A Flexible Distributed Authorization Protocol

Jonathan T. Trostle
CyberSAFE Corporation

B. Clifford Neuman
Information Sciences Institute
University of Southern California

Abstract

While there has been considerable effort in creating a single sign-on solution for interoperability among authentication methods, such interoperability across authorization methods has received little attention. This paper presents a flexible distributed authorization protocol that provides the full generality of restricted proxies while supporting the functionality of and interoperability with existing authorization models including OSF DCE and SESAME V2. Our authorization protocol includes a delegation method that is well suited for certain electronic commerce applications.

1 Introduction

Authentication and authorization are essential functions for the security of distributed systems. Authentication protocols have been studied at great length; this paper concentrates on distributed authorization and describes a protocol that can be applied to address a variety of distributed authorization problems. Such protocols must be flexible in order to satisfy the needs of a diverse set of applications.

For the purposes of the paper, we define a client to be a process that makes requests to another process called a server. The server is either a security server or an application server. An application server can also become a client if it makes requests to another application server; these additional requests can occur during delegation.

1.1 Description of Problem

Authentication is the means of verifying the identity of a user or process; authorization is the means of determining whether a user or process is permitted to perform a particular operation. More specifically, the client presents credentials and a request for an operation on an object to an application server, and the application server passes this data as arguments to an access control function. The access control function returns a binary output; either the requested operation is granted or denied.

Currently, most applications make authorization decisions using application specific access control functions in conjunction with local files. The advantage of a general approach to the problem is that administrative tools can be created that ease the management of authorization information. Also, authorization information can be more easily shared in a distributed environment. For example, a group, or privilege server, can certify whether principals are members of certain groups. In short, the more general approach provides developers with an infrastructure for integrating authorization with applications and managing authorization information.

A distributed authorization infrastructure should allow principals to securely delegate rights to other principals. Secure delegation involves a temporary transfer of rights from a requesting principal to a second principal called the delegate. As a result of the interaction with the requesting principal, the delegate may itself delegate rights to a third principal. The third principal is also a delegate of the first principal. This process continues until one of the delegates makes a request to a final target server which does not itself delegate any rights to another principal. The resulting chain is called a delegation chain. The initial requester in the chain is called the initiator. Requesters may limit the access rights of their delegates with restrictions that are called delegate restrictions.

A common example of distributed delegation consists of a client C, a print server P, and a file server F. The client desires to give the print server temporary read access to a file X so that the print server can download X from the file server and print it (Figure 1). Currently, this authorization problem might be solved by giving the print server permanent read access to all files on a particular file server; this solution is insecure since a compromise of the print server gives an attacker read access to all files on the file server.

We now consider another distributed authorization problem that can be viewed as a delegation problem. We have two corporations, A and B. Executives in Corporation A desire to grant certain individuals in Corporation B temporary read access to Corporation A sensitive documents. Furthermore, it is not known ahead of time exactly which documents are to be viewed (depending on negotiations, some documents may or may not be viewed). One solution is to have a system administrator modify the access control lists (ACL's) for each document as needed; this solution is unsatisfactory since it requires too much effort on the part of the system administrator. In addition, there is the possibility that the system administrator will make a mistake and leave an access control list entry in the wrong state. We discuss solutions to these problems in the remaining sections.

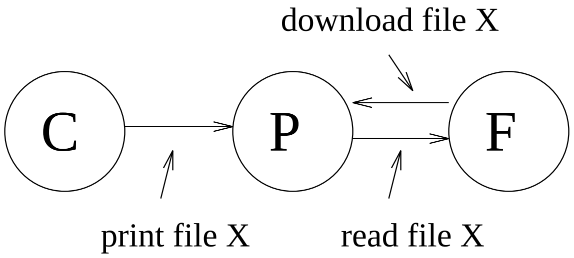


Figure 1: Distributed Delegation Problem

1.2 Existing Solutions

We now briefly examine two implemented solutions to the distributed authorization problem. These solutions are OSF DCE [2, 10] and SESAME V2 [6]. Both solutions use Kerberos V5 [5, 9] as the underlying authentication protocol. The OSF DCE privilege server supports some of the functionality proposed for Kerberos proxies as described in [8]. The SESAME V2 solution uses a variant of Kerberos V5 that has been extended to use public key cryptography for interrealm authentication.

OSF DCE uses a privilege server to issue privilege attribute certificates (PAC's) that list the principal identity and groups of the principal to which the PAC was issued. The PAC allows users to acquire different network identity roles (a user can separate a system administrator role from another role); to assume a new role, the user obtains a new PAC from the privilege server. OSF DCE also provides a distributed delegation method [2]; in Section 4 we will discuss the strengths and limitations of DCE delegation with respect to the examples above.

The flexible authorization protocol introduced in this paper includes the proxy delegation method which is based on the proxy concept in [8]. The proxy delegation method has advantages over DCE delegation with respect to the particular examples above. It is also a preferable method for certain electronic commerce applications such as NetCheque [7]; for NetCheque, roughly half as many messages are required using proxy delegation versus OSF DCE delegation.

SESAME V2 also uses a privilege server to issue PAC's; in SESAME V2, the PAC is signed by the privilege server using asymmetric cryptography. Also, delegate restrictions can be included in an extended PAC (EPAC); delegate restrictions are activated by passing values that can be used as input to a one way function to obtain values visible in the EPAC (the CV/PV method). Although potentially fewer messages are needed than for DCE when making subsequent delegated requests (since DCE requires an exchange with the privilege server to chain a new requester's identity with the intermediate server), the main drawbacks of this method are the CPU performance penalty associated with public key cryptography and the potential difficulties in administering and implementing the CV/PV method.

