculated from the VSS public values as: $Y_j = \prod_{i \in SEL_j} g^{s_i}$.

Thus the user can encrypt a message *M* with Y_j by using random $r \in Z_p$, $(A, B)=(g^r \bmod p, M*Y^r \bmod p)$.

4.7 Message Decryption

When decryption of the encrypted message (*A*, *B*) is necessary, the user (or a receiver of the encrypted message) requests *t+1* servers to reconstruct the secret. Only the first part of the encrypted message ($A=g^r$) needs to be attached to the request. If the servers agree to the request, they send the partial decryption to the user. They can be used only to decrypt the message containing g^r, and give no information on the secret maintained in the system. Desmedt and Frankel constructed a protocol for implementing a reusable shared secret [DF89] and we extended it to obtain partial decryption for the j-th secret: Any polynomial *f(x)* of degree *t* can be represented by using its *t+1* points: *f(i), i=1, .., t+1,* as

$$f(x) = \sum_{i=1}^{t+1} f(i) \prod_{j=1, j \ne i}^{t+1} \frac{x-j}{i-j}$$

Thus, if we represent server *i*'s modified shadow for the *j*-th secret as

$$a_i^{(j)} = s_i \prod_{k=1, k \ne i}^{t+1} \frac{e_j - k}{i-k}$$

then the *j*-th secret can be represented by:

$$x_j = f(e_j) = \sum_{i=1}^{t+1} a_i^{(j)}$$

Thus the partial decryption for *A* from server *i* is $A^{a_i^{(j)}} \bmod p$ and the user can reconstruct a message M as $M = \frac{B}{\prod_{i=1}^{t+1} A^{a_i^{(j)}}}$.

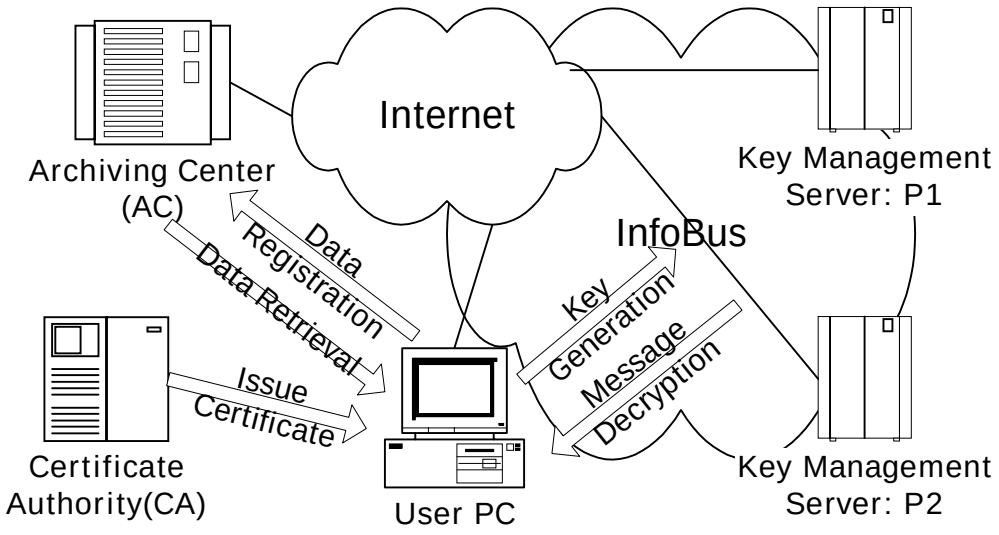

Figure 2. Internet Archiving Service Framework

5. Implementation

On the basis of the key management protocol, we designed an Internet-based secure data archiving service, in which the key recovery function is designed for individual users who lose the decryption key, and for the corporate users in cases where confidential data may only be accessed by authorized employees. We designed the system on top of current industry-standard Internet architectures, such as the public key infrastructure (PKI), Java, HTTP, and SSL/X.509, in order to fully utilize WWW tools such as Web servers that support SSL and Servlets, and Web browsers that support SSL and Applets.

5.1 Archiving Server Framework

An Internet archiving server framework is shown in figure 2. It consists of a certificate authority (CA), an archiving center (AC), multiple key management servers (KMS), and user's PC, all of which are connected to the Internet. The KMSs and the user's PC are also interconnected by the InfoBus Repeater Publish/Subscribe environment [InfoBus98] [MU98], which we will describe in section 5.2.

