27

Security Architecture based on Secret Key and Privilege Attribute Certificates

Yoshiki Sameshima
Research & Development Department,
Hitachi Software Engineering Co., Ltd.
6-81, Onoe-cho, Naka-ku, Yokohama, 231 Japan
email: `same@ori.hitachi-sk.co.jp`

Abstract

An authentication server which employs the secret-key cryptography holds the secret keys of user clients and application servers in a local database, and this leads to attacks on the database, key propagation from a master server to slave servers and the management from a remote console. The situation of a privilege attribute server is same. In order to solve the problems the author introduces secret key certificate and privilege attribute certificate, which can be handled same as the public key certificate. The certificates can be used not only for authentication and privilege attribute services but also delegation of privilege and messaging system.

Keywords

network security, authentication, privilege attribute, delegation of privilege, certificate, secret-key cryptography

1 INTRODUCTION

In a typical distributed system today, the protection of resources is archived by user logon to a host and mutual trust among the hosts of the distributed system. However, the system is very vulnerable to network eavesdropping, replay attack, etc., since there is no protection of sensitive information transmitted through the network, such as user identifier, user password, file contents. As a result it is possible for an attacker to impersonate a user or to access resources of the user, etc.

Several security architectures, most of which provide authentication service, are proposed as solutions to the above problem, such as Kerberos (Kohl & Neuman, 1993), Secure European System for Application in a Multivendor Environment (SESAME) (Kaijser, Parker & Pinkas, 1994), Open Software Foundation's Distributed Computing Environment (OSF/DCE) (Rosenberry, Kenney and Fisher, 1993) and X.509 Authentication Framework (ISO 1988). The systems employ cryptographic techniques and the management of cryptographic keys is the most important technical and operational matters as well as the protocol which realizes the security service. All such systems need central servers or authorities that all principals in the system trust, such as Authentication Server (AS) of Kerberos and DCE, Certification Authority (CA) of X.509 Public Key Cryptography System (PKCS) Authentication Framework.

The X.509 CA issues a public key certificate which includes a public key of a user or another CA and its owner name along with the digital signature generated with the private key of the issuer CA. Once the public key certificate is issued, a principal needs the public key certificate of a peer principal and the public key of the issuer CA in order to verify the certificate and to get the correct public key of the peer principal. The CA itself does not need to be connected to the distributed system, because the signing process can be done in a off-line manner, and the issued certificates, which is protected with the digital signature of the CA, can be stored in a public database such as a directory and freely distributed from the database. As a consequence the off-line CA and its management are not targets of direct attack through the network.

The situation of a server which employs the secret-key cryptography is very different; because the AS holds all secret keys of application servers and user clients in a local database, the database and its management, which is supported by a special protocol between the server and a remote management console, are targets of attack. In addition, for the purpose of high availability and performance of the authentication service, the AS is duplicated and the key propagation from a master server to slave servers is another target of attack.

While the AS of Kerberos and OSF/DCE distribute tickets used for authentication, Privilege Attribute Server (PAS) and Privilege Server (PS) distribute authorization information and they are vital for the distributed environment. After a user logons to the network and the user's client gets a ticket for the PAS or the PS, the client requests Privilege Attribute Certificate (PAC) or Privilege Ticket Granting Ticket (PTGT) containing privilege information of the user, such as authenticated user name, group memberships, role, capabilities, clearance. The information is presented to an application server, and is used to decide whether access of the client to the target system or an object in the system is permitted or rejected. The situation of the PAS/PS is exactly same as the AS; in practice the servers come packaged with the AS and the servers are targets of the attack as well as the management from the remote console and the privilege information propagation from a master to slaves.

The author introduces two kinds of certificates as a solution of the problems; a secret key certificate includes a user's secret key, and a user PAC encloses a user's privilege. Since the certificates are protected with cryptographic techniques and the handling of the certificate is same as one of the public key certificate, the local databases which holds the secret keys and the privilege attributes is taken away from the servers, and as a result the attacks mentioned above are avoidable.

In the paper, the author introduces the two certificates and shows how they are applied to the services of authentication, privilege attribute, delegation of privilege in a distributed environment as well as to a messaging system.

First Section 2 shows the notations appeared in the paper, Section 3 introduces the two certificates and explains how they are used. Next Section 4, 5 and 6 describe application of the certificates to authentication and privilege attribute, delegation of privilege and messaging system respectively, and the advantages of the application are presented.

