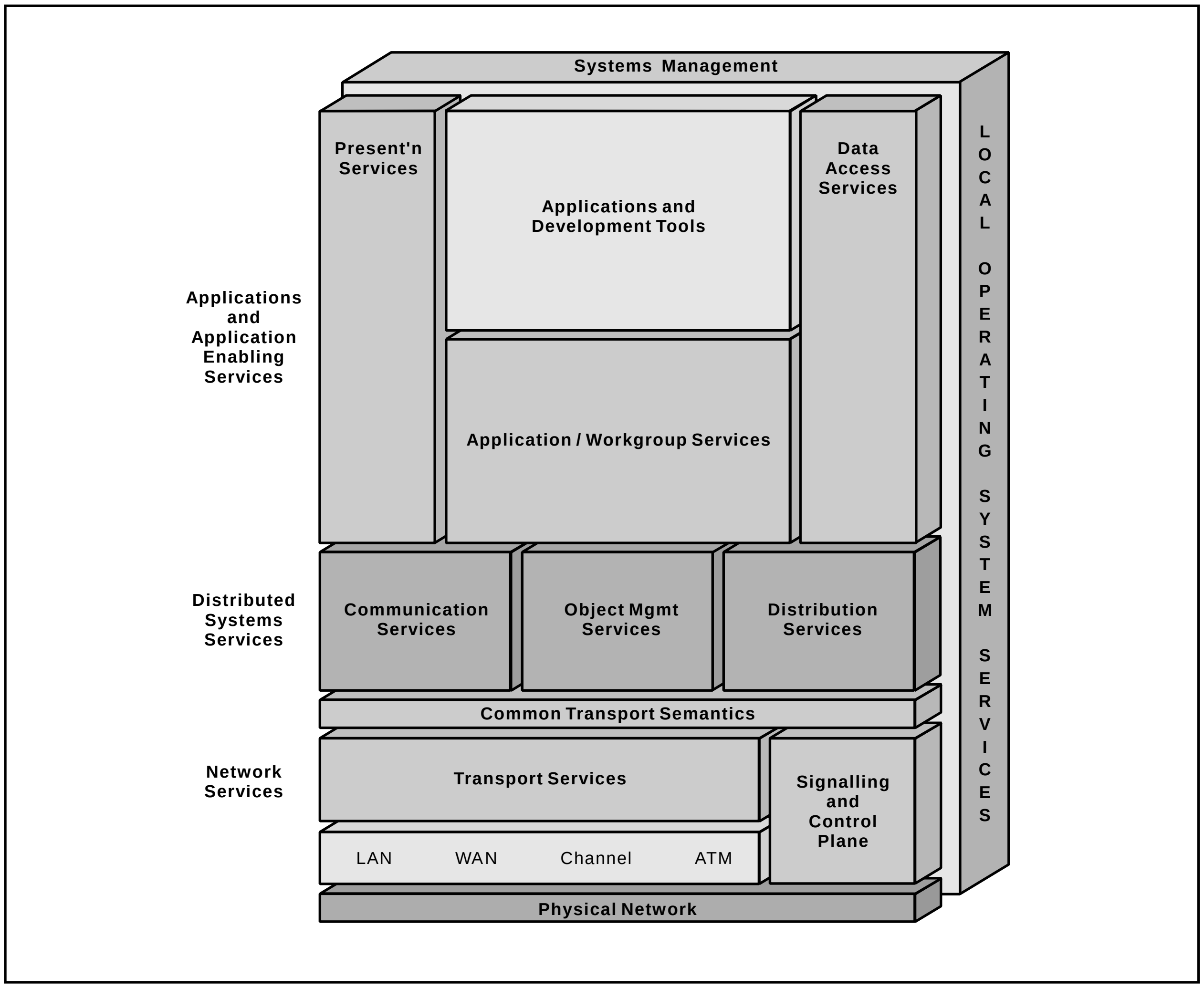
Open Blueprint

IBM

Security in the Open Blueprint


Systems Management
Present'n Services
Applications and Development Tools
Data Access Services
Applications and Application Enabling Services
Application / Workgroup Services
L O C A L O P E R A T I N G S Y S T E M S E R V I C E S
Distributed Systems Services
Communication Services
Object Mgmt Services
Distribution Services
Common Transport Semantics
Network Services
Transport Services
Signalling and Control Plane
LAN
WAN
Channel
ATM
Physical Network

GC23-3936-01

Open Blueprint

Security in the Open Blueprint

GC23-3936-01

About This Paper

Open, distributed computing of all forms, including client/server and network computing, is the model that is driving the rapid evolution of information technology today. The Open Blueprint structure is IBM's industry-leading architectural framework for distributed computing in a multivendor, heterogeneous environment. This paper describes the Security component of the Open Blueprint and its relationships with other Open Blueprint components.

The Open Blueprint structure continues to accommodate advances in technology and incorporate emerging standards and protocols as information technology needs and capabilities evolve. For example, the structure now incorporates digital library, object-oriented and mobile technologies, and support for internet-enabled applications. Thus, this document is a snapshot at a particular point in time. The Open Blueprint structure will continue to evolve as new technologies emerge.

This paper is one in a series of papers available in the *Open Blueprint Technical Reference Library* collection, SBOF-8702 (hardcopy) or SK2T-2478 (CD-ROM). The intent of this technical library is to provide detailed information about each Open Blueprint component. The authors of these papers are the developers and designers directly responsible for the components, so you might observe differences in style, scope, and format between this paper and others.

Readers who are less familiar with a particular component can refer to the referenced materials to gain basic background knowledge not included in the papers. For a general technical overview of the Open Blueprint, see the *Open Blueprint Technical Overview*, GC23-3808.

Who Should Read This Paper

This paper is intended for audiences requiring technical detail about the Security in the Open Blueprint. These include:

- Customers who are planning technology or architecture investments
- Software vendors who are developing products to interoperate with other products that support the Open Blueprint
- Consultants and service providers who offer integration services to customers

Contents

Figures

Summary of Changes

This revision includes:

- Updates to the security architecture, which integrate Internet security technologies, in particular, public key
- Restructuring of the security architecture into additional discrete resource managers
- Examples of security flows

Security Overview

Security is concerned with identification and authentication of users, access control to resources, integrity and privacy of information, audit facilities, and management of security information.

In the Open Blueprint structure, security function is provided by a set of discrete resource managers.

- The Identification and Authentication resource manager
- The Security Context Management resource manager
- The Identity Mapping and Credential Transformation resource manager
- The Certificate Management resource manager
- The Cryptographic Services resource manager
- The Access Control resource manager
- The Audit resource manager

In addition, security function is provided in the Open Blueprint Network Services. For information about Network Services, see the *Open Blueprint Network Services* component description paper.

The Requirement for Security

Figure 1 on page 4 illustrates an organization's view of secure computing; an organization owns a set of resources that are accessible through a computing system, and employs (or is otherwise associated with) a group of people who are authorized to use those resources in various ways. Organizations require that no unauthorized access to any resource can be accomplished through the computing system, and that all authorized accesses can be accomplished through the system. Organizations might also require that all accesses to a resource be recorded in an audit log to determine accountability for actions accomplished through the computer system.

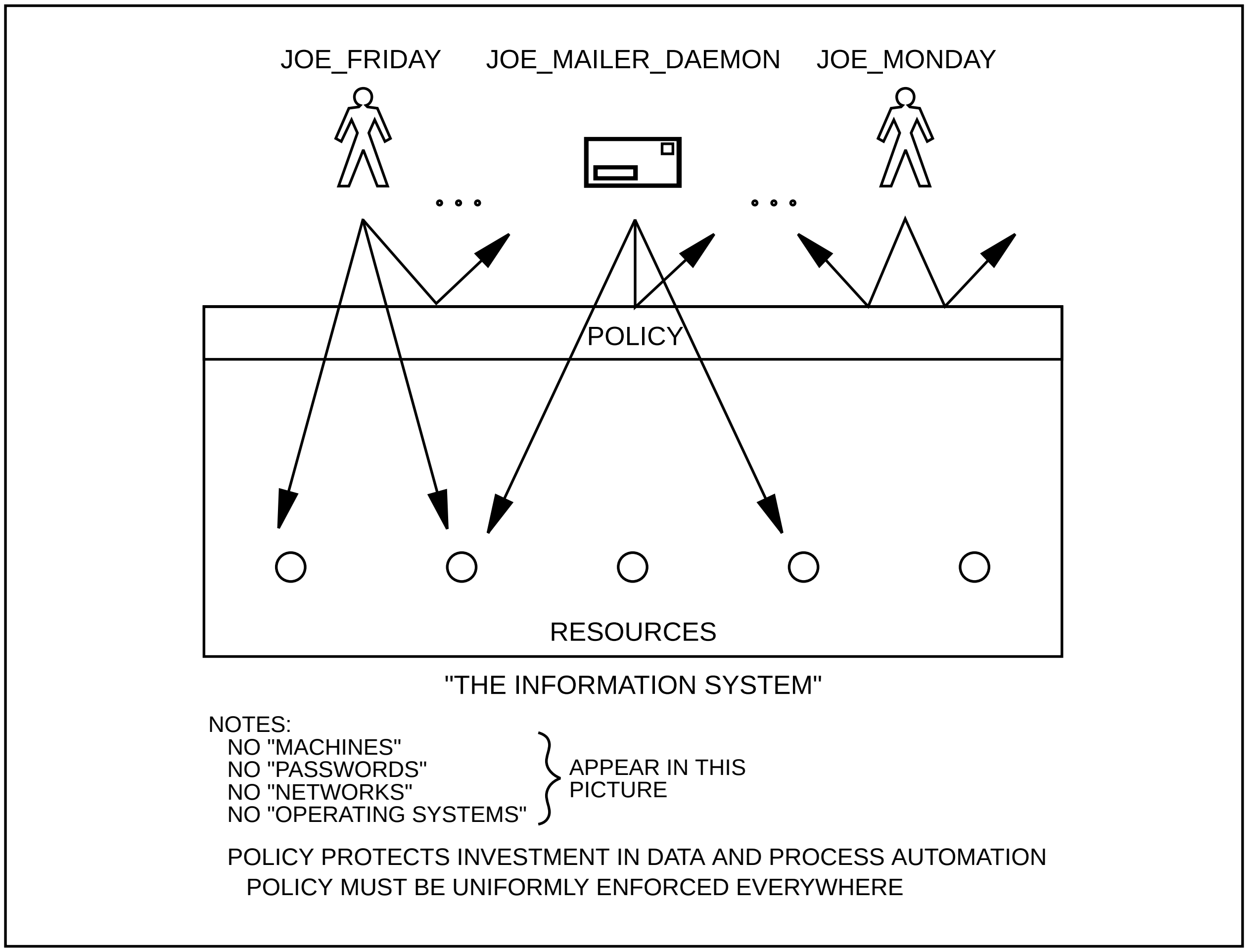


Figure 1. An Organization's View of Secure Computing

Uniformity and non-circumventability of policy enforcement are critical properties of secure systems; an organization is not interested in hearing that it is very difficult to protect resources that might, at various times, be stored on a wide variety of machines with different architectures, operating systems, applications, and communications connectivities—the organization views this problem as the vendors' problem. The only acceptable solution to this problem, from an organization's point of view, is a system that enforces the resource protection policy uniformly, regardless of user and resource location and system heterogeneity.

When an organization installs a computing system, it makes an *information investment*. The resources stored in the system, or the portions of the business that are automated through the system, are worth money to the organization. This information investment generates the requirement for security.

This security requirement is proportional to the value of the investment—not all information stored in a computing system is equally valuable. An organization will not pay more to secure an information investment than the investment is worth.