Both OSF DCE and SESAME V2 tend toward a static view of delegate restrictions; the performance of these two methods is likely to suffer when delegate restrictions change frequently. In Section 4, we describe how the proxy delegation method solves this problem.

1.3 Overview

This paper describes a flexible distributed authorization protocol (the flexible authorization protocol). The protocol allows the strengths of several authorization models to be combined, while accommodating existing (OSF DCE and SESAME V2) authorization systems. Though the flexible authorization protocol we describe can be implemented over a variety of mechanisms, including both symmetric and public key mechanisms, this paper describes our implementation using Kerberos V5. Version 5 of Kerberos [5, 9] is used widely for authentication and key management for distributed systems, and it provides a natural foundation upon which to build an authorization service. The extensible authorization data fields in Kerberos tickets and authenticators can be used to carry authorization information.

The next section provides an overview of the functionality that exists in our flexible authorization protocol. In Section 3, we discuss how initial security credentials (PAC's and TGT's) are obtained in OSF DCE and SESAME V2, and we show how our solution requires fewer messages. Our flexible authorization protocol also allows finer grained privilege attributes, since access control entries can be placed in the PAC. Section 4 describes the OSF DCE and SESAME V2 solutions to the two delegation problems presented earlier in this section. We present our proxy delegation method which has some advantages with respect to these examples. We then describe the NetCheque application [7] to demonstrate the benefits of the proxy delegation method for electronic commerce applications. We note that some applications can use the DCE delegation method and then switch to proxy delegation for subsequent requests in order to improve performance. In Section 5, we discuss related work in this area. Section 6 is a summary.

2 Flexible Authorization Protocol Functionality

In this section, we briefly describe some of the main features of our flexible authorization protocol.

2.1 PAC's and Privilege Servers

Both OSF DCE and SESAME V2 use a trusted privilege server that issues privilege attribute certificates (PAC's) at login time. For the same reasons, to allow users to maintain different roles and to implement the principle of least privilege, we use a privilege server that issues PAC's. The privilege server supports the concept of least privilege since principals can obtain PAC's with only the privilege attributes that are needed to accomplish the tasks for the current session. If such a principal is compromised, the attacker is more limited with respect to potential actions.

The PAC is a client selected list of groups for which the client principal is a member. By contacting the privilege server which issues the PAC's, the client can obtain additional PAC's if the client wants to change its role.

We have followed the lead of SESAME V2 in moving the PAC outside the ticket and adding a signature or seal (symmetric key cryptographic checksum) over the PAC. As stated in [2], there are two advantages to having the PAC outside the ticket. First, as the PAC is extended to contain extensible delegate restrictions, legal issues surrounding encryption of arbitrary data are avoided if the PAC is outside the ticket (if client generated data is allowed in the encrypted ticket, then the ticket can also serve as an encrypted message). Second, performance is slightly improved since less data has to be encrypted.

2.2 Capabilities

We have expanded the PAC to contain arbitrary access control list entries (ACEs) in addition to the principal's identity and list of groups for which the principal is a member. An ACE can be roughly described as a triple (requester, operations, target): the requester is a principal or group; the operations field is a bit vector giving the operations that the requester is allowed to perform; and the target is the object that the operations are to be performed on. The motivation behind the ACE is to allow capabilities (proxies with restrictions) to replace application server maintained access control lists (ACL's) where appropriate. In addition, this extension to the PAC allows for finer grained levels of privilege to be associated with principals.

The flexible authorization protocol allows applications to control the point in time at which authorization information used by a server is obtained. The authorization information can be in the form of ACL's that are managed by the server, or it can be provided in client obtained capabilities. We do not require that application servers store or be forced to obtain the authorization information that they will use to make authorization decisions. Consequently, the flexible authorization protocol allows application servers with limited resources to be more easily supported.

2.3 Centrally Managed ACL's

The flexible authorization protocol allows ACL's to be managed either locally or centrally. In some cases, it is beneficial to have central management of ACL's. Centrally managed ACL's provide a way to support a common policy across an organization. In addition, centrally stored ACL's require less sophisticated management than distributed ACL's.

2.4 Support for Delegation

We have examined the delegation problem from the viewpoint of applications.

Our authorization protocol includes a delegation scheme similar to the one planned for OSF DCE 1.1 [2]. For DCE delegation, initiators of the delegation do not have to know the location of resources necessary for the delegated request, or even which resources are needed. This form of delegation is well-suited for distributed object oriented systems. This form of delegation also allows end servers to determine the identities and privileges of all the intermediate servers, since the extended PAC (EPAC), a chain of PAC's from an initiating client and its delegate servers, is sealed using a symmetric key.

Our flexible authorization protocol also supports proxy delegation, which is based on the concepts in [8]. In proxy delegation, each client obtains an end server proxy ticket and authenticator which are used as a proxy and sent to the immediate delegate. These proxies, and subsequent proxies, are accumulated along the delegation chain and presented to the end server. The limitation of this delegation scheme is that the initiating client must know the identity of the end server. For many electronic commerce applications like NetCheque [7], the client knows the identity of the end server. Thus this requirement is not a limitation for those applications. Our approach also has an advantage over DCE delegation for these applications, since the initiators share a key with the end server and repeated requests with different restrictions do not require interaction with a central server (the session key from the ticket is used to seal new delegate restrictions).

