Toward A Safe and Secure Distributed Computing Environment

Jeffrey I. Schiller
Manager, Systems and Operations
Distributed Computing and Network Services

Introduction

The last few years have brought a revolution in computer networking. Computing has gone from batch oriented processing, to time-sharing and now we are entering the age of the distributed computing system. Distributed Computing is literally a new paradigm in computation. Rather then having one massive CPU which is shared among a number of users (either via batch or time-sharing) we now have a network of computers which work in a cooperative fashion to solve problems.

Computing in the past was dominated by centrally managed systems. In this new age of distributed computing, not only is processing distributed, but so is system management.

Providing protection for information in a distributed processing environment presents new and different challenges. However solutions do exist. This document describes the issues, proposes principals and discusses technology for a distributed computing security architecture for MIT.

Administrative Computer Security at MIT

Traditionally computer security for administrative systems at MIT has been based on the principal that only authorized users can physically get access to terminals. Dial up access is guarded with dial-back modems, modems which accept calls from authorized users and call them back at their known home phone numbers before granting them access to administrative information. In essence protection of administrative information has been accomplished by security at the perimeter of our systems. This perimeter security has been augmented with other security mechanisms, including access control restrictions on data and passwords on user accounts. However the primary protection was, and still is, physical access.

Isolation is becoming less and less an option

However a security perimeter is harder to define in a distributed environment where different departments, in fact different individuals, have administrative control over the pieces of the environment.

Instead of defining a security perimeter, we need to define security borders at many places within a distributed environment and arrange for appropriate security controls when information crosses these internal "borders."

We also need to include within our scope computing that has until recently not been considered, namely administrative computing that takes place within academic departments and research labs. Today each department sets up its own system, usually using whatever facilities that are available.

MIT is a leader in distributed systems security

MIT's Project Athena was one of the first and largest client/server distributed computing environment. From the very beginning the project took into account the security requirements imposed by the networked computing architecture that Athena was based on.

This work led to the development of the Kerberos cryptographically based authentication system. Recognized the world over as a leading technology, Kerberos permits client programs to authenticate their access to server systems without sending any information over the network that would permit a third party eavesdropper from learning any information that would permit him to then later impersonate the client.

However you may have heard about other distributed computing security technologies. One time passwords, digital signatures and token cards are all technologies that are also used to provide network security.

The following section describes the basic problems of providing security within a networked environment and shows how these various technologies address the different problems that arise.

<u>The Network Security Problem</u>

A traditional time-shared computing system can be protected with a security perimeter that in essence surrounds the entire system. Figure 1 graphically depicts such an arrangement.

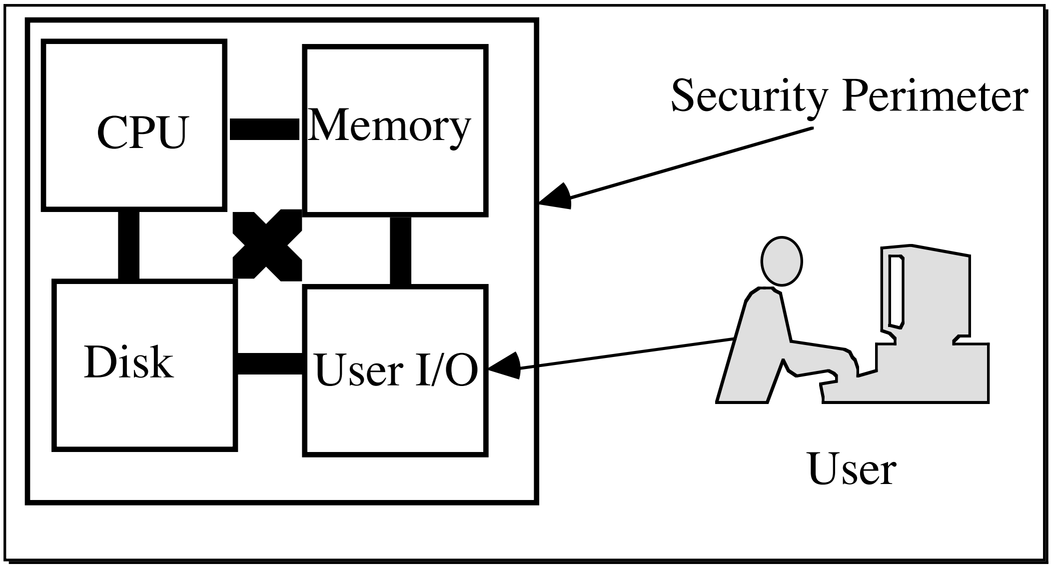

Figure 1

In a distributed computing environment we essentially separate the various functional components of a computer and connect them together by a network.