Table 1 notation

notation	description
S	AS, PAS (in Section 3) or application server
K_X	secret key of principal X
$\{I\}_K$	information I encrypted with key K
$\{I\}^K$	information I sealed with key K
P	privilege attribute
L	control information of a certificate
C	user client (in Section 4, 5)
AS	Authentication Server
PAS	Privilege Attribute Server
$K_{X,Y}$	session key used between principals X and Y
$T^K_{X,Y}$	ticket used between principals X and Y encrypted with K
l	validity time of a ticket
ts	time stamp of a ticket
$KeyCert_{S,X}$	key certificate of principal X issued from S
$PrivCert_{S,X,P}$	PAC of P of principal X issued from S
ST	service type
I	intermediate
R	restriction on a privilege (in Section 5)
O	originator
R	recipient (in Section 6)
C	context (in Section 6)
MSG	message
h	hash (message digest) of MSG
k	data encryption key of MSG

Implementation status is described briefly in Section 7 followed by short discussion on problems of the implementation in Section 8, and finally section 9 concludes the paper.

2 NOTATIONS

Table 1 shows the notations used in the paper in the order of appearance.

3 SECRET KEY AND PRIVILEGE ATTRIBUTE CERTIFICATES

A secret key certificate (Davis & Swick, 1990) includes a user's secret key encrypted with the secret key of the AS or the PAS; the structure is similar to one of the public key certificate defined in the X.509 Authentication Framework (ISO 1988). The secret key certificate of a principal, X, which secret key, K_X, is encrypted with the key of the AS or the PAS, S, is written as $\{X, \{K_X\}_{K_S}, L\}^{K_S}$ where $\{I\}^K$ denotes information, I, sealed with key, K, and $\{I\}_K$ stands for I encrypted with K; the key certificate includes the principal name, the secret key of the principal encrypted with the server's key and control information such as a serial number, a validity time, which is abbreviated to L. When the server uses the certificate, first the server verifies the seal, secondly checks the validity time and confirms that the certificate is not revoked by comparing the serial number against a revocation list (ISO 1988), and finally decrypts $\{K_X\}_{K_S}$ with the server's key, K_S, to get the authentic secret key of X.

A PAC includes an attribute representing a privilege of a user or a service type of an application server, which is used mainly for authorization in application level (Kaijser, Parker & Pinkas, 1993). The format of the certificate is almost same as one of the key certificate: $\{X, P, L\}^{K_S}$ which certifies that X has a privilege attribute, P, such as a role, a title, a group to which the user belongs. With verification of the seal, check of the validity time and the serial number, the server can confirm that P is given to X.

Since the certificates are protected with the seal of the server and additionally the secret key is encrypted with the server's key, any modification of the certificate can be detected, no principal except the server can retrieve the secret key. As a result the certificates can be stored and distributed freely; they might be stored in a public database such as a directory or a local cache of a client or an application server. The certificates can be issued in a off-line manner and neither the AS nor the PAS needs to hold the local key/privilege database. As a consequence the attack on the database and its management is avoided, and the management cost can be reduced.

In the following, the application of the certificates to services of authentication, privilege attribute, delegation of privilege and a messaging system is described as well as its benefits.

4 AUTHENTICATION AND PRIVILEGE ATTRIBUTE SERVICE

4.1 Basic Protocol

The combination of the two certificates can be employed for authentication and privilege attribute services; the following protocol illustrates that first a client, C, holding a secret key, K_C, requests the AS, AS to get a session key and tickets for the PAS, PAS, next requests PAS to get a session key and a ticket for an application server, S, and finally accesses S for mutual authentication and passing a privilege attribute, P. The structure of the ticket, $T^K_{X,Y}$, is almost same as one of the Kerberos: $\{X, Y, K_{X,Y}, l, ts\}_K$ where l is a validity time and ts is a time stamp; the principal which uses the ticket needs to check the validity time against the current time. The certificates $\{X, \{K_X\}_{K_S}, L\}^{K_S}$, $\{X, P, L\}^{K_S}$ are abbreviated as $KeyCert_{S,X}$ and $PrivCert_{S,X,P}$ respectively for readability.

1. **authentication request**: C requests AS to get a ticket for PAS. The difference against Kerberos is that C sends the request with the key certificates of C and PAS.

