

5 Authorization, Authentication, Accountability, and Availability Technologies – The 4 A's

5.1	INTRODUCTION	5-2
5.2	AUTHENTICATION	5-3
5.2.1	Overview.....	5-3
5.2.2	IDs and Passwords	5-4
5.2.3	Token Cards: Secure ID, Smart Cards, and Java Rings.....	5-5
5.2.4	Biometrics: Fingerprints, Eye Scans, and Speech Recognition.....	5-6
5.2.5	Using Cryptography for Authentication –Digital Signatures and Session Keys ..	5-7
5.2.6	Digital Certificates for Authentication.....	5-8
5.2.7	Using Kerberos for Authentication	5-10
5.2.8	Single Signon (SSO) Systems.....	5-10
5.3	AUTHORIZATION AND INTRUSION DETECTION	5-12
5.3.1	Authorization and Access Controls	5-12
5.3.2	Intrusion Detection	5-13
5.3.3	Real-time Intrusion Detection Systems (IDSs)	5-13
5.3.4	Data Mining for Intrusion Detection	5-15
5.4	ACCOUNTABILITY AND NON-REPUDIATION	5-17
5.4.1	Overview.....	5-17
5.4.2	Non-Repudiation – The Legal Meaning	5-18
5.4.3	Non-Repudiation – The Crypto-Technical Meaning.....	5-18
5.4.4	Technical Vulnerabilities for NR in Existing Systems.....	5-19
5.4.5	Technologies and Approaches for Non-Repudiation.....	5-20
5.5	AVAILABILITY AND INTRUSION TOLERANCE	5-22
5.5.1	High Availability – Dealing With Natural and Denial of Service Attacks	5-22
5.5.2	Denial of Service Attacks.....	5-23
5.5.3	High Availability, Fault Tolerance, and Intrusion Tolerance	5-24
5.5.4	Replication as a Backbone for Increased Availability	5-26
5.5.5	FRS (Fragmentation, Replication, Scattering).....	5-27
5.5.6	Fragmentation Considerations	5-28
5.5.7	Scattering Considerations	5-28
5.5.8	Combining FRS with Cryptography	5-28
5.5.9	Special Considerations	5-29
5.5.10	The Reality Check on FRS.....	5-29
5.6	SHORT CASE STUDIES AND EXAMPLES	5-29

5.6.1	<i>Australian Agency Uses Digital Certificates to Sign Contract with Fujitsu</i>	5-29
5.6.2	<i>How Computer-Savvy Investigators Operate</i>	5-30
5.6.3	<i>City of Edmonton Upgrades Access Control</i>	5-31
5.7	CONCLUSIONS	5-32
5.8	SUGGESTED REVIEW QUESTIONS	5-32

5.1 Introduction

Authentication, authorization, and accountability (AAA) represent an important aspect of system security. We have also added availability, the 4th A to this equation. As mentioned previously, PIA3 indicates security and PIA4 indicates intrusion tolerance. This chapter reviews the basic technologies and approaches needed to support the following questions related to the 4A's:

- Authentication: Who are you and how do I know that you are who you say you are?
- Authorization: What rights do you have ?
- Accountability: What have you done or not done?
- Availability: Can you do what you want to do?

Chapter Highlights

- The 4A's are important aspects of system security and intrusion tolerance.
- Authentication technologies include the following;
 - ID, also known as a personal identification number (PIN), and a password.
 - Token cards such as secure ID, smart cards, and Java rings.
 - Biometrics such as fingerprints, speech recognition, and eyeball scans.
 - Cryptographic techniques such as digital certificates and digital signatures.
- Multi-factor authentication combines more than one technology, e.g., PIN plus secure ID card, for stronger authentication.
- Authorization is concerned with assuring that only permitted users can access a particular system resource.
- Authorization relies heavily on access control, typically enforced through access control lists (ACLs).
- Intrusion detection is closely related to authorization because intruders are essentially unauthorized users.
- Intrusion detection systems (IDSs) use real-time techniques as well as data mining to capture and detect intruder behavior.
- Intrusion detection systems are based on two major principles:
 - Anomaly detection: tries to determine whether a deviation from an established normal profile can be flagged as an intrusion.
 - Signature detection (also known as misuse detection) uses patterns of known intrusion to match and identify an intrusion.
 - Intrusion detection is an evolving area of research and development.
 - Accountability is synonymous to answerability and indicates responsibility.
 - Accountability requires tracking who or what accessed and/or made changes to the

system.

- Logs and audit trails are used to support accountability.
- Non-repudiation (NR) is the ability to provide proof of the origin or delivery of data.
- Legal and cryptographic technical views do not coincide.
- In the paper-based environment, the signatory has complete control over the signing mechanism.
- In the digital environments the signatory has to rely on the cryptographic technologies, and documents have to be transmitted over networks and then stored on computers – all subject to a variety of attacks.
- Main technologies to support NR consist of audit trains, digital signatures, digital certificates, and a trusted computing environment.
- Availability: percentage of time a system can be used by a user (human or program).
- Hackers and intruders can make a system unavailable by launching *denial of service* attacks.
- Replication is a common approach. A Fragmentation-Redundancy-Scattering (FRS) scheme is a good approach to increase availability plus security.

5.2 Authentication

5.2.1 Overview

Simply stated, authentication is the process of proving someone is who she claims she is. In practice, authentication is synonymous to positive identification. As we will see, authentication is also closely related to authorization. Authentication is required to limit access to resources, to identify participants in transactions, and to create seamless personalization of information based on identity.

It is naturally important to make sure that only the right users access and manipulate the right information. Digital enterprises heavily rely on remote communications and therefore require that all parties involved authenticate one another. In e-business, it is crucial to identify and authenticate the consumers who buy your products or services, employees who access internal systems from remote locations via the public Internet, or business partners who are tightly integrated into your supply chain and ERP systems.

Authentication can be classified in terms of the following factors described in the next sections:

- Something you *know*: an ID, also known as a personal identification number (PIN), and a password.
- Something you *have*: token cards such as secure ID, smart cards, and Java rings.
- Something you *are*: fingerprints, speech recognition, or other biometric identifications.
- Something you *belong* to: digital certificates and digital signatures indicate, for example, that you belong to a digital enterprise infrastructure that uses cryptography to identify you.

Each one of these techniques, discussed in the following sections, has its strengths and weaknesses, as we will discover. To provide stronger authentication, some of these factors can be combined. For example, a two-factor authentication uses a PIN and a secure ID card to access a network. This is very common in many VPNs and is also used in ATM banking. You first insert your ATM card (something you have) and then enter a PIN (something you know). A really strong authentication could combine all three: a PIN, a secure ID, plus fingerprints. Using multiple factors for authentication is known as *multi-factor authentication* (obviously!).

Theoretically, you can design an extremely strong authentication system by using 5 or 6 factors (a PIN, a password, a secure ID card, an eye scan, and a fingerprint). However, there are a few tradeoffs:

- Cost: Supporting more factors increases the cost (in dollars and time) of supporting a system.
- User convenience: More factors create more hassle for the users and decrease the usability of a system.

Despite these reasons, strong authentication should be used as much as possible. One of the main drivers is *deferred liability*, a relatively recent development [Andress 2002]. For example, if a computer owned by your company is not well protected and a hacker uses this computer to launch an attack on another company, then your company could be liable for this attack. The current law holds that the third party can not only sue the perpetrator (the hacker) but also other parties involved in the act, including the party that served as a jumping-off point (your company). This type of liability has instilled fear in many organizations.

5.2.2 IDs and Passwords

For authentication, a large number of systems employ the venerable **user ID and password (PW)** as a basis for authentication. We are all used to accessing emails, websites, ISPs, applications, databases, and desktop computers by using IDs and passwords. The major problem with ID-PW authentication is that it can be hacked relatively easily because people choose PWs that are somewhat obvious to guess. This limitation can be addressed by using strong passwords that are at least 7 digits long, plus requiring use of upper- and lower-case characters, numeric codes, special characters, and words that are not part of proper names or common dictionaries. In addition, strong PWs typically expire frequently to irritate the hackers. Some systems use passphrases instead of passwords for added security (see the sidebar “Passwords versus Passphrases”).

It is important not to send ID and password in clear text (un-encrypted). Clear text PWs can be read by intruders and are not safe even if you use a 200-character PW or a passphrase with very contorted numbers.

Due to the known problems with IDs and passwords (i.e., hackers guessing the passwords or reading them as clear text), some applications choose to make use of one-time passwords. However, the use of such one-time passwords often requires the deployment of token cards such as secure ID or smart cards. Deployment and maintenance of these cards, as discussed later, is an expensive and labor intensive effort. This is why some systems use cryptographic techniques such as session keys, digital

signatures, and digital certificates. Many good authentication systems use a combination of ID-PW, token cards, and cryptography.

Passwords versus Passphrases

A *password* is a unique string of characters that a user types in as an identification code. A *passphrase* is a longer version of a password, and in theory, a more secure one. Passphrases are typically composed of multiple words; thus they are more secure against standard *dictionary attacks*, where the attacker tries all the words in the dictionary in an attempt to determine the password. For security purposes, passphrases should be relatively long and contain a mixture of upper and lowercase letters, numeric and punctuation characters. Some packages, such as PGP, use a passphrase instead of a password to encrypt the private key on the user's machine.