Any action that decreases the value of the organization's information investment is called an *information compromise*. A malicious action that seeks to compromise information is called an *attack*.

Computing systems are subject to the following types of information compromise:

- **Disclosure** of data
- **Destruction** of data or programs

- **Modification** of data or programs
- **Denial of Service** to legitimate users

Not all information compromises devalue an organization's information investment equally. For example, the destruction of a set of launch codes for a ballistic missile is much less costly than the disclosure of the same set of codes. Thus, a government might be willing to spend more to protect these codes against disclosure than to protect them against destruction.

Some attacks are less costly if they are detected promptly. For example, missile launch codes can be changed to prevent an unauthorized launch if their disclosure is detected. Thus, an organization might be willing to pay for the auditing of security-relevant events.

Some information stored in a computing system can be compromised (devaluing the owner's investment) in the absence of any breach of computer security. For example, an employee with authorized access to the information can reveal it to a competitor in a telephone conversation. Typically, an organization does not spend a large amount of money on strong computer security to protect information that is weakly protected against other classes of compromise.

The security requirements for a computing system take the following forms:

- In a secure computing system, it is more difficult to devalue the organization's information investment through the computing system than through some other means.
- The security features of a computing system should cost less to install and operate than the value of the organization's information investment.

Challenges to System Security in a Distributed Environment

Securing standalone systems is a well-understood problem; documents such as the US Department of Defense's “Orange Book”[1] provide excellent guidance. Most modern systems, however, are distributed and support client/server or network computing or both. Distributed and networked systems introduce many new problems, some that are not yet well-understood.

A modern, distributed computer system spans multiple *security domains*. A security domain defines a set of objects that are subject to a *security policy*. The security domain structure of a distributed system is illustrated in Figure 2 on page 6.

Each type of security domain protects a different set of objects.

- The **hardware domain** protects the physical components of a single system. For example, the hardware domain can prevent an unauthorized user from starting the system by requiring the user to enter a power-on password. In addition, most systems have mechanisms to prevent users from opening the case of the system; some protect the system against addition of unauthorized components or cables.
- The **operating system domain** protects resources in a single, standalone system. For example, the operating system domain protects files on local disks from unauthorized access through the operating system file system interface.
- The **network domain** protects the network against unauthorized use. Network security mechanisms include link encryption, which protects the network cabling against wiretapping, and gateway authentication, which protects the network from the introduction of packets by unauthorized users.
- The **network operating system domain** protects distributed resources. For example, files in a distributed file system frequently reside in caches on various machines; both the caches and the

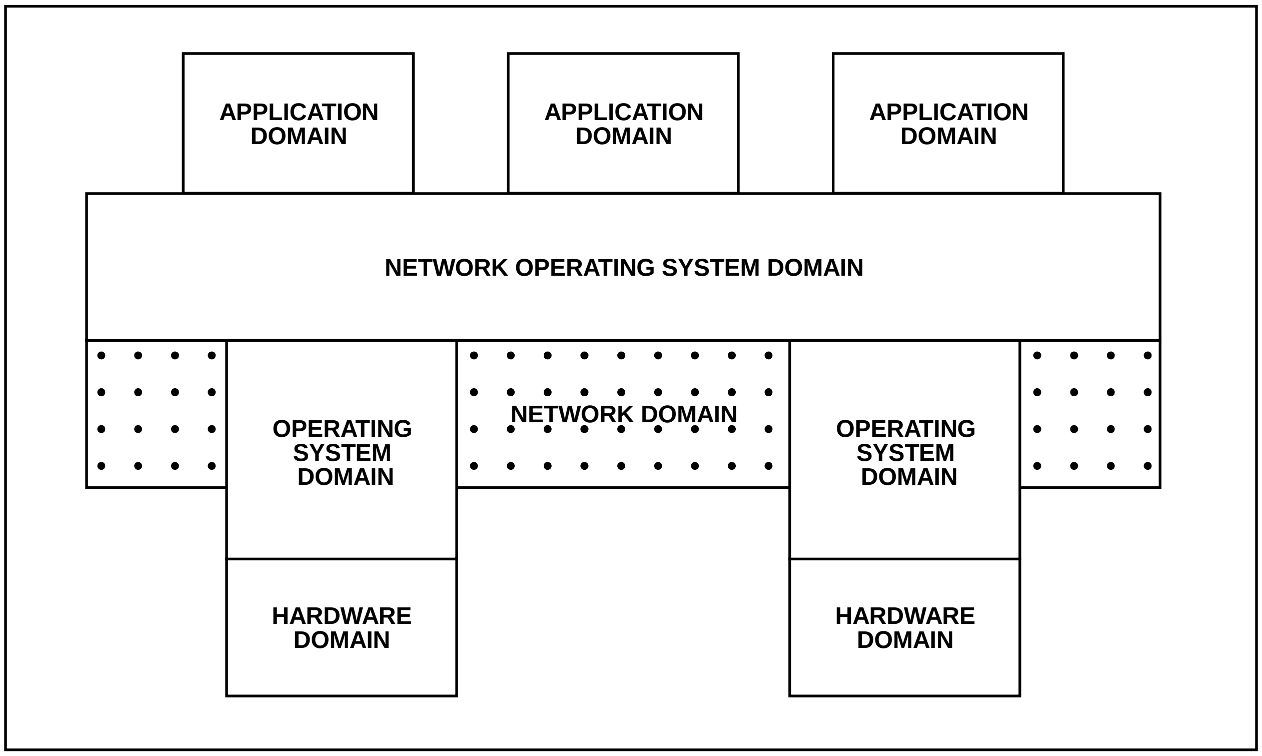


Figure 2. Security Domains in a Distributed System

servers' disks must be protected, even though the existence of the caches is usually transparent to users.

- The **application domain** protects resources whose semantics are understood only by the applications that process them. For example, a database must sometimes provide inference control to prevent users from getting information about individuals by forming highly specific queries; an electronic mail system must be able to provide user-specific encryption to protect messages from disclosure to anyone except the intended recipients.

Few configurations today have security that is homogeneous across all domains. In most commercial systems, the different domains' security mechanisms are designed independently (in many cases by different vendors).

Because of this variety of security mechanisms, truly secure systems are difficult to produce. Configurations in which the security mechanisms of the domains are not designed to work well together usually have the following problems:

- **Weak system integrity**

  Because the security mechanisms of different domains are not designed to work together, they do not protect one another well, and attacks are often possible at the seams between domains.

- **Multiple signons**

  Each domain usually has its own user database and its own signon, or logon, procedure. Multiple signon procedures can create a very complex environment for the user, who might be required to have many different identities and many different passwords. This environment is also very difficult to administer.

- **Unintegrated policy management**

Each domain can have its own policy database, its own policy syntax and semantics, and its own policy management interface. Each domain's policy is expressed in terms of its own set of user identities, which creates a very complex environment for system administrators. Policies expressed in one domain might not be semantically valid in other domains, so consistent resource protection is not possible across the entire system.

- **Unintegrated audit logs**

  Each domain uses its own audit log, its own audit event formats, and its own auditing interface. Each audit log's event records contain only the domain's version of a user's identity, so correlating events from logs in different domains is difficult. Different domains might not have consistent notions of system time, further complicating the correlation task and making sequence of events and causality difficult or impossible to reconstruct.

To produce manageable, secure systems, the security mechanisms of all the domains must be designed to be integrated, where feasible, and federated (united) elsewhere. The most important considerations for integrating or federating (or both) domains are:

- A coherent set of architectures that define the security functions that belong in each domain and how the domains should be either integrated or federated, or both.
- A single notion of user identity (*unified identity*) that is common across domains and administered in a single database. Until all security-aware elements of the domains can recognize the unified identity, new services that map identities from one form to another are required, for example, services that map a Distributed Computing Environment (DCE) *principal* to an MVS *userid*.
- A single semantic model for authorization policy.
- A single audit log that uses an integrated user identity and a consistent, global system time.
- Standard programming interfaces and management interfaces to the security functions.

Internetworking Considerations

Although trusted network domains can be connected to other trusted network domains through a secure communications link or intermediate network, they are often connected through a set of untrusted networks such as the Internet. Multiple mechanisms exist to provide a secure link through untrusted intermediate networks.

Secure links between trusted network domains can be established with hardware or software or both. Most secure links depend on cryptographic methods to protect information being exchanged between the trusted network domains. Some of the methods that are currently available include:

- Internet Engineering Task Force (IETF) Internet Protocol Security (IPSec), which provides identification and authentication, integrity, and privacy at the IP packet level. IPSec is defined by IETF Requests for Change (RFC) 1825-1829
- Secure Sockets Layer (SSL) Version 3
- Generic Security Services Application Programming Interface (GSSAPI), which is defined by IETF RFCs 1508 and 1509
- Distributed Computing Environment (DCE) intercell support

Additional information about network layer security can be found in the *Open Blueprint Network Services* component description paper.

Trusted network domains that must connect to untrusted networks usually use a *firewall* to provide the necessary connectivity to the untrusted network while preventing unauthorized access from outside networks. As shown in Figure 3 on page 8, a firewall is a blockade between a trusted, internal, private

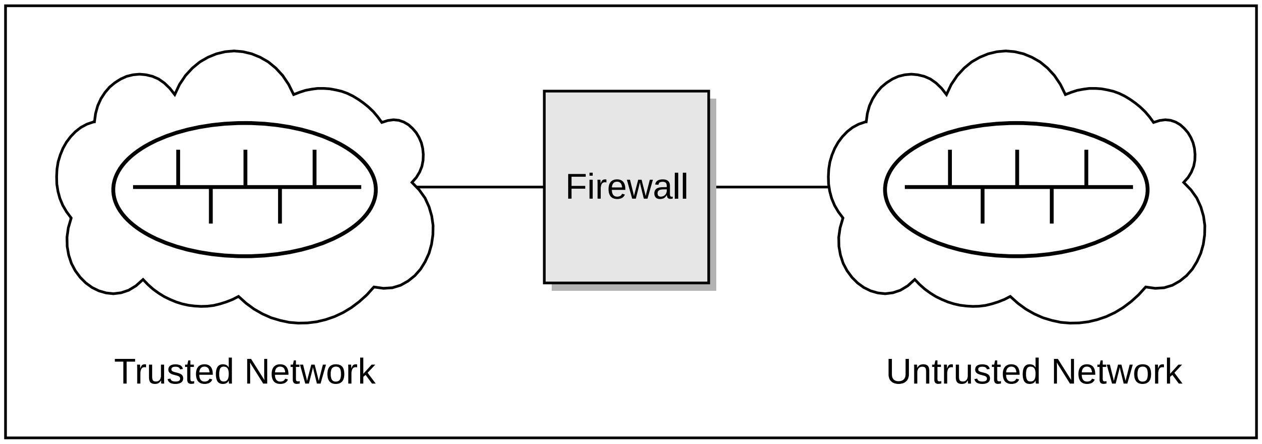


Figure 3. A Firewall Protecting a Trusted Network

network such as an intranet and another untrusted network or the Internet. A firewall prevents unwanted or unauthorized communication into or out of the trusted network. The firewall has two jobs:

- The firewall allows users in the trusted network to use authorized resources from the untrusted network without compromising the data and other resources in the trusted network.
- The firewall prevents users outside the trusted network from attacking or compromising the trusted network.

A firewall uses configurable filtering of network traffic to allow restricted access to services that a host provides to the network (TCP/IP port numbers) and blocks access to other services.

Firewall technology is being incorporated into other products such as LAN switches and is used for additional applications such as those which partition or “harden” subnetworks within a trusted network. Additional functions such as virus scanners and filters are being added to firewalls on a regular basis to keep pace with new requirements.