However the network is fundamentally different then the internal bus of a computer. Specifically whereas no one can snoop at the internal bus of a computer, a network is vulnerable to several possible security attacks. To protect against these attacks we have to place virtual security perimeters around the individual components of our "computer", we have to individually protect clients and servers, and we have to protect each and every data transaction that crosses the network.

Figure 2 depicts this situation graphically.

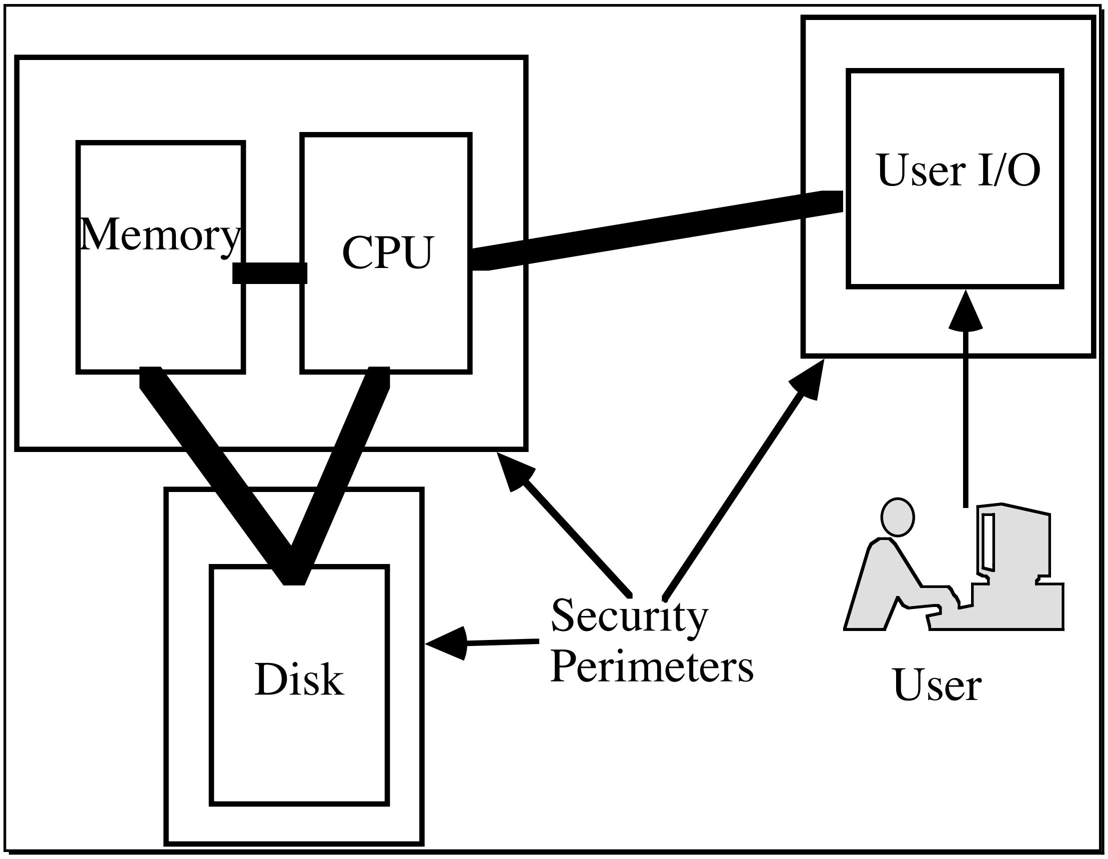

Figure 2

Attacks against networked computing come in two basic categories, active and passive.

Passive attacks, as their name implies, do not alter data as it travels across the network. Instead the passive attacker merely listens in on information, hoping to gain information that might be useful later. Confidential information may be revealed to a passive attacker or information such as passwords may be compromised that later permit the passive attacker to become move invasive.

An active network attack is an attack which actually interferes in some fashion with the distributed computing environment. This interference may be as simple as attempting to login with guessed passwords or it may be a sophisticated attack whereby legitimate information is modified while it traverses the network.

As networks become more ubiquitous and more individuals have access to them, the level of sophistication of actually witnessed attacks increases. For example two years ago the primary attack on the Internet was a "Dictionary Attack."