   $C \rightarrow AS$: C, PAS, $KeyCert_{AS,C}$, $KeyCert_{AS,PAS}$

2. **authentication reply**: AS verifies the key certificates to extract K_C and K_{PAS}, and generates a session key, $K_{C,PAS}$, used between C and PAS. Then AS creates tickets

for C and PAS including the client name, the PAS name, the generated session key, a validity time and a time stamp, and encrypts them with K_C and K_{PAS} respectively, and sends them back to C.

$AS \rightarrow C$: $T_{C,PAS}^{K_C}$, $T_{C,PAS}^{K_{PAS}}$

Notice that AS does not hold the secret keys of C nor PAS; it gets them from the key certificates send from C with the request.

3. **privilege request**: C recovers $K_{C,PAS}$ from $T_{C,PAS}^{K_C}$ with its secret key and creates a privilege attribute request including an application server name, S, a privilege attribute used for accessing S, a validity time of the privilege attribute and the time stamp included in the ticket, and then encrypts it with $K_{C,PAS}$. The encrypted request is send with the ticket for PAS, the PAC of P and the key certificate of S.

   $C \rightarrow PAS$: $T_{C,PAS}^{K_{PAS}}$, $\{S, P, l, ts\}_{K_{C,PAS}}$, $PrivCert_{PAS,C,P}$, $KeyCert_{PAS,S}$

4. **privilege reply**: PAS recovers $K_{C,PAS}$ from $T_{C,PAS}^{K_{PAS}}$, decrypts the request with it, checks the time stamp in the request against one in the ticket, and regards the accessing client as C that is requesting a ticket for S. Verifying the PAC PAS confirms that C has P and its validity time, and then creates tickets for C and S including the client name, the server name, a newly generated session key, a validity time which might be shorter than the requested one and a time stamp, and encrypts them with $K_{C,PAS}$ and K_S got from the key certificate of S. PAS also generates a temporary PAC for S which certifies that C has P, and sends it with the tickets.

   $PAS \rightarrow C$: $T_{C,S}^{K_{C,PAS}}$, $T_{C,S}^{K_S}$, $\{C, P, l\}_{K_S}$

   Note that PAS does not have the privilege of the client; it gets the information from the PAC send from the client.

5. **server request**: The client recovers $K_{C,S}$ from $T_{C,S}^{K_{C,PAS}}$, makes the request including the server name, a validity time and the time stamp in the ticket, encrypts it with the session key, and sends S it with the ticket for S and the temporary PAC for S.

   $C \rightarrow S$: $T_{C,S}^{K_S}$, $\{S, l, ts\}_{K_{C,S}}$, $\{C, P, l\}_{K_S}$

6. **server reply**: S gets the session key from the ticket, decrypts the temporary PAC and the request, and then authenticates that the accessing principal is C which has P. The server sends back the incremented time stamp encrypted with the session key.

   $S \rightarrow C$: $\{ts + 1\}_{K_{C,S}}$

7. **mutual authentication**: C decrypts the reply, checks the time stamp, which completes mutual authentication and sharing the session key between C and S.

Comparing with Kerberos, SESAME and DCE the major difference is that neither the AS nor the PAS holds the secret key information and the privilege attribute information of the clients; the information is included in the secret key certificates and the PACs, and the servers can get the information by verifying the certificates with its secret key. In the above protocol a *push* model (Kaijser, Parker & Pinkas, 1993) is used; the client *pushes* certificates with a request. It is also possible that the server *pulls* the certificates from a public database such as a directory or a local cache which is not necessary to be secure.

4.2 Extended Protocol

In the basic protocol the client specifies the server explicitly, however in some situations like host anycasting service (Partridge, Mendez & Milliken, 1993), the client might specify only a service type, ST, not a server name. The following extended protocol supports such kind of demand.

1. **privilege request**: It is assumed that C has finished the authentication procedure against AS and has a ticket for PAS. C creates the privilege request for ST same as the basic protocol except not sending the key certificate of an application server.

   $C \rightarrow PAS$: $T_{C,PAS}^{K_{PAS}}$, $\{ST, P, l, ts\}_{K_{C,PAS}}$, $PrivCert_{PAS,C,P}$

2. **privilege reply**: PAS generates a session key for C and ST, and makes tickets and encrypts them with $K_{C,PAS}$ and a newly generated session key, $K_{ST,PAS}$. PAS also generates a temporary PAC which certifies that C has P, seals it with $K_{ST,PAS}$, and creates a ticket including $K_{ST,PAS}$ for itself. The three tickets and the temporary PAC are returned to C.