In conjunction with cryptography, firewall technology is used to establish virtual private networks (VPNs) over untrusted networks. Two or more firewalls can establish secure inter-firewall communications through untrusted networks to form a trusted network that uses a public, untrusted network infrastructure.

Firewalls are only part of a complete security infrastructure. Some examples of security issues not addressed by firewalls are:

- Dial-up connections from workstations that bypass the firewall. These connections must be addressed by usage policy and vigorous enforcement.
- Untrusted or potentially ill-behaved software such as software obtained from the Internet using the File Transfer Protocol (FTP). Ill-behaved software can be addressed through policy, for example using quarantine facilities, and evolving standards for certifying trusted software, for example digital signatures.

The Trusted Computing Base

A widely accepted concept in trusted system technology is that of the *trusted computer base* (TCB). The TCB is defined as the software and hardware elements that collectively enforce an organization's computer security policy. Any element or part of an element that can effect security policy enforcement is security-relevant and part of the TCB. The TCB can be described as an object that is bounded by the security perimeter. The mechanisms that implement the security policy must be non-circumventable, and programs must be prevented from gaining access to system privileges to which they are not authorized.

Non-circumventability is a very difficult requirement to satisfy, but it is a key security requirement because security policy rules are usually defined as follows:

```
Joe must never be allowed to read the file "payroll.dat"
```

In this example, and in most security policies, never actually means never. No matter what Joe does (no matter what tool he uses, what program he buys or writes, what disk he steals), he must be prevented from reading the payroll data file. This policy is part of an organization's business policy and is designed to protect the organization's information investment. If the policy is *ever* violated, the organization can lose money.

Creating non-circumventable security policies involves:

- Identifying all the code used to define, store, or enforce the security policies. This collection of code is called the computer system's trusted computing base (TCB).

  A distributed system's trusted computing base can reside on one machine or can be distributed across many machines.
- Assuring, through careful design analysis, formal verification, testing, or some combination of these methods, that the code in the TCB enforces the required policies.
- Assuring that the hardware and software integrity properties of the system guarantee that TCB functions can only be invoked through their published interfaces.
- Assuring the functional correctness of the TCB code.
- Assuring that execution of each TCB function while the system is in a secure state leaves the system in a secure state.
- Assuring that all data upon which the system's security depends is protected by the TCB against unauthorized access.
- Assuring that all authentication and authorization decisions are made within the TCB.

Non-TCB code is not subject to any restrictions for reasons of security. Non-TCB code can be incorrect without endangering security (though, from the organization's point of view, it is a poor practice to allow incorrect code on the system). In other words, the code outside the TCB is *untrusted*. Untrusted code might not only be incorrect, it might be malicious. If the TCB is trustworthy, even untrusted, malicious code cannot violate the computer system's security policy.

All access to protected resources by untrusted code must be mediated by the TCB's authentication and authorization mechanisms.

System Components in the Trusted Computer Base

The TCB contains several kinds of components:

- **Security Resource Managers**

  Any component that stores security policy information or performs security-sensitive function is a *Security resource manager*.

  The instantiation of the Open Blueprint resource managers that take part in the enforcement of security policies are components of a distributed system's TCB. There are seven Open Blueprint Security resource managers:

  - The Identification and Authentication resource manager
  - The Security Context Management resource manager
  - The Identity Mapping and Credentials Transformation resource manager
  - The Certificate Management resource manager
  - The Cryptographic Services resource manager
  - The Access Control resource manager
  - The Audit resource manager

- **Privileged Components**

  Any component that runs with system privilege (that is, that runs under the system's identity or exempt from security checking) is a *privileged component*. Typically, kernel components and some trusted services are privileged.

- **Trusted Resource Managers**

  Any resource manager that maintains protected information that is not security-sensitive is called a *trusted resource manager*. The distinction between Security resource managers and trusted resource managers is defined as follows:

  - Security resource managers define system security policy.

    If a Security resource manager's protection fails, the security policy of the system can be *changed* (for example, new users can be added, passwords can be changed and authorization profiles can be corrupted).

  - Trusted resource managers enforce system security policy.

    If a trusted resource manager's protection fails, the security policy of the system can be *violated* (for example, access to resources can be allowed in violation of the security policy and some security-relevant events can not be audited), but the policy is not changed.

  Trusted resource manager protocols are typically built on secure associations (so the resource manager can be assured of the identity of the party requesting access to a resource). Trusted resource managers use Security resource managers to store their authorization and audit information. They are trusted because their code does the following:

  - Retrieves authorization profiles
  - Checks the requester's identity against a resource authorization profile to determine whether request is authorized
  - Updates the audit log

  Trusted resource managers can be either *trusted servers* or *trusted proxies* (or both).

  - **Trusted Servers**

    A server that manages protected resources must be trusted to ask a Security resource manager to make an access control decision and must abide by that decision (that is, to deny access if the access control check fails). Examples of trusted servers include file systems and database managers. Most server code is resource manager code.

By using careful modularization, it is possible to implement trusted servers using code that is not completely contained in the TCB; code that is not used for security-sensitive operations can be untrusted.

– **Trusted Proxies**

  A server that accesses protected resources on behalf of the principals that call it must be trusted to ensure that the sever is always using the correct identity when accessing resources. As with trusted servers, by using careful modularization, it is possible to implement trusted proxies using code that is not completely contained in the TCB.

Systems management applications that are part of the TCB would use the authentication, confidentiality and integrity functions provided by the underlying distributed system to protect against unauthorized modification of TCB recognized objects (programs and data) in transit between components of the distributed trusted computing base.

Services are provided on local platforms that enable the detection of viruses. If viruses are detected, platform-unique corrective action must be taken.

Untrusted Programs

All programs that are not security-sensitive are referred to as *untrusted*. Untrusted programs are not contained in the TCB. Untrusted programs can manage information about users, manage resources and use resources. What distinguishes an untrusted program from security-sensitive, or trusted, code is that untrusted program code has no responsibility for defining or enforcing system security policies. This implies that no action of an application (including a malfunction) can compromise the system's security.

System Integrity

The security of a system rests fundamentally on the correct functioning of the security mechanisms and on the correctness of the following two kinds of data:

- **Security policy data**, which is persistent data that defines the system's security policy
- **Security context data**, which is transient data that describes the identity and security-relevant attributes of a human user or other identified principal for which the system is currently doing work

To insure the correctness of the security mechanisms and to protect security policy data and security context data, systems must be designed for high integrity. The US National Computer Center defines integrity[2] as follows:

> The assurance, under all conditions, that a system will reflect the logical correctness and reliability of the operating system; the logical completeness of the hardware and software that implement the protection mechanisms; and the consistency of the data structures and accuracy of the stored data.

The following general rules help to ensure system integrity:

- The system must ensure that the operation of non-privileged code cannot force the system to enter an undefined or insecure state (for example, as a result of a non-privileged program passing “illegal” data through the interface of a privileged system component).
- The system must ensure that system TCB components' code can only be modified by authorized administrators (for example, to upgrade to a new software release).
- System TCB components have the following specific system integrity responsibilities:

  - If a TCB component is a Security resource manager, it must insure that its transient and persistent security data can only be changed by authorized users.
  - If a TCB component is privileged, the following conditions apply:
    - Before passing data to a caller outside of the trusted computing base, the privileged component must check the caller's authorization.
    - Before changing data in response to a request from outside the trusted computing base, the privileged component must check the caller's authorization.
    - The privileged component must not execute untrusted code while operating with privilege.
    - The privileged component must audit all security-relevant operations.
  - If a TCB component is a trusted resource manager, the following conditions apply:
    - A trusted server must authorize every resource access request before granting access. Authorization consists of checking the authenticated identity and credentials against an authorization profile such as an access control list.
    - A trusted server must audit every resource access request in conformance with the audit policy.
    - A trusted proxy assumes the identity of the requester before performing access, or it can access the protected resources under its own identity, but only after determining that the resource access policy permits access by its client (the original requester).

TCB Enforcement of Security Policies

A secure computing system TCB enforces security policies by ensuring that:

- Every user and identified program in the system is required to establish a security context through an authentication, or signon, procedure before accessing protected resources. Security contexts include protected data and are maintained in the TCB.
- Every action affecting protected system resources takes place in a security context. No available action enables a user or identified program to become associated with a security context to which it is not entitled. No available action can enable a user or identified program to escape from a security context and initiate actions outside of any context.
- Whenever an action is initiated that can affect the state of a protected resource, the TCB components ensure that the system's resource access policies are not being violated. Typically, the TCB compares the security context of the user or program initiating the action against an authorization profile such as an access control list for the resource.

  Authorization profiles for protected resources are maintained within the TCB (usually by the Access Control resource manager), and the code that implements the authorization check is part of the TCB. This code is usually contained in a trusted resource manager.
- The system records some administrator-specified set of attempts to initiate actions that can affect the state of protected system resources in an audit log. The code that generates audit log entries is part of the TCB, and the audit log is a protected resource.

Cryptography and Certificates

Cryptography is the base technology for security. Dictionaries define cryptography as the act of writing in and deciphering secret characters. Cryptography includes encryption and decryption. *Encryption* refers to the transformation of data into a form that is useless to anyone without the decryption key. Encrypted information is unusable to everyone but the individual for whom it is intended.

The most important uses of cryptography are to support:

- **Confidentiality**. Prevents unauthorized eyes from seeing information. For example, users who want to send confidential e-mail messages can encrypt messages so only recipients who possess the encryption keys can decode and read it. Anyone else intercepting the message sees only gibberish.
- **Integrity**. Guarantees that information is not changed during transit.
- **Authentication**. Verifies the identity of a user and the user's eligibility to access and use information.
- **Non-repudiation**. Proof, with authority, of the origin, delivery, submission or transmission of information. Cryptography can be used to provide undeniable proof that, for example, a certain customer actually placed an order several weeks back.

Cryptography has been around almost as long as writing itself. In fact, history tells us that Julius Caesar used a rudimentary letter replacement scheme to encode the military orders he dispatched to his troops via messengers. Anyone who played with a cereal box “decoder ring” as a child will be familiar with this technique: if E's are replaced by U's, M's by V's, T's by M's, W's with Y's, and O's with A's, “meet me tomorrow” becomes “VUUM MU MAVALLAY.”

Simple methods of cryptography got the job done for centuries. But with the advent of computers came the need for more sophisticated cryptography.

Secret Key Cryptography

Secret key encryption is based on a symmetric key that is used by the encrypting and decrypting programs.

In the early 1970's IBM developed the core technology for the Data Encryption Standard (DES). DES uses a single 56-bit key both to encrypt and decrypt information. When encrypting an electronic document, DES breaks the document into 64-bit chunks, then encodes each chunk. Using DES, the only way to decode a document without the key is to try every possible key. On average this comes out to 36,028,797,018,963,968 keys when a 56-bit DES key is used for encryption. If a supercomputer is programmed to try one key every nanosecond, it would still take an average of 400 days to pinpoint the right key.

Public Key Cryptography

Public key encryption, which is implemented by the RSA Corporation and others, is the other form of cryptography commonly used today. Public key cryptography uses two mathematically-related keys: one public key and one private key. Using public key cryptography, an electronic document can be encoded with a public key (which the owner can make freely available), but can only be decoded using the private key. Alternately, the private key can be used to sign a document, and the public key for verification, a useful model in the implementation of digital signatures.

Even though the public and private keys are mathematically related, meaning that, theoretically, the code could be broken, doing so would require the mathematical equivalent of factoring very large integers—an almost impossible feat.

Key Management

Modern cryptography is dependent on keys. Keeping track of these keys is a primary concern. Traditionally, because both DES and public key cryptography are virtually impossible to break, once a key is lost, the data cannot be decoded. As a result, two techniques have evolved to ensure that a backup key can be located if the primary one is lost. These techniques are key escrow and a trusted third party.

