Security Functions for a File Repository

Arne Helme
Faculty of Computer Science / SPA
University of Twente, Enschede
The Netherlands
Email: *arne@acm.org*

Tage Stabell-Kulø
Department of Computer Science
University of Tromsø
Norway
Email: *tage@acm.org*

November 1996

Abstract

When personal machines are incorporated into distributed systems a new mixture of threats is exposed. The security effort in the MobyDick project is aimed at understanding how privacy can be protected in this new environment. Our claim is that a two-step process for authentication and authorisation is required, but also sufficient. The research vehicle is a distributed file repository.

1 Introduction

A personal machine is more than just a small machine used by one person at the time. Small, portable machines are like traditional personal organizers and they contain a wealth of private information. If users are to incorporate them into a distributed system, they must be confident that the system is trustworthy. But no-one likes to entrust personal information to a third party, e.g., the technical staff that happens to maintain the local infrastructure. If personal machines are to be truly useful for the owner, however, they simply must exchange information with the personal machines of others. Therefore, in order to unleash the full potential of small, personal machines, the systems must be constructed with the user in control over where his information flows, and who can access it.

The aim of the MobyDick project[1] is to exploit the availability of small (mobile), personal machines by including them in larger systems. Several aspects are investigated, such as replication algorithms, protocols for variable bandwidth communication, and seamless switching between networking technologies. The research vehicle used is a distributed file repository (FR). This paper, however, is only concerned with the security effort of the MobyDick project.

The security effort is targeted at mechanisms that enable users to implement their own policy for security. Access to files is granted in a two-step process. First the quality of the authentication credentials is evaluated, not merely whether they are valid or not. Then, an access control decision is made. Ensuring the privacy of users is an important goal of the security effort of MobyDick. Our claim is that this two-step process is required, but also sufficient, to meet this goal.

In this paper we present our views on: personal computing in the MobyDick project, personal computing from a security point of view, how personal computing changes the rôle of third parties and a short description of the research vehicle (FR). In Section 2 we give a functional description of the services FR provides. The security aspects of these services are considered in Section 3. At the end we present some related work in Section 4 and the current status in Section 5.

1.1 Private data

Assume that an employee has been assigned a palm-top computer to replace his paper-based personal organiser. He will enter his entire agenda and phone directory, his upcoming appointments and other similar information. A phone number will be entered regardless of whether it belongs to

[1] MobyDick (Esprit Long Term Research 20422) is a project of the University of Pisa, Italy, University of Twente, Netherlands, and University of Tromsø, Norway.

a customer or to a family member. This is how this type of personalised equipment is generally used.

Since the machine is personal, it will hold a mixture of private and work-related data, and the privacy of the owner depends on how this data is handled, and by whom. In particular, someone he trusts to access/store data in his normal business activities, may not be trusted to access his personal data. This aspect of computing is not an artifact of the introduction of personal machines per se, but more a result of us bringing computers into our daily lives.

There are two important security aspects. The first has already been mentioned: third parties that used to be trusted for all purposes, might not be trusted to perform quite a few number of tasks where the users' privacy is at stake. We will elaborate more on this below. Second, the user guards resources (his files), and access to a resource is granted at the discretion of the user, based on *two* policies: the user's policy for authentication and the user's policy for authorization. What the policy for authorization says will (naturally) depend on the request, but so will the policy for authentication. For example: Any friend can read my phone dictionary, but I am very strict on how I view the credentials of a friend. At the other extreme, any colleague can read the report I am writing. If the department Kerberos server says that the requester is a colleague I will accept those credentials. The point of interest is that the user would like to base the decision on which credentials to accept at the time of their presentation.

1.2 The rôle of third parties

Traditionally, computers were used in one domain only. For example, a workstation owned and operated by a university only needed access to university resources. In this setting, a single trusted third party (TTP) is sufficient. This is the typical setting where Kerberos is used to enhance security.

Kerberos is controlled by the same authority that owns and controls the resources. There is an implicit link between the policy of authentication done by Kerberos (how to obtain a user ID) and the policy for authorization. The resources will not (have the possibility to) question how the authentication was done.

For example, still, the best way to pay for any merchandise without passing (real) money is utilizing a credit-card company (such as American Express or VISA) as a trusted third party. The credentials the company provides—the card with accompanying expiration date—has no value outside its domain, but within it, it is as "good as gold". There will be one (or more) TTP for practically every service.

However, there will not always be a trusted path[2] between these TTP's,

[2]At least not in the X.509 meaning of the word.

not even a connected graph. Therefore, the user needs to be involved twice before access to a resource is granted: First to consider the authentication credentials in order to decide if certificates from this particular TTP is trusted for this purpose, then to consider authorisation and whether to grant access or not.