We allow an application to switch from using DCE delegation to proxy delegation for subsequent requests. The advantage of switching to proxy delegation is that repeated requests with different delegation restrictions requires less than half as many messages as with DCE delegation.

Our flexible authorization protocol can also support a delegation method such as the one in SESAME V2. In this form, the privilege server signs PAC's and delegate restrictions (extended PAC's or EPAC's) with its private key. In addition, our protocol can be easily extended to support other public key based delegation schemes.

2.5 Other Features

Other features of the flexible authorization protocol include:

1. It supports applications that do not use DCE RPC.
2. If a centralized authorization server is used, ACL's may be cached at the application servers or stored in capabilities at the clients.
3. The structure of ACL entries is flexible enough to support immediate revocation of capabilities, since clients or administrators can contact end servers and revoke the client's delegates by placing special temporary access control entries.

3 Obtaining Initial Credentials

OSF DCE [10] clients obtain initial credentials (PAC's and TGT's) using a protocol that requires six messages. SESAME V2 [6] clients obtain initial credentials using a protocol that requires four messages. Both protocols attempt to achieve as much separation as possible between the authorization and authentication services, so that the existing authentication or authorization services can be replaced more easily. We believe that our protocol allows for as much separation between the authentication and authorization services, but it contains an option that allows clients to

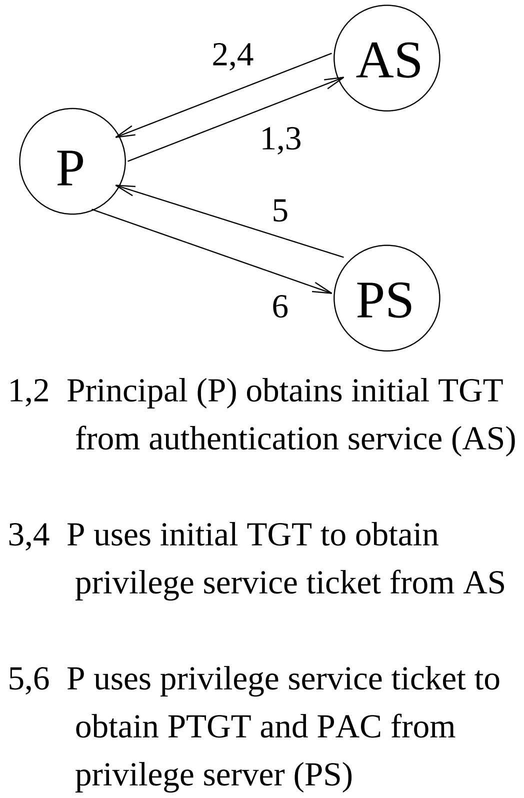


Figure 2: Obtaining Initial Credentials in OSF DCE

obtain initial credentials in two messages. In addition, we allow finer grained levels of privilege attributes (access control entries or ACE's) to be placed into the PAC.

We now describe how a principal P obtains initial credentials in OSF DCE. When P logs in or initializes, it obtains a ticket granting ticket (TGT) which it can then use to obtain additional service tickets (the TGT is obtained through an exchange with the authentication service (AS)). The principal P then uses the initial TGT to obtain a service ticket for the privilege server (PS) from the authentication service. This ticket (and the accompanying session key) allows the client to authenticate to the privilege server. In the last exchange with the privilege server, the client obtains a PTGT (privilege ticket granting ticket) and PAC. A PTGT is a ticket granting ticket with the privilege server as the client; in addition, it contains a seal over the PAC. These steps are illustrated in Figure 2. The PTGT and PAC allow the principal to authenticate to other principals, and they allow the principal to securely pass its PAC to its peers.

SESAME V2 principals obtain initial authorization credentials in four messages. As in OSF DCE, principals first obtain an initial ticket granting ticket. Since the authentication service and the privilege server share the same long term key, principals are able to present their initial TGT directly to the privilege server in order to obtain a signed PAC and PTGT.

We present an example showing how initial authorization credentials can be obtained in two messages in the flexible authorization protocol. As in SESAME V2 and OSF DCE, our privilege server shares a long term key with the authentication service. In this case, the authentication service is Kerberos V5. As in SESAME V2, principals may present their initial TGT to the privilege server. Thus a principal can obtain initial credentials in four messages. We now show the two message option.

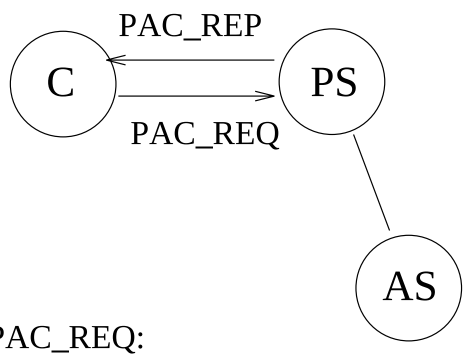
A client can create its request message (PAC_REQ message) by entering a character string identifier identifying the PAC it desires to obtain; in this example, the requesting user desires a bank officer PAC. The client also enters the Kerberos V5 authentication server request (AS_REQ) fields including the desired validity period for the ticket (starttime and endtime fields). The authentication mechanism type is set equal to Kerberos V5. Thus we combine a privilege server request with a Kerberos V5 authentication server request (AS_REQ) (Figure 3).