- In the key escrow system, the private key is given to another party to hold. Alternatively, the key can be divided into part, and the parts can be given to several individuals. Either way, the protection of the key is in another party's hands.
- A trusted third party is a service that creates a key, distributes it to the proper recipients, and stores a copy. Security of the key is the third party's responsibility, and again the responsibility of keeping the key secret is in another party's hands.

These systems can be used to maintain both DES and RSA keys. But there are significant drawbacks: both the expense of engaging a third party and a lack of control over how the keys are stored.

Key recovery is a new technology that will address concerns with existing key management systems. This technology will support a process that will associate recovery information with an encrypted message or file, perhaps as header information. Key recovery technology would allow an organization to store and transmit data using strong encryption without concern if the keys used to encrypt the data get lost or destroyed.

In addition, this key recovery technology could facilitate the export of strong encryption function to the global marketplace. Key recovery technology is being designed to meet the cryptographic import and export requirements of any country. Furthermore, this key recovery technology is intended to be interoperable between countries with different import and export rules or key length requirements. In addition, this technology will be open; it will not depend on secret algorithms and will work with the user's encryption algorithm of choice.

Visit URL: http://www.ibm.com/security for additional information about key recovery and other cryptographic topics.

Public-Key Certificates

Digital certificates, which are also known as digital IDs, digital passports, digital credentials or public-key certificates, are defined by an International Telephone Union-Telecommunication (ITU-T) standard known as X.509v3. Certificates are the digital equivalents of drivers licenses, passports, and so on. Like their paper counterparts, certificates enable their owners to prove their identities. They also include important information like the validity period of the certificate and proof of the certificate issuer's identity and authority to issue the certificate.

Certificate Content: An X.509v3 certificate is a small structure (collection of bits), stored in a file or smartcard. Certificates contain information, including:

- **Subject's Distinguished Name (DN)**. A name that uniquely identifies the certificate owner. For example, `C=US, O=IBM, OU=DSS, CN=Dave Hemsath`
- **Issuer's Distinguished Name**. A name that uniquely identifies the certification authority that signed the certificate. For example, `C=US, O=IBM, CN=Entrust root`
- **Subject's Public Key**. The certificate owner's public key. (The corresponding private key of the asymmetric key-pair is typically held only by the certificate owner.)
- **Issuer's Signature**. The certification authority's digital signature, which programs use to validate the authenticity of the certificate. To validate the authenticity of the certificates, programs require:

  - A copy of the certification authority's public key
  - Access to a robust certificate infrastructure

- **Validity Period**. Dates between which the certificate is valid.
- **Serial Number**. A unique number generated by the certification authority for administrative purposes.

The Open Blueprint Security Strategy

A central goal of the Open Blueprint is to enable a single system image. An environment that is structured using the Open Blueprint appears to its users and administrators to be a single, logical system—not a collection of unintegrated or semi-integrated systems—regardless of the actual number, location, and type of machines, operating systems, LANs, and so on.

The Open Blueprint security strategy calls for a *distributed C2 single system image*. This means that the entire collection of hardware and software in an enterprise should look to users like a single system which meets, and in selected areas, exceeds the US Government's “Orange Book” C2 (commercial) security requirements for a single system.

The C2 requirements include the following:

- The system must identify all system users uniquely and must authenticate their identities before allowing them access to any system resources (authentication).
- The system should enable users to restrict access to resources they own by specifying allowable accesses for each system user (authorization).
- The system should irrevocably destroy information in all deleted resources (object reuse protection).
- The system should keep a tamper-resistant audit log of security-relevant events,[3] which can be examined for evidence of attacks on the system's security (audit).

The single system image requirement further implies that:

- Users must be able to gain access to all resources in the distributed system through a single signon (that is, a single presentation of a user name and password to the system), regardless of where in the system the resources are stored and which resource manager protects them.
- Users should be able to change passwords from any machine in the system and be assured that the new password is valid for signon at all machines in the network shortly after the password is changed.
- Users should be able to administer access permission information for all resources they own, regardless of where in the system those resources are stored and which resource manager protects them.
- System administrators should be able to administer users and policy in a single logical database through a single interface.

The Open Blueprint structure includes several advanced security features that are not mandated by government C2 requirements, including:

- An application-level nonrepudiation service
- Trusted path protection on all Open Blueprint platforms
- Local media data confidentiality protection for servers and portable machines
- Support for separation of duties and least privilege

To achieve the equivalent of C2 single system image security in a distributed environment, it is necessary to:

- Secure each host
- Secure the enterprise distributed systems using local security services and an appropriate combination of Open Software Foundation (OSF) Distributed Computing Environment (DCE) technology and standard Internet security technology

- Secure end-to-end client/server and peer-to-peer computing through any number of trusted and untrusted networks, including the Internet, based on emerging Internet standards.

  As Internet technologies continue to evolve, the concept of an enterprise will expand to incorporate instances of single system images that span sets of trusted and untrusted networks. The enterprise will not necessarily own or control the underlying system resources. The key enabling elements that make this possible are:

  – Secure networking at the packet level, for example, IPSec

  – Secure directory services such as Lightweight Directory Access Protocol (LDAP) and X.500

  – Secure credentials, for example, X.509v3 certificates and the corresponding certificate infrastructure

This strategy includes the use of the Tivoli Management Environment 10 Framework (TME 10)[4] to provide single security administration and management services.

The Open Blueprint Security Architecture

The Open Blueprint security architecture provides the means to implement the Open Blueprint security strategy. The security services described in the Open Blueprint architecture are provided by multiple Security resource managers.

Figure 4 illustrates the Open Blueprint security architecture. The figure represents an enlargement of the Distributed Systems Services and includes components from the Local Operating System Services.

The Open Blueprint security architecture can be applied to both standalone and distributed systems. The architecture also provides end-to-end security through the Internet.

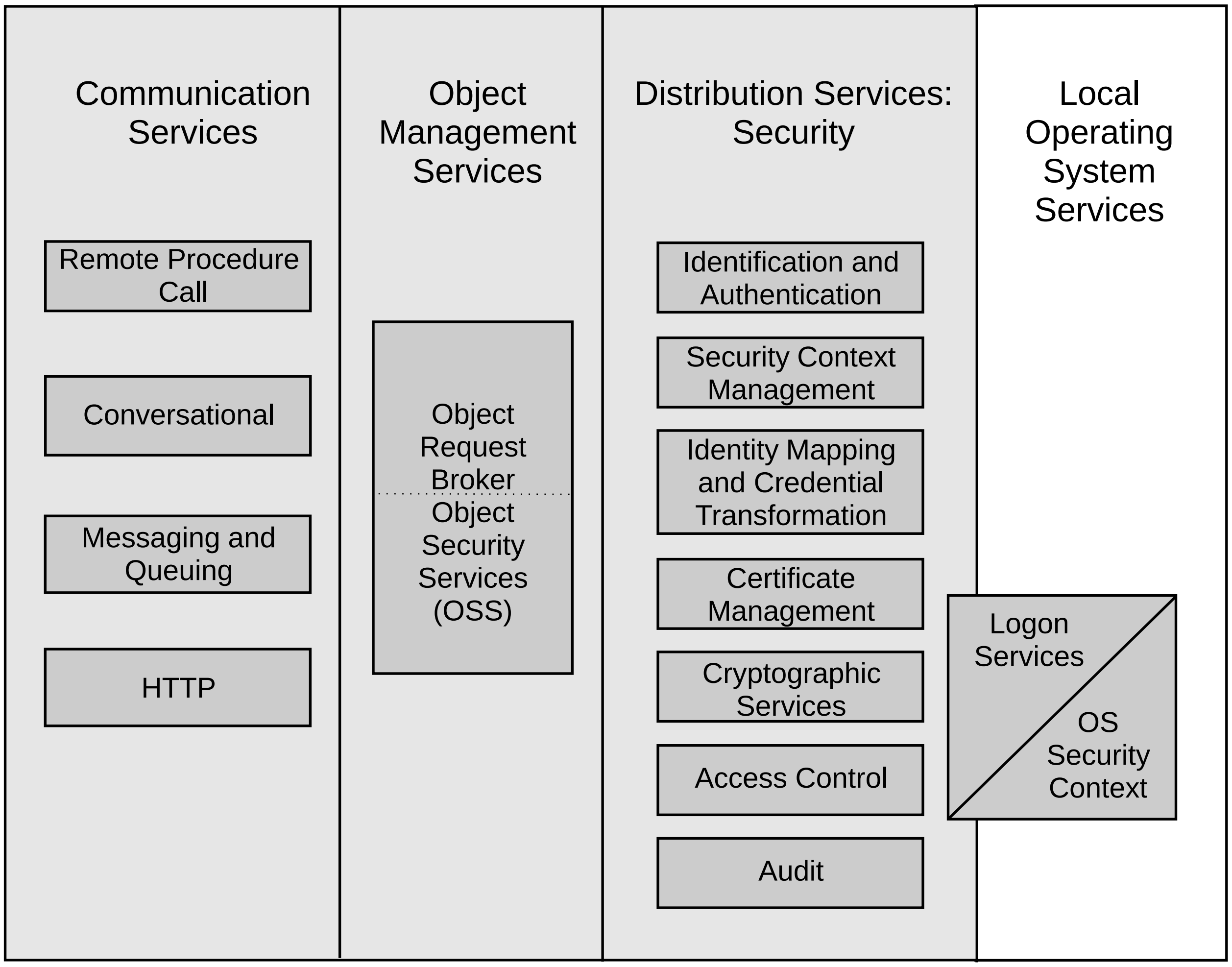

Communication Services
Remote Procedure Call
Conversational
Messaging and Queuing
HTTP
Object Management Services
Object Request Broker
Object Security Services (OSS)
Distribution Services: Security
Identification and Authentication
Security Context Management
Identity Mapping and Credential Transformation
Certificate Management
Cryptographic Services
Access Control
Audit
Local Operating System Services
Logon Services
OS Security Context

Figure 4. Open Blueprint Security Architecture

Open Blueprint Distributed Systems Services Security Components

The left three columns of Figure 4 illustrate the Open Blueprint Distributed System Services security components. These components support the public key infrastructure (PKI) required to extend enterprise systems into public (untrusted) networks such as the Internet. Secure communications are provided by Open Blueprint Communications Services, and secure object messaging is provided by the Open Blueprint Object Management Services through the use of OMG Object Security Services. The Object Security Services access the Open Blueprint Security resource managers.

The architecture allows for choices in implementations. For example, DCE Kerberos support could be used as the authentication mechanism in some configurations, and a public-key-based mechanism could be used in others. However, not all mechanisms provide equivalent levels of function.

Security Resource Managers

The third column of Figure 4 on page 19 illustrates the Security resource managers contained in the Open Blueprint Distribution Services. These resource managers serve both an enterprise-wide distributed security single-system image and Internet end-to-end security needs.[5]

Identification and Authentication Resource Manager: *Authentication* proves a user's entitlement to use a particular system-defined identity. Only after a user is authenticated will the system build a security context representing the user's identity. Every action the user asks the system to perform is performed in the context created at the time of authentication.

When authentication is performed successfully, a user is assigned *credentials*. In addition to the user's identify, credentials contain:

- Public and private key information
- Secret key information.
- Information that can be used by the Access Control resource manager to make access decisions. For example, credentials can contain a user's group membership information.

When credentials are received from the Identification and Authentification resource manager, they are placed in the local security context of the user.