1.3 The research vehicle

In MobyDick, a distributed file repository (FR) is used as the research vehicle. FR provides services to users through a set of servers. These cooperate, and maintain a distributed repository in which users can store files. Research into the area of replication protocols and policy is part of the project at large, but is beyond the scope of this article.

Each user can run his own personal FR. An implication is that the trust placed in a FR is constrained by the user's own ability to maintain this service. When information is to be shared, the user can ship files to another user's FR (see below).

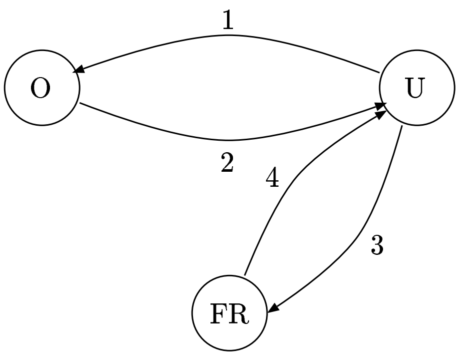

Figure 1: Conceptual model

The owner can grant other users access to his files. How this is done, conceptually, is shown in Figure 1, where the "O" represents the owner, "U" the other user, "FR" the file repository, and arrows denote messages. Message 1 is a request to the owner for an access to one of his files. This "message" might very well be "sent" as part of a normal conversation. Message 2 is the response, containing credentials that the user can use in order to access the file. Message 3 is a request to the FR. This request contains the credentials provided in Message 2 plus other credentials that FR needs in order to make the required access control decision. Message 4 represents the file being sent to the user. In our implementation, Message 3 and Message 4 are part of a single special-purpose file transfer protocol that also transfers credentials.

At this conceptual level, we make no assumptions about how communication is done. The exchange of Messages 1 and 2 does not require a computer network. In particular, we have included means to convey Message 2 orally. Consequently, credentials can be "sent" by the owner as part of a conversation, e.g., when speaking in the phone. This makes it possible to fulfill a request for access even when the owner is disconnected from networking infrastructure.

1.4 The challenge

A modern system that includes personal machines must provide good facilities to protect the privacy of users. To provide facilities is quite different from providing privacy, and also from enforcing a policy that ensures the privacy of users. At the same time, the traditional infrastructure with one single TTP (or a hierarchical structure as in X.509) will not be available. The challenge is to build an infrastructure under these constraints; this is the crux in the security effort in the MobyDick project.

2 Services provided by FR

In the world of portable computing network connectivity can not always be relied upon. This makes it necessary to facilitate smooth working even when disconnected. FR provides services for basic file operations, file sharing, off-line delegation of access rights and file shipping.

Basic operation: FR is as a simple file repository. Over time, files have many versions. In addition, copies of files can be distributed over servers; one copy is the master copy. By providing storage, and the possibility to store state about a file's whereabouts, FR makes it possible *for the user* to decide whether it is safe to proceed in any situation where the existence of several copies creates a consistency problem. The user is kept in the control loop and must inform FR about the decisions he make. When the user does not know, the system will not be able to decide either.

File sharing: Every file has an owner. When a file is to be shared, the owner must grant the borrower access to the file. The credentials that give access can grant either read-access, write-access, or both. Usually it will be one read-once and one write-once. When the file has been checked in by the borrower, the owner can commit the new version as the official version of the file.

Off-line delegation: In FR there is a need to transfer a delegation certificate without the means of a computer network. That is, in order to transfer a certificate, the owner has to do so orally (e.g., over a phone line). It is highly impractical to dictate several hundred hexadecimal digits.

File shipping: Files can be shared by shipping them out of the personal domain. Shipping files to different domain also implies a different location.

In order for this service to be available, the local FR must cooperate with FR's at other locations. In particular, the file owner must be able to name the remote FR in a way that ensures that the file owner's and local FR's understanding of where files are to be shipped actually coincide.

When the file owner is finished, the files are shipped to his home server. If so desired, the (home) FR will buffer all files that are "returning" so that the owner may authorize the deletion of the originals later.

3 Security considerations

When considering a new service the user must know what is at risk[3] if he uses it and "who" is capable of unleashing the different scenarios. This amounts to knowing which channels he believes will provide secrecy when communicating with other principals. Knowing the effects of trust includes also knowing what can happen if a particular principal turns out be be unworthy the trust that has been placed upon it. In particular, users must understand what long-term effects the use of a service has. If a secret becomes known at some time in the future (as opposed to having already being leaked), can it be used to intrude on their privacy.

3.1 Principals

In FR, users (humans) are viewed as principals. Principals have one or more encryption keys associated with them. These keys can be viewed as proxies on which statements from the principal will appear. It is then, ultimately, left to the user to make a binding between the information the system can provide about the encryption key that has been used and the human the user wants to relate to. The problem users have to consider is whether the "correct" principal controls the channel in question, and who is responsible for statements appearing on a particular channel.