5.2.3 Token Cards: Secure ID, Smart Cards, and Java Rings

Secure ID cards have become a de facto standard for token-based authentication. A secure ID card is a small hand-held device with a LCD that shows a number that changes every minute. The number is a unique, one-time key that can be used to authenticate the user because only the owner of the card can know this number. The other player in secure ID is an Authentication Server (AS) that generates the same number as the secure card through software. The AS is preprogrammed so that the same keys are generated on both sides (i.e., synchronized), based on a common key such as an employee number. A secure ID is used for authentication as follows:

- An employee is issued a secure ID card by a company. The card is unique to each employee. At the same time, the AS is synchronized with the employee ID.
- The employee accesses the company system through dial-up or other remote access mechanisms.
- The system prompts the employee for an ID. The employee types his or her ID (usually an employee number).
- The system then asks for a token key. The employee looks at his or her token card and enters the unique key being shown by the secure ID card.
- The AS on the other side compares the numbers typed by the user with the ones generated by the AS. If the numbers match, the user is authenticated, otherwise not.

The key generated by secure ID can be used just for authentication or it can be used as a key for encryption. This is common in VPNs.

The main criticism of secure ID is that it requires yet another device for us to carry around. For this reason, secure ID functionality can be imbedded in cellular phones, laptops and smart cards if needed.

Smart cards look very much like credit cards but contain a chip for special program processing. Depending on the type of program that runs on the chip, smart cards can be used for authentication of users, credit card processing, and many other transactions. It is

possible that in the future, we will have only one card that will behave as our ID, driver's license, and several credit cards.

Smart cards can be very simple memory cards which only keep track of some information. For example, Metro cards in New York city can be purchased for a fixed amount (\$20, \$30, \$40). Every time a customer inserts the Metro card through a train entrance, it deducts \$2 from the card and updates the balance on the card. Phone cards also use the same principle. Other smart cards come with a complete microprocessor that can run a program. An example is a card that downloads and runs a Java program to allow the same card to behave as Visa, Mastercard, American Express, or other credit card.

Smart cards are potentially very powerful – their use is only limited by the imagination of developers to build small programs that can be downloaded and run on the chip. As mentioned previously, one smart card can potentially replace your entire wallet. Although the smart cards have a great deal of potential, they have not been as popular as initially thought. There are several reasons for this:

- Smart cards need special readers that require extra investment for the participants.
- Standards are needed so that different services can be provided from one card. Otherwise, we may end up carrying different smart cards for different services (not very smart!).
- The cost of smart cards should be reduced.

A great deal of information about smart cards can be obtained from the website (www.smartcardcentral.com).

Java rings were developed by Sun Microsystems as a convenient token device. A Java ring looks like a regular ring (slightly thicker than the wedding ring but closer to a college ring). The rings have a chip that can run Java code, so in principle Java rings are similar to smart cards. These rings can be used conveniently for quick ID checks through electronic doors and other detectors. A Dallas-based company has commercialized the Java ring concept under the name “iButton” (www.ibuttob.com).

5.2.4 Biometrics: Fingerprints, Eye Scans, and Speech Recognition

The purpose of biometrics is to use physical attributes of a person for positive identification. The main question is what parts to use. The choice depends on the ease of use, comfort to the users, and the accuracy of the technology in uniquely identifying the individuals. Examples of biometrics used so far are:

- **Fingerprints.** This is one of the oldest biometric identification systems and has been used in police and legal cases for centuries. Significant improvements have been made in fingerprinting over the years (see the sidebar “Improvements in Fingerprinting Technologies”).
- **Eye scans.** These include retina scans and iris (around pupil) scans. These are human attributes that are very unique to us and can be used for identification. Recent developments in eye scan technologies have led to extremely accurate systems that are used in high security areas.
- **Voice patterns.** Different humans have different voice patterns that can be identified through sophisticated speech recognition systems for identification.

- **Signature dynamics and handwriting.** We have different writing styles that can be recognized by handwriting expert systems.
- **Other features** such as shape of a face, height, and weight can be used in conjunction with other factors for a stronger authentication.

Many commercial systems are available and are becoming available for biometric identification. Some security experts argue that biometrics represent the only reliable identification systems. A major advantage of biometric systems is that you do not have to remember or carry things to be identified. Suitability of biometric systems depends on:

- Accuracy (false rejections/acceptances)
- Speed and throughput
- Acceptability to users (social implications)
- Resistance to counterfeiting
- Storage requirements

Current biometric systems range from \$100 to \$400. Given all these considerations, fingerprints are the most cost-effective and accurate biometric systems. Eye scans are extremely accurate but are quite expensive. For additional information on this topic, see D. Richards, "Biometric Identification," published in *Information Security Management Handbook*, edited by H. Tipton and M. Kraus, 4th and 5th editions, Auerbach, 2000.

Improvements in Fingerprinting Technologies

Fingerprinting is one of the most difficult-to-defeat biometric access control methods. But acquiring and matching fingerprints is not easy when dealing with large populations. Traditionally, fingerprinting required ink prints – a messy, time-consuming, and inconvenient process. More recent systems use the optical fingerprinting technology which solves the problems connected with ink by snapping a picture of fingers. The main problem with this technology is that dirty fingers alter fingerprint images in ways that make it difficult for software systems to find accurate matches, thus making them unreliable. People have to wash their hands and the fingerprint recognition software has to be enhanced to adjust for inconsistencies created by dirt and grease on the fingerprint image. A better technology, developed in 1996 by Ultra-Scan, uses sound waves to scan fingerprints. Called ultra-sonic imaging, it is essentially the same technology used to scan pregnant women. Sound waves penetrate dirt, grease, and other contamination on the finger and create an accurate image of the fingerprint ridge structure. This provides a system that is easy to use and implement and is also quite accurate.

5.2.5 Using Cryptography for Authentication – Digital Signatures and Session Keys

Cryptography is used mainly for confidentiality and privacy by encrypting messages. However, public key cryptography provides a method for authentication also through *digital signatures*. A digital signature, discussed in the previous chapter, enables the recipient of information to verify the authenticity of the information's origin. Thus,

public key digital signatures provide *authentication*, and digital signature technology can be used to authenticate the source of a message instead of, or in addition to, the traditional ID and password. Digital signatures are especially considered vital in e-commerce transactions where funds are transferred and business commitments are made over the network.

Many systems enforce authentication by developing a *session key* that establishes the identity of partners at the start of a session and is used throughout a session. But then this session key needs to be encrypted. Should a private or public key system be used? Given the advantages and disadvantages of these approaches in practice (a private key is efficient but not very secure, and public keys are not efficient but secure), a public key system is used to exchange the session key between the two sides. Then this key is used in a private key system only for that session. Many current systems, such as SSL (Secure Socket Layer) use this technique.

But nothing is beyond controversy and debate (life would be so boring without controversy or debate!). For example, one of the primary technologies proposed for strong authentication is the digital signature, especially for the investment and finance communities. A possible advantage of digital signature technology concerns the issue of “non-repudiation” between the signer of an electronic document and a relying party. There is some controversy about digital signatures giving the same rights to the parties involved as compared to traditional signatures (see the discussion on non-repudiation in section 5.4).

5.2.6 Digital Certificates for Authentication

A digital certificate is data that functions much like a physical certificate such as a passport or driver’s license. A digital certificate includes a person’s public key along with other information that verifies that a key is genuine or valid. *Just as passports and driver’s licenses identify a person, a digital certificate is used to identify a person with his or her public key.* Thus a digital certificate can be used for authentication. For example, Joe can show his digital certificate, very much like he shows his passport, whenever he needs to prove that he really is Joe.

Digital certificates simplify the task of establishing whether a public key really belongs to the purported owner, and consists of:

- A public key
- Certificate information (“identity” information about the user, such as name, user ID, and so on)
- One or more digital signatures from CAs

Once created, the certificates can be stored in a secure *certificate server*, also called a *key server*. A certificate server usually provides some administrative features that enable a company to maintain its security policies – for example, allowing only those keys that meet certain requirements to be stored (see Figure 5-1).

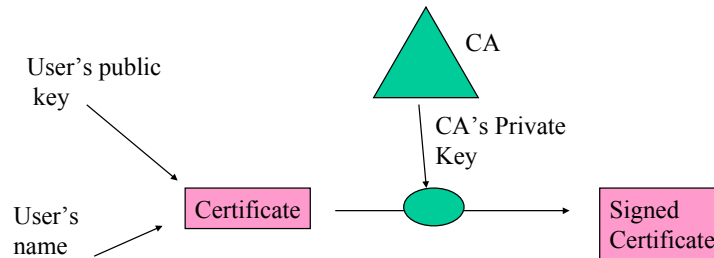


Figure 5-1: Digital Certificates

A digital certificate can exist in a number of different formats. X.509 is the most common format. All X.509 certificates comply with the ITU-T X.509 international standard; thus (theoretically) X.509 certificates created for one application can be used by any application complying with X.509. In practice, however, different companies have created their own extensions to X.509 certificates, not all of which work together. Although X.509 is used widely in many systems, perhaps the most widely visible use of X.509 certificates is in Web browsers. See the sidebar “X.509 Certificate Format” for a sample certificate format. As can be seen, a certificate is a text file (in reality, it is a record in a certificate server).