The Identification and Authentification resource manager provides the following services:

- **Logon Service**. Logs a user onto all necessary local and network services. This service is also responsible for providing *single signon* (SSO), which is a single, integrated view of the logon process.
- **Network Security Context Management Service**. Provides services to Communication resource managers for authentication of parties and session key exchange during communication session establishment. The GSSAPI programming interface externalizes these functions. (Refer to “Generic Security Service Application Programming Interface” on page 21 for information about GSSAPI.)
- **Message Protection Service**. Uses session keys to protect data that is passed between communication partners. The GSSAPI programming interface externalizes these functions. (Refer to “Generic Security Service Application Programming Interface” on page 21 for information about GSSAPI.)

Two distinct cryptographic technologies can be used to implement the Identification and Authentification resource manager in a distributed, networked environment:

- Shared secret key (symmetric key)
- Public key (asymmetric key-pair)

These two authentication and identification technologies require different implementations of the Identification and Authentification resource manager services. (For information about symmetric and asymmetric keys, refer to “Cryptography and Certificates” on page 13.)

Two additional Identification and Authentification resource manager services are used in a shared secret key environment:

- **Registry Service**. Stores and maintains user information, including user passwords. The registry service typically stores passwords in a one-way encrypted format. The DCE Registry is federated as part of the Open Blueprint directory. The Open Blueprint directory is accessed through the use of the LDAP protocol. (For information about the Open Blueprint directory, see the *Open Blueprint Directory Resource Manager* component description paper.)

- **Network Authentication Service**. The trusted third party that is invoked for all authentication requests. The network authentication service distributes tamper-resistant security credentials to clients. Clients then provide these credentials to resource managers with resource requests. The network authentication service ensures that the security credentials are readable by resource managers (so that the security context information can be used to authorize resource access requests); and ensures that the credentials are neither readable nor modifiable by users or malicious software on network client systems.

Generic Security Service Application Programming Interface: The Generic Security Service Application Programming Interface (GSSAPI) provides a programming interface to the network security context management and message protection services of the Identification and Authentification resource manager. The GSSAPI is based on IETF and X/Open standards and is included in release 1.1 of Open Software Foundation (OSF) Distributed Computing Environment (DCE).

Applications can use the GSSAPI to transmit their own data across the network. GSSAPI provides tokens for initial authentication and for encapsulating data so the data can be sent out securely (either with data privacy or integrity) across the network. GSSAPI is used by network applications that have secure association requirements that cannot be met by any of the secure communication services. These requirements include:

- Complex pre-negotiations of mechanism capabilities and application secure channel requirements. GSSAPI has a high degree of flexibility in specifying security quality of service parameters such as mutual or one way authentication, replay detection, and checking for out of sequence messages.
- Advanced features such as delegation.

Security Context Management Resource Manager: The Security Context Management resource manager maintains the security state information that is associated with execution contexts (processes and threads). Each execution context can be associated with security state information belonging to many secure subsystems. For example, a single thread can have a DCE login context, a Netware Credential, and a local operating system context. Figure 5 on page 22 illustrates overall execution context support, including the structure of the data managed by the Security Context Management resource manager.

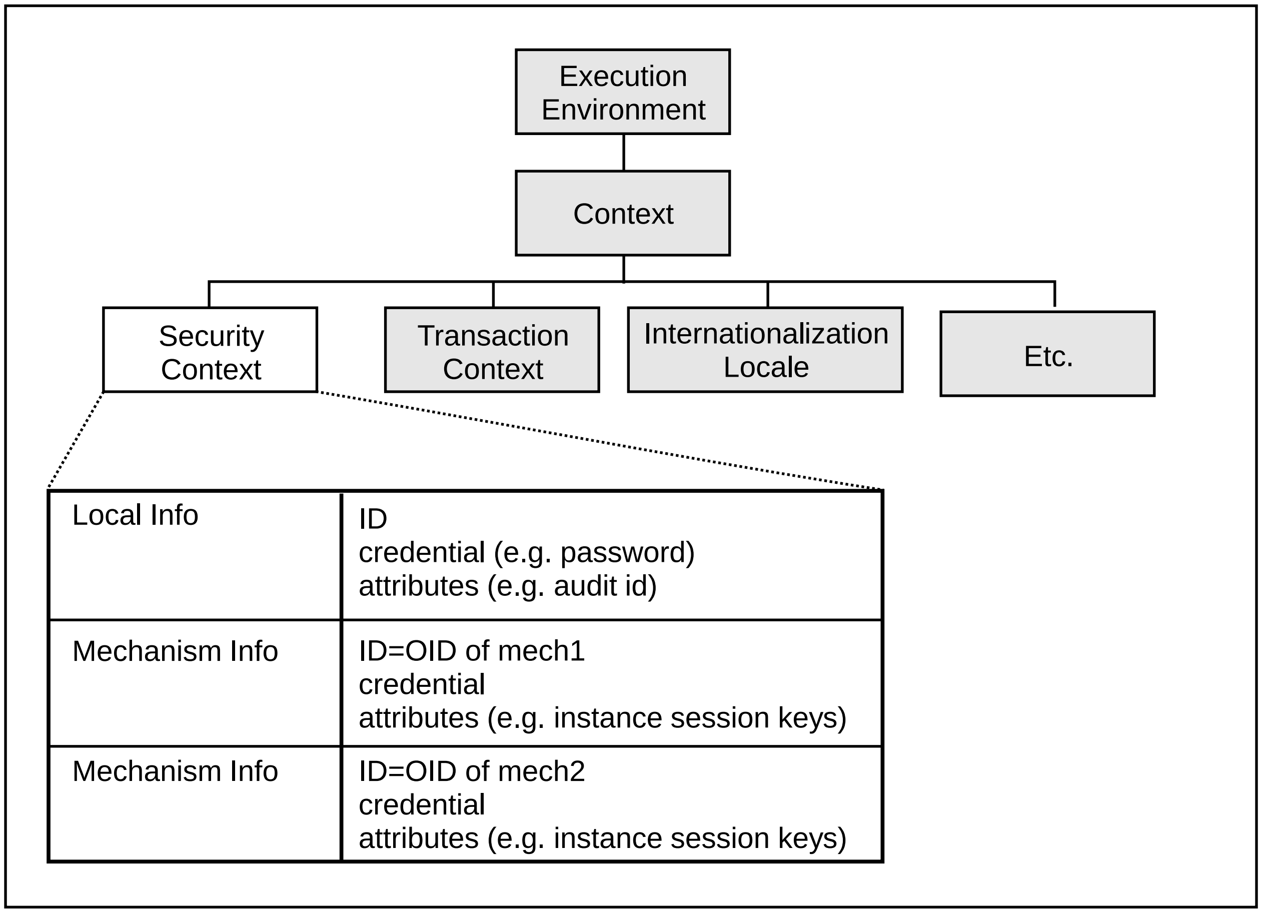

Execution Environment
Context
Security Context
Transaction Context
Internationalization Locale
Etc.
Local Info
ID
credential (e.g. password)
attributes (e.g. audit id)
Mechanism Info
ID=OID of mech1
credential
attributes (e.g. instance session keys)
Mechanism Info
ID=OID of mech2
credential
attributes (e.g. instance session keys)

Figure 5. Data Structures Managed by the Security Context Management Resource Manager

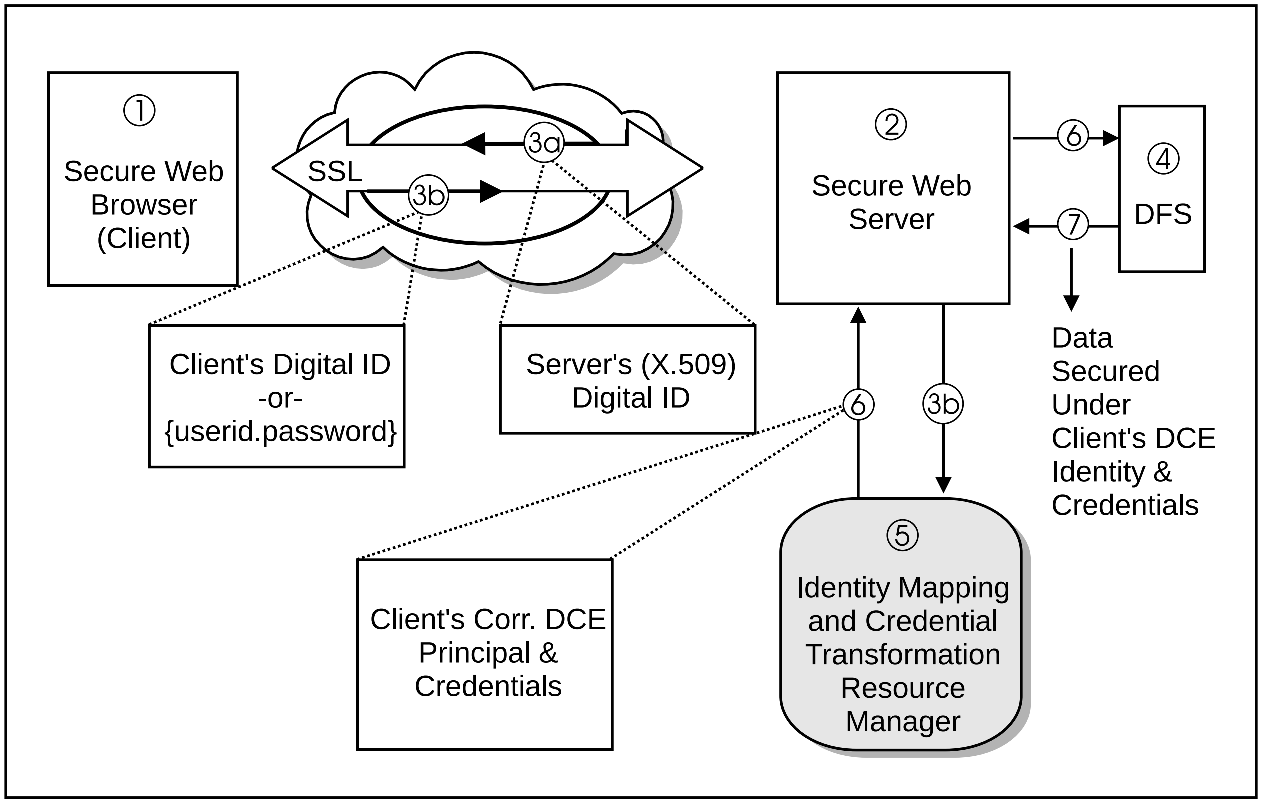
The Security Context Management resource manager can store different types of state information, including:

- Local IDs
- Local ID attributes such as logon password, logon user name, logon domain, and so on
- Security mechanism IDs such as DCE Kerberos, and so on
- Credentials
- Session keys, which are used in a network context

Identity Mapping and Credential Transformation Resource Manager: The Identity Mapping and Credential Transformation resource manager provides any-to-any security context (identity+credentials) transformations. For example, trusted servers can use this resource manager to derive a useable, security context from a different context. This capability provides the ability to recognize and honor new types of security contexts.

Usage Scenario: Figure 6 on page 23 shows an example of some of the interoperability services needed to provide end-to-end security services through public networks. In this example, a secure Web browser (1) establishes a secure session (a mutual authentication and encrypted “tunnel” through the Internet) with a secure Web server (2), using Secure Sockets Layer (SSL) Version 3. The establishment of the secure session includes mutual authentication between the client and the server. First, the server transmits its Digital ID (3a) to the client, which the client then authenticates. The client then sends its Digital ID or a userid/password pair to the server for authentication (3b). In this scenario, the client sends a Digital ID. The ID is a X.509v3 certificate, which includes the client's distinguished name and public key.