Two Years Ago: Dictionary Attacks

The Dictionary Attack was a mostly passive attack. Basically the attacker would somehow obtain a password file from a target system and attempt to "crack" the contained encrypted passwords off-line. This attack was facilitated because many networked computer systems permitted outsiders access to their password files. The encryption of the passwords was thought to provide sufficient protection against intrusion.

Normally this is the case, the best known attack against a password file filled with encrypted passwords is one where the attacker guesses a password, encrypts it with the same algorithm used to store passwords in the file, and then compares the result to see if the guess was correct. For a typical 8 character password, where lower case and numeric characters are permitted, up to 36^8 passwords may need to be tried. That is 2.8 ***trillion*** possible combinations. However few people use passwords that contain a random combination of letters and numerals. Many people choose passwords which are words that you might find in a dictionary. A "Dictionary Attack" is then an attack where the attacker doesn't try every 8 character combination but instead only tries all the words in a dictionary. Special dictionaries have been constructed that contain not only words (English or other languages) but also "hack" phrases, misspellings and proper names. Such dictionaries typically contain about 200,000 entries and are usually very successfully used to guess passwords. 200,000 is a number that is a lot smaller then 2.8 trillion!

That was two years ago. Since then many modern systems have been modified to make it harder to obtain unauthorized copies of password files. In fact the chain of events is representative of many security scenarios. First you have a security "vulnerability", in this case the password files with the encrypted passwords. Then you have a "threat", a person or institution that decides to take advantage of a vulnerability. The "threat" mounts an "attack", an actual attempt to make use of the vulnerability. Eventually a "countermeasure" is developed, in this case making password files unreadable to unauthorized users.

One Year Ago: System Modification

Last year we saw on the Internet the first of the serious active attacks. This attack involved modifying legitimate system programs so that in addition to their intended function they also recorded passwords for later harvest by the attacker. In security parlance we had a situation where an attacker installed a "Trojan Horse."

On compromised systems the "login" program and the "telnet"[1] program were modified to record user names and passwords in a file that the attacker

---

[1]Telnet is the name of the program that permit remote login on Internet systems.

hid on the compromised system. These log files gave the attackers many account names and corresponding passwords that permitted them to then break into other systems and place Trojan Horses there. How the first system was compromised may never be known, but once this technique was put into action thousands of computers have been effected.

Why was it so effective? One answer is that many people have accounts on multiple computers. Because mere mortals can only memorize only so many passwords (after all, most people are told not to write their passwords down), people tend to use the same password on many different systems. Once an attacker found your password he/she could then use this knowledge to get into other systems.

Today's Problem: Password Sniffers

Recently the Internet has been attacked with a more sophisticated attack. This attack obtained national attention and is the subject of congressional hearings.

Basically the password "sniffing" attack (as it is called now) is a version of the Trojan Horse attacks of last year, with a twist. Last year's attack only broke into systems that were directly used by people whose password was compromised on a previously attacked system. Systems that were not used by people who used a lot of other systems were reasonably safe. For example if an administrator used a PC or Macintosh to get to an administrative application such as EREQ, they were not subject to having their password compromised simply because research machines in their department may have been compromised (unless they themselves used those research machines and used the same password as they use on EREQ or they telneted to EREQ from the research machine).

The new twist this year is that the Trojan Horse programs not only steal passwords on the effected system, but they also install a monitor program that looks at the local area network which is connected to the compromised system and steal password from it. If a compromised system is connected to a network with 100 other systems on it, *all* of those systems are now compromised as well.

Sophistication Not Required

One misconception that some people have is that it requires a sophisticated cracker to perform these types of attacks. Turns out this isn't true. Although the first attack (or more properly the programs required to make the attack) requires some sophistication, once the programs are written they are shared over cracker bulletin board systems and other means. Literally overnight hundreds of neophyte crackers can be in possession of the necessary programs to launch such an attack. Very often these programs are very easy to use.

What do we do?

As the paragraphs above demonstrate, we are at risk. We need to do two important things. The first thing we need to do is recognize that simple passwords are not longer viable as an authentication strategy.[2]

The second thing we need to do is move toward "strong" authentication and security systems. The word "strong" has a technical meaning here. A strong system is one which cannot be compromised by simple means. Typically breaking a strong system requires solving an intractable mathematical problem or breaking a ciphering system such as the Data Encryption Standard (DES).

The use of strong systems minimizes the iterative loop of "vulnerability" -> "attack" -> "countermeasure" which is characteristic of security systems today.

