CITI Technical Report 92-1

Third-Party Authentication in the Institutional File System

Bill Doster
Bill.Doster@umich.edu

Jim Rees
Jim.Rees@umich.edu

Institutional File System Project
Center for Information Technology Integration
The University of Michigan
519 West William Street
Ann Arbor, MI 48103–4943

ABSTRACT

The use of intermediate translators in the Institutional File System presents the problem of authenticating the translator to the file server where the client's private key is not known to the translator. We have implemented a modification to the Kerberos authentication exchange that allows our translators to securely acquire the rights necessary for them to access files and other services on behalf of their clients.

February 2, 1992

Third-Party Authentication in the Institutional File System

Bill Doster
Bill.Doster@umich.edu

Jim Rees
Jim.Rees@umich.edu

Introduction

The Institutional File System (IFS) [1] is a large, heterogeneous distributed file system developed at the Center for Information Technology Integration (CITI) and based on AFS [2]. Authentication in AFS between clients and servers is provided by the Kerberos Authentication System [3]. Eventually we expect to support tens of thousands of file system clients. While many of these clients are UNIX workstations capable of participating directly in AFS and Kerberos services, many others are personal computers (mostly Macintosh or IBM PC), which may not have the necessary resources to support AFS directly. These clients obtain file services from *translators* that translate IFS services into services the client can understand.

For example, a Macintosh may want to access the IFS through the AppleTalk Filing Protocol (AFP) [4]. In this case, the translator will accept file requests from the Macintosh in AFP form, communicate with the IFS server to satisfy the requests, and present the file back to the Macintosh in AFP form.

The translator is a separate Kerberos principal, situated between the client and the file server. The client communicates only with the translator, and not with the Kerberos authentication server or the file server. The translator also acts as an intermediate file server, and caches files for the client [5]. The network connections among these principals is shown in Figure 1.

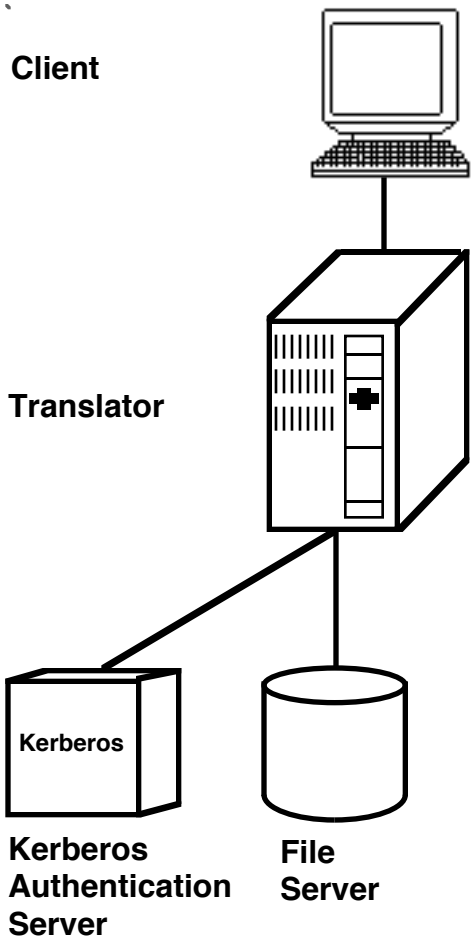

Figure 1

The use of translators presents an authentication problem. The translator

does not have access to the client's private key, yet it must be able to authenticate itself to the IFS server in such a way that it can perform file system operations on behalf of the client. We have solved this problem through the use of an intermediate authentication service [6]. This paper describes the additions we have made to the Kerberos Version 4 authentication exchange to support the use of translators.

Goals

We wanted to preserve all of the guarantees that Kerberos provides to its clients. In particular, we assume the existence of *attackers,* hostile users who attempt to impersonate others in order to gain unauthorized access to files [7]. We assume that an attacker has physical access both to the network and to a workstation capable of reading any message from the network and sending any message to any host. We also assume that the attacker has full knowledge of the network protocols.

Clients must be able to use IFS services without being required to communicate with any part of IFS except through a translator. Some clients may not even have TCP/IP, which is currently the only way to communicate directly with the Kerberos servers.

Translators must not share keys with clients, and in particular must not store, acquire, or use client private keys. This restriction simplifies key distribution and limits the damage should a translator become compromised. Translators are Kerberos principals, and therefore have their own private keys.