3.2 Trusted computing base

In FR the only principal a user *must* trust is himself. This trust, however, covers the following two topics:

[3]In secure systems, the assumptions one makes properly reflect the risks of using the system.

Intention and ability: Each user must trust his own good intentions. This includes not doing something that jeopardises his own privacy, e.g., by trusting someone who is not trustworthy.

When a user places trust in his own secrets also, he implicitly trusts his own ability to safeguard them. The reason is that placing trust in a secret is, of course, a meaningless statement by itself. One places trust in the safekeeping of a secret.

Ability and intention are closely interwoven. If a decision is made that turns out to be wrong, it is impossible from the "outside" to judge whether it was lack of ability or intention that was the cause.

File Repository: A user must trust the repository to perform its tasks according to specification. However, since the FR is only used to store files, it is only trusted to not disclose the contents of these files; see below for a detailed description of the assumptions a user must make.

Notice that a user only has to trust his local FR server; when exploiting other servers these must be trusted explicitly. In addition, placing trust in a FR server only affects the objects (files) the user entrust to it.

We do not include any form of trusted third party in the list, so unlike Kerberos and other centralized systems, the FR is not able to "impersonate" the user by misusing the trust placed upon it.

3.3 Key distribution

In general we can not assume that communication with a third party is possible for users of FR. In this situation it is impossible to contact a third party without incurring high costs, particularly in time. Precautions must be taken to ensure that the user either has the server's key available, or has the possibility to infer trust in the server's key. This must also be the case when the credentials are presented by (someone who claims to be) the server itself.

When the user has sufficient connectivity, the key-distribution can be performed by a variety of protocols, see [5] for examples. In the case of sufficient communication bandwidth, verification of certificates can also be done, for example by an on-line agent, as described in [4, section 5.1].

Each user must build up trust in the keys he assembles, and assemble keys he believes belong to users that are willing and able to protect it. In centralized systems there is a certification authority (CA) that will issue certificates about other users keys. This is possible since every user signs a certificate saying $CA \Rightarrow U$. This makes protocols for key distribution less hard, but the CA may impersonate U. This might be tolerable in a centralized system where those controlling the CA also control the resources. It is definitely not acceptable in a world of personal computing.

FR uses a widely accepted format for storing and exchanging public keys and certificates, namely PGP [9]. We have written software to interface FR to existing PGP key rings, key servers, and certificates used in FR are expressed in a format compatible with PGP.

3.4 Trusting the File Repository

When a user places trust in the FR, the trust is limited to the following:

> FR is trusted to ensure that access to the files entrusted to FR will only be granted to the owner, or to those the owner grants access.

Since trust is placed in the willingness and ability of those in charge of the FR, it is impossible to enumerate what this really implies. However, by limiting ourself to trust the FR as described above, the danger becomes less hard to live with.

In less general terms, the trust amounts to (at least) the following:

- When an issued certificate states that the holder of some key might have access to a particular file, the FR will ensure that the requester indeed holds the key.
- FR will destroy temporary keys (e.g., session keys that provide encryption channels) after they have been used.
- When presented with a once-only certificate (see below) the FR will hold a copy of the certificate until it expires (to ensure at-most-once semantics), or, if this is impossible for some reason, reject it.[4]
- Ensuring that if files are taken off-line (backup, for example) they will be protected at least as good as the versions stored on-line (see e.g., [2]).

Users of centralized systems will be well acquainted with these assumptions, as they apply.

We assume that there is a secure way for the user to obtain the key of FR and vice versa; recall that we are not speaking about a remote or infrequently used service. With these strong assumptions it is relative simple to exchange a key for each session.

From the users point of view, the FR is trusted to be honest in that it will perform its part of the protocols faithfully; in particular, the FR is trusted

[4]In this case, denial of service is correct. As opposed to other situations, where it is a threat [6].

to do so *now*. In the future the situation might be different. For example, if a session key was exchanged under the secrecy provided on a channel from FR, exposure of the encryption key some time in the future will make it possible to decrypt any recorded session. In other words, although the FR can be trusted *now*, relying on its key to be secret for all time to come might be unrealistic. What is required is that when the session ends, the encryption channel is properly closed. The protocols used to interact with FR is constructed so that if the FR is honest (now), disclosure of its secrets at a later time will not make a recording of this session available to the (illegitimate) holder of the secrets.

3.5 File sharing

Files are protected by FR, and sharing a file with some other user U requires the owner O to delegate some of his authority to U. This is done by issuing a delegation certificate. When O delegates access to a file, he trusts U, in the case of a read certificate, to protect the content of the file, and, in the case of a write certificate, not to let any third user write to the file.