   $PAS \rightarrow C$: $T_{C,ST}^{K_{C,PAS}}$, $T_{C,ST}^{K_{ST,PAS}}$, $T_{ST,PAS}^{K_{PAS}}$, $\{C, P, l\}_{K_{ST,PAS}}$

3. **service request**: The client makes a service request to an application server which supports ST or anycasts the request with the service type; the request is same as the basic protocol except sending $T_{ST,PAS}^{K_{PAS}}$, $T_{C,PAS}^{K_{PAS}}$ and the service type.

   $C \rightarrow S$: $T_{C,ST}^{K_{ST,PAS}}$, $\{ST, l, ts\}_{K_{C,ST}}$, $\{C, P, l\}_{K_{ST,PAS}}$, $T_{ST,PAS}^{K_{PAS}}$, $T_{C,PAS}^{K_{PAS}}$, ST

4. **certificate request**: A server, S, which receives the request sends a certificate request to PAS using $K_{S,PAS}$ got through an authentication process, with the PAC indicating that S supports the service type. S sends also all got from the client except the service type.

   $S \rightarrow PAS$: $T_{S,PAS}^{K_{PAS}}$, $\{PAS, l, ts\}_{K_{S,PAS}}$, $PrivCert_{PAS,S,ST}$,
   $T_{C,ST}^{K_{ST,PAS}}$, $\{ST, l, ts\}_{K_{C,ST}}$, $\{C, P, l\}_{K_{ST,PAS}}$, $T_{ST,PAS}^{K_{PAS}}$, $T_{C,PAS}^{K_{PAS}}$

5. **certificate reply**: PAS gets $K_{ST,PAS}$ from $T_{ST,PAS}^{K_{PAS}}$, decrypts $T_{C,ST}^{K_{ST,PAS}}$ with the key to gets $K_{C,ST}$, and gets to know that C is accessing an application server which supports a service which type is ST, and the session key is $K_{C,ST}$. Verifying the PAC of S and the temporary PAC, PAS confirms that S supports ST and C has P, and then makes a reply including the accessing client, its privilege, a new session key and its validity time as well as the time stamp included in the request from C. PAS sends back the reply encrypted with $K_{S,PAS}$, and a ticket including the new session key encrypted with the session key between C and PAS.

   $PAS \rightarrow S$: $T_{C,S}^{K_{C,PAS}}$, $\{C, P, K_{C,S}, l, ts\}_{K_{S,PAS}}$

6. **service reply**: S decrypts the reply to verify the accessing principal and its privilege, and sends back a service reply with the new ticket.

   $S \rightarrow C$: $T_{C,S}^{K_{C,PAS}}$, $\{ts + 1\}_{K_{C,S}}$

7. **mutual authentication** C decrypts the new ticket with $K_{C,PAS}$, gets a new session key, $K_{C,S}$, with its validity time, and authenticates the accessing server by processing $\{ts + 1\}_{K_{C,S}}$. In this stage C and S have finished mutual authentication and shares the session key. S also verifies the privilege of C as well as C verifies the service type of S.

Comparing Kerberos, SESAME and DCE, an important difference is that the client is accessing to a service, not a server in the first stage. Any server that supports the service can reply the service request, however in the final stage, the client authenticates the server. A typical examples of services which get the benefit of the protocol are: Network Information Service (Stern, 1991) which multiple servers provide information on user, host address, etc., an anycast type service which is supported by multiple servers, or a task broking service which allocates a requested task to a host in view of load balance, resource, etc.

5 DELEGATION OF PRIVILEGE

Delegation of privilege appears when an initiator asks an intermediate to act for the first principal. A typical example is a printing service; a user requests a printer server to print a file with the file name not the file contents, and then the printer server reads a requested file on a file server and prints it. In this case the intermediate needs the permission to read the file, which the intermediate does not have normally and should be delegated from the initiator.

Recent years much attention is paid in the delegation mechanism. In the Distributed System Security Architecture (DSSA) (Gasser & McDermott, 1990) an initiator generates and signs a certificate to allow the intermediate to act on the initiator's behalf. The SESAME architecture has adopted the PAC chaining method where the chained PACs, which represent the delegated privileges, are included in the PAC of the intermediate.