As we will discuss later, the CA system can be run as a function inside an organization or by an outside company such as VeriSign (www.verisign.com). A digital certificate system, wherever it runs, enables partners to verify each other's identity before proceeding. For example, a credit card user and merchant can validate that their digital certificates were issued by an authorized and trusted third party before they exchange data.

Certificates are created with a scheduled *validity period*: a start date/time and an expiration date/time as shown in the sidebar “X.509 Certificate Format.” When the certificate expires, it will no longer be valid, as the authenticity of its key/identification pair are no longer assured. There are also situations where it is necessary to invalidate (revoke) a certificate prior to its expiration date, such as when the certificate holder terminates employment with the company or suspects that the certificate's corresponding private key has been compromised. It is much more important to detect a revoked certificate than an expired one. Expired certificates are unusable, but do not carry the same threat of compromise as a revoked certificate.

To summarize, a digital certificate binds an entity's identification to its public key and is issued by the Certification Authority. Digital certificates, typically based on the X.509 standard, enable Internet applications and other users to verify the identity of an entity. Unfortunately, certificates produced by one vendor product may not inter-operate with another vendor's because X.509 does not define the formats of the certificate entries and other necessary provisions. PKIX, the X.509 standard by IETF, defines the contents of public key certificates and is intended to resolve these inter-operation issues.

X.509 Certificate Format

The X.509 standard defines the information that goes into the certificate. All X.509

certificates have the following data (many fields are self-explanatory):

- X.509 version number
- Certificate holder's public key
- Serial number of the certificate – a unique serial number to distinguish it from other certificates it issues
- Certificate holder's unique identifier – this is a unique name across the Internet. The uniqueness is achieved by several subsections that indicate the user's Common Name, Organizational Unit, Organization, and Country).
- Certificate's validity period – indicating when the certificate will expire
- Unique name of the certificate issuer – the unique name of the CA that signed the certificate
- Digital signature of the issuer
- Signature algorithm identifier – identifying the algorithm used by the CA to sign the certificate

5.2.7 Using Kerberos for Authentication

Kerberos (<http://www.mit.edu/kerberos/>) is a cryptographic authentication scheme developed at MIT. It uses a third-party authentication server to grant cryptographic “tokens” that authenticate users to a given service. Kerberos is used quite heavily for user authentication because it supports, in addition to the venerable user ID and password authentication, additional authentication schemes such as certificate-based public key systems, asymmetric-key cryptography, smart cards, and token cards. We will re-visit Kerberos in the next chapter.

Digital Certificates Versus ID/Passwords

IDs and passwords are commonly used for authentication and are easy to use and support. But passwords and user names carry a security risk as they can be guessed or cracked by adversaries. It can also be very difficult for the users to remember passwords and user names if they use multiple systems. Although digital certificates are expensive to set up, they have several benefits. Each user can be issued a unique digital certificate that can be recognized by multiple systems. In addition, personalized content can be delivered based on the information contained in the digital certificate, because a certificate could show what services are typically used by the holder. See the Prudential Case Study in the next chapter.

5.2.8 Single Signon (SSO) Systems

We all use different systems with different authentication systems requiring different signon IDs and PWs. These IDs and PWs expire at different times and impose different

requirements for IDs and PWs; thus it is not possible to use the same signon and ID everywhere. From an end-user point of view, it is highly desirable to have a single signon that can be used on multiple systems. From a system administrator point of view, it is also desirable to have single signons to avoid the problems of maintaining security on many systems with many different options.

A means of sharing the fact that authentication has been performed successfully is to allow “single signon.” An example is a travel portal offering destination information, flight schedules, sight-seeing tours, the ability to make reservations, and other services. To a customer, it should appear as a single website, but in fact different suppliers may be cooperating to provide the service. A customer should only need to authenticate once to enter the portal, and information on the successful authentication should be shared with the different underlying systems, with some validity period.

Different approaches for SSO have been developed and deployed over the years. A recent example is the Microsoft .NET Passport (see the sidebar “Microsoft .NET Passport for Single Signon”). The main concept underlying these systems is that of a security proxy that acts on behalf of the user to sign on to different systems. A security proxy is a program that stores the different signon IDs and PWs in its own storage and then produces these when a user needs to log on to a system. Where does the proxy reside? It can reside on the user client, a server, or both (see Figure 5-2). Most common browsers (user clients) such as Netscape and Internet Explorer act as your security proxies by simply remembering and prompting you with IDs and PWs. However, there is a potential security problem with client-based proxies – the proxy resides on your machine and anyone who can log on to your machine can also log on to all of your back-end systems. For this reason, a server-based proxy may be placed on an intermediate trusted machine. But the intermediate machine has to be very highly trusted. We will visit SSO again when we discuss Web and Web Services security.

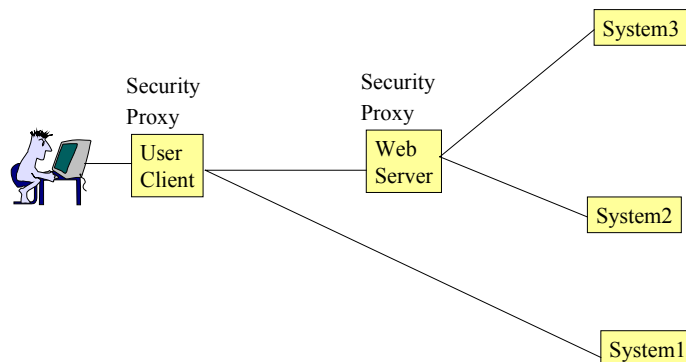


Figure 5-2: Conceptual View of Single Sign-On

Microsoft .NET Passport for Single Signon (SSO)

Microsoft .NET Passport was launched in 1999 to provide a single-signon (SSO) facility for Internet users. By now, it is one of the largest online authentication systems in the world because it eliminates the need for users to remember numerous passwords and signon names in the digital age. The .NET Passport is designed for users who want to establish their identity once and then move smoothly among various websites. It also

facilitates confirmation of the consumer's identity as the consumer moves from one site to another. We will discuss Passport in a later chapter.

5.3 Authorization and Intrusion Detection

5.3.1 Authorization and Access Controls

Authorization is concerned with assuring that only permitted users can access a particular system resource. Authorization relies heavily on access control – the process of checking whether an authenticated user's privileges permit the execution of a particular operation on a particular protected resource. In addition, authentication is the foundation of authorization. For example, can Alice withdraw money from account zc-11-35? To authorize this withdrawal, first Alice has to be authenticated, then it has be checked whether Alice can withdraw from this account. The access control is typically enforced through access control lists (ACLs) that may look something like the following table.

Table 5-1: Sample Access Control List

User name	Resource Name	Access Type Allowed
Joe	Payroll	Read Only
Alice	Account zc-11-35	Read, Add, Withdraw
Sam	Customer Database	Read only
Tim	Inventory control	Read and Update

Scalability of ACLs is a major issue because modern applications may scale to dozens or hundreds of Web servers and potentially tens of millions of end users. The administration of ACLs can be very complex if they must be configured on each Web server system. Authorization to back-end data or subsystems must be handled as well, including systems that have existing authorization mechanisms. In addition, authorization to other key e-business resources such as objects and message queues must be incorporated.

Due to the complexity of managing ACLs, many applications provide access control on their own because it is not always possible to provide intra-application access control using Kerberos or public-key schemes. Some products have been released that make use of the Distributed Computing Environment (DCE) access control policies. These products, such as HP's Praesidium, make use of the fine-grained access control capabilities of DCE and link them to the deployment of Kerberos within a system. Other products such as the Tivoli Secureway Policy Director provide a centralized authorization service that is the point for administering access controls for Web servers, Web applications servers, firewalls, EJBs (Enterprise Java Beans), and other systems.

5.3.2 Intrusion Detection

Simply stated, an intrusion is an unauthorized action because an authorization is an agreement between two parties about some actions. For example, according to Table 5-1, Sam is authorized to only read a customer database. But if Sam attempts to update the customer database, then he is an intruder. From this point of view, hackers and other adversaries are intruders who attempt to eavesdrop and/or perform other activities that they are not authorized to perform. Naturally, there are several types of intrusions. Some examples include:

- stealing a password and impersonating its owner
- guessing a password by repeated attempts
- flooding a network and/or a server to cause denial of service
- abusing access privileges by internal users
- using pre-packed scripts, often found on the Internet, to attack a network

This list of intrusions is not exhaustive; and, in fact, the classification of all possible intrusions is an ongoing research topic in this area (especially due to the fact that new intrusion techniques are continuously discovered).

Good intrusion detection schemes emphasize early detection of intrusions for quick actions. Should an extranet, a corporate database, or any internal system be compromised, you need to detect that fact early, and take necessary actions to prevent the launching of further attacks into the private network.

Virus detection is an example of intrusion detection. Computer viruses can enter your systems in a variety of ways: via email attachments, from software installs, from files brought by employees from home, etc. They can quickly proliferate from system to system and user to user, and cause damage to data, applications and networks. Viruses must be quickly identified and isolated, and damage must be promptly repaired.

Two types of approaches are used for intrusion detection: real-time detection for immediate action, and the after-effect analysis through data mining. These two techniques are briefly reviewed in the next sections.