We wanted to minimize the changes necessary for clients to gain access to the IFS. Where client software already has hooks in place for access controls, we have used those hooks whenever possible.

And we wanted to avoid changing the file servers at all. The global AFS community contains many servers over which we have no control; we want our users to have access to these servers without making special arrangements ahead of time.

Other Approaches

Neuman [8] describes a way to use proxy authentication to delegate authorization for use of a service to a third party. Kerberos Version 5 has features added to it to provide better support for restricted proxies. This method is not directly useful for us, because it requires that the client acquire a ticket directly from Kerberos, then grant a restricted ticket (a *proxy*) to the translator. Our clients must be able to gain access to their files by communicating only with the translator and not directly with either the Kerberos or file system services.

Blakley [9] considers four options for the integration of PC class machines with AFS servers. His requirements are more restrictive than ours. In particular, he requires that the client machines not run any modified software, and that users of the translator have identities separate from their identities as users of direct AFS clients. He proposes that all users of the translator be members of a separate authentication realm, and that the translator be the authentication server for that realm.

Unlike Blakley, we are willing to make additions to client software where necessary to meet our other objectives, especially in the authentication system. More importantly, we prefer that our user's identities be the same whether they access AFS directly or through a translator.

The Authentication Exchange

The end result of Kerberos authentication is that the client acquires a (random) *session key* and a *ticket* with which it can communicate with the server. The session key and ticket are issued to the client by a

ticket-granting service. We use the following notation to describe message exchanges:

a→b:	message passed from *a* to *b*
c	client
x	translator
s	server
#	timestamp
tgs	ticket-granting service
as	authentication service
K_a	*a*'s private key
$K_{a,b}$	session key for *a* and *b*
$\{p,q\}K_a$	*p,q* encrypted in *a*'s key
IP_a	*a*'s IP address
$T_{a,b}$	*a*'s ticket to use *b*
	$(a,b,IP_a,\#,lifetime,K_{a,b})$
A_a	authenticator for *a* $(a,IP_a,\#)$

IP_a is the IP address of the workstation at which Kerberos principal *a* wants to gain access to the file server.

The complete ticket acquisition sequence for the normal client-server case, not involving translators, can be written as:

c→as:	c,tgs
c←as:	$\{K_{c,tgs},\{T_{c,tgs}\}K_{tgs}\}K_c$
c→tgs:	$s,\{T_{c,tgs}\}K_{tgs},\{A_c\}K_{c,tgs}$
c←tgs:	$\{\{T_{c,s}\}K_s,K_{c,s}\}K_{c,tgs}$
c→s:	$\{A_c\}K_{c,s},\{T_{c,s}\}K_s$

The goal of intermediate authentication is to provide the translator with the session key $K_{c,tgs}$ and the ticket-granting ticket $\{T_{c,tgs}\}K_{tgs}$. This must be done without the client communicating directly with the authentication server. Also, the IP address in the ticket must be that of the translator, not that of the client, as the server would otherwise reject requests from the translator.

Our modified Kerberos authentication proceeds as follows. First, client *c* indicates to translator *x* that it wants *x* to acquire tickets on its behalf.

c→x:	c,tgs	(1)

Now *x* gets a ticket-granting ticket from authentication server *as*:

x→as:	x,tgs	(2)
x←as:	$\{K_{x,tgs},\{T_{x,tgs}\}K_{tgs}\}K_x$	(3)

Next *x* gets a service ticket for *c*. Note *x* is acting as a client and *c* as a server for the purpose of mutual authentication.

x→tgs:	$c,\{T_{x,tgs}\}K_{tgs},\{A_x\}K_{x,tgs}$	(4)
x←tgs:	$\{\{T_{x,c}\}K_c,K_{x,c}\}K_{x,tgs}$	(5)

Now *x* has the session key $K_{x,c}$ that it will use to communicate securely with the client. Next *x* gets a ticket-granting ticket on behalf of *c*. Kerberos principals are not tied to any particular IP address, and tickets are encrypted in the client's key, so it is safe (and not unusual) for the translator to do this.

x→as:	c,tgs	(6)
x←as:	$\{K_{c,tgs},\{T_{c,tgs}\}K_{tgs}\}K_c$	(7)

Now *x* has the necessary session key and the ticket-granting ticket, but they are encrypted by the client's private key, which the translator does not have. The translator also needs to authenticate itself to *c*. To do this, it generates an authenticator A_x.