Now the client invokes a request for secure data supported by the Distributed File System (DFS) (4). Currently, DFS is not aware of X.509v3 certificates for purposes of authentication, identification, and authorization decisions. DFS requires its clients to have DCE security credentials. Therefore, the secure Web server (2) must use the Identity Mapping and Credential Transformation resource manager (5) which accepts the client's credential (3b) as input, and returns a DCE security credential (6). The DCE security credential is then used to make the DFS request on behalf of the client and to return the data (7).



①
Secure Web Browser (Client)
SSL
3a
3b
②
Secure Web Server
⑥
④
DFS
⑦
Client's Digital ID -or- {userid.password}
Server's (X.509) Digital ID
6
3b
Data Secured Under Client's DCE Identity & Credentials
⑤
Identity Mapping and Credential Transformation Resource Manager
Client's Corr. DCE Principal & Credentials

Figure 6. Example of Identity Mapping and Credential Information

Certificate Management Resource Manager: Certificates, also referred to as digital IDs, are the cornerstone of Internet security. They provide a robust mechanism to associate a name with a public key. An infrastructure that supports certificates must be in place to enable the use of certificates.

Some of the following information regarding the Certificate Management resource manager services is based on The Open Group Architecture for Public Key Infrastructure (APKI) initiative. APKI information can be found at URL: `http://www.xopen.org/public/tech/security/pki/index.htm`. Additional information about certificate basics can be found in “Public-Key Certificates” on page 14.

Run Time: The run time component handles the verification, caching and storage of certificates. The run time component also provides parsing functions to extract distinguished names and public keys from certificates.

Certification Authority: The certification authority component provides two fundamental certificate services: generation and revocation. The certification authority is architecturally subdivided[6] into the following sub-components:

- **Certification authority server**: Issues certificates based on requests. Certification authority security precautions vary depending on the certification authority's scope. For a small company, a software solution on a controlled-access server might be sufficient. For a root certification authority for several countries, physical security of the computer, cryptographic hardware protection of keys, and multilevel, mandatory access control operating systems might be appropriate. The certification authority server must also be capable of generating and distributing keys pairs and of generating and distributing certificate revocation lists (CRLs). The certification authority server can optionally place generated certificates and certificate revocation lists in a certificate repository.

- **Certification authority front end (agent)**: Required when extra security is required for the certification authority server. The certification authority front end deals with the local registration authorities (described below) and provides certificate requests to be forwarded to the certification authority server under careful controls.
- **Local Registration Authority**: The local registration authority is the client agent that provides interfaces for requesting the generation of key pairs and corresponding certificates, requesting certification of existing public keys, and requesting revocation of existing certificates. Communications between the location registration authority and the certification authority server (or certification authority agent) must be secure.
- **Publication Authority**: The publication authority provides interfaces through which the certification authority servers and agents can place certificates and certificate revocations lists into repositories or transmit them directly to requestors.

Cryptographic Services Resource Manager: The Cryptographic Services resource manager provides all cryptographic services. These services include:

- Encryption and decryption (multiple algorithms and key lengths)
- Hash generation and validation
- Signature generation
- Certificate validation
- Key generation
- Random number generation
- Key import, export and distribution

Many of these functions are applicable to both symmetric (secret) key and asymmetric (public/private) key-pair cryptography. For additional information about cryptography, refer to “Cryptography and Certificates” on page 13.

Access Control Resource Manager: The Access Control resource manager supports authorization. *Authorization* is the process of checking the credentials associated with a resource access request against an authorization profile to determine whether to grant or deny the request. The system's collection of authorization profiles is its security policy. There are several different kinds of authorization profiles. One type of authorization profile used in the Open Blueprint is called an access control list. An access control list describes the authorization policy that governs all system users' access to a single resource.

The Access Control resource manager enforces security policies by evaluating user and programmatic principal access requests against an access control policy and returning an access decision. Applications and system components that own protected resources consult the Access Control resource manager before servicing callers' requests for access to protected resources, and enforce the decision it returns.

Access control allows an organization to protect critical resources by limiting access to authorized and authenticated users. Depending on the environment, access can be controlled by the resource owner, or it can be controlled automatically by the operating system. The resource owner can specify who can access the information, how it can be accessed, when it can be accessed, and under what conditions it can be accessed (for example, when executing specific applications, programs, or transactions). The functional goal is to assure that security is maintained for resources, whether they are in central, distributed, or mobile systems.

Audit Resource Manager: Security-relevant events are recorded in a security audit log to help detect attempts (successful or unsuccessful) to compromise security. The Audit resource manager provides an audit application programming interface (API) that enables the collection of audit records from other resource managers. These other resource managers provide auditing services that monitor security-relevant events on the resources that they manage. A set of auditable events is provided from which the administrator can select appropriate activities to be audited.

The Audit resource manager supports access to audit records by audit reduction tools that integrate audit records from multiple nodes and resource managers. Audit records have defined formats and are stored in logs to which access is controlled by relevant security policy. Audit records include one or more time stamps. Ideally, the time stamp reflects a consensus time value that is shared across the network domain. (The consensus time value is provided by the Time resource manager.) At a minimum, the consensus time value includes the locally-available times from the users and the provider of the audit service. IBM is an active participant in the X/Open Distributed Audit Services (XDAS) workgroup.

Open Blueprint Local Operating System Services Security Components

The rightmost column of Figure 4 on page 19 illustrates the Open Blueprint local security components.

- **Local Security Context Services**. Local security context services provide the foundation on which Open Blueprint security is built. Each platform's local security context component is responsible for ensuring the following:
  - No user gains access to the system without first being authenticated (logging on) and being associated with a security context.
  - Each action initiated within the system is identified by the security context of the initiating user or identified program.

  To authenticate users, the local security context component often consults a local or remote registry service.
- **Logon Services**. Refer to “Identification and Authentication Resource Manager” on page 20 for information about logon services.

Secure Communication Services

When two machines share a user's network security context, it is possible to use that context's cryptographic keys to establish a secure association between the machines on the user's behalf. A secure association is a virtual communication channel that uses cryptography to protect the information passing through it against modification or disclosure (or both).

Designing a secure virtual communications channel is difficult. There is no reliable, systematic methodology for designing secure communication protocols. Experience has shown that a very large number of commercially deployed protocols that were originally presumed to be secure are in fact subject to a wide range of effective attacks. Generally, application programmers should not be required (or perhaps even allowed) to design secure protocols; instead, the Open Blueprint provides secure communication services that implement secure associations “under the covers.”

The Open Blueprint security architecture includes a variety of secure communication services. Each of these services provides the following:

- **Location transparency**.

The programs using a secure association to communicate with one another only need to know their partners' names; they do not need to know their partners' machine names or transport network addresses.

- **An application-level programming abstraction**

  Writing a program using a secure communication service for communication does not require any communications programming expertise; the task of communicating over a secure association is an application-level function such as calling a procedure or placing a data structure on a queue. In addition, using a secure communication service does not require security programming expertise; all that is required of application programmers is that they specify what quality of protection should be applied to the communication. For example, programmers might set up a secure conversation, specifying at the time the connection is allocated that they want the conversation to provide strong data integrity protection but no data confidentiality protection.

Secure communication services are provided by the following resource managers:

- **Remote Procedure Call**, which implements the OSF DCE Authenticated RPC specification.
- **Conversational**, which implements the IBM Common Programming Interface for Communications (CPI-C) interface.
- **Messaging and Queuing**, which is based on the IBM message queue interface (MQI).
- **HyperText Transfer Protocol (HTTP)**, which includes HTTPS[7], a de facto standard developed by Netscape to make HTTP flows secure. HTTPS is the use of HTTP over the Secure Sockets Layer (SSL). SSL Version 3 is based on the Internet Engineering Task Force (IETF) Transport Layer Security (TLS) specification.

Using a secure communication service isolates applications from both the underlying transport protocols and underlying network security protocols (but not from security credential formats).

Some applications cannot take advantage of the secure communication services. For example, one application might require the use of a transport protocol that is not supported by secure communication services. Another might require an application-level programming abstraction that is different from that provided secure communication services.

Protocol stacks and applications that cannot use secure communications services can use the GSSAPI to establish secure associations without having to deal with the complexities of communications programming and secure protocol design.

Secure Object Management Services

In the Open Blueprint structure, secure object messaging is based on the Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) Version 2.0 Object Security Services. Figure 7 on page 27 illustrates how an application uses the CORBA Security APIs. In the Object Security Services environment, the CORBA security functions are built using Open Blueprint Security resource managers. (See URL: http://www.omg.org for additional information about CORBA security.)

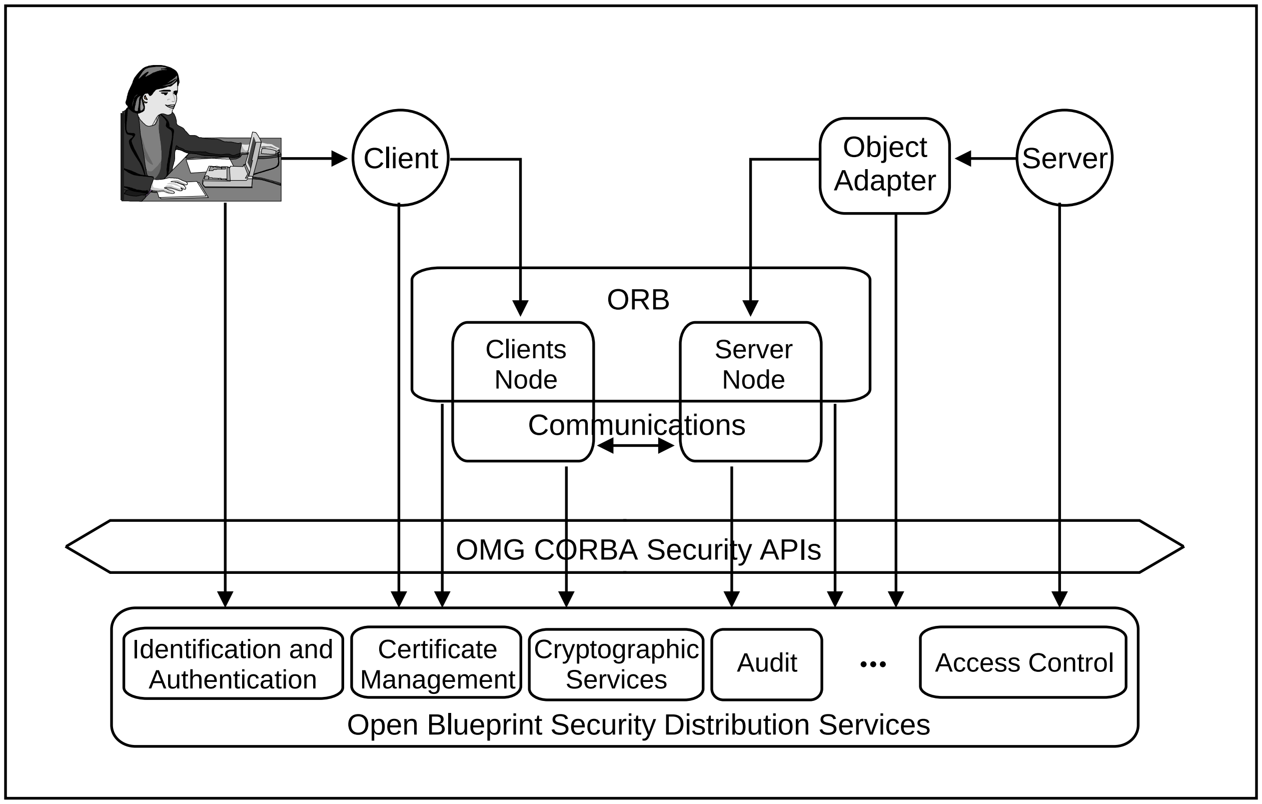


Figure 7. Overview of OMG CORBA Security

Example Security Flows

This section describes the interaction between the various Security resource managers that occurs in providing support for common security flows.