It should be noted that intrusion detection research has not provided a final solution preventing all types of intrusions. There is, however, a large set of detection principles that need to be used by networked systems. Plus, this set is in continuous expansion, as newer intrusion methods are discovered (notable examples are worldwide Internet attacks that have been largely covered by the press in the last few years). A basic tool that is used by most, if not all these detection principles, is using audit data, and, most typically, system access log data.

5.3.3 Real-time Intrusion Detection Systems (IDSs)

Intrusion detection systems (IDSs) are often considered to work analogously to “intrusion alarms,” as follows. First, every time an intrusion happens – or, more generally, whenever the security of the access to the system is compromised – the alarm sounds. Second, after the alarm sounds, some entity responds to the alarm in various possible ways: either some automatic software is executed that disallows the intruder any action, or some external authority (for instance, a system administrator) is informed.

These systems attempt to trigger alarms if they detect an intrusion. Intrusion detection systems (IDSs) are based on two major principles:

- **Anomaly detection:** determines whether a deviation from an established normal profile can be flagged as an intrusion. This principle identifies anomalies with possibly suspicious activities; a detector may give an alarm whenever it sees some unusual behavior of the traffic in question. The construction of such a detector starts by forming a structure as to what is considered “usual,” and then decides how to make the particular assessment that a specific action is not compatible with this structure. A reason why this principle may not be satisfactory is that it does not necessarily detect intrusions; or, the number of false alarms can be too high.
- **Signature detection** (also known as misuse detection): uses patterns of known intrusion to match and identify an intrusion. This principle identifies suspicious activities with behavior similar to previously observed intrusions. Detectors following this principle do not need to care about the normal behavior of the observed system; but they are not able to detect intrusions that are unknown generally or even just to the specific system.

In both cases the intrusion detection process goes through the following phases: first, audit data collected by sensors is analyzed according to a first set of criteria, then some output is generated according to a second set of criteria and passed to some external authority, and finally the authority decides according to a third set of criteria on some action to be taken. It is important to use data mining and knowledge discovery (discussed in the next section) to build better criteria in all phases of the intrusion detection process.

Intrusion Detection Systems (IDS) as currently available use anomaly detection and/or signature detection techniques. These and other types of detection schemes are embedded in *sensors* that operate at different levels of a system. Examples of the system level are:

- **Application-level IDS.** These sensors recognize that an unauthorized user is attempting to run an application or is actually running the application. The applications can be e-commerce applications, mobile applications, supply chain management systems, or any other modern or legacy applications. The sensors can detect this by a real-time analysis of logs or by challenging the sensitive application periodically.
- **Middleware-level IDS.** This topic has not received a great deal of attention in the past but is becoming increasingly important due to the vital role middleware services are playing in modern enterprises. These IDSs make sure that the directories that contain routing information are not contaminated or modified. In addition, XML Document Translation Definitions (DTDs), used in sensitive XML-based e-commerce, are monitored for corruption and modifications. The IDSs monitor errors or sudden changes on such systems and trigger alarms to invoke corrective actions.
- **Network-level IDS.** These IDSs attempt to sense intrusions mostly at the network routing and transmission levels. Intruders at this level can cause denials of service or eavesdrop on information transmission. These issues are especially acute in wireless networks because of the current weaknesses of network security (we will discuss this in detail in a later chapter). The IDSs at this level monitor the network traffic patterns and route the information to alternate routes if they suspect intrusions.

The current state of IDS is not as mature as it should be. A great deal of DARPA research on advanced IDSs that could detect and react to intrusions quickly is currently

underway. The future of the current generation of IDSs is also not clear (see the sidebar, “Are Standalone Intrusion Detection Systems Dead?”). We will look at salient results in parts III and IV of this book.

Are Standalone Intrusion Detection Systems Dead?

Many IDSs have been developed in the industry in the past few years. In a summer 2003 report called “Hype Cycle for Information Security, 2003,” Gartner stated that “intrusion detection systems (IDS) are a market failure.” Gartner is advocating that the technology be incorporated into other products such as firewalls instead of being implemented as a stand-alone solution. Some vendors have started renaming their IDS software as “intrusion prevention systems (IPS)” to emphasize active prevention of attacks instead of just alerts. However, this re-labeling is not helping either.

It does seem that IDS/IPS functionally is moving into firewalls, which are now performing sophisticated packet inspection in addition to antivirus activities. Companies such as ISS have introduced all-in-one security devices with combined firewalls and IDS capabilities. Most of these systems employ “deep packet inspection” techniques to closely examine the content of packets before letting them enter their corporate perimeter.

5.3.4 Data Mining for Intrusion Detection¹

A recently introduced approach to network intrusion detection is that of using data mining, in the following two-stage process. In the first stage, data mining algorithms are used to find useful information from audit data. In the second stage, this information is used to improve the design and success of intrusion detectors. Thus data mining software analyzes the data that has been collected in system logs, and from those, attempts to detect past intrusions and then use these findings to correct future intrusions. The data mining algorithms also use the two major principles (anomaly detection and signature detection) used in real-time IDSs. The main challenge is how to use the existing data mining techniques to detect intrusions.

The currently available data mining tools use a variety of underlying technologies such as neural networks, decision trees, statistical analysis, and machine learning to detect:

- associations (e.g., linking a user site with intrusions)
- sequencing (e.g., tying events together such as break-ins at certain times of day)
- classifications (e.g., recognizing patterns such as the attributes and profiles of intruders)
- forecasting (e.g., predicting future intrusions based on past patterns)

¹ The information in this section is based on a research project that I was involved in at Telcordia Technologies.

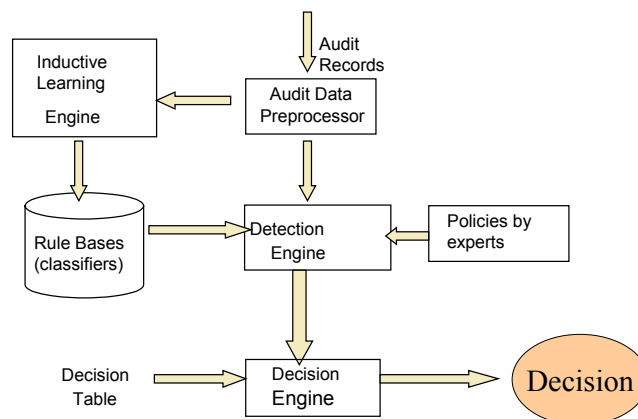
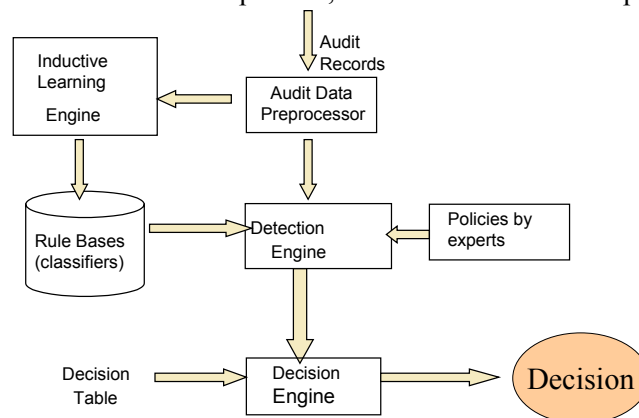


Figure 5-3: An Adaptive Model of Data Mining (based on [Lee 1999])

Data mining for IDS uses anomaly detection and/or signature detection techniques, as mentioned previously. However, encoding and maintaining of usage profiles and known intrusion patterns are mostly carried out via manual and ad hoc means. Not only new intrusion patterns are being invented at present, but “normal profiles” are changing as time progresses. For example, after September 11, 2001, new attitudes and models of intruders have been developed. These developments make the job of IDS’s even more difficult. It is best to use adaptive intrusion detection techniques for analyzing audit data. A natural candidate for such techniques is an artificial neural network, which not only takes into account historic intrusion patterns, but also learns and adapts new and/or



evolving patterns.

Figure 5-3, based on an extension of view presented by Lee [1999], illustrates the basic model. The actual data mining techniques used in this model may be a combination of clustering, classifications, neural networks, pattern recognition, and decision trees.

Due to space limitations and ongoing developments in this field, detailed discussion of these issues is beyond the scope of this book. The interested reader should pursue the literature shown in the sidebar, “Some References for Intrusion Detection.”

Some References for Intrusion Detection

- E. Bloedorn, A. Christiansen, W. Hill, C. Skorupka, L. Talbot and J. Tivel, “Data Mining for Network Intrusion Detection: How to Get Started,” available at http://www.mitre.org/support/papers/tech_papers_01/bloedorn_datamining/index.shtml.
- S. Axelsson, “Intrusion Detection System: A Survey and Taxonomy,” available at <http://citeseer.nj.nec.com/axelsson00intrusion.html>
- W. Lee, S. Stolfo, and K. Mok, “A Data Mining Framework for Building Intrusion Detection Models,” in Proc. of the 1999 IEEE Symposium on Security and Privacy.
- M. Goebel and L. Gruenwald, “A Survey of Data Mining and Knowledge Discovery Software Tools,” in *SIGKDD Explorations* 1, no. 1 (1999), 20-33.
- Piatetsky-Shapiro, Brachman, Khabaza, Kloesgen, and Simoudis, “An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications,” KDD-96, 89-95.
- E. King, *Data Warehousing and Data Mining: Implementing Strategic Knowledge Management*, Computer Technology Research, 2000.
- A. Berson et al., “Building Data Mining Applications for CRM”, McGraw Hill, 1999.
- U. Fayyad, et al, (ed.), *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

5.4 Accountability and Non-Repudiation

5.4.1 Overview

The term *accountability* is used to indicate responsibility to someone or for some activity. Accountability is synonymous to answerability; for example, in a corporation top management is accountable to the stockholders. In the security context, accountability means that the security system should be able to tell who did what, and when, and how. It is the ability of a system to keep track of who or what accessed and/or made changes to a system. Modern digital enterprises must provide assurance that the infrastructure and application resources, including computing platforms, networks, and data, are only being used by authorized individuals in the right way. This entails keeping track of the enterprise network and systems usage, and also requires that the communications between the consumer/business partners and the enterprise application – the path – is properly tracked.