The privilege server receives the client request for initial credentials. It checks the database to see if the client is authorized to obtain a bank officer PAC. If not, an error message is sent back to the client. If the client is authorized, the privilege server calls the appropriate Kerberos V5 function with the Kerberos V5 AS_REQ part of the client message as input. The authentication service returns the Kerberos V5 AS_REP fields (Figure 3), including the initial TGT, the session key encrypted for return to the client, and the sesssion key unencrypted for use by the privilege server. The privilege server will use the key to compute a checksum over the return (PAC_REP) message fields, but first, it makes another call to the authentication service. The same fields are sent, except the forwarded flag is requested for the new TGT, and the client is listed as the privilege server itself. The requested PTGT will have the privilege server as the client, but it will have the requesting client's address. The PTGT will be returned, with the privilege server's seal over the PAC in the PTGT. Alternatively, the PAC can be signed with the private key of the privilege server. These fields are then returned in the reply message (Figure 3).

At this point, the client has obtained a PAC, a PTGT, and its session key which can be used to obtain credentials for other application servers. These initial steps do not have to be repeated until the client wants to assume a new role (obtain a different PAC).

4 Delegation

In this section, we discuss existing solutions to the two authorization problems (print server/file server and temporary document sharing) that we presented in the introduction. These existing solutions are OSF DCE and SESAME V2. We then present the proxy delegation solution to these problems. The flexible authorization protocol supports all three approaches. Lastly, we briefly discuss NetCheque and its delegation needs. The proxy delegation method is the optimal method for NetCheque, and it also has some advantages for the first two examples.



PAC_REQ:

authentication mechanism type:

Kerberos V5

Kerberos V5 AS_REQ fields:

client, realm, server, starttime,...

authorization request fields:

bank_officer

PAC_REP:

authentication mechanism type:

Kerberos V5

Kerberos V5 AS_REP fields:

client, realm, initial TGT,

[TGT session key, starttime,...] C’s key

Kerberos V5 AP_REP

PTGT (TGT with PS as client)

PAC:

"bank officer"

authenticated flag

client name, realm

group G1,...,group Gn

Checksum over PAC_REP

Figure 3: Obtaining Initial Credentials in the Flexible Authorization Protocol

4.1 OSF DCE Delegation Solution to Print Server Problem

We now discuss the OSF DCE solution to the print server delegation example presented in the introduction; our description is based on the OSF DCE plans described in [2]. We focus on the security mechanisms rather than the RPC (remote procedure call) details.

The reader should recall from the introduction that a client user C desires to print file X on print server P. The file X is managed by the file server F. The client C must obtain a privilege service ticket for the print server P (a ticket with P as the server and the privilege server as the client). In addition, the client C obtains a delegate restriction "P can read file X only" that will be associated with its PAC. The client C forwards its PAC, PTGT, and delegate restriction to the privilege server (Figure 4). After the client request has been received and validated by the authentication service, the privilege server validates the ticket seal over the client PAC. The privilege server then computes the new seal using the session key for the new print server ticket. The new seal is computed over the PAC and delegate restriction, or extended PAC (EPAC); it is placed in the ticket for the print server. This seal allows the print server to validate the client EPAC. Since the seal is inside the print server ticket, the client cannot modify its PAC without being detected by the print server (the print server ticket is encrypted in the print server’s long term key).

In addition, the privilege server computes a seal using a non-shared symmetric key over the client EPAC. This seal is placed in the EPAC; the EPAC is returned to the client. When the print server presents its PAC and the client’s EPAC to the privilege server (the print server will have to present the two credentials in order to become the delegate of the client), the privilege server will check the seal in the client’s EPAC. The purpose of the seal is to prevent the client’s delegates (the print server in this case) from modifying the client’s delegate restriction. Thus a compromised print server is prevented from removing the client delegate restriction. The privilege server returns the client EPAC and print server ticket to the client (Figure 4).

Upon receiving the EPAC and print server ticket, the client creates a message containing the print server ticket and EPAC to send to the print server. The print server receives the client request; after authenticating the client, the print server validates the inside ticket seal over the client’s EPAC. The appropriate error message is sent to the client in case the validation fails; otherwise, the print server processes the client request to print file X. The print server may check an access control list (ACL) to determine if the client is authorized to print file X. If the client passes the authorization check, then the print server contacts the privilege server (Figure 5).

The print server sends its PTGT, its PAC, and the EPAC from the client in the request message. Upon receiving the message, the privilege server validates the seal in the PTGT over the print server PAC. It then validates the seal in the client EPAC over the EPAC. The privilege server now creates a new chained

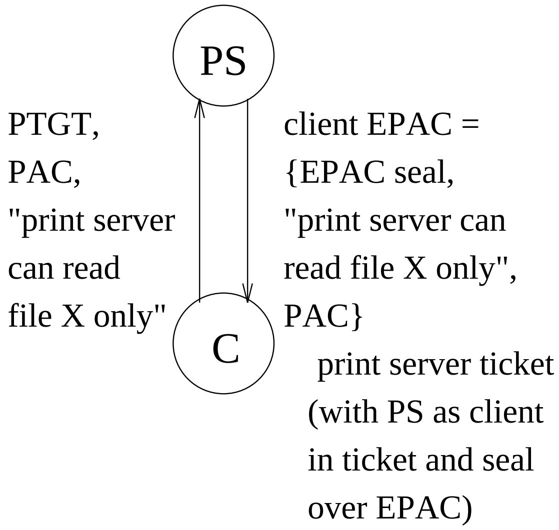


Figure 4: The Client C Obtains an EPAC (DCE)