The translator now sends three things to the client. The first is the session key and ticket-granting ticket that it got in step 7 above. The second is the ticket $T_{x,c}$ that it got in step 5, and the third is the authenticator A_x, encrypted by the session key $K_{x,c}$.

x→c:	$\{K_{c,tgs},\{T_{c,tgs}\}K_{tgs}\}K_c$, $\{T_{x,c}\}K_c,\{A_x\}K_{x,c}$	(8)

The client uses its private key K_c to decrypt the ticket $T_{x,c}$. This ticket contains the client-translator session key $K_{x,c}$ that *c* will use to communicate with *x*. The client uses this key to decrypt the authenticator A_x and verify *x*'s identity.

The client now has the ticket-granting ticket and session key that *x* needs. It uses its private key to decrypt these, and re-encrypts them in the client-translator session key $K_{x,c}$. Then it sends them back to *x*.

$$c \rightarrow x: \quad \{K_{c,tgs}, \{T_{c,tgs}\}K_{tgs}\}K_{x,c} \tag{9}$$

Now *x* has the session key $K_{c,tgs}$ and the ticket-granting ticket $\{T_{c,tgs}\}K_{tgs}$, and can acquire a service ticket and communicate with the file server in the normal way.

$$x \rightarrow tgs: \quad s, \{T_{c,tgs}\}K_{tgs}, \{A_c\}K_{c,tgs} \tag{10}$$

$$x \leftarrow tgs: \quad \{\{T_{c,s}\}K_s, K_{c,s}\}K_{c,tgs} \tag{11}$$

$$x \rightarrow s: \quad \{T_{c,s}\}K_s, \{A_c\}K_{c,s} \tag{12}$$

Figure 2 shows a summary of the complete sequence of transactions. In this figure, the vertical lines represent the principals involved, and the horizontal lines represent messages. A message enclosed in a box represents encryption, with the key written above the box.

Experience and Conclusions

Translators must be able to act as Kerberos principals to exchange secure data (keys and tickets) with their clients, so the requirement that the translators have their own private keys seems unavoidable.

This authentication method is in daily use on several translators serving dozens of workstations here at CITI. We use translators with third-party authentication to provide AFS file services to all of our Macintosh computers and NFS clients. We also use intermediate caching servers with third-party authentication for AFS clients, for example to provide file services to a set of machines located on the remote end of a low-speed network connection.

Future Work

We plan to use this authentication scheme for several additional types of translators in the future. We would also like to use it for services in addition to remote file access, for example to obtain tickets for mail or printing service on a remote server machine.