A similar method, which is based on proxy, is proposed by Neuman (1993); a proxy is a certificate that allows the intermediate which has the proxy key to operate with the privilege of the initiator that granted the proxy.

The following protocol illustrates that an initiator, C, asks an intermediate, I, a job with a capability containing a privilege attribute, P, and a restriction, R, on P. I passes the capability to PAS, gets a new capability, a session key and a ticket for a final target, S, and requests S the service. Next S asks PAS to check the capability and gets back the checking result. The restriction on the privilege in the capability, such as *read-only* access to a file, is specified to make sure the intermediate cannot use the capability for other purposes.

1. **delegate request**: C asks PAS to create a capability; the request includes the intermediate name, the privilege and the restriction, and is encrypted with $K_{C,PAS}$. The initiator sends also the key certificate of I and a PAC to proof that the initiator has the privilege.

   $C \rightarrow PAS$: $T_{C,PAS}^{K_{PAS}}$, $\{I, P, R, l, ts\}_{K_{C,PAS}}$, $PrivCert_{PAS,C,P}$, $KeyCert_{PAS,I}$

2. **delegate reply**: PAS decrypts the request with the session key in $T_{C,PAS}^{K_{PAS}}$ and gets to know that C is requesting delegation of P with R to I and its validity time is l. With verification of the PAC PAS confirms that the requesting client has the privilege and issues a capability including the intermediate name, the privilege, the restriction, the validity time and the initiator name, $\{C\}$, which is used for accounting and audit. The created capability is encrypted with the key of PAS and sends back with tickets used between C and I.

   $PAS \rightarrow C$: $T_{C,I}^{K_{C,PAS}}$, $T_{C,I}^{K_I}$, $\{I, P, R, l, \{C\}\}_{K_{PAS}}$

3. **service request**: C sends I a request as well as the capability, the ticket for I, P and R.

   $C \rightarrow I$: $T_{C,I}^{K_I}$, $\{I, l, ts\}_{K_{C,I}}$, $\{I, P, R, l, \{C\}\}_{K_{PAS}}$, P, R

4. **service reply**: I sends back a reply for mutual authentication.

   $I \rightarrow C$: $\{ts + 1\}_{K_{C,I}}$

5. **privilege request**: I sends PAS a privilege request with the capability and the key certificate of the final target, S; the request includes S, the privilege, the restriction, the validity time and the time stamp in the ticket, and is encrypted with the session key between I and PAS got through an authentication process with AS.

$I \rightarrow PAS$: $T_{I,PAS}^{K_{PAS}}$, $\{S, P, R, l, ts\}_{K_{I,PAS}}$, $\{I, P, R, l, \{C\}\}_{K_{PAS}}$, $KeyCert_{PAS,S}$

6. **privilege reply**: PAS decrypts the request with the session key in $T_{I,PAS}^{K_{PAS}}$ and verifies the capability to get to know that I is requesting P for S, which is delegated from C. PAS replies the request with tickets used between I and S and a new capability including P, R, l, $\{C, I\}$ encrypted with the server's key; the last component, $\{C, I\}$, is used for audit or accounting, which indicates that the privilege is originated from C through I.

   $PAS \rightarrow I$: $T_{I,S}^{K_{I,PAS}}$, $T_{I,S}^{K_S}$, $\{S, P, R, l, \{C, I\}\}_{K_S}$

7. **service request**: I sends a service request with the new capability.

   $I \rightarrow S$: $T_{I,S}^{K_S}$, $\{S, l, ts\}_{K_{I,S}}$, $\{S, P, R, l, \{C, I\}\}_{K_S}$

8. **service reply**: S sends back a reply for mutual authentication.

   $S \rightarrow I$: $\{ts + 1\}_{K_{I,S}}$

As well as the case of the authentication service in the previous section, the privilege server does not need to hold the privilege attributes of the client and the secret keys of the client, the intermediate and the server. It is also possible to extend the protocol to specify only the types of the intermediate or the target group, not the name of them explicitly. Note that the delegation can be extended to multiple intermediates; in that case the capability has the following form: $\{S, P, R, l, \{C, I_1, I_2, \ldots\}\}_{K_{PAS}}$.

6 PRIVACY ENHANCED MESSAGE SYSTEM

The previous sections show that the key certificate and the PAC can implement authentication, privilege attribute and delegation services, however, the certificates can also be employed by an upper layer. There are two advantages of the application of the certificates to a messaging system; one is authentication of the originator's privilege attribute and sending context, and the other is recipient particularization with her / his privilege attribute and receiving context.