Principals

The first step is to define a set of principals for guiding how we provide security services within the MIT community. What follows are my proposed set of principals:

1. All individuals have names

Each individual who in any way interacts with information at MIT needs to have an authenticated name. We should provide names (accounts) to whomever requires access for whatever reason. Failing to do so results in people sharing their accounts (names). From a security standpoint this is bad business. People who share their identities have less accountability then those who do not. Although within the Athena environment we have a rule against sharing a password, people do so anyway. When an inappropriate action is traced back to an individual they can often dodge the consequences by stating that many people have their password (their "identity").

It is not sufficient to have a rule, we must develop systems and policies that make it sufficiently easy for individuals to have their own "name" within the system so that no one feels obligated to share their password as an expedient way of solving a particular problem.

Although this should sound like an obviously good thing to do. Giving each person their own name often requires that they be granted a computer "account". This granting implies the granting of access to scarce resources

[2]Simple passwords are those which are sent over the network in the clear. There are more sophisticated authentication systems, which we will discuss later, that still prompt a user for a passoword. However these systems are not *simple* password systems because they do not expose the password itself to interception.

which is often used as the reason why accounts (names) cannot be granted trivially.

As we move toward more modern computing architectures, the granting of an authenticated name (called a "principal" in the Kerberos world) does not usually imply that scarce resources are also being committed.

To this end it is important that we not use the fact that someone has a name known to us as a way to discriminate for purposes of access control. In other words we should grant names readily, and we should not assign any meaning to having a name.

One good example where we can easily get tripped up is for us to license a software package for use at MIT and set it up so that only people with MIT Kerberos principals can access the software. This implies that all people with Kerberos principals are members of the MIT community based on the definition of MIT community that appears in the software license agreement. We should not make such implied assumptions.

2. All transactions are individually authenticated

Any time information traverses the network the receiving software needs to check the authentication of the transaction. Many systems only check authentication at login time. From that point forward all information on the established network "connection" is considered to be authentic.

This is not a good assumption as we move into the future. Although today we have not yet seen connection "stealing" attacks on the Internet, it doesn't mean that they won't happen at some point.

Last year Novell was embarrassed by a security problem within Netware 3.1X. Turns out the nature of this problem was that if a "supervisory" user was logged in, a cracker could run a program (distributed by some folks at the Delft University of Technology in Delft, The Netherlands) that would steal the supervisor's connection and use it to send a command that would grant them, the cracker, supervisor access.

Netware 4.X solves this problem by using a cryptographically based authentication system (similar to Kerberos) to authenticate every network transaction. We should do likewise.

3. Use "strong" mechanisms wherever possible

4. Attempt to minimize vulnerabilities rather then use extensive countermeasures

By basing our security on "string" mechanisms we can avoid the attack, countermeasure loop. This is important because one of the intermediate steps in the attack, countermeasure loop is one where you *are* compromised!

5. Security Technology will be applied where it fits best

Different security technologies are available. In some cases different technologies are direct competitors, solving the same set of problems. However there are complimentary sets of technology where each technology addresses a somewhat different problems. We should use technology appropriately to meet the requirements of our applications. One size does not necessary fit all.

6. Our information systems will be available to a wide community

7. Security will not be enforced by physically limiting who can gain access to network "entry points"

We need to realize that we can no longer protect our environment by denying physical access to "outsiders." The very definition of insider vs. outsider is unclear as we extend our administrative computing environment to departments and even individual students and employees.

Client Server Model

One of the fundamental differences between client server computing and more "traditional" computing has to do with the format of information that traverses a network. In the more "traditional" approach what traverses the network are keyboard input (what you type) and screenfuls of information (which is displayed on your terminal).

In client/server systems the user interface (the keyboard and screen) are part of a client computer system that interacts with the human user. Often this interaction is enhanced by the use of a graphical user interface (GUI). Client applications then make discrete network requests to a server system which acts upon them and returns a discrete response. These discrete transactions are usually "binary" requests that follow a protocol which is defined by the client/server system designer.

The discrete aspect of these requests make them well suited for the application of our principal of "all transactions are individually authenticated." Consider what would happen if we attempted to apply this principal to keyboard typing. We would have to provide authentication information for each keystroke. Adding authentication often comes with some overhead. Applying that overhead on every keystroke would probably come at an unacceptable performance cost.

However client/server transactions are usually large enough that the cost of adding authentication is reasonable.