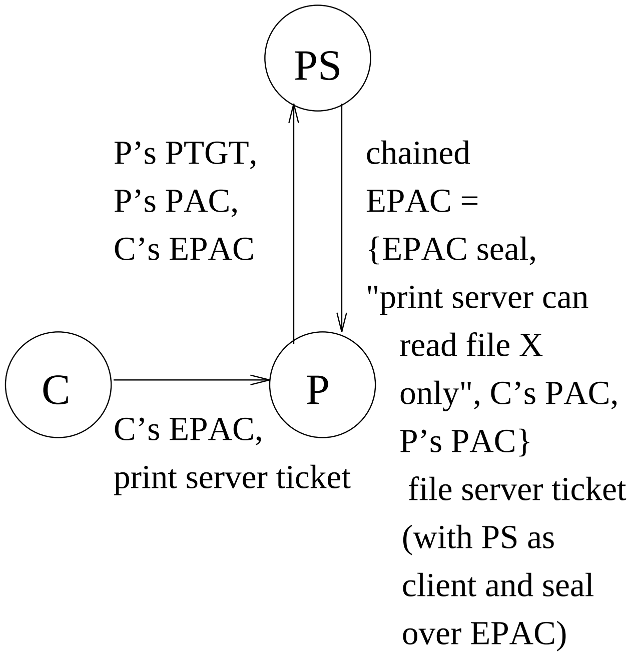


Figure 5: Print Server Obtains Chained EPAC (DCE)

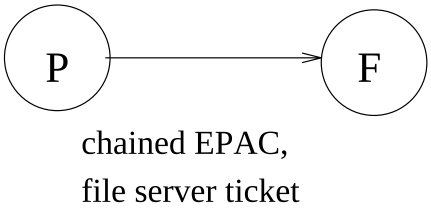


Figure 6: Print Server Request to File Server (DCE)

EPAC that consists of the client EPAC and the print server's PAC; a seal is computed over the chained EPAC using a non-shared symmetric key and is placed in the EPAC. In addition, the privilege server computes a seal over the same fields using the session key from the to be issued file server ticket. This seal is placed in the file server ticket (the authorization data field of the ticket is used to hold the seal). The privilege server then returns the file server ticket and chained EPAC to the print server (Figure 5).

Upon receiving the ticket and chained EPAC from the privilege server, the print server sends a request to the file server. The file server authenticates the print server. When the file server receives the print server's read request for file X (passed as application data), it uses the local access control list combined with the chained EPAC from the print server to determine if access is to be granted. The file server first validates the inside ticket seal over the EPAC using the session key from inside the ticket. Three tests must be passed before access can be granted. The access control list entries (ACLE's) are divided into two sets: one set for initiators of requests and the other set for their delegates. Initiators are defined to be the initiating principal in any delegation chain. The principals of a delegation chain after the initiator prior to the end server are called delegates, or intermediates. In order for the file server to grant read access to file X for the print server, the client, who is the initiator, must obtain read access to file X as an initiator (using the access control algorithm with the initiator ACLE's). The file server then checks whether the print server has read access as either an initiator or a delegate (both sets of ACLE's are checked). The print server must pass the access check as either an initiator or a delegate, or else the request is rejected. Finally, the print server request may be rejected if it violates the delegation restriction placed by the client. Since the print server is requesting to read file X, and the client's delegate restriction is "print server can read file X only," the delegate restriction is not violated. The file server grants access to this print server request (Figure 6).

4.2 Strengths and Weaknesses of OSF DCE Delegation Approach

We now examine the strengths and weaknesses of OSF DCE with respect to the above example. Suppose the client subsequently requests to print file Y. In order for it to place the restriction, "print server can read file Y only", it must go through the same steps as above. New tickets must be obtained; the procedure again requires at least seven messages. The reason that seven messages are again needed is that new EPAC's must be obtained with the new delegate restriction. Thus, the OSF DCE delegation method requires a lot of messages when delegate restrictions change frequently.

If we examine performance from the point of view of the print server, we see that the print server must obtain a new ticket and chained EPAC for the file server each time a new client requests to print a file. The print server can reuse its chained EPAC when a previous client requests to print a new file with the same delegate restrictions. Since the proxy delegation type that we introduce below requires fewer messages, it follows that organizations that rely solely on the DCE delegation type may require additional security servers to handle the additional requests. In addition, if delegated requests are common, there is a question as to how well the DCE delegation type scales with respect to performance.

One way to reduce the number of requests that the privilege server has to handle is to use looser delegate restrictions. In the above example, the client could restrict the print server to reading files in a certain directory, thus enabling itself to reuse the same EPAC for repeated requests in the same directory. The print server must still contact the privilege server each time a new client has a delegated print request.

An advantage of the OSF DCE delegation method is that the client does not have to know the identity of the server that is managing file X. This feature eases administration if there is no directory service. In addition, the client may not know which objects must be accessed by its delegates; the OSF DCE delegation method accommodates these delegations which can occur in distributed object oriented systems [2].