The function and trust models of each party are as follows:

- CA issues the user's and server's certificates, which are necessary for mutual authentication. CA guarantees the uniqueness of the user's name, which means that it does not issue certificates with the same name to other users. Thus, if it becomes necessary to reissue the certificate, the CA checks the identity of the owner by examining physical credentials such as the owner's driver's license. The certificate format is based on X.509.
- AC is a kind of file server that is responsible for maintaining the data only from the physical point of view. Since the data has already been encrypted at the user's site, AC does not need to maintain confidentiality.
- KMSs are key recovery agents that maintain the key-recovery key in a (t, n)-threshold scheme. As described in section 4, KMSs collectively generate the public key and maintain the corresponding secret key. They also provide a message decryption service.
- The user's PC runs Web browser which supports SSL client authentication. Every service is available only when the client certificate is valid. Message encryption with a key-recovery field is also performed at the user's PC, and the encrypted message is then sent to AC.

The data and the key require different maintenance policies, and therefore they are maintained by different servers: AC and KMSs.

5.2 Key Management Framework

Since communication among multiple servers is necessary for the key management protocol, it is inefficient to establish peer-to-peer connections one by one. Publish/subscribe is a programming model whereby a message sender does not need to specify the address of a receiver; instead, the sender publishes an event with a subject, and the receivers who are subscribing to the subject receive an asynchronous event.

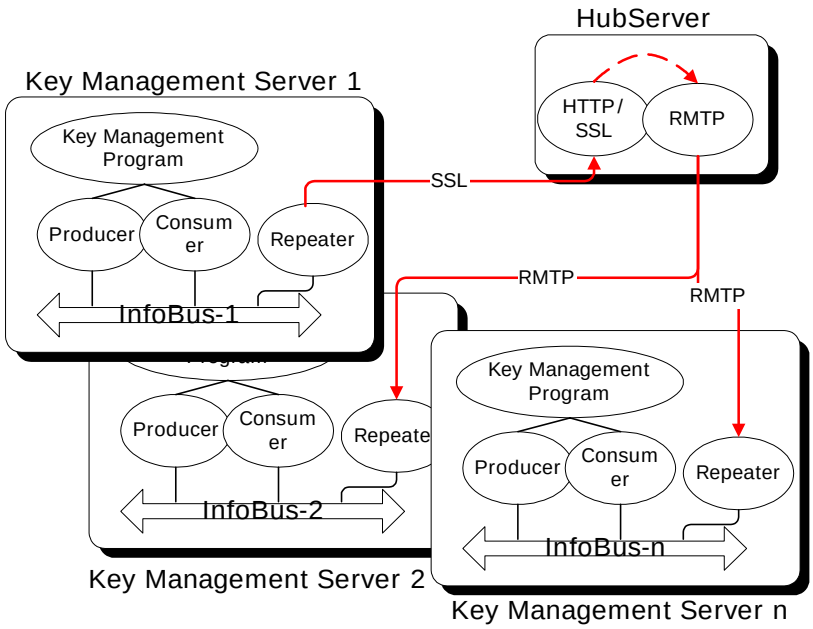

Figure 3. Key Management Framework

InfoBus Repeater [MU98] is a Java middleware program that support the publish/subscribe model. It is based on the InfoBus API [InfoBus98]. We designed our key management framework by using the InfoBus Repeater as shown in figure 3. The InfoBus API consists of three Java classes: Producer, Consumer, and InfoBus. When a producer (an instance of Producer) generates a new event, it notifies the bus (InfoBus), which in turn announces the event to every consumer (Consumer). Repeater is an additional class defined by [MU98] for connecting local InfoBuses over the Internet. This is a kind of gateway to the Internet: when an event occurs, Repeater sends it to a remote InfoBus. A hub server is also introduced for broadcasting events to other InfoBuses. For an uplink to send a message to the hub, an HTTP/SSL connection is used, and for a downlink to broadcast a message from the hab, the Reliable Multicast Transport Protocol [SSTYYK97] is used.

We now describe how messages are exchanged between the key management servers. The typical verifiable-secret-sharing (VSS) message from server *i* at time *t* is:

$$VSS_{i,t} = \left(i, t, \{\alpha_{ij}\}, \{e_{ij'}\}\right)_{SIG_i},$$

where $\alpha_{ij}, 1 \le i \le n$ is a secret share encrypted by the public key of server *j*, and $e_{ij}, 0 \le j' \le t$ are public values for VSS checking. The entire message is signed with the private key of server *i* to maintain consistency and non-repudiency. Thus, at in each update phase, each server broadcasts one message, and a total of *n* messages are exchanged.