However even in a client/server worlds we have to consider the case where the client processor and the end user are separated by a more traditional "telnet" or login connection. This may happen for example when an MIT person is traveling and accessing our administrative systems from a location where the client application is not available, but a "telnet" connection (remote login) is. We need to provide a solution to this authentication problem that does not depend on sending clear text (stealable) passwords over the network.

Three modes of authentication

In essence there are three distinct authentication problems that we are likely to encounter. They are represented graphically in figure 3.

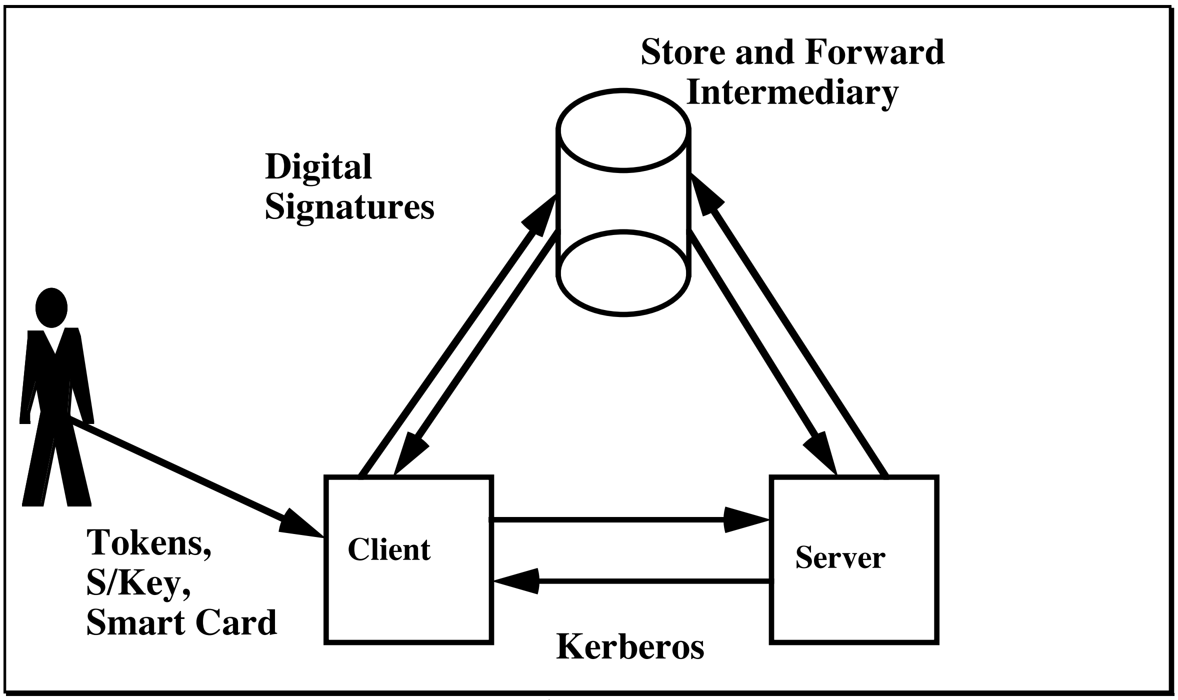

Figure 3.

Person to client system

The "person to client system" problem is the traditional "telnet" (remote login) connection. The key definition of this scenario is where the end user does not have access to local computation. Security and authentication technology that requires complicated computations by the end user may not be used because no local computation is available.

There are two approaches to providing authentication in this scenario. The first is to use a "token card". A token card is a card, typically credit card sized, that has a display an optionally a keyboard on it. To login a person types in the value displayed on the card (which in some models changes over time). If a password sniffer learns this value, it is of no consequence for that value will never be accepted again.

The second approach is to use a "one time password" system. Like its name implies a password is only valid for one login. Once used it is no longer useful to a password sniffer. In its most primitive form a one time password system requires each person to carry around with them a list of passwords. However advanced technology such as S/Key from Bellcore (freely available) supports either a manual list of passwords, but also a application that a person may run on a laptop computer which computes the necessary password when needed.

Client system to server system, real time

When a client communicates with a server in real time, cryptographic solutions such as Kerberos come into play. With Kerberos a person "logs" into their local client system. At login time a DES key is exchanged between the client workstation and the server system. This key is used to compute cryptographic checksums on each data transaction. This checksum permits the server system to verify that a request came from a legitimate client and similarly the client can confirm that the response came from the appropriate server. A person (read: cracker) who does not have access to the shared "session" DES key cannot forge these checksums.