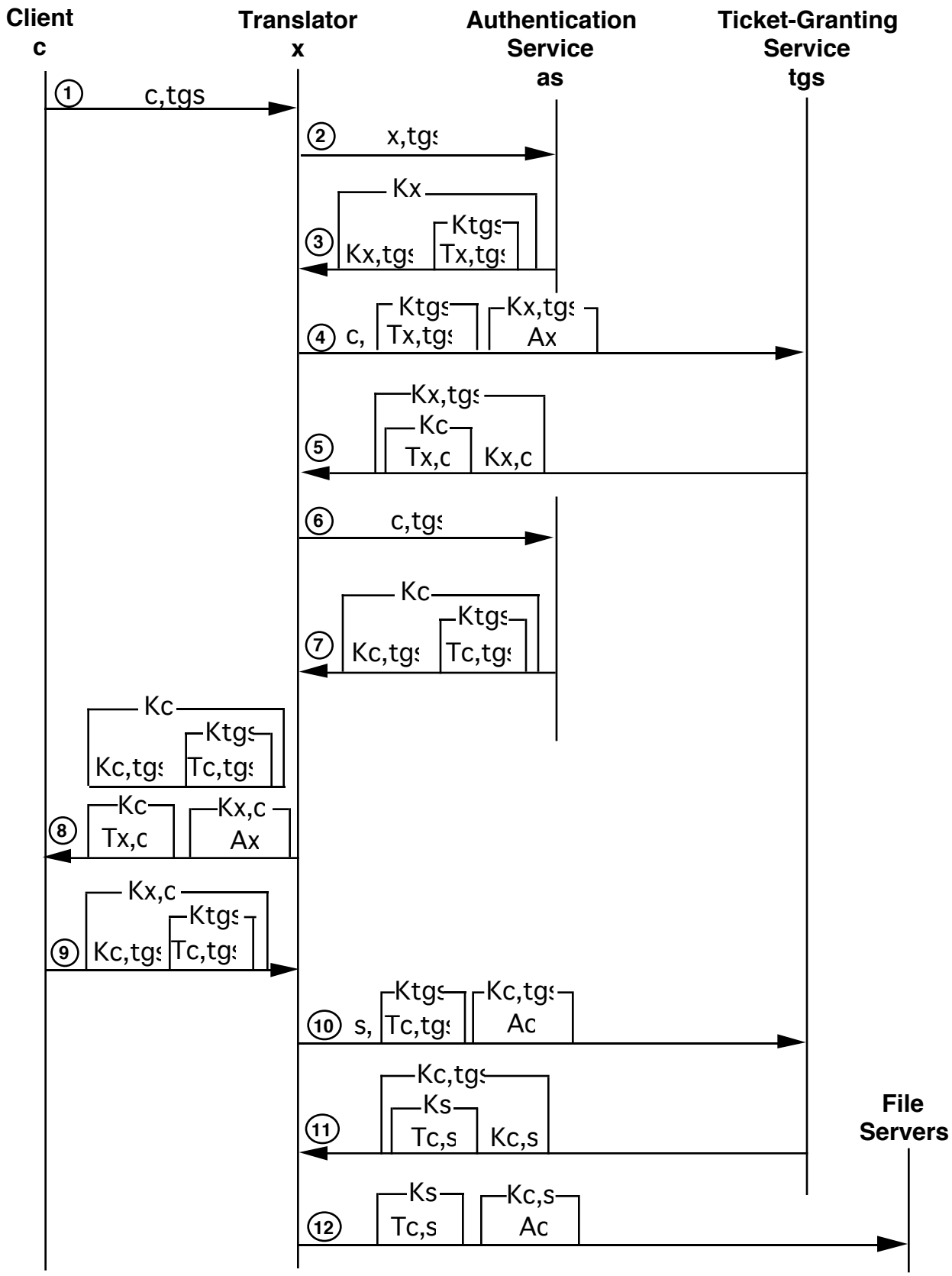
Client
c
Translator
x
Authentication
Service
as
Ticket-Granting
Service
tgs
File
Servers
1
c,tgs
2
x,tgs
Kx
3
Kx,tgs
Ktgs
Tx,tgs
4
c,
Ktgs
Tx,tgs
Kx,tgs
Ax
Kx,tgs
Kc
5
Tx,c
Kx,c
6
c,tgs
Kc
Ktgs
7
Kc,tgs
Tc,tgs
Kc
Ktgs
Kc,tgs
Tc,tgs
Kc
Kx,c
8
Tx,c
Ax
Kx,c
Ktgs
9
Kc,tgs
Tc,tgs
10
s,
Ktgs
Tc,tgs
Kc,tgs
Ac
Kc,tgs
Ks
11
Tc,s
Kc,s
Ks
Kc,s
12
Tc,s
Ac

Figure 2

Acknowledgements

Bruce Howard and Richard Campbell did much of the implementation work. Peter Honeyman showed us why secure authentication is necessary.

References

1. T. Hanss, "Institutional File System Overview," *AIXTRA: The AIX Technical Review*, pp. 25–32 (January 1992).

2. J. H. Howard, "An Overview of the Andrew File System," pp. 23–26 in *Winter 1988 USENIX Conference Proceedings*, Dallas (February 1988).

3. J. G. Steiner, B. Clifford Neuman, and J. I. Schiller, "Kerberos: An Authentication Service for Open Network Systems," pp. 191–202 in *Winter 1988 USENIX Conference Proceedings*, Dallas (February 1988).

4. G. S. Sidhu, R. F. Andrews, and A. B. Oppenheimer, *Inside AppleTalk, Second Edition,* Addison-Wesley Publishing Co., 1990.

5. J. W. Howe, "Intermediate File Servers in a Distributed File System Environment," CITI tech report in preparation.

6. B. Howard, "Intermediate AFS Authentication Service," unpublished CITI internal document, September 1990.

7. P. Honeyman, L. B. Huston, M. T. Stolarchuk, "Hijacking AFS," pp. 175-181 in Winter *1992 USENIX Conference Proceedings*, San Francisco (January 1992).

8. B. Clifford Neuman, "Proxy-Based Authorization and Accounting for Distributed Systems," Technical Report 91-02-01, University of Washington Department of Computer Science and Engineering (March 1991).

9. Bob Blakley, "LAN Coexistence with DCE: Options," in preparation.