4.3 SESAME V2 Delegation

We now examine the SESAME V2 approach to the print server/file server delegation problem. In the delegation scheme described in [6], the initiator obtains a delegatable PAC and passes it to the intermediate server (we recall from above that PAC's are signed with the private key of the privilege server). The delegatable PAC contains protection values (PV's) that are obtained from randomly generated control values (CV's) using a one way function (these values are created by the privilege server). The initiator passes the appropriate control values to the intermediate server; these values are encrypted in the shared session key since they must remain secret as they are passed over the network. The protection values are associated with delegate restrictions in the delegatable PAC; in this case, one of the restrictions would identify the print server as a legitimate delegate of the client (see Figure 7). The print server then passes the client's delegatable PAC and the control values to the file server. The control values are encrypted in the session key that is shared by the print server and the file server. The file server can verify that the print server is a legitimate delegate of the client by passing the control values into the one way function and obtaining the appropriate protection values that are visible in the client PAC. In addition, the file server validates the client PAC using the public key of the privilege server.

Since the intermediate server (the print server in this case), does not pass its PAC, the end server cannot make use of the intermediate's PAC in the access control function. In addition, if there are more than three principals in the delegation chain, then the end server cannot determine the identities of all of the intermediate servers. Fairthorne [3] describes extensions to the delegation method of [6]; these extensions are called traced delegation since each principal in the delegation chain passes its EPAC along with the received EPAC's from previous delegates to the immediate target (the immediate target is the next principal in the chain). Each member of the chain links its EPAC to the immediate target principal by placing the target principal's identity in the EPAC. Multiple target principals can be placed in the EPAC; the CV/PV method is used to activate a particular target principal. The flexible authorization protocol can easily be extended to support the SESAME V2 traced delegation method. For the remainder of the paper, unless stated otherwise, SESAME V2 delegation will mean the SESAME V2 traced delegation method.

PV1; print server P is delegate

PV2; delegate restriction2

. .
. .
. .

Figure 7: CV/PV Protection Method for PAC's

4.4 Strengths and Weaknesses of SESAME V2 Delegation

The delegate restriction "print server can read file X only" is a fine-grain delegate restriction. In SESAME V2, clients have to revisit the privilege server if they wish to obtain delegate restrictions that are this fine grained, since the CV/PV method would be too slow if these type of restrictions were issued for every file. Thus for fine grained restrictions, the CPU performance penalty associated with public key cryptography is present, but there are still a lot of messages that have to be exchanged. The only savings occurs for the print server; it does not have to contact the privilege server again for new clients as in OSF DCE. The proxy delegation method that we describe below allows for much better performance when fine-grain delegate restrictions are being used.

We now consider SESAME V2 performance when loose delegate restrictions are being used. The delegate restriction "print server can read files from directory D only" is a more loose delegate restriction than the delegate restriction of the previous paragraph. In this case, the CV/PV method can be used to activate different delegate restrictions (see Figure 7) without having to return to the privilege server. Thus, significantly fewer messages are needed than in the OSF DCE delegation method, but there is still the CPU performance penalty associated with public key cryptography. The proxy delegation method also requires less messages but without the need for public key operations.

The CV/PV method requires extra work on the part of the system administrator in order to set up the necessary restrictions for principals and their commonly used targets. There is a cleaner separation between authentication/ key management and authorization in SESAME V2 than in the other methods, since tickets contain reusable principal identifiers (unique to the principal) rather than seals over PAC's. These identifiers are copied from ticket granting tickets into service tickets by the Kerberos V5 KDC; thus the process of obtaining new service tickets does not involve the authorization service. Another advantage of the SESAME V2 architecture is that for interrealm operations, the privilege server signature of the originating realm improves non-repudiability.

4.5 Proxy Delegation Solution to Print Server Problem

We now present the proxy delegation method and show how the print server/file server delegation problem is solved using proxy delegation. In proxy delegation, each member of the delegation chain obtains an end server ticket with a seal over the PAC. In addition, the principal uses the session key from the ticket to compute a seal over the delegate restrictions. If principals are not able to determine the end server in the delegation chain, they can pass a ticket granting ticket instead of an end server ticket. In this case, the proxy delegation method is very similar to the OSF DCE delegation method with respect to performance and administration. When principals know the end server in the chain, fine grain controls and better performance can be achieved.

In the print server/file server scenario above, the client obtains both a proxy ticket for the file server and a proxy ticket for the print server. In order to determine the file server principal identity where file X is stored, a directory service could be queried if one is available. These tickets both contain seals over the client PAC. The client computes a seal using the file server ticket session key over the delegate restriction "print server can read file X only." This seal is placed in the authenticator that is associated with the file server proxy ticket. The client then authenticates to the print server and passes the file server proxy, authenticator, PAC, and delegate restriction to the print server (Figure 8). The print server obtains a proxy ticket for the file server; it then authenticates to the file server. The proxy ticket has a seal in it using the session key from the ticket; the seal is over the print server PAC. The print server also passes the proxy ticket, authenticator, PAC, and delegate restriction from the client to the file server. The file server can now validate the client PAC, the print server PAC, and the delegate restriction from the client. If the print server removed or modified the client delegate restriction, the file server would detect it.