6.1 Privilege and Context Authentication

The first advantage of the application is the authentication of the originator's privilege attribute and sending context information; not only the originator's identifier such as personal name, but also its role, title and context information such as sending time, network location or type of the message can be authenticated by the recipient.

The following protocol shows how the advantage is realized; an originator, O, having a privilege, P, asks PAS to certify the privilege and sending context, C, with the hash of a message, MSG, for authentication of message origin and integrity check of the message, and a recipient, R, verifies the fact:

1. **authentication request**: O creates MSG and calculates its hash, h. The hash with P and C to be authenticated is encrypted with the originator's key and send to PAS with the key certificate and the PAC of O. The context C is either a context type such as time, network location, or a context attribute (type and value) such as message type, message identifier.

   $O \rightarrow PAS$: $\{h, P, C\}_{K_O}$, $KeyCert_{PAS,O}$, $PrivCert_{PAS,O,P}$

2. **authentication reply**: PAS decrypts the request with the originator's key got from the key certificate, and gets the content that the originator is requesting to authenticate P and C. Verifying the PAC PAS can confirm that O has P and creates authentication information including the message hash, the originator, the verified privilege attribute and the context, and encrypts it with PAS's key. The process of C depends on its type. When C in the request is a type, C in the reply is an attribute consisting the type and a value; a typical example is that the request is *time* and the result is *time = 12.34.34 9th June 1994*. In the case that C in the request is an attribute, C in the reply is the attribute itself; a typical example is *messageType = PostScript*.

   $PAS \rightarrow O$: $\{h, O, P, C\}_{K_{PAS}}$

3. **message sending**: The originator sends MSG with the authentication information.

   $O \rightarrow R$: MSG, $\{h, O, P, C\}_{K_{PAS}}$

4. **certification request**: The recipient, R, sends the authentication information with the key certificate of R.

   $R \rightarrow PAS$: $\{h, O, P, C\}_{K_{PAS}}$, $KeyCert_{PAS,R}$

5. **certification reply**: PAS decrypts the request with its key, encrypts it again with the recipient's key got from the key certificate, and sends it back to R.

   $PAS \rightarrow R$: $\{h, O, P, C\}_{K_R}$

6. **message verification**: R calculates the hash of MSG and compares it with h got by decrypting the reply. If the two hash values are same, R can make sure the integrity of MSG, the message originator, the originator's privilege, and the sending context.

With this protocol the recipient can verify not only the source and the integrity of message but also the originator's privilege such as role and the context such as sending time, network location, message type.

6.2 Recipient Particularization with Privilege and Context

The second advantage of the application of the certificates to the messaging system is that the originator can particularize the recipient not with the recipient identifier but by specifying the privilege attribute of the recipient and the receiving context such as permitted decryption time.

1. **message sending**: O creates MSG, generates randomly a data encryption key, k, and sends the encrypted message, the data encryption key (DEK) information including k, recipient's privilege attribute, P, and receiving context, C, encrypted with the originator's key to the recipient. O also sends P, C to let the recipient know the receiving condition as well as the key certificate of O.

   $O \rightarrow R$: $\{MSG\}_k$, $\{k, P, C\}_{K_O}$, P, C, $KeyCert_{PAS,O}$

2. **decryption request**: R sends PAS the DEK information, the originator's key certificate, the PAC corresponding to the receiving condition and the key certificate of itself.

   $R \rightarrow PAS$: $\{k, P, C\}_{K_O}$, $KeyCert_{PAS,O}$, $PrivCert_{PAS,R,P}$, $KeyCert_{PAS,R}$

3. **decryption reply**: The server decrypts the DEK information with the originator's key retrieved from the key certificate of O, and gets k with the receiving context. Verifying the recipient's PAC and the current context such as time, the server checks whether the condition is satisfied or not. If the condition is satisfied, the server encrypts k with the recipient's key got from the key certificate of R and sends it back to R.

   $PAS \rightarrow R$: $\{k\}_{K_R}$

4. **decryption process**: The recipient recovers k and decrypts the encrypted message to get MSG.

A typical usage of the messaging system is that a secretary reads messages send to the boss during the holiday of the boss with a certificate claiming that the secretary has the privilege of the boss during the holiday. A care must be paid; a normal PAC admits full privilege and a restriction must be added to the use of the attribute, such that the privilege is only granted in case of decryption of the DEK information, not authentication service of privilege.