Mechanisms to support accountability include reporting systems and logs, naturally. A secure system needs to log all attempts to access corporate resources to ensure that only authorized people are accessing the system. This logging can also facilitate management decisions by allowing analysis of usage patterns. A comprehensive, distributed logging and audit facility for Internet-based applications is needed in modern enterprises.

Non-repudiation (NR), the ability to provide proof of the origin or delivery of data, is an important aspect of accountability. This issue has gained popularity due to the increase in e-commerce of the use of digital signatures. A great deal of information on non-repudiation is becoming available at present. The following discussion is based on the thoughts by McCullagh and Caelli [McCullagh 1998, McCullagh 2000].

5.4.2 Non-Repudiation – The Legal Meaning

The term “non-repudiation” has legal as well as cryptographic meanings. The traditional legal meaning of “Non-Repudiation” is that an alleged signatory to a document is always able to repudiate (i.e., deny) a signature that has been attributed to him or her [McCullagh 1998]. The reasons for a repudiation of a traditional signature may include the following:

- The signature is a forgery.
- The signature was obtained via unconscionable conduct by a party to a transaction, fraud instigated by a third party, or undue influence exerted by a third party.

The general rule of evidence in a legal sense is that if a person denies a particular signature then it falls upon the relying party to prove that the signature is truly that of the person denying it [McCullagh 1998]. The term “deny” and the term “repudiate” are synonymous in this discussion. To overcome a false claim of non-repudiation, witnessing has been introduced in the common law. A witness is an independent adult who certifies to the signing of a document and thus reduces the risk of denials at a later date. Even with witnesses, the signatory can still deny the signature on other grounds such as undue coercion.

How should non-repudiation work in the digital economy? Basically, the current digital environment should not have different rules from those in the traditional paper-based environment. These rules have been developed and tested for centuries to protect all parties in a transaction.

In a traditional forged paper-based signature (also known as “wet signature”) case, the onus lies upon the party wishing to rely upon the signature. The relying party is required to establish that the signature is not a forgery. In particular, the relying party has to prove that the signature is in fact that of the alleged signatory if the alleged signatory disputes the signature as belonging to him or her.

5.4.3 Non-Repudiation – The Crypto-Technical Meaning

From a cryptographic point of view, the term “non-repudiation” in authentication means a service that provides proof of the integrity and origin of data, both in an unforgeable relationship, which can be verified by any third party at any time. NR protects the sender against a false denial by the recipient that the data has been received. It also protects the recipient against false denial by the sender that the data has been sent. In other words, a receiver cannot say that he/she never received the data and the sender cannot say that he/she never sent any data.

There are many problems related to NR in the digital environment. In the paper-based environment, the signatory has complete control over the signing mechanism and there is no reliance on any technology. This is simply not true in the digital environment because

the signatory has to rely on the cryptographic technologies – i.e., the public and private keys that are used to create the digital signatures. In addition, the signature and other documents have to be transmitted over networks and then stored on computers – all subject to a variety of attacks.

In addition, NR in digital environments either shifts the onus of proof from the recipient to the alleged signatory or entirely denies the signatory the right to repudiate a digital signature. For example, if a digital signature indicates that Joe's private key was used to create the digital signature, then Joe has the onus of proving that it is not his digital signature. Thus there is a shift in the burden of proof. This crypto-technical position does not correspond with what occurs in the paper-based environment [McCullagh 1998].

This shifts the burden to a Trusted Third Party (TTP) for verification of the digital signature. The role of TTPs has been formalized by the International Organization for Standardization (ISO) as regards to non-repudiation services [Granito 1997]. The purpose of non-repudiation is to provide verifiable proof or evidence of:

- Approval: Non-repudiation of approval service provides proof of who is responsible for approval of the content of a message.
- Sending: Non-repudiation of sending service provides proof of who sent a message.
- Submission: Non-repudiation of submission service provides proof that a delivery authority has accepted a message for transmission.
- Transport: Non-repudiation of transport service provides proof for the message originator that a delivery authority has given the message to the intended recipient.
- Receipt: Non-repudiation of receipt service provides proof that the recipient received a message.
- Knowledge: Non-repudiation of knowledge service provides proof that the recipient recognized the content of a received message.

In summary, the Electronic Commerce Environment (Article 13 Model Law) shifts the onus of proof to the signatory to prove that the digital signature is a forgery. However, in a paper-based environment, the onus of proof is upon the relying party to prove that the signature is not a forgery. This change in position is quite controversial, as can be imagined.

5.4.4 Technical Vulnerabilities for NR in Existing Systems

There are several vulnerabilities in the existing systems that make non-repudiation very difficult. Here are some examples:

- Most computers are permanently connected to the Internet through DSL, cable modem or other connections instead of dial-ups. The use of permanent IP addresses increases the vulnerability to attacks by hackers.
- Computers that are not located behind firewalls are increasingly exposed to outside attacks. However, many personal computers do not operate behind firewalls.
- Many viruses are being introduced on an ongoing basis. Most of these viruses are examples of mobile code that operate covertly on computer systems. Trojan horses are an example.
- Increased use of wireless communications further increases the chances of eavesdropping and integrity of information being transferred between source and destination.

For example, a virus could be designed to steal private keys. This is relatively easy if the private key is stored in a commonly known file, such as “PrivateKeys” and “PGPPrivateKeyRing.” This virus could search storage devices for the private keys and then FTP them to a remote location. To avoid detection, the program could turn off display functions to eliminate dialogue boxes and delete any relevant entries to be traced, before destroying itself. Security policies can be implemented to avoid this situation. For example, private keys could be stored in password protected files with non-obvious names, and programs could be denied the ability to start FTPs without an OK from a security system.

In these situations, how can a signatory deny that he or she did not sign the document? The alleged signatory has the onus of proof in demonstrating that it was not him or her who signed the document. In addition, how can a relying party prove that a particular signatory signed the document? Trusted computing techniques are needed to help in this area.

5.4.5 Technologies and Approaches for Non-Repudiation

As discussed previously, the main technologies to support accountability and non-repudiation are:

- Extensive logs and audit trails
- Digital signatures
- Certificate authorities
- Strong security measures in the computer systems to protect the private keys. These include firewalls, password protection, and obfuscation of directory names to avoid easy detection by mobile code.
- Secure communication channels between the parties to assure safe transfer of certificates. Thus, problems such as the “man in the middle” should be avoided.

Many of these topics are discussed under the general heading of “trusted computing.” Many definitions of trusted computing have been introduced since the 1970s. The best known are the Trusted Computer System Evaluation Criteria (TCSEC) – a collection of criteria to grade or rate the security offered by a computer system product (<http://www.radium.ncsc.mil/tpep/process/faq-sect4.html>). The TCSEC defines the following six fundamental requirements of any computer system that aims for a level of trustworthiness:

- Security Policy: There must be an explicit and well-defined security policy enforced by the system.
- Marking: Access control labels must be associated with objects.
- Identification: Individual subjects (users) must be identified.
- Accountability: Audit information must be selectively kept and protected so that actions affecting security can be traced to the responsible party.
- Assurance: Computer system must contain hardware and software mechanisms that can be independently evaluated to provide sufficient assurance that the system enforces the security requirements.
- Continuous Protection: The trusted mechanism enforcing these basic requirements must be continuously protected against tampering and unauthorized changes.

Other work has refined these requirements to further specify trusted computing. For example, the British ITSEC (www.itsec.gov.uk/) defined the following seven assurance levels to form a “trust hierarchy”:

E0; Inadequate assurance

E1: Informal description of architectural design of product/system exists and functional testing used to confirm target is met.

E2, or E1 plus: Informal description of detailed design exists:

- Evidence of functional testing to be evaluated
- Configuration control system exists
- Approved distribution process exists

E3, or E2 plus:

- Source code and/or schematics for hardware to be evaluated
- Evidence of testing of these must be evaluated

E4, or E3 plus:

- Underlying formal model of security policy supporting the security target exists; and,
- Security-enforcing functions, architectural design, and detailed design are specified in semi-formal style

E5, or E4 plus:

- Close correspondence between detailed design and software source code/engineering hardware design drawings.

E6, or E5 plus:

- Security-enforcing functions and architectural design must be specified formally – consistent with formal model of security policy.