4.6 Strengths and Weaknesses of Proxy Delegation

We now analyze the strengths and weaknesses of the proxy delegation method. With respect to the

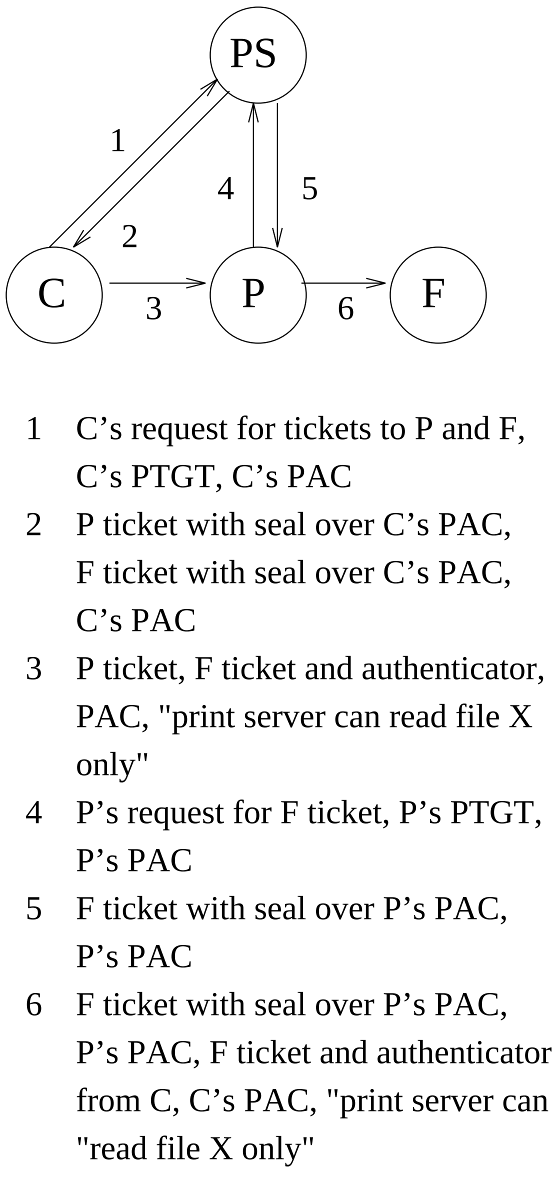


Figure 8: Proxy Delegation Solution to Print Server Problem

above example, the print server does not have to recontact the privilege server as additional clients contact it. Clients do not have to revisit the privilege server in order to create new delegate restrictions (e.g., the client can create the restriction "print server can read file Y only" without having to recontact the privilege server).

The proxy delegation method is the preferred method when initiators either know the end server identity or can easily find out the end server identity, since the number of messages needed is substantially less than with the OSF DCE delegation method. It is also more efficient than the SESAME V2 method with respect to CPU performance. The proxy delegation method is the optimal method for the NetCheque application that we describe below. For distributed object oriented applications and other applications where clients do not know which resources must be accessed by their delegates, the proxy delegation method requires passing ticket granting tickets; it then has no advantage over the OSF DCE method. In fact, it is slightly less efficient in this case.

4.7 Document Sharing Problem

We now describe the OSF DCE solution to the temporary document sharing problem. Since both initiators and delegates must obtain access from the access control lists, these lists must be modified by the system administrator. The administrator can modify the lists initially to allow delegate view access by Corporation B for all of the documents. For each document that is to be viewed by Corporation B, an initiator in Corporation A can obtain a delegate restriction to limit Corporation B to viewing that document. The delegate restriction is passed to Corporation B as part of an EPAC as described above for the print server delegation problem. There are three problems associated with this solution. First, an exchange with the privilege server is required for both parties when a new document is to be viewed, since delegation restrictions must be changed. Second, another principal in Corporation A could also allow access to the same principal in Corporation B (unknown to the first principal) for another information sharing task. This second principal would not impose the document viewing delegate restriction above, so in this case Corporation B would be able to view all of the documents. Third, access control lists must be modified in the beginning and at the end of the process; these modifications are additional work for the administrator (a mistake can also occur).

Within our flexible authorization protocol, a better solution exists. We use the proxy delegation scheme; the initiator in Corporation A obtains a proxy that allows the Corporation B delegate to view a document. The proxy acts as a capability and gives the delegate an access right that the access control lists do not give. The initiator can pass such a capability to a delegate as long as the access control list for the object gives the initiator the access right that it passes and does not explicitly deny the delegate the same right. With this solution, the three problems mentioned above are avoided.

4.8 NetCheque

We now briefly describe the requirements of NetCheque [7] which is an electronic banking application developed at the University of Southern California. NetCheque allows users to securely write checks or other orders for payment. As part of the implementation of NetCheque, USC implemented part of the proxy framework that we have described in this paper.

A NetCheque client, acting on behalf of a user, sends a check which includes delegated authentication in the form of a proxy ticket and authenticator. The proxy ticket is for the payer's accounting server, and the authenticator contains a checksum over the other fields of the check (including the amount, currency, payee, and expiration date). The check is sent to the payee; the payee then obtains a ticket for the payer's accounting server and generates an authenticator endorsing the check in the name of the payee. Thus the client passes an accounting server proxy to the payee, or intermediate server, and the payee endorses and passes the accounting server proxy to the accounting end server (after authenticating to the accounting server).

Using NetCheque, a client can write many checks to different payees after obtaining the single accounting server ticket. The proxy delegation scheme below supports applications like NetCheque since it allows the client to place new delegate restrictions (check amount and check payee) without contacting the authorization server again. The client knows the identity of the accounting server, so it is able to obtain a ticket for it and thus share a key with it; this key is used to seal the delegate restrictions. The proxy delegation scheme is much more efficient than either the OSF DCE or SESAME V2 delegation schemes for applications like NetCheque (roughly half as many messages are required for each subsequent check), where the intermediate servers are untrusted and the delegate restrictions can change frequently. It requires that the initiating client know the end server principal identity which is true for applications like NetCheque, since the writer of a check knows the accounting server that maintains the account against which the check is written.