Logon (Secret Key)

The following scenario reflects a user logging on to both the local operating system and a DCE environment through a single logon process. The flows through the security resource managers to support logon in this environment are illustrated in Figure 8.

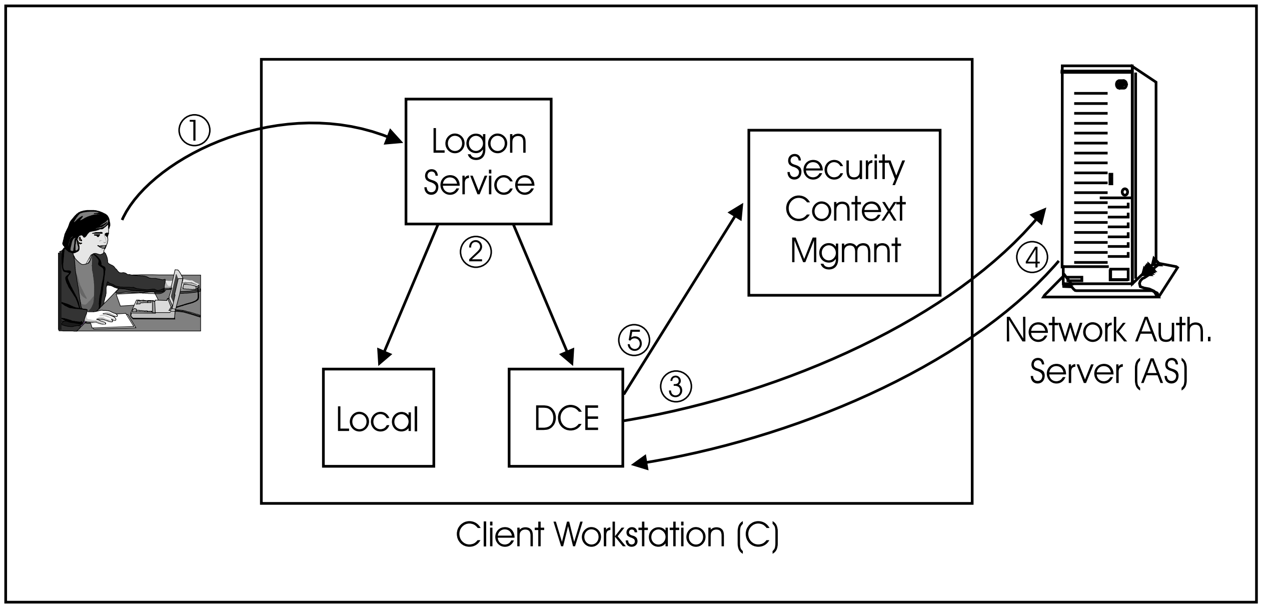


Figure 8. Logon Flow for Secret Key Environment

1. A user logs on to the system by invoking the local operating system logon service. The user's ID and password is input to the logon service.
2. The logon service retrieves the user's logon profile, which defines which logon services to invoke for a specific user, invokes the local logon service, then invokes the DCE network logon service.[8]
3. The DCE network authentication service is invoked to authenticate the user and obtain a DCE-based network security context for the user. The password never flows in the clear across the wire during the authentication flows between the client and the DCE authentication service.
4. A DCE-based credential (ticket) is acquired after successful authentication. This credential allows the client to issue subsequent requests to the network authentication service for credentials for other resource managers.
5. On successful completion, the Security Context Management resource manager is invoked to create a network security context in which the credentials are deposited.

Logon (Public Key)

The flows through the security resource managers to support logon in a public key environment are illustrated in Figure 9 on page 30. Retrieval of public and private key information is, in many cases, integrated with the local logon service.

In the following scenario, a smartcard hardware device is used to store a user's password and private and public key information locally.[9]

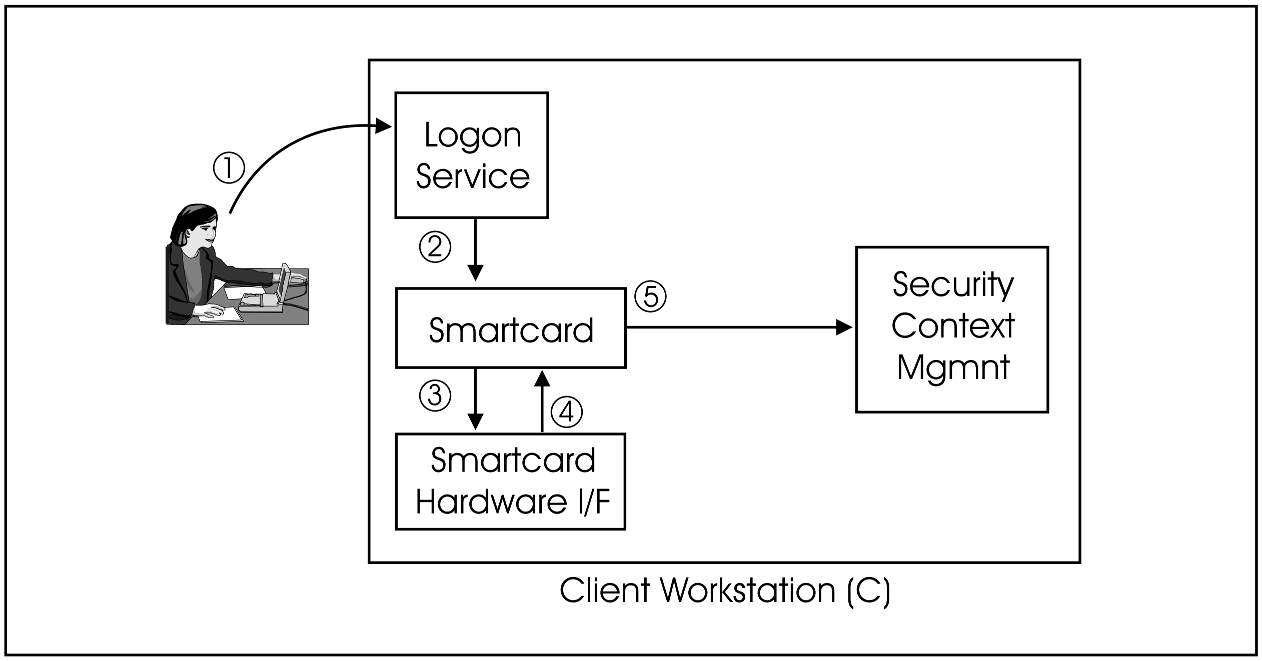


Figure 9. Logon Flow for Public Key Environment

1. A user logs on to the system by invoking the local operating system's logon service. The user's ID and password is input to the logon service.
2. The logon service retrieves the user's logon profile and invokes the smartcard logon service to validate the user. The logon service defines which logon services to invoke for a specific user.
3. The smartcard hardware interface is invoked with the ID and password.
4. A successful validation indication and a handle to the user's private and public key information is returned to the smartcard logon service.
5. The Security Context Management resource manager is invoked to create the security context, which contains the handle to the private and public key information.

Client-Server Authentication (Secret Key)

After logon is completed successfully and a security context is established, requests can be issued to resource managers that are running on different machines. Before requests are accepted, authentication takes place between the requester (for example, the resource manager client running on behalf of the user) and the resource manager. Figure 10 on page 31 illustrates the authentication flow for the secret key environment:[10]

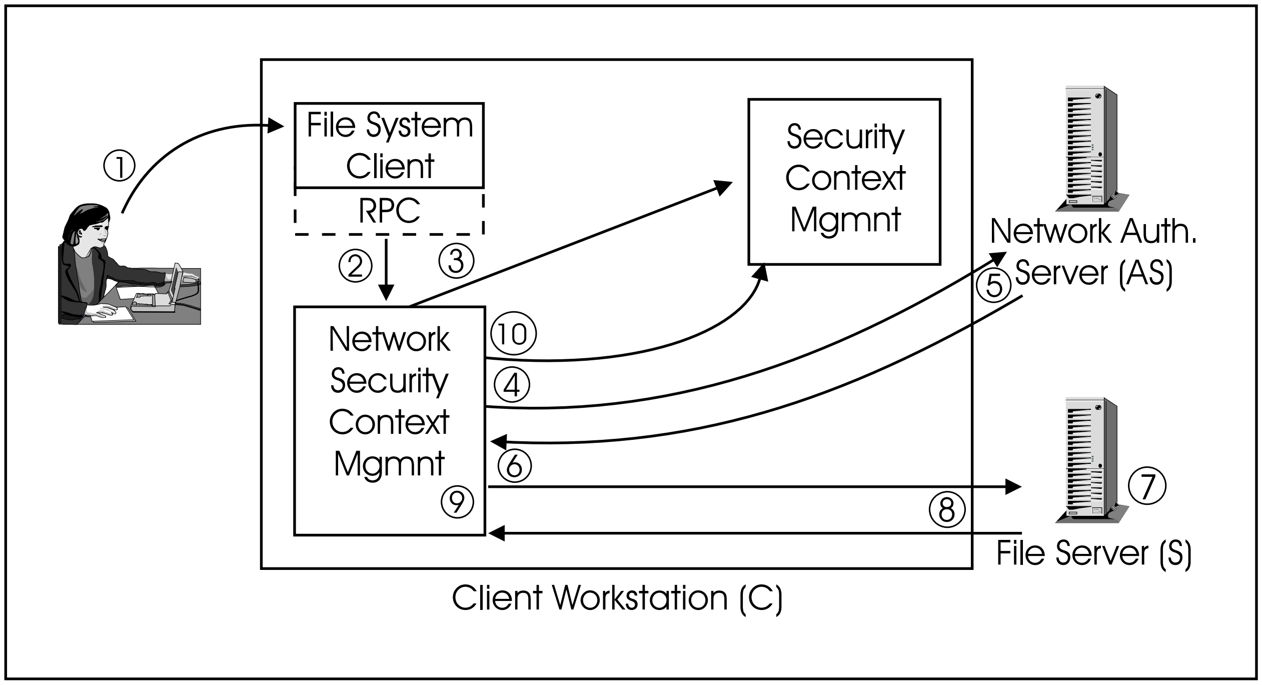


Figure 10. Client/Server Authentication Flow for Secret Key Environment

1. A user issues a request to open a file on a remote server S. The file system client uses authenticated remote procedure call (RPC) to communicate with the remote file server.
2. Because this is the first time Server S has been invoked, an RPC session does not exist between Client C and Server S. As a result, the RPC component invokes the network security context management service of the Identification and Authentification resource manager to mutually authenticate Client C and Server S.
3. The Security Context Management resource manager is invoked to obtain the credential for the network authentication service (AS) that was created during logon.
4. Client C requests credentials from the network authentication server for Server S.
5. The required credentials for Server S arrive in two parts:
   - One part for Client C, which is encrypted in Client C's session key, which is shared with the network authentication service
   - One part for Server S, which is encrypted in Server S's secret key

   A session key is included in both parts. A session key is used to securely transmit data between Client C and Server S after authentication is established.
6. Client C sends Server S the authentication credential along with an authenticator, which is encrypted in the session key. The authenticator uniquely identifies Client C.
7. Server S obtains the session key from the authentication credential using its secret key for decryption. Server S decrypts the authenticator and authenticates Client C.
8. Server S returns an encrypted response to Client C.
9. Client C decrypts the response using the shared session key and authenticates Server S.
10. The Security Context Management resource manager is invoked to add the session context, which includes the server name, session key, and so on. File-system-specific requests can now flow between Client C and Server S.