The certificates are once-only and U need not be trusted not to misuse the certificates. Giving away a read certificate is almost, but not quite equivalent with giving away the file itself. In the former case U will not have a copy of the file available. Giving away a write certificate might imply writing garbage in the file. Such changes made by means of a write certificate must be committed (see above), and this breach of trust will probably be detected by O.

The important parts of certificates for delegating access to a file are:

- The name of the server on which the file can be accessed;

  This is important since files in FR might be replicated. To ensure the once-only semantics of a certificate it should be valid at one server only.

- To which channel the file can be given;

  Usually, certificates are issued to a public key, that is, the holder of the secret part of the key-pair will be able to present the credentials on the correct encryption channel. The "name" of the key can be expressed in any form suitable for indexing at the server. However, when U wants to remain anonymous, O can generate a new key-pair and include the public part in the certificate and hand the secret part over to U. FR will honour a request arriving on the channel that the key represents. When delegation is performed to an anonymous key, no trace will remain in the system of the identity of U, neither at FR nor by O.

When FR has fulfilled the request, it will hold on to the certificate until it expires. This way FR can enforce the once-only semantics.

O may keep a copy of the certificate and use it for revocation purposes. FR will honour a demand from O to revoke a certificate. FR keeps a copy of the revoke order for as long as the certificate is valid.

3.6 File shipping

When a user travels to a remote site, he can utilize the distributed aspects of FR and ship some of his files to a remote site. In order to use this feature, the user must trust the remote FR to protect his files as good as the local FR does. The problem, of course, is that the user can not be expected to know first hand those in charge of a remote FR, and this gives rise to the question of how to exchange keys. Unless the user has some other means to obtain the key, the local FR must supply a certificate stating that some key belongs to the remote FR. The indirect nature of the certificate reflects the problem of using an unknown service. Likewise, since the remote FR does not have any knowledge of the "physical" user and can only relate to a certificate from the local FR (local to the user, that is) that ties the user's key to a statement about the users status as user of the local FR.

3.7 Off-line Delegation

Off-line delegation is a means to convey certificates orally. In order for this service to be available to the user, a secret must be shared with FR. This secret is used to create a secret channel between the two. However, when such a channel is created, the FR can construct certificates and claim they originate from the user. There are a great many security problems related to channels based on shared secrets. Consequently, off-line delegation service is an exercise in the tradeoff between ease of use and security.

4 Related work

The view on principals expressed in this paper is similar to the one presented in [7]. The once-only semantics FR enforces on delegation certificates is related to, but different, from the one found in [3], and is historically related to capabilities [8]. Related to PolicyMaker [1], our system is essentially performing similar functionality, but we have embedded it in an application and not as a general solution. Well established solutions such as Kerberos provide valuable impulses, although they are designed for centralized environments.

5 Current status

An implementation of FR is in use and available. The security architecture described in this paper is currently under implementation, incorporated into FR.

Acknowledgements

The authors would like to thank the participants of the 1996 DIMACS Workshop on Trust Management in Networks for valuable discussions and feedback. Sape Mullender gave us critical comments to improve this paper.

References

[1] Blaze, M., Feigenbaum, J., and Lacy, J. Decentralized Trust Management. In *IEEE Conference on Security and Privacy* (Oakland, CA, May 1996).

[2] Boneh, D., and Lipton, R. J. A Revocable Backup System. In *6th USENIX Security Symposium* (San Jose, CA, July 1996), pp. 91–96.

[3] Gong, L. A Secure Identity-Based Capability System. In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, California, May 1989), pp. 56–63.

[4] Lampson, B., Abadi, M., Burrows, M., and Wobber, E. Authentication in Distributed Systems: Theory and Practice. *ACM Transactions on Computer Systems 10*, 4 (November 1992), 265–310.

[5] Liebl, A. Authentication in Distributed Systems: A Bibliography. *ACM Operating Systems Review 27*, 4 (October 1993), 31–41.

[6] Needham, R. M. Denial of Service: An Example. *Communications of the ACM 37*, 11 (November 1994), 42–46.

[7] Rivest, R. L., and Lampson, B. SDSI—A Simple Distributed Security Infrastructure. `http://theory.lcs.mit.edu/~rivest/sdsi-10.ps`, 1996. (Version 1.0).

[8] Tanenbaum, A. S., van Renesse, R., van Stavern, H., Sharp, G. J., Mullender, S. J., Jansen, J., and van Rossum, G. Experience with the Amoeba Distributed Operating System. *Communications of the ACM 33*, 12 (December 1990), 46–63.

[9] Zimmerman, P. *PGP User's Guide, Revised for PGP Version 2.6.1*, August 1994.