7 IMPLEMENTATION

Currently the following systems have been implemented:

- An authentication and privilege attribute server and a client library that support security services introduced in Section 4. The system supports very basic functions and has not integrated with an application program interface such as Berkeley socket interface, remote procedure call nor GSSAPI (Linn, 1993).
- A privacy enhanced message described in section 6 has been realized on a PC environment and integrated with an existing mail system.

8 PROBLEMS AND FUTURE WORKS

8.1 Multiple Domains and Servers

The current implementation supports only single domain (cell) and server. In order to support the inter-domain service each pair of servers of different domains needs to share a same secret key and the key is used to creat a ticket that a client can present to the foreign AS (Rosenberry, Kenney and Fisher, 1993). Another approach is introduction of an inter-domain service; the SESAME architecture provides an inter-domain server which can verify seals of PACs of other domains and if necessary translates privileges and restrictions to the local representation.

The author does not have a new idea and is extending the message system which employs the DCE model where two servers share a key which might be included in a key certificate. This approach will work up to middle scale network, say tens of domains, however, the introduction of the public-key cryptography is inevitable for a large scale network such as the Internet.

8.2 Protection of Server's Key from Attack

The key of AS and PAS is used to seal the certificates and to encrypt the users' and application servers' secret keys, and this leads to known-plaintext attack or chosen-plaintext attack. The situation is same in the case of the public-key certificate, however, it is believed that the public-key cryptography is stronger than the secret-key cryptography. One of solutions is to employ strong encryption algorithms with longer size key such as Triple-DES (Kaliski, 1993). Another is introduction of work key; randomly generated work key is used to encrypt a user key in a key certificate and the seals of the certificates, and the work key is encrypted with the server's key. This method protects the server's key form the direct chosen-plaintext attack, however, the verification cost of a certificate increases.

8.3 Audit

The current implementation records which principal requested to certify what attribute and context, who requested a ticket for which server with how long validity time, and so on. The author has not yet enough experience to assure that these logs are sufficient for audit and accounting, and it needs to investigate audit and accounting services in the certificate-based security framework.

9 CONCLUSION

The author has described the secret key certificates and the PACs that solve the problems on the storage and the propagation of secret keys and privilege attributes of user clients and application servers in the current authentication and privilege attribute system, and how the certificates are useful for authentication, privilege attribute, delegation of privilege in a distributed system as well as a messaging system. It has also been presented that a client can specify a service type instead of specifying a server name explicitly as well as authentication of privilege attribute of originator and context, and recipient particularization in the messaging system.

The author has not yet considered how this work can be integrated with the OSF/DCE facilities, however, he believes his protocols should be included in a future release of DCE.

REFERENCES

Davis, D. and Swick, R. (1990) Network Security via Private-Key Certificates. *Operating Systems Review.* 24, 4, 64-7.

Gasser, M. and McDermott, E. (1990) An Architecture for Practical Delegation in a Distribution System. *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy.* Oakland, California, United States of America.

International Standards Organization. (1988) *Information processing – Open Systems Interconnection – The Directory – Authentication Framework. IS-9594.* International Standards Organization, Geneva Switzerland.

Kaijser, P., Parker, T. and Pinkas, D. (1994) *SESAME: The solution to security for open distributed systems.* Computer Communications, 17, 7, 501-18.

Kaliski, B. (1993) *Triple-DES: A Brief Report.* Technical report, RSA Laboratories.

Kohl, J. and Neuman, B. (1993) *The Kerberos Network Authentication Service (V5).* Internet Requests for Comments 1510.

Linn, J. (1993) *Generic Security Service Application Program Interface.* Internet Request for Comments 1508.

Neuman, C. (1993) *Proxy-Based Authorization and Accounting for Distributed Systems.* Proceedings of the 13th International Conference on Distributed Computing Systems. Pittsburgh, Pennsylvania, United States of America.

Partridge, C., Mendez, T. and Milliken, W. (1993) *Host Anycasting Service.* Internet Requests for Comments 1546.

Rosenberry, W., Kenney, D. and Fisher, G. (1993) *Understanding DCE.* O'Reilly & Associates, Inc., California, United States of America.

Stern, H. (1991) *Managing NFS and NIS.* O'Reilly & Associates, Inc., United States of America.