A possible extension to NetCheque is to allow clients to sign checks using asymmetric cryptography. There is a CPU performance penalty associated with this extension, but the advantage is that a compromised security server may be easier to detect.

4.9 Using Both Delegation Types

Our flexible authorization protocol supports both the DCE delegation type and the proxy delegation type as described above. It is possible for an application to start out using the DCE delegation type (which it would use if it did not know the end server principal identity) and then switch to the proxy delegation type. The advantage of switching to proxy delegation is that it is more efficient when there are repeated requests using the same delegation chain and the delegate restrictions need to be changed. The application does not have to be aware of this switch.

5 Comparison to Other Work

This section describes other work in the area of authorization for distributed systems. There are many papers that address the topic of delegation in distributed systems. Some of these systems have already been described, and other discussions can be found in [4, 11, 12]. The main influences on our flexible authorization protocol are [2, 6, 8]. Our proxy delegation type is based on the concepts described in [8]. We have discussed the OSF DCE approach to authorization [2, 10] in detail in this paper.

As discussed above, we accommodate both the OSF DCE and SESAME V2 approaches to authorization, but we add new functionality in the form of the proxy delegation scheme, extending the PAC to contain access control entries, allowing additional authentication mechanisms, and reducing the number of messages needed to obtain initial credentials.

SESAME V2 has also implemented authorization functionality over Kerberos V5 [6]. We have followed their lead in moving the PAC outside the ticket; we allow the PAC to be signed with the private key of the privilege server as in SESAME V2, or a seal can be placed over the PAC as in OSF DCE. We discussed the SESAME V2 approach to delegation in detail in Section 4.

6 Summary

In this paper, we have presented some existing problems in distributed authorization. We have also presented the major existing solutions to these problems: OSF DCE and SESAME V2. We have described the flexible authorization protocol solutions to these problems.

The main features of the protocol include the ability to operate over different authentication mechanisms (including Kerberos V5 and public key based mechanisms), a privilege server that issues PAC's, extending the PAC to include access control entries, accommodating both the OSF DCE and SESAME V2 approaches to delegation, the proxy delegation scheme, allowing ACL's to be managed centrally, and reducing the number of messages needed to obtain the initial credentials. Further information is available in [1].

With respect to OSF DCE, our protocol supports both existing and planned OSF DCE security functionality. Thus new applications that are developed to run over an implementation of our protocol will be able to utilize DCE security functionality. In addition, we extend OSF DCE security functionality in a flexible manner as described above. In particular, the proxy delegation scheme is better suited than the OSF DCE delegation scheme for certain electronic commerce applications [7]. The proxy delegation scheme also has some advantages for the two authorization problems in the introduction: the print server/ file server delegation problem and the document sharing problem. In addition, our authorization protocol does not require that applications use DCE RPC.

Acknowledgements

We wish to thank Dan Webb, Joe Kovara, and Glen Zorn for some very helpful conversations. We also thank Dennis Glatting and the referees for some helpful comments. We thank Laurie Anderson for her help in preparing an initial draft.

References

[1] CyberSAFE Corporation. "Practical Distributed Authorization for Large Networks," Technical Report 95-47, CyberSAFE Corporation, 2443 152nd Avenue NE, Redmond, WA 98052 USA; tr-request@cybersafe.com

[2] Marlena E. Erdos and Joseph N. Pato, "Extending the OSF DCE Authorization System to Support Practical Delegation," *In Proceedings of the PSRG Workshop on Network and Distributed System Security*, pages 93-100, February 1993.

[3] S. B. Fairthorne, "Security Extensions for DCE 1.1," OSF-DCE RFC 19.

[4] M. Gasser and E. McDermott, "An Architecture for Practical Delegation in a Distributed System," *In Proceedings of the 1990 IEEE Symposium on Security and Privacy*, pages 20-30, May 1990.

[5] John T. Kohl and B. Clifford Neuman, "The Kerberos Network Authentication Service (Version 5)," Internet RFC-1510. September 1993.

[6] P. V. McMahon, "SESAME V2 Public Key and Authorization Extensions to Kerberos," *In Proceedings of the 1995 Symposium on Network and Distributed System Security*, pages 114-131, February 1995.

[7] B. Clifford Neuman and Gennady Medvinsky, "Requirements for Network Payment: The NetCheque Perspective," *In Proceedings of IEEE Compcon'95*, San Francisco, March 1995.

[8] B. Clifford Neuman, "Proxy-Based Authorization and Accounting for Distributed Systems," *In Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283-291, May 1993.

[9] B. Clifford Neuman and Theodore Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications*, 32(9), September 1994.

[10] OSF DCE 1.02 Application Development Guide, Volume 2. Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142. 1993.

[11] Karen R. Sollins, "Cascaded authentication," *In Proceedings of the 1988 IEEE Symposium of Research in Security and Privacy*, pages 156-163, April 1988.

[12] V. Varadharajan, P. Allen, and S. Black, "An Analysis of the Proxy Problem in Distributed Systems," *In Proceedings of the 1991 IEEE Symposium on Security and Privacy*, IEEE Computer Society, 1991.

Jonathan Trostle may be reached at CyberSAFE Corporation, 2443 152nd Ave, Redmond, WA 98052, USA. Telephone +1 (617) 883-8721, email jtt@cybersafe.com. B. Clifford Neuman may be reached at USC/ISI, 4676 Admiralty Way, Marina del Rey, CA 90292-6695, USA. Telephone +1 (310) 822-1511, email bcn@isi.edu