To be trustworthy, the systems must operate at a minimum level of E3, which proves the functionality of the signing mechanism, thus preventing unauthorized access to the private key. Compliance with E3 also assures that the source code for digital signatures is evaluated, and thus it is possible to show that the signing mechanism will only perform the desired function and no other. Implementation of E3 and higher levels of assurance can ensure that the private key has not been stolen.

Development of technologies and environments to support non-repudiation is an ongoing area of research. For example, the paper titled “A Software Framework for Non-repudiation Service” by Sung Woo Tak and Eun Kyo Park² proposes a secure and efficient software framework for non-repudiation service based on an adaptive secure methodology. The paper proposes an explicit security framework that supports non-repudiation of service for a successful e-commerce transaction, and proposes an adaptive secure methodology to support secure and efficient non-repudiation of service in the proposed framework. Additional work is needed in this area.

² “A Software Framework for Non-repudiation Service” by Sung Woo Tak and Eun Kyo Park, *Information Systems Frontiers Journal*, Special Issue on Object-Oriented Client/server Internet Environments (January 2003), ed. A. Umar.

5.5 Availability and Intrusion Tolerance

5.5.1 High Availability – Dealing With Natural and Denial of service Attacks

As modern enterprises increasingly rely on digital networks for their revenue and operations, they need to take additional steps to ensure that their systems and applications are always available to support their business. The IT infrastructures of digital firms must provide a continuous level of service availability across all systems interconnected through a network. Firms such as those in the airline, financial, and telecommunications service industries with critical applications requiring on-line transaction processing have traditionally used robust computer systems for many years to ensure 100 percent availability. In online transaction processing systems, transactions entered online are immediately processed by the computer system, and numerous changes to databases, reporting, or requests for information occur each instant. Consider, for example, the impact of network or website failures on companies such as Amazon.com, where 100% of the business is conducted online.

From a user point of view, availability reflects the percentage of time a system can be accessed by a user (human or program). Many factors impact the availability of a system. For example, network transmission errors and component failures reduce the availability of a network to the user. Websites can be impacted by network failure, heavy Internet traffic, and exhausted server resources. Transmission errors can occur due to the distances between components, number of components and environmental factors (e.g., weather). The electronic causes can be white noise, impulses, crosstalk and attenuation. In wireless networks, fading and losses due to scattering and background noise can result in reducing the availability of the network to its users. Similarly, disk crashes and CPU failures can make a computer unavailable to the users.

In addition to the natural reasons for system unavailability, hackers and intruders can make a system unavailable by launching *denial of service* attacks (don't these people have anything better to do?). In these attacks, adversaries can flood a network segment, or tie up or crash a server so that the authorized users cannot use it. Several denial of service attacks have been launched over the years. These attacks reduce the availability of a system. See section 5.5.2 for further discussion.

System failures, interruptions, and downtime can translate into disgruntled customers, millions of dollars in lost sales, and the inability to perform critical internal transactions. The availability A of a system should be translated into business impact. For example, let us assume that the availability A of a large packet switching network for September 2003 is 0.97. This sounds good. However, let us translate this into potential business loss due to network "unavailability" of 0.03. Assuming that the network operates 24 hours a day and 7 days a week, then the network was unavailable for $0.03 \times 720 = 21.6$ hours per month (we are assuming 720 hours per month). This is almost a full day! Assuming that the network handles an average traffic of 5,000,000 packets a day, and the users pay 1 cent per packet to the network owner, this indicates a financial loss of almost \$50,000 to the network owner. Even if the unavailability could not be measured in terms of dollars,

the loss of one day of work activity is much clearer to the user than an availability figure of 0.97.

Although availability is not typically discussed in a security context, the denial of service attacks from the same type of hackers who also launch viruses have linked the two. This section reviews the commonly used techniques employed to address availability attacks.

5.5.2 Denial of Service Attacks

Denial of service (DoS) attacks generally fall into the following two categories:

- The attacks that crash parts of a system
- The attacks that may not crash a system but may keep it busy so that it cannot do any productive work

Some attacks, in the knowledge that a system cannot properly handle some requests, deliberately cause a system crash by sending those requests. For example, many older systems had flaws in their network stack and could not handle certain types of packets. The attackers would continually send these packets to the network devices, causing crashes. To keep systems busy, attackers usually flood the system with a large number of erroneous messages. For example, a packet can be sent to a server with an erroneous return address. The server returns the response and waits for the other machine to respond back. This does not happen, thus the server ties up a link waiting for a response until it times out (see figure 5-4). If thousands of such packets are sent to the server in a second, then naturally the server will be tangled up in waiting for responses that never show up.

Distributed denial of service attacks are an evolution of the standard DoS that flood the system by exploiting distributed environments. Thus instead of one attacker launching a DoS attack, several join hands and drive the victim crazy. For example, many clients can simultaneously send erroneous requests to a victim and keep sending more and more requests as time goes on (“closing in on the enemy”). The participants in a distributed DoS attack may be other weak machines that send these requests on behalf of the attacker without knowing about it. A well known example of distributed DoS is the series of attacks launched against some of the most visible sites such as Amazon.com, CNN.com, eBay, Excite, Yahoo!, ZDNet and others.³ These attacks sent up to 1 gigabit per second to the victim sites and brought them to their knees, causing millions of dollars of revenue losses. A Canadian youth was apprehended and prosecuted to launch these attacks.

³ Ghosh, A., *Security and Privacy for e-Business*, Wiley, 2001

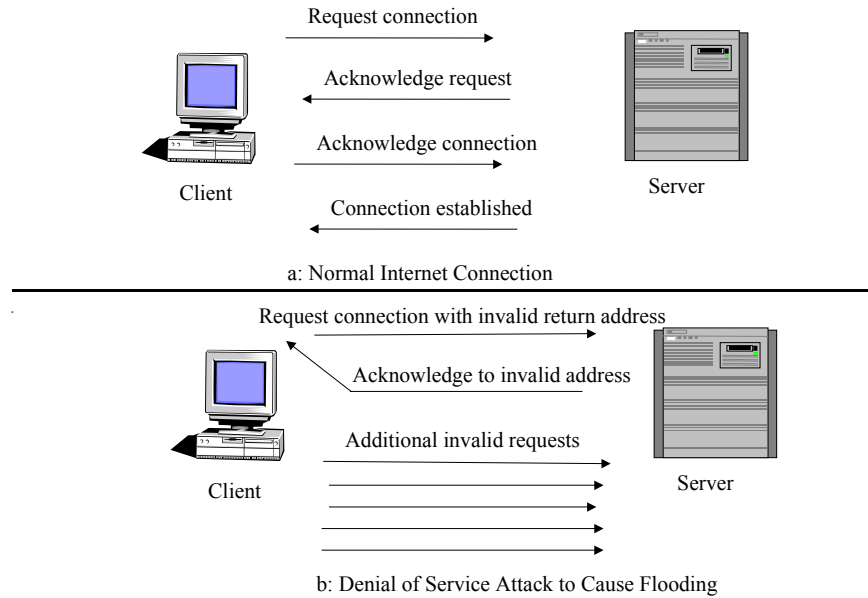


figure 5-4: Standard Denial of Service Attacks to Flood a Server

What can be done to prevent the variety of DoS attacks. The DoS attacks that cause crashes exploit weaknesses of the victim software. Most of these type of attacks are taken care of by newer releases of systems and eventually disappear (we hope!). The standard DoS attacks that cause flooding are much harder to prevent because they exploit the vulnerabilities of the Internet protocols itself and not their implementation. The commonly used approaches are:

- Use powerful servers that cannot be easily flooded – many DoS attacks do not generate enough traffic to disturb very powerful servers.
- Use backup and distributed servers that take over in case of intense attacks.
- Harden your servers by adding/upgrading software that plugs current system vulnerabilities. Since many DoS attacks are copycats, closing well-known holes is a good general defense.
- Install special software in routers that filters packets for invalid origin addresses. This software can be installed at the routers and should ideally be installed by ISPs. A variety of rules can be used to detect invalid origin address. For example, if a packet arrives from outside a network, then its origin address should not be the same as internal network address.

5.5.3 High Availability, Fault Tolerance, and Intrusion Tolerance

High availability, fault tolerance, and intrusion tolerance are sometime used interchangeably. In fact, these represent stages of building highly robust systems needed by the digital enterprises. Let us clarify the differences.

First, fault tolerance should be distinguished from high-availability computing. Although both are designed to maximize application and system availability and both use backup hardware resources, there are some differences. High-availability computing helps enterprises recover quickly from a failure, whereas fault tolerance promises elimination of recoveries altogether by providing non-stop services. Thus fault tolerance goes beyond

high availability. Let us now include intrusion tolerance. Simply stated, a system is intrusion tolerant if it is fault tolerant plus secure. Thus intrusion tolerance goes beyond fault tolerance. It is important to view these three as stages of availability where the availability increases as we move from one stage to the next (see Figure 5-5).