The publish/subscribe model makes it easy for a user to join in the framework. Since a user can see e_{ij} public values broadcast by server *i*, he can choose them and construct a public key as a key-recovery key, as described in section 4.5 and 4.6.

6. Summary

We have described a secure and robust protocol for key registration based on the proactive secret-sharing scheme. Through combination of implicit key generation of an ElGamal-based encryption scheme with the share update protocol of proactive secret sharing, a public key can be determined through interaction with the user, while the corresponding secret key is implicitly maintained in a (*t*, *n*)-threshold scheme. We also enhanced the system to support (1) multiple-secret-sharing for multiple users and (2) reusability of secrets. These two features are very useful for actual operational systems because (1) enables some secrets to be revoked without affecting other secrets, and (2) ensures that the secrets themselves are never revealed even at decryption phase, which enhances the total security of the system.

We also presented an Internet-based system architecture for a data-archiving service. Since it uses multi-casting technology, our protocol can be easily implemented on HTTP, and the user can access the system via Web browser, which makes it very practical.

7. References

[BF97] Boneh, D. and Franklin, M., "Efficient Generation of RSA Keys," CRYPTO '97, 1997.

[BSCGV94] Blundo, C., De Santis, A., Di Grescenzo, G., Gaggia, A. G. and Vaccaro, U., "Multi-Secret Sharing Schemes," CRYPTO '94, LNCS 839, 1994.

[CGHN97] Canetti, R., Gennaro, R., Herzberg, A., and Naor, D., "Proactive Security: Long-term protection against break-ins," CRYPTOBYTES, RSADSA, 1997.

[CGNA85] Chor, B., Goldwasser, S., Micali, S. and Awerbuch, B, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults," 22th FOCS, IEEE, 1985.

[Denning96] Denning, E. D., "A Taxonomy for Key Escrow Encryption Systems," CACM, Vol.39, No.3, 1996.

[DF89] Desmedt, Y., and Frankel, Y., "Threshold Cryptosystems," CRYPTO '89, LNCS 435, 1989.

[DH76] Diffie, W., and Hellman, M., "New Directions in Cryptography," IEEE Trans. Information Theory, Vol. 22, 1976.

[Feldman89] Feldman, P., "A Practical Scheme for Non-interactive Verifiable Secret Sharing," IEEE FOCS '87, 1987.

[FY92] Franklin, M. and Yung, M., "Communication Complexity of Secure Computation," ACM STOC '92, 1992.

[HJKY95] Herzberg, A., Jarecki, S., Krawczyk, H. and Yung, M., "Proactive Secret Sharing or: How to Copy With Perpetual Leakage," CRYPTO '95, LNCS 963, 1995.

[InfoBus98] InfoBus 1.1.1 Specification, available from: http: www.javasoft.com/beans/infobus

[IS90] Ingemarsson, I. And Simmons, G. J., "A Protocol to Set up Shared Secret Schemes without the Assistance of a Mutually Trusted Party," EUROCRYPT '90, LNCS 473, 1990.

[JMOK95] Jackson, W-A., Martin, K. M. and O'Keefe, C. M., "Efficient Secret Sharing without a Mutually Trusted Authority," EUROCRYPT '95, LNCS 921, 1995.

[Maher96] Maher, D. P., "Crypto Backup and Key Escrow," CACM, Vol. 39, No. 3, 1996.

[MU98] Maruyama, H. and Uramoto, N., "InfoBus Repeater: A Java-Based Publish/Subscribe Middleware," IBM Research Report RT0245, 1998.

[OY91] Ostrovsky, R. and Yung, M., "How to Withstand Mobile Virus Attacks," 10th PDOC, ACM, 1991.

[Pedersen91] Pedersen, T. P., "A Threshold Cryptosystem without a Trusted Party," EUROCRYPT '91, LNCS 547, 1991.

[Ped91] Pedersen, T., "A Threshold Cryptosystem without a Trusted Party," EUROCRYPT '91, LNCS 547.

[Shamir79] Shamir, A., "How to Share a Secret," CACM Vol. 22, No. 11, 1979.

[VT97] Verheul, E.R., and van Tilborg, H. C. A., "Binding ElGamal: A Fraud-Detectable Alternative to Key-Escrow Proposals," EUROCRYPT '97, LNCS1233, 1997

[Walker96] Walder, S. T. et al, "Commercial Key Recovery", CACM, Vol. 39, No. 3, 1996.

[SSTYYK97] Shiroshita, T. et al., "Performance Evaluation of Reliable Multicast Transport Protocol for Large-scale Delivery," Proc. of IFIP 5th International Workshop on Protocol for High Speed Networks, 1996.