The exchanged DES key may also be used to encrypt data between client and server if the confidentiality of the data needs to be protected. With Kerberos the only cost to doing full blown data encryption is the time that is required by the CPUs in the client and server systems to perform the actual encryption.

Because session keys issued by Kerberos (in what are called kerberos "tickets") are only valid for a period of hours, Kerberos is best suited for real time client/server applications where the client communicates directly with the server system.

Client system to other systems, store and forward

Many administrative applications are suited to a less real-time approach. For example electronic mail may be used as a transport for routing electronic documents such as requisitions, request for personnel and other similar documents.

In this problem space a very important technology can be brought into play, Digital Signatures. Invented by researchers at Stanford and MIT in the late 1970's, Digital Signatures make it possible to "sign" documents electronically in an unforgeable manner. Such signatures can be verified hours, days and even years after they were first produced.

A Word or two about Digital Signatures

Some special mention is appropriate here about Digital Signatures. This is because they are some of the newest technology to come to the computer security arena, and because their application and implications are both subtle and profound.

A digital signature is created by making use of what is called "Public Key" encryption technology. Conventional cryptography makes use of a common key to both encrypt and decrypt information. In public key cryptography two keys are employed. One key is used to perform one operation and the other key can then undo that operation. For example if one key is used to encrypt information, then the other key is used to decrypt information.

What is novel about public key cryptography is that having one of the keys, the "public" key does not give one knowledge of the corresponding private key.

An example may make this clear. Suppose I want you to send me confidential information. I can send you, in the clear, my public key. You can then encipher the information using my public key and send me the resulting output. I can then use my private key, which only I know, to decipher the information.

Where digital signatures arise is where I take my private key and use it to *encrypt* information. Anyone who then has my public key (which I may publish for this purpose) can then use it to decrypt the information. If what they get is not nonsense, then they know that only I, because only I have my private key, could have authored the information.

Modern use of public key cryptography makes use of other algorithms such as one time hash algorithms in order to make digital signatures which are the same size regardless of the size of the information document being signed.

What makes this so important is that it enables a new paradigm in information security.

The Old Paradigm

Prior to public key technology and digital signatures the only way to know if information within a computer system had been tampered with relied upon running the computer system in a secure manner.

This depends on a vigilant technical staff. Documents that have been on-line a long time become more vulnerable the longer they are on-line. Of course frequent system backup and the maintenance of archival backups aids in proving later that a particular document was not altered.

However proving that a document hasn't been altered (when you are not certain) involves making arguments about the relative security of the computer the document is stored in. It may be necessary to reload backup tapes to verify a document (and of course one then has to argue about the integrity of the storage facility where the tapes are kept).

The New Paradigm

In the new paradigm of digital signatures, it no longer matters where a document is stored. The digital signature itself proves who signed a document and that it has not been tampered with in storage. The security of the storage medium is no longer a concern.

What is important is that the private key used to make a signature has not been compromised. However a key is a much smaller data element to secure. It is also worth noting that we can periodically destroy private keys (and of course create a new public/private key pair for future use) while preserving the public keys and the documents that are signed. In this fashion it becomes impossible to forge a document.

An example: A Requisition

A traditional requisitioning system, like the one in use at MIT today, depends on the security of the computers used for storing requisitions in order to prove later that a particular individual wrote a particular requisition.[3]

However using digital signatures end-users can generate an electronic requisition document and then sign them on their own personal computers, assuming that their private key is stored only on their personal computers. Using their public key, purchasing officials can verify the originator of the electronic requisition and act upon it. If in the future the requisition is questioned, the presence of the digital signature proves who issued it. Not even the staff of the purchasing office, no matter how high, mighty or technically skilled, can forge a requisition using these techniques.

Tying them Together

Although these three approaches to providing security and authentication address different environments, they are not antagonistic. In fact a good distributed computing environment should make use of all three types of techniques.

It is possible to design an environment where the S/Key one time password scheme may be used by an individual to prove their identity to a client system, and as part of the exchange Kerberos tickets are deposited on the

---

[3]Of course there are other non-electronic confirmation techniques that are used by purchasing. However the point of this example is to show how digital signatures may be used, so we will not cover the non-electronic aspects.

client. These Kerberos tickets can then be used by the client for authentication to the various server computers that it makes use of.

Similarly Kerberos authentication may be used to run a service which certifies public keys for the use of digital signatures.[4]

[4] Knowing a public key permits you to verify a signature on a digital document. However before you can know that a particular person signed the document you need to have proof that the public key you have does in fact belong to the named person.