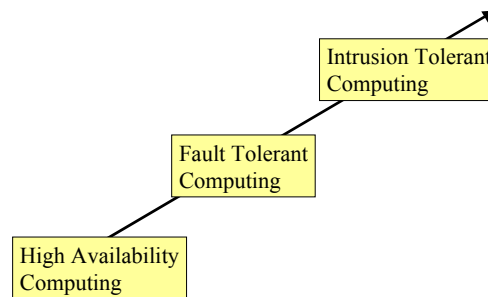


Figure 5-5: Stages of Availability

High-availability computing is a minimum requirement for most digital firms. It is essential for firms with heavy e-commerce processing or for those that depend heavily on digital networks for their internal operations. Companies such as Amazon and eBay are examples. These firms require more than 90% availability but tolerate outages by using good recovery plans. To maximize availability, high-availability computing requires an assortment of tools and technologies such as redundant servers, mirroring, load balancing, and clustering. Basically, the computing and communication platform must be extremely robust with scalable processing power, storage, and bandwidth. A good disaster recovery plan is essential. An example of good disaster recovery plan is the Merrill Lynch plan (see the opening case study about World Trade Center Disaster in chapter 1).

Fault tolerant computing is the next stage where the need for recoveries is eliminated by tolerating attacks and recovering from them in real-time. Of course, a good recovery plan is needed for disasters, but the idea is that the systems should be able to handle failures automatically and modify their behavior to continue processing. These systems are highly desirable in mission critical applications and command control systems. These systems use highly reliable and redundant hardware, including power supplies, with extensive load balancing to achieve non-stop processing. In addition to extra hardware, they use special software routines or self-checking logic built into their circuitry to detect hardware failures and automatically switch to a backup device. Parts from these computers can be removed and repaired without disruption to the computer system. Companies such as Stratus provide such fault-tolerant systems. Despite efforts, systems do fail, thus fault tolerance is in fact very high availability computing (above 99%).

Intrusion tolerant computing goes beyond fault tolerant to include security. In other words, an intrusion tolerant system must be able to withstand attacks from hackers plus natural failures and still keep on operating. Thus:

$$\text{Intrusion tolerance} = \text{security} + \text{availability}$$

From our perspective, PIA3 (Privacy, Integration, Authorization, Authentication, Accountability) signifies security and PIA4 (PIA3 + Availability) refers to intrusion tolerance. Since we are focusing on PIA4, in fact we are concentrating on intrusion tolerance.

Naturally, making a system intrusion tolerant is much more challenging than making it secure. The main reason is that tradeoffs exist between security and availability. For example, a system can be made highly secure by centralizing everything that is kept under tight controls. But centralization threatens the availability of the system because if the site goes down, then the system is not available. In reality, *fault tolerance requires high redundancy which may lower security*.

A great deal of research on intrusion tolerance has been undertaken by DARPA. For example, the DARPA Broad Agency Announcement (BAA) 00-15 on Intrusion Tolerance Systems funded almost 20 projects to look at different aspects of intrusion tolerance.⁴ The research included different replication schemes, self-reconfigurable systems, and mobile agents, among others. We will only discuss replications and its variants in this chapter. Other topics are beyond the scope of this book.

5.5.4 Replication as a Backbone for Increased Availability

A large number of techniques, mentioned above, are used in high availability and fault-tolerant computing. Load balancing, a popular one, distributes large numbers of access requests across multiple systems. The requests are directed to the most available system so that no single device is overwhelmed. If one machine starts to get overloaded, requests are forwarded to another with more capacity. Mirroring, another common approach, uses a backup system that duplicates all the processes and transactions of the primary system. If the primary system fails, the backup system can immediately take its place without any interruption in service.

Mirroring can be used in databases or servers. However, extensive mirroring is very expensive, because each system must be mirrored by an identical system whose only purpose is to be available in the event of a failure. Thus mirroring is more common in databases (requiring just an extra disk) than servers. Instead of mirroring, clustering offers a less expensive technique for ensuring continued availability. High-availability clustering links two computers together so that each computer can act as a backup to the other computer. Each computer can have a primary as well as a secondary (backup) role. If the primary computer fails, the second backup computer picks up its processing, and vice versa. Many computers can also be clustered together as a single computing resource, with at least one backup for each.

In all of the techniques above, replication is the common denominator in increasing system availability. Common examples of replication are:

- Providing more than one copy of a file (mirroring) so that if one fails, the other can take over
- Using more than one server to handle Web traffic (load balancing). Alternate servers may be assigned by a proxy whenever the primary server becomes unavailable.
- Network devices such as routers and network segments can be clustered to provide alternate paths.

Although replication is quite effective in increasing system availability, it introduces security vulnerabilities. For example, if you have ten copies of a file, then all ten files have to be secured at the same level. Otherwise, one weak copy of the file could be

⁴ My project on Intrusion Tolerant Middleware services, funded by DARPA, was one of these projects.

accessed by an intruder. Thus replication can lead to increased availability but decreased security. Consequently, replication alone is not sufficient for intrusion tolerance. FRS (Fragmentation, Replication, and Scattering) is a variation of replication that can help in intrusion tolerance.

5.5.5 FRS (Fragmentation, Replication, Scattering)

A Fragmentation-Redundancy-Scattering (FRS) scheme [Deswarte 1988, Deswarte 1991, Silva 1998] is a good approach to increase availability plus security. The FRS technique consists of three activities:

- Fragmentation: Cutting all the sensitive information into several fragments such that no significant information is contained in any isolated fragment.
- Redundancy: Multiple copies might be introduced by copying the fragments to tolerate accidental or purposeful destruction or alteration of fragments.
- Scattering: The fragments along with their copies may be scattered amongst the different sites of the distributed system.

For example, a sensitive file could be split into fragments F_1, F_2, \dots, F_n where no fragment has complete information. Then three copies, say, of each fragment can be created so that if one copy is destroyed, the other two can be used instead. The copies of the fragments can be further scattered around a network of computers in such a manner so that an intruder finds it extremely difficult to develop a complete picture of the document. Figure 5-6 shows an example of FRS. In this case a file F is split into 3 fragments (F_1, F_2, F_3) and then these fragments are scattered and replicated across computers C_1, C_2 , and C_3 .

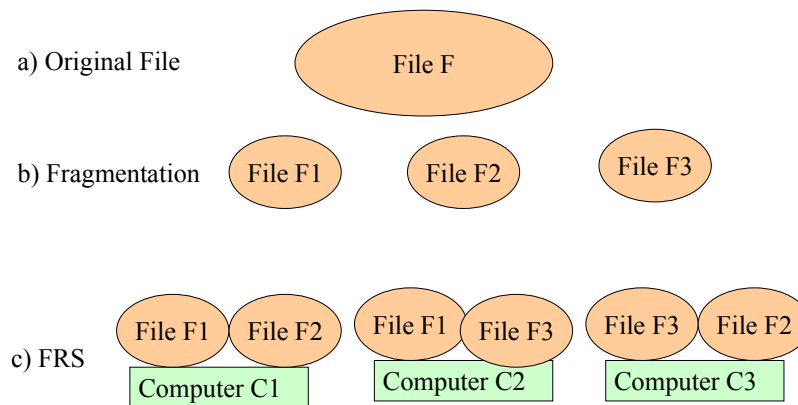


Figure 5-6: Fragmentation-Redundancy-Scattering (FRS) Example

This scheme increases the availability as well as security of a document. To adjust security and availability levels, more fragments could be created with more copies that are scattered around the network in a mysterious fashion. To further secure highly sensitive documents, the fragments could be also encrypted before replication and scattering. For even more security, the scattered fragments could be moved around periodically to further confuse the intruder. Thus, practical use of FRS raises several questions such as the following:

- How many fragments should a document be split into?
- How many copies should be created of each fragment?

- How should the fragments be scattered?
- Should the fragments be encrypted for added security?
- Should the scattered fragments be re-scattered every now and then?

The main tradeoff is that more sophisticated the FRS scheme, the harder it is to reconstruct the fragments for normal use. For example, if a file is highly fragmented and scattered, then every time a user needs to access this file, all fragments have to be presented by the system software to the user as if the file was never fragmented. The cost of creating and accessing multiple fragments can be quite high. If a FRS scheme is not properly designed then the supposed advantages would be lost, making the system more susceptible to intrusions. More research is needed in this area. The following sections highlight the key ideas.

5.5.6 Fragmentation Considerations

Attention has to be paid also to how fragmentation can be used and benefit gained from an object-oriented model of system structuring. One possibility is to fragment based on the objects created, such that different objects correspond to different fragments. The user might also be given the flexibility of declaring the amount of security required for each object. A more extreme form of fragmenting could be to fragment the code based on the operations and then execute these operations on different machines.

5.5.7 Scattering Considerations

The problems of updating and scattering do also require more attention. Problems that should be addressed include efficient schemes to recollect the fragments when necessary, as well as efficient schemes to update these fragments. The efficiency could be studied in terms of the network link usage as well as the use of computing resources of the system. Scattering schemes should not only take into consideration the security at each of the server sites, but also should ensure that a single site contains unrelated fragments. Investigations into dynamic scattering schemes (whereby the set of sites to which the fragments are sent is not constant but variable over time) as well as static scattering schemes are also needed. Detection of corrupted copies of fragments has to be done efficiently. In particular, when the original document is to be reconstructed, we need to determine when we need to compare different fragments to detect any unauthorized changes and what are some optimal schemes to do this. When the user requires just a portion of the original document, then algorithms to reconstruct just the required portion most efficiently have to be designed.