Client-Server Authentication (Public Key)

After logon is completed successfully and a security context is established, requests can be issued to resource managers running on different machines. Before requests are accepted, authentication takes place between the requester (for example, the resource manager client running on behalf of the user) and the resource manager. Figure 11 on page 32 illustrates the authentication flow for the public key environment[11]

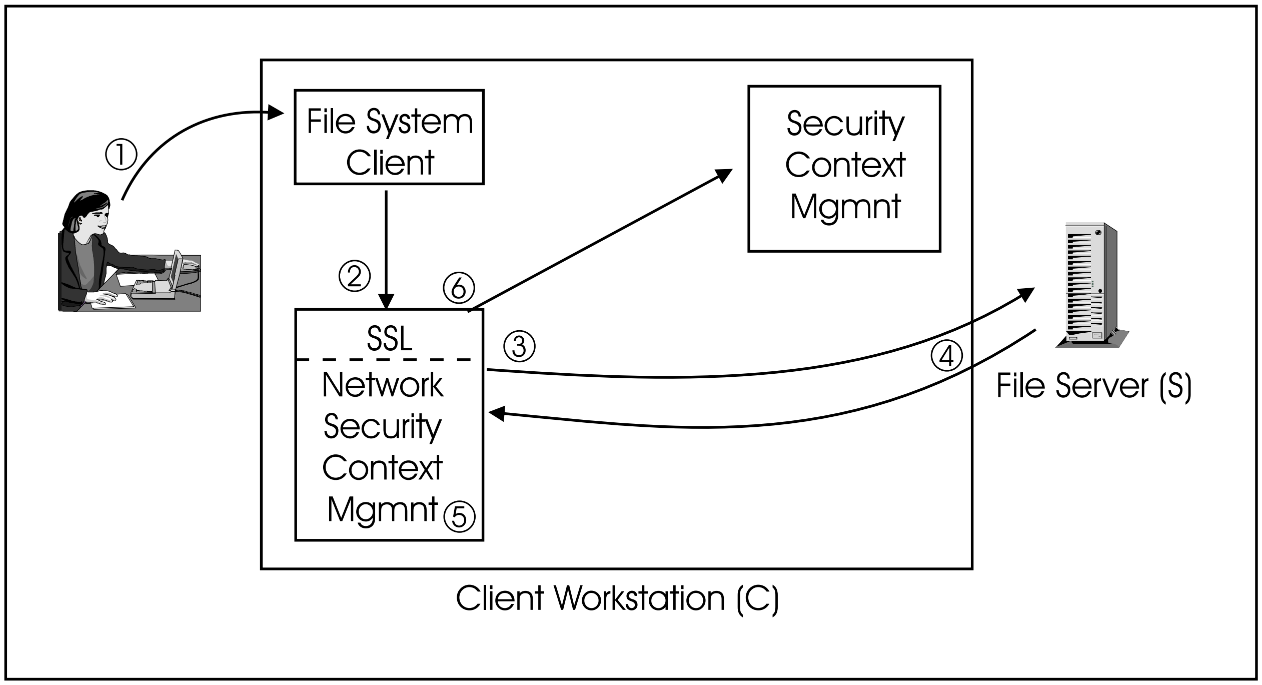


Figure 11. Client/Server Authentication Flow for Public Key Environment

1. A user issues a request to open a file on a remote server S.
2. The file system client uses the secure SSL communication protocol to communicate with the remote file server. Because this is the first time Server S has been invoked, an SSL session does not exist between Client C and Server S. As a result, the Network Security Context Management component of SSL is invoked to authenticate Server S to Client C.[12]
3. Client C communicates in the clear with Server S to obtain the public key certificate of Server S.
4. Server S returns its public key certificate along with auxiliary information. The auxiliary information contains the session key that will be used to securely transmit data between Client C and Server S after authentication is established. The auxiliary information is encrypted using Server S's private key and Client C's public key.
5. Client C receives the data from Server S and:
   - Validates the public key certificate of server S. Client C uses the public key of a certification authority, which it has cached locally, to validate the public key certificate of Server S. For authentication to be successful, Client C and Server S must share a common certification authority.
   - Obtains the session key by decrypting the auxiliary information using its own private key and Server S's public key.

6. The Security Context Management resource manager is invoked to add the public key certificate and the session key for Server S to the security context. File system specific requests can now flow between Client C and Server S.

For Server S to authenticate Client C, steps 3-6 are repeated with the client/server roles reversed.

Creating a Confidential File

The following scenario illustrates the interaction of various Security resource managers during the creation of a confidential file such as a file that contains the salaries of members in a department. The scenario has the following characteristics:

- Access to the directory where the file is stored is restricted.
- The information must be protected when it flows over the wire to the file server.
- Access to the directory where the file is stored is audited.

This scenario assumes that a user has logged on to the network and that mutual authentication has already taken place between client C and file server S. Figure 12 illustrates the security flows that take place to provide data protection, access control, and auditing of a request to create a remote, confidential file:[13]

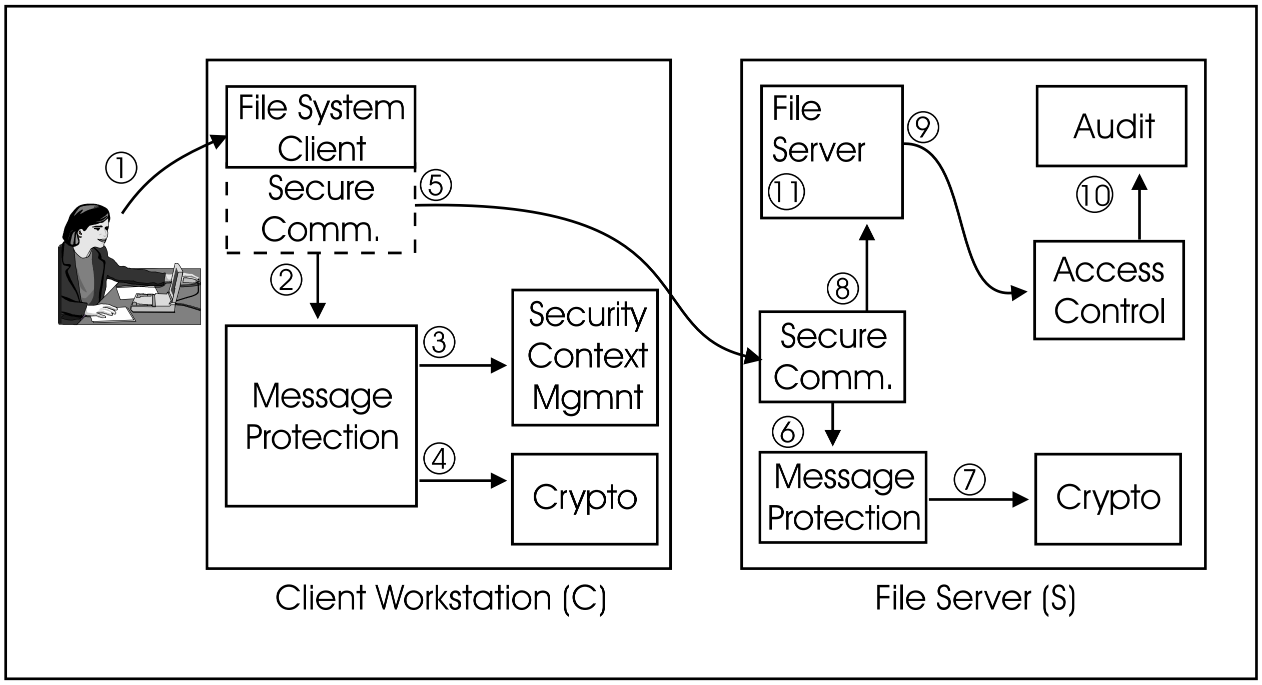


Figure 12. File Creation Security Flows

1. A user issues a request to create a confidential file on Server S.
2. The secure communications protocol that is used by the file system client invokes the Identification and Authentification resource manager message protection service to ensure that data is encrypted when it flows over the wire to Server S.
3. The Security Context Management resource manager is invoked to obtain the shared session key for Client C and Server S.
4. The Cryptographic Services resource manager is invoked to encrypt the file data using the shared session key.

5. The "create" request and file data are sent to the communication services of Server S. User credentials are also sent on the request. These credentials are used during authorization on the server.
6. The communication service of Server S invokes the message protection service to decrypt the file data.
7. The Cryptographic Services resource manager is invoked to decrypt the file. The message protection service passes the shared session key for Client C and Server S as input.
8. After the data is decrypted, the "create" request is sent to the file server.
9. The file server invokes the Access Control resource manager to determine if the user has the authority to create the file. The Access Control resource manager uses the credentials passed by Client C to determine access rights.
10. If auditing of the directory where the file is being created is active, the Access Control resource manager invokes the Audit resource manager to audit the event. If auditing of the file itself is required, a separate administrative request to enable audit events for this file must be issued.
11. If the user has appropriate access rights to the directory where the file is being created, the file server invokes the local file system to create the file. In most cases, the file inherits the access control list of its parent directory. If a different access control list for this file is required, a separate administrative request to change the access control list must be issued.

---

[1] *Department of Defense Trusted System Computer Evaluation Criteria*, DOD 5200.28STD, Department of Defense, Washington, D.C. (also referred to as the Orange Book).

[2] This definition of integrity is taken from the *COMPUSECese: Computer Security Glossary*; First Edition, NCSC publication NCSC-WA-001-85. October 1, 1985.

[3] See the "Orange Book," protection class C2, for information about what events should be considered security-relevant.

[4] For additional information about the Tivoli Management Environment 10 Framework (TME 10), visit URL: http://www.tivoli.com.

[5] A number of other services and applications provide additional or alternative security functions. For example, SSL and IPSec are Open Blueprint Network Services functions that provide identification and authentication functions.

[6] The certification authority sub-components can be delivered as discrete distributed software entities, or they can be packaged as sets of sub-components.

[7] SHTTP, or Security HTTP, is another protocol that was developed by the CommerceNet consortium to address the lack of security on the World Wide Web. However, SHTTP is being replaced by HTTPS.

[8] For simplicity, the flows for the local logon are not illustrated.

[9] In the absence of a smartcard, a local logon service can authenticate a user and retrieve the user's public and private key information from password-protected storage.

[10] For simplicity, flows to the Cryptographic Services resource manager to encrypt and decrypt data are not illustrated.

[11] For simplicity, flows to the Cryptographic Services resource manager to encrypt and decrypt data are not illustrated.

[12] The Network Security Context Management component of SSL is a logical, integrated function that does not externalize a general-use interface.

[13] These flows apply to both secret key and public key environments.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM
IBMLink
MVS
Open Blueprint

The following terms are trademarks of other companies:

CORBA	Object Management Group, Incorporated
DCE	The Open Software Foundation
Java	Sun Microsystems, Incorporated
Lotus Notes	Lotus Development Corporation
Open Software Foundation	Open Software Foundation, Incorporated
OSF	Open Software Foundation, Incorporated
Tivoli Management Environment	Tivoli Systems Inc., an IBM Company
X/Open	X/Open Company Limited in the U.K. and other countries

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the U.S. and other countries.

Communicating Your Comments to IBM

If you especially like or dislike anything about this paper, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply. Feel free to comment on specific error or omissions, accuracy, organization, subject matter, or completeness of this paper.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

- If you prefer to send comments by FAX, use this number:

  United States and Canada: 1-800-227-5088.

- If you prefer to send comments electronically, use one of these ID's:
  - Internet: **USIB2HPD@VNET.IBM.COM**
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink: **CIBMORCF at RALVM13**

Make sure to include the following in your note:

- Title of this paper
- Page number or topic to which your comment applies

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC23-3936-01