5.5.8 Combining FRS with Cryptography

It is quite intuitive that using a combination of cryptographic and FRS schemes should be more efficient than using either alone. This is because use of scattering might mean that the ciphers used can be much simpler than conventional ciphers. Cryptographic techniques along with FRS can be used to address the problem of corrupt system administrators. But how do we combine these schemes to get different levels of protection? The basic approach in this case would be to use less of the system resources while providing the same level of difficulty for the intruder. One option is to combine different cryptographic keys with different FRS schemes and quantify the cost/benefit to

the system of interest. This selects the schemes which are the most efficient plus also satisfy the constraints on the level of intrusion tolerance required by the system. This should allow the system flexibility to choose different intrusion tolerance levels based on the threat perception.

5.5.9 Special Considerations

How do the scattering-updating-reconstructing schemes work when some of the server sites are disconnected? This might mean that the user in some cases is unable to access his file. This can be considered as the downtime of the system, and guarantees will have to be provided to the user as to the maximum amount of downtime that he or she will be subjected to using a given algorithm. Note that the lower the downtime guaranteed, the more copies of each fragment would have to be made. These copies would then have to be distributed over sites such that the probability of all such sites failing is less than the probability of guaranteed downtime. The problem becomes more complicated when one also considers the fact that some of these sites might have been compromised, and hence each fragment needed by the user would have to be checked for accuracy. We understand that the different questions raised above do not constitute an exhaustive set.

5.5.10 The Reality Check on FRS

Theoretically, you can create thousands of fragments that can be replicated and scattered around hundreds of computers in a large corporate Intranet. However, the following realities of life need to be considered:

- Every time an authorized user needs to access a file, the fragments have to be assembled into a whole so that the user can make some sense out of them.
- The performance overhead of ambitious FRS can be very high.
- Special purpose middleware services will be needed to handle FRS requests because the users should not have to locate and assemble the fragments themselves. Such middleware is not commercially available at the time of this writing. However, commercial database management systems, such as Oracle, do support database partitioning that can be used to support simple FRS schemes.
- The middleware to handle FRS could be quite complex and could itself be target for viruses and bugs.

Based on these factors, FRS should be undertaken for only extremely sensitive data stores.

5.6 Short Case Studies and Examples

5.6.1 Australian Agency Uses Digital Certificates to Sign Contract with Fujitsu

The Victoria's Transport Accident Commission (TAC) in Australia has embarked on e-commerce as a way of increasing both efficiency and service delivery to customers and other stakeholders. To help achieve its vision, TAC chose Fujitsu as its information technology service provider for the company's ability to deliver end-to-end e-services.

The \$20 million contract with Fujitsu will allow TAC to transform its interactions with its clients, stakeholders and the broader community.

Besides the work itself, the contract signing was a landmark event because the contract was signed by using digital certificates. It was the first time high-grade Gatekeeper certificates had been used to sign a digital contract under Australian law. The digital certificate technology was supplied by eSign Australia. eSign was responsible for the registration authority services, including validation and authentication of the individuals signing the contract. In addition, eSign created and issued the digital certificates containing the TACs and Fujitsu's electronic credentials.

Using the digital certificates streamlines and simplifies the contract signing process, and is expected to be tamper-proof – if anyone tries to change the document, it will indicate to all parties that it has been modified since it was signed. In addition, the identity of a signatory can be verified anytime by querying the independent certifying authority – in this case, eSign.

Source: <http://au.fujitsu.com/FAL/CDA/0,1531,322~978,00.html>

5.6.2 How Computer-Savvy Investigators Operate

Police detectives are increasingly using computers at the crime scene to gather additional information. These cases involve interesting legal and information security issues. The following is an example of a computer-savvy investigator who followed a network trail from a murder probe to child pornography.

On Oct. 16, 1998, Lt. J. J. McLean, a Massachusetts police officer, arrived on a crime scene where John Hinds lived with his 87-year-old mother, after Hinds had gunned down two of his family members in the street outside his house. McLean, a computer forensics expert, started looking for emails exchanged between John Hinds and the victims that would show a motive for the killings. McLean found a computer in John's house and noticed that there were two other computers – Takedown and Chuck – on the network. McLean later learned that these two computers were in the adjacent house where John Hinds' nephews lived.

McLean knew that any or all of the three computers on that network could hold what he was looking for. He also knew that his search warrant to John Hind's house did not automatically allow him to investigate the rest of the network. To search the other two computers, the police had to get a warrant for the house next door or ask each of the owners for permission to search the computers for email evidence. The two brothers gave them the permission. After getting the permission, McLean disconnected the network cables from murder suspect John Hinds's computer so that no files could be transferred or compromised remotely.

Having secured the computers from intrusions, McLean started searching the machines for email evidence for the murder case. While searching for the emails, he ran into a great deal of child pornography material on one of the computers in the second house. The child pornography on this machine gave McLean grounds to declare the system contraband, seize it without a warrant and ship it, along with John Hinds's computer, to the Attorney General's High Tech Crime Unit lab in Boston.

The process of packing computer evidence is also quite intricate. The step-by-step procedure included: a) photos of the system before disassembling it, b) disconnection of internal data and power cables to the hard drives to ensure that the drives were not accessed and possibly tampered with before being removed, c) placement of a write-protected boot disk in the disk drive to ensure write protection, and d) a detailed catalogue of what was seized and by whom.

What was the result? The email evidence on John Hinds's computer resulted in a conviction of two counts of first-degree murder and a mandatory life sentence. To pursue the child pornography case, McLean obtained a second search warrant specifically to examine the computer owned by Chuck Hinds, nephew of John Hinds. Using sophisticated search tools, McLean found a great deal of child pornographic information that resulted in conviction and sentencing of Chuck.

Naturally, this type of search and seizure raises several questions. Can a search for emails for murder start searching for pornographic materials? Where does it end? A superior court judge upheld McLean's search, citing that Chuck Hind had permitted the search for email on his computer. The judge noted that since an email file could be masked by changing names and extensions, McLean acted legally when he opened some files that turned out to be child porn files. In addition, the judge noted that the network was unprotected and that the poor network security opened the door to the case. In other words, if there had been good security – restricted print and file sharing, encrypted files and drive, intrusion detection – McLean might not have ever seen the suspect drive or files. If the suspects had used sophisticated protection, the government may not have made the case. The suspect might have denied access to the police and had time to erase the evidence.

Source: N. Roiter, "Cybercop," *Information Security Magazine* (April 2002).

5.6.3 City of Edmonton Upgrades Access Control

Many cities have developed electronic access control systems to protect facilities while being monitored from one central location. These control systems have card readers that the entrants use to gain access. The entrants insert their card into the card reader and the reader transmits the card information to a central location for verification. But installation of such a system is a non-trivial task. For example, the city officials in Edmonton, the capital of Alberta, Canada, wanted an electronic access control system to help protect dozens of municipal facilities. A local security dealer was hired for the purchase and installation of equipment in about 50 sites, including maintenance yards, swimming pools and ice skating arenas. However, the job was not done right and a host of problems plagued the system at a number of the sites.

For many years, the city tried to use the system but could not. Eight years later, some officials were about to scratch the project but decided to hire a systems integrator to implement the project. The system integrator, Antar-Com Inc. (ACI), a White Plains, N.Y.-based systems integrator, switched Edmonton to a new access system, wrote system installation standards, and also trained city employees on system administration. The new system worked as expected, and the city added new sites, totaling about 100 city-owned facilities that included the city hall and other administrative office space, bus garages, childcare centers and many other city-owned recreation and work sites. The

system includes approximately 650 card readers and 5,000 input points. The system activities are monitored around the clock from a command center, and alarm information is transmitted to the command center via Edmonton's wide area network (WAN). About 4,000 of the city's 9,000 employees carry access cards., with that number scheduled to expand significantly.

The city relies on the system heavily and added a redundant server to ensure that the access system will continue to function without interruption in case of a failure. In addition to increasing availability, the new system allows authorized city employees to dial into the main access server from laptops in their homes in the event of an alarm. The city is also considering integrating cameras with card readers so that the security guards can use the cameras to match a face with the access card being used to gain entry into city facilities.

Source: "Access Control & Security Systems" (May 1, 2003), retrieved from http://govtsecurity.securitysolutions.com/ar/security_edmonton/

5.7 Conclusions

This chapter has scanned the developments in the 4A (authentication, authorization, accountability, and availability) technologies. The systems that support the 4A's are considered highly trusted computing systems. These systems support strong authentication schemes, digital signatures, digital certificates, intrusion detections, audit trails, and some levels of replications. These trusted systems provide an environment in which the users are properly authenticated and authorized. In addition, any deviations from normal behavior are detected. In addition, a trusted computing system is needed for a fair non-repudiation system. Without a trusted computing system, neither party – the signer or the recipient – can prove their respective case.

5.8 Suggested Review Questions

- 1) What are the differences between authentication and authorization?
- 2) Prepare a table that lists the various authentication schemes along with their strengths and weaknesses.
- 3) What exactly is strong authentication? Give examples. Explain why strong authentication should not be used in all cases.
- 4) How is intrusion detection related to authorization?
- 5) What are different types of IDSs? Which one appears to be most promising?
- 6) What are the main issues in accountability, and what techniques are used to support accountability?
- 7) What is non-repudiation (NR) and why is it important?

- 8)** What are the different views on NR, and what is the best way to resolve these issues?
- 9)** Why is availability important in the context of security? How can FRS schemes help in increasing the